EFFICIENT ALGORITHMS FOR HIGH-DIMENSIONAL DATA-DRIVEN SEQUENTIAL DECISION-MAKING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yilun Chen

May 2021

EFFICIENT ALGORITHMS FOR HIGH-DIMENSIONAL DATA-DRIVEN SEQUENTIAL DECISION-MAKING

Yilun Chen, Ph.D.

Cornell University 2021

The general framework of sequential decision-making captures various important real-world applications ranging from pricing, inventory control to public healthcare and pandemic management. It is central to operations research/operations management, often boiling down to solving stochastic dynamic programs (DP). The ongoing big data revolution allows decision makers to incorporate relevant data in their decision-making processes, which in many cases leads to significant performance upgrade/revenue increase. However, such data-driven decision-making also poses fundamental computational challenges, because they generally demand large-scale, more realistic and flexible (thus complicated) models. As a result, the associated DPs become computationally intractable due to *curse of dimensionality* issues.

We overcome this computational obstacle for three specific sequential decision-making problems, each subject to a distinct *combinatorial constraint* on its decisions: optimal stopping, sequential decision-making with limited moves and online bipartite max weight independent set. Assuming sample access to the underlying model (analogous to a *generative model* in reinforcement learning), our algorithm can output ϵ-optimal solutions (policies/approximate optimal values) for any fixed error tolerance ϵ with computational and sample complexity both scaling polynomially in the time horizon T, and essentially independent of the underlying dimension. Our results prove for the first

time the fundamental tractability of certain sequential decision-making problems with combinatorial structures (including the notoriously challenging high-dimensional optimal stopping), and our approach may potentially bring forth efficient algorithms with provable performance guarantee in more sequential decision-making settings.

BIOGRAPHICAL SKETCH

Yilun Chen was born and grown up in Shanghai, China. After obtaining his B.S. degree in Mathematics from Peking University in 2014, he started his PhD at Georgia Tech in Atlanta. In 2017, he transferred to Cornell University and earned his PhD degree there in 2021.

To my family and my advisor.

ACKNOWLEDGEMENTS

There are many people I would like to extend my gratitude to, without whom the completion of this thesis would not be possible.

First, I want to thank my advisor, David Goldberg, for being a fantastic mentor and a great friend. I feel fortunate to have him as my PhD advisor.

I would also like to thank the rest of my committee members, Sid Banerjee, Jim Dai, Bobby Kleinberg and Shane Henderson for their insightful comments on this thesis.

I am grateful to the School of ORIE at Cornell and the School of ISyE at Georgia Tech. They provide me with the opportunity to conduct research, which leads to this thesis.

I want to thank all my friends, at Georgia Tech and at Cornell, from my undergrad and from my high school. I am lucky to have them throughout my PhD journey.

Finally, I would like to thank my parents, Mingliang Chen and Jian Zhou, and my girl friend Ruyu Chen, for their unconditional love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Many real-world decision-making processes are fundamentally *sequential*, where a sequence of decisions must be made in a chronological order, each based only on currently available information. Typical examples include investors' portfolio management, companies' pricing/inventory control and governments' pandemic management. Such sequential decision-making problems have long been studied in the operations and the computer science literature, often boiling down to solving stochastic dynamic programs (DP) or Markovian decision processes (MDP). Recently, amidst the ongoing data revolution, new, profound challenges are posed by the increasingly common desires of exploiting relevant data streams (e.g. past sales records, customers' personal information) to make better decisions. Such data-driven decision-making generally demands large-scale, more realistic and flexible models, which renders the associated DPs/MDPs computationally intractable due to *curse of dimensionality* issues. This raises the natural questions that, for what types of tasks can we hope of designing computationally tractable algorithm that yields provably near optimal decision-making policies? This thesis aims at providing a solution to this question.

To advance our rigorous study of the "hardness" of a task/the "complexity" of an algorithm, we first specify a general mathematical framework for sequential decision-making. We consider a discrete-time problem. Let T be the finite time horizon. Suppose $\mathbf{X} = (X_t,\ t = 1, ..., T)$ is a stochastic process that models the exogenous data streams. Suppose a decision maker (DM) can observe the evolution of $\mathbf{X}$ throughout T. In each time period $t \in [1, T]$, he/she needs to pick an action a_t from some action set $A_t(a_1, ..., a_{t-1}, x_1, ..., x_t)$, based only on

$x_1, ..., x_t$, the realized trajectory. A reward/cost $r_t(a_1, ..., a_t, x_1, ..., x_t)$ is generated following the action choice. A feasible policy $\pi : (\pi_1, ..., \pi_T) \in \Pi$ is formally defined as a set of T functions, where π_t maps a realized trajectory $(x_1, ..., x_t)$ to a probability distribution over the action set $A_t(a_1, ..., a_{t-1}, x_1, ..., x_t)$. The sequential decision-making problem concerns finding the best policy π, i.e.

$$\text{OPT} \triangleq \sup_{\pi \in \Pi} \left(\inf_{\pi \in \Pi} \right) E\Big[\sum_{t=1}^{T} r_t(a_1, ..., a_{t-1}, \pi_t(X_1, ..., X_t), X_1, ..., X_t) \Big], \tag{1.1}$$

where the expectation is taken over the randomness of $\mathbf{X}$ as well as π. This is a general framework for sequential decision-making: it captures almost all models studied in the literature, including MDPs in particular.

Within such a mathematical framework for sequential decision-making, an algorithm $\mathcal{A}$ should be thought of as a function that maps an input, i.e. a task instance, to an output, i.e. some decision-making policy. The complexity of such an algorithm $\mathcal{A}$, in line with the theory of computational complexity, is measured by the total computational time relative to the size of the input, subject to a given precision requirement on the output. Loosely speaking, when the computational time scales polynomially in the size of the input, the algorithm is considered efficient, and the task is considered (efficiently) approximable.

The notion of "efficiency" or "approximability" depends crucially on the means by which algorithm $\mathcal{A}$ accesses the task. Which piece of information does the algorithm takes as input? We need to first specify the answer to this question before conducting any rigorous analysis. Historically, the study of the MDPs (in the so called *tabular* case) assumed the algorithm possesses a complete knowledge of the task. Namely, the state space, the action space, the transition probabilities, the reward/cost functions and the time horizon/discount factor of the MDP are all known to the algorithm. However, this assumption is highly impractical especially in the data-driven setting. Indeed, for many tasks,

the precise knowledge of the underlying exogenous data such as its joint distribution is barely available. Even when all information is available, the often prohibitively large/infinite state space of $\mathbf{X}$ renders encoding such information impossible.

More realistic alternatives were proposed in the reinforcement learning (RL) literature, where the algorithm is assumed to only have *sample access* (in various ways) to the underlying process $\mathbf{X}$. A popular such sampling model is the **generative model**, first introduced in [164]:

Generative model: There is a simulator that, for any $t \in [1, T]$ and any realized trajectory $x_1, ..., x_{t-1}$, can draw independent samples of X_t conditional on $X_i = x_i, i = 1, ..., t-1$. Additionally, for any action sequence $a_1, ..., a_t$, it can evaluate reward/cost $r_t(a_1, ..., a_t, x_1, ..., x_t)$.

In this thesis, we assume the generative model, although it's worth pointing out that there exists a handful of other sampling models under which various complexity and hardness guarantees were obtained ([160]). With the generative model, we can now formally state the central question of concern:

For what types of sequential decision-making tasks does there exist an algorithm that can find an "ϵ−optimal" policy with probability at least $1 - \delta$*, with* (1) *the number of calls to the generative model and (2) the computational effort required to manipulate the samples both depend* ***polynomially*** *on* T *and* ***independent*** *of the size of the state space of* $\mathbf{X}$ *?*

This question has been studied for over two decades in the theoretical RL literature ([160]). Perhaps unsurprisingly, an efficient approximation algorithm that can universally solve all sequential decision-making tasks does not exist.

Indeed, we have

Theorem 1.0.1 (Lower bound of [164])**.** *Suppose $\mathcal{A}$ is any algorithm for MDP tasks that is given access to only the generative model, and that takes as input a state s and ϵ (hence output ϵ-approximation of the to-go value at state s). Then there exists an MDP instance on which $\mathcal{A}$ must make at least $\Omega((\frac{1}{\epsilon})^T)$ number of calls to the generative model.*

Due to this negative result, one natural research direction is to study special structures of sequential decision-making tasks/MDPs that permits the existence of efficient approximation algorithms. Such a direction remains highly active in RL research. People have studied problems with linear parametric/low rank structures (see e.g. [156], [256], [257], [158], [204], [19], [238], [110], [157], [183] among others), characterizing precise structures under which efficient approximation algorithms exist. In operations management, researchers also made significant contribution in this direction. A number of works investigated problems restricted to simple (e.g. i.i.d.) underlying dynamics, which in many cases guarantees the existence of efficient approximation algorithms (e.g. [15, 71, 249], for a detailed literature review check Chapter 3). There are also a number of heuristic methods, including the approximate dynamic programming (ADP) method, first introduced by [243], and information relaxation methods, first introduced by [68], among many others (see later chapters for detailed literature review). These methods enjoy a great empirical success, but they often do not come with strong theoretical guarantee.

The approach taken in this thesis deviates significantly from all previously mentioned works. Indeed, our study focuses on sequential decision-making tasks where **the decisions are subject to certain combinatorial constraints**. On the one hand, various combinatorial structures such as cut, matching, packing

etc. arise frequently in real world decision-making scenarios across all application domains. On the other hand, imposing such combinatorial constraints on decisions reduces the total number of feasible policies, which may potentially reduce the computational effort required, leading to efficient algorithms. A typical example that we study is the **optimal stopping problem**. In an optimal stopping problem, the DM irrevocably determines whether to stop/continue in each time period based on the realized trajectory. Here the implicit combinatorial constraint is that, given any realized trajectory $x_1, ..., x_T$, the number of total stopping actions chosen throughout T periods equals one. Optimal stopping is a special case of sequential decision-making with important practical applications, and we are able to show the existence of efficient approximation algorithms for this problem (see Chapter 2). We want to remind our reader that the combinatorial (stopping) structure plays a vital role in the problem's approximability here. Indeed, consider a famous sequential decision-making task known as the *binary trajectory tree* ([164]). This task has a binary action set just like the optimal stopping problem, but without the stopping constraint, so that the DM is allowed to choose arbitrarily one of the two actions in each time period. It turns out that this seemly simple task is actually the hard instance for Theorem 1.0.1!

We want to comment that imposing combinatorial constraints generally does not trivialize the computational problem, especially in the data-driven setting where the underlying process $\mathbf{X}$ can be high-dimensional and with complicated dynamics. Indeed, the optimal stopping problem with a high-dimensional $\mathbf{X}$ is long believed to be computationally challenging and intractable ([243, 193]). The theoretical (in)approximability of such a problem, or problems with various other combinatorial structures has not been systematically studied in the

literature. Determining the approximable and coming up with efficient approximation algorithms for such problems are meaningful from both an applied and a theoretical point of view.

1.1 Overview

This dissertation thesis investigates three specific decision-making tasks, each subject to a distinct combinatorial constraint on decisions. The existence of efficient approximation algorithms is guaranteed in all three cases, positively answering our central research question. In Chapter 2, we study the fundamental problem of high-dimensional optimal stopping. This problem is with wide applications across operations management, economics, computer science and control, and of central importance to the fields of applied probability and mathematical finance. we develop a new methodology for efficiently approximately computing the optimal value/stopping policy, contrasting the popular belief that such problems are intractable. This surprising result is built on a novel discovery of an equivalent formulation (in the form of an infinite expansion) of the optimal stopping DP. In Chapter 3, we study a fairly general class of problems subject to a "limited-move" constraint, effectively generalizing our result for optimal stopping. The class of problems we consider captures a large number of important real-world decision-making examples, including the dynamic pricing of a product under the *limited price change* business rule, and the evaluation of high-dimensional financial derivatives(swing options) with a restricted number of exercising opportunities. Viewing each time point of action change as an optimal stopping time, we leverage the results of Chapter 2 to build efficient approximation algorithms with provable performance and computational time

guarantees. In Chapter 4, we focus on the problem of *online maximum weight independent set* in bounded-degree bipartite graphs. This sequential decision-making problem is subject to a network-type constraint, i.e. the independent set constraint. Combining several sophisticated techniques, we propose a novel efficient approximation algorithm whose computationl time scales only *linearly* in the time horizon for any fixed error tolerance, and is effectively *independent* of the underlying dimension.

CHAPTER 2

BEATING THE CURSE OF DIMENSIONALITY IN HIGH-DIMENSIONAL OPTIMAL STOPPING

2.1 Introduction

2.1.1 Overview of problem

Optimal stopping is a fundamental model of sequential decision-making under uncertainty. It finds wide applications across a variety of areas including operations management, finance, economics, statistics and computer science, and has been extensively studied for over 70 years. The basic setting of a *discrete-time* optimal stopping is as follows: a decision maker (DM) observes a random-evolving process. At each time period, an irrevocable decision of whether or not to stop must be made by the DM. A STOP decision results in a reward/cost, which depends on the underlying stochastic process. The goal is to find the optimal policy that maximizes the expected reward/minimizes the expected cost.

The study of *high-dimensional* optimal stopping was initiated over 30 years ago in the field of mathematical finance, motivated by the need to price complex, multivariate financial derivatives (see e.g. [26]). Due to the big-data revolution and the corresponding, growing demand for data-driven decision-making, today's real-world optimal stopping tasks are becoming increasingly complicated. For example, modern epileptic seizure detection is driven by the electroencephalogram(EEG) data stream, which is generated by multiple electrode "sensors" (in some cases more than 20) and is typically highly non-stationary ([228, 227]). As another important example, the vital pandemic management

decisions of lockdown/reopen needs to be based on massive amount of information such as the dynamics of the daily infected cases. In such optimal stopping tasks, the underlying stochastic processes are often high-dimensional and with non-i.i.d., discontinuous, and path-dependent dynamics. Such problems typically have no simple analytical solution, thus substantial attention was diverted to numerical/computational methods.

Unfortunately, it is generally understood that such high-dimensional path-dependent optimal stopping problems are computational intractable, suffering from the *curse of dimensionality*. Indeed, such problems can have a massive state space with exponentially (in the dimension, time horizon, and/or both) large size and discontinuous dynamics, which is unbearable for vanilla dynamic programming (DP) to handle. This is in sharp contrast to the one-dimensional, i.i.d. "secretary" type of optimal stopping problem that is easy to solve. Coping with the curse of dimensionality has thus become a central theme in the literature of optimal stopping in the past two decades.

Considerable effort has been devoted to the design of fast and near-optimal *approximation algorithms* for high-dimensional optimal stopping. Popular approaches include approximate dynamic programming (ADP) methods ([193],[244] etc.), dual-based methods ([225], [142], [13], [79], [108] etc.) and deep learning methods ([32, 33] etc.). To sidestep the computational challenge, these existing approaches usually implement certain dimension reduction heuristics (such as introducing basis functions/basis martingales). Many such heuristic-based approaches enjoy excellent performance on test examples/ in practical tasks. However, they typically don't come with strong approximation guarantees in theory (often with fundamental dependence on e.g. the quality of the basis functions etc.).

Motivated by the need to rigorously describe the performance of approximation algorithms, a systematic framework was developed in computer science and operations research. In particular, for a general optimization problem, *polynomial-time-approximation-schemes (PTAS)* refer to such algorithms that for any fixed ϵ, the running time for achieving $\epsilon-$optimality is polynomial in the problem size (independent of specific problem instances). Algorithms which guarantee the above with high probability are called *polynomial-time-randomized-approximation-scheme (PRAS)*. An optimization problem is considered tractable if PTAS/PRAS exist. These classical definitions were later extended to sequential decision-making problems (with optimal stopping a special case) by the reinforcement learning (RL) literature (see [160]), to categorize the various existing Monte Carlo based RL algorithms. Such algorithms assume the generative model, in which (as we recall from Chapter 1) the access to the problem instance (i.e. the underlying stochastic process and the reward/cost functions) is only through a system simulator that can generate independent trajectories/samples. In such settings, an analogy of PRAS refers particularly to an algorithm such that (1). taking any fixed ϵ, δ as input, it can output an ϵ-approximation with probability at least $1 - \delta$, and (2) the computational cost/ sample complexity (number of independent calls made to the simulator) scales only polynomially in the horizon and the dimension, independent of the specific underlying distribution/ dynamics. We shall simply refer to such algorithms as generative-model PRAS hereinafter, and defer a more detailed discussion to the literature review in Section 2.2.

The generative-model PRAS and related concepts provide us with a rigorous framework within which Monte Carlo based approximation algorithms for optimal stopping can be compared and analyzed. As we indicated in the pre-

vious section, to our best knowledge, there has been no such generative-model PRAS for optimal stopping appearing in the existing literature, nor is it clear whether such algorithms even exist. In fact, although various works (particularly those in mathematical finance) have achieved great empirical success, the general belief prior to this work is that high-dimensional path-dependent optimal stopping is theoretically intractable (in the precise sense of polynomial time inapproximability), at least without positing additional continuity or prior knowledge of basis functions.

The main contribution of the current work is the (somewhat surprising) discovery of a generative-model PRAS for optimal stopping, proving that high-dimensional path-dependent optimal stopping problems are fundamentally tractable (in the aforementioned sense). More precisely, for any fixed error tolerance (ϵ, δ), the algorithm we propose can output $\epsilon-$optimal solutions (both stopping policies and approximate optimal values) with high probability (at least $1-\delta$) with computational and sample complexity both scaling *polynomially* in the time horizon T, and essentially independent of the dimension D (beyond the cost of generating samples from the simulator). Our algorithm works in total generality except for a boundedness requirement on the cost/reward, allowing for arbitrary high-dimensional, non-stationary, non-Markovian underlying processes and time/path-dependent cost/reward functions. In fact, beyond the ability to draw independent trajectories/samples from the simulator, the algorithm requires no prior knowledge of the probability law/transition probability that drives the underlying process, a feature which may be desirable in applications ([63]).

Our algorithm is based on a novel expansion representation of the optimal value of the stopping problem. This expansion is by nature different from all

existing representations, which typically rely on standard backward inductive logic. In particular, the first term of the expansion coincides with the expected path-wise minimum/maximum (i.e. the hindsight optimal value). The subsequent terms are derived in a recursive manner, themselves expectations of certain minima conceptually corresponding to various notions of regret (and regret of regret). Our algorithm is derived from truncating this expansion after a small number of terms. To yield any theoretical performance guarantees, many DP based approaches would necessitate the computation of deeply nested conditional expectations, where the depth of nesting scales with the time horizon T. In contrast, our expansion yields a (normalized) error of $\frac{1}{k}$ with only k levels of nesting. This key fact endows our algorithms with a runtime complexity that is independent of the time horizon T. Furthermore, we believe that this expansion-based technique is potentially applicable to other more complicated/general sequential decision-making and control problems.

2.1.2 Organization

The rest of the chapter is organized as follows. Following a literature review, in Section 2.3, we define a general optimal stopping problem and describe our main theoretical and algorithmic results: a novel expansion representation for the optimal value, the rate of convergence of this expansion, and a (simulation-based) generative-model PRAS inspired by the expansion (for both the optimal value and the policy). Specifically, we also extend our algorithmic result to the more practical maximization setting. In Section 2.4, we prove the validity of the expansion. In Section 2.5, we prove several results related to the rate of convergence. We provide in Section 2.6 the detailed algorithmic analysis. Numerical

examples are presented in Section 2.7. Finally, we conclude the chapter in Section 2.8.

2.2 Literature review

As discussed earlier, there is a vast literature on optimal stopping in operations research, mathematical finance and computer science. Traditionally, one large stream of works takes a continuous-time perspective and is intimately connected to the theory of partial differential equations (PDE). Related computational methods include numerical PDE (finite difference, finte element etc.) or variational approaches. They generally aim at continuous-time problems with certain model structures (e.g. the underlying process is Markov/evolves according to an Ito process), which is not the main focus of the current work. Besides, in the presence of high-dimensionality and path-dependence, these methods generally suffer from similar complexity related issues and/or require additional continuity assumptions to achieve strong theoretical guarantees. We refer the readers to e.g. [6], [3], [213], [215], [151], [198], [32], [31] for additional background.

We mainly focus on the discussion of methods based on Monte Carlo simulation. Monte Carlo methods rose in popularity within the literature on high-dimensional American option pricing/ optimal stopping in the 1990's. These methods use sampling, and certain dimension reduction techniques to approximate the dynamic programming (DP) equations and yield tractable algorithms. From the late 1990's to the mid-2000's (and continuing today), one of the main approaches taken was regression/approximate dynamic programming (ADP). Here one fixes a family of basis functions to approximate the DP value functions.

Seminal papers in this area include [244], and especially [193]. There was much subsequent work on e.g. nonparametric regression, smoothing spline regression, recursive kernel regression, integer programming, reinforcement learning, (deep) neural network and etc. (see [179], [111], [112], [172], [28], [173], [42], [92], [29], [240], [56], [81], [133], and [33] etc.). These methods are largely heuristic whose performance typically relies on how well the choice of basis functions can approximate the true value functions. Most existing theoretical analyses for these methods focus on their convergence to the best approximation *within the fixed set of basis functions*. Such analyses appear in e.g. [93], [131], [60], [55], [36], [34], [40], [237]. These methods don't have generative-model PRAS-type theoretical performance guarantees, which is the key contribution of the current work.

Building on the seminal work of [106], significant progress was made simultaneously by [142] and [225] in their formulation of a dual methodology for optimal stopping. Instead of finding an optimal stopping time, it formulates and solves a dual optimization problem over the set of adapted martingales. Other dual representations were subsequently discovered ([153, 159, 186]), and the methodology was extended to more general control problems ([68, 222, 78, 45]). Simulation approaches via dual formulation (e.g. methods approximating the optimal dual martingale by appropriate value function approximations/basis martingales/convex optimization techniques) have since led to substantial algorithmic progress (see [13, 79, 41, 150, 174, 39, 62, 108, 91, 37, 184, 224] among many others). Dual simulation approaches can yield upper bounds (in the context of maximization/option pricing) which can be used as benchmarks to measure stopping policies. However, the quality of these upper bounds typically depends on the expressiveness of the set of basis functions/martingales, or the

quality of the initial estimation of the value functions. Consequently, they also don't have PRAS-type theoretical performance guarantees.

More relevant to our approach, a handful of iterative Monte Carlo methods were proposed and analyzed in the literature. Similar to our own, these methods don't use basis functions/martingales or a set of initial estimation of value functions, instead they rely on policy-iteration type of recursive procedures to gradually improve and approach optimality. [175] proposed a method that recursively improves the set of stopping times, approaching the optimal stopping policy. [39] exploited the dual representation and developed a recursive primal-dual method that maintains both upper and lower bounds. A similar idea appears in [79], where the authors defined an iterative procedure based on so-called supersolutions which leads to an expansion for the optimal dual martingale. Other relevant works include [54], [230] and [223]. Some of the algorithms proposed in these papers have impressive performance on numerical experiments, yielding tight bounds after only a few iterations, where one more iteration often means evaluating one more level of nested conditional expectation with simulation. However, in theory these algorithms usually don't have a performance guarantee until the optimal value is achieved after T iterations, which is the same level of nesting/complexity required by implementing a naive backward induction (thus in theory impractical in the high-dimensional and path-dependent settings). As a comparison, the expansion proposed in the current work can output an approximate solution with explicit theoretical performance bound after any *finite* number of iterations (independent of both T and D), which is the key reason why it leads to theoretically efficient and near-optimal algorithms.

More recently, a line of works utilizes *deep neural network* techniques to solve

reinforcement learning (RL) problems, and in particular, high-dimensional American option pricing/optimal stopping problems (see [173], [234], [32], [33], [235], [180], [121] etc.). These works exhibit excellent numerical performance when applied to optimal stopping problems. In fact, we use the extensive numerical experiments of [33] as benchmarks in our own numerical experiments (as we discuss in Section 2.7). Similar to the methods mentioned previously, deep learning based techniques generally don't have strong theoretical performance guarantees, especially without additional continuity assumptions.

Our work is also generally related to the theory of approximation algorithms and RL/machine learning. The formal concept of a PTAS/PRAS can be found in standard textbooks, e.g. [254], while analogous notions of complexity for sample-based algorithms in stochastic control/stochastic optimization can be found in e.g. [140], [239], [187] and [100]. Certain work in reinforcement learning is of particular relevance. [163] introduced the concept of a "generative model", which is a black-box simulator only providing sampling access to the underlying process/state. A formal description and discussion of the generative-model PRAS in sequential decision-making/RL can be found in [160]. Since it's well known that the optimal stopping problem can be formulated as a Markov decision process (MDP) (even if the underlying process is non-Markov), it can thus fit into the RL framework assuming the generative model. It turns out that for general generative-model RL problems, in a certain precise sense an algorithm's complexity needs to either depend on the number of state-action pairs (typically formalized in models requiring one to apriori specify an action for every state, see e.g. [20], [233]), or to scale exponentially in the time horizon T (see e.g. [163],[165]), hence suffering from the curse of dimensionality. Our work, however, shows that generative-model PRAS ex-

ists for high-dimensional path-dependent optimal stopping. We believe that there is a deeper connection between the current work and the related works on generative-model RL problems, and we leave a further investigation as an open problem for future research.

Finally, although PTAS and PTAS-related concepts build the most popular framework for assessing approximation algorithms in computer science and operations research, there are also other frameworks proposed in other literatures. In certain specific scenarios generative-model PRAS may not be the most appropriate algorithmic requirement. As an example, [38] introduced the concept of semi-tractability, which requires an algorithm to return an ϵ approximation with a sample complexity growing slower than ϵ^{-D} as $\epsilon \to 0$ and $D \to \infty$, and exhibits stochastic mesh methods with such a property under additional continuity assumptions. Such alternative notions of tractability may be especially relevant in settings where the discrete-time problem arises as a discretization of an underlying continuous problem (as in some financial models for options pricing).

2.3 Main Results

In this section we rigorously define the general optimal stopping problem of interest, and state our main results: a novel expansion representation of the optimal value, the rate of convergence of the expansion, and its algorithmic implications.

2.3.1 Formulation of problem and notation

Let $\mathbf{X} = \{X_t, t \in [1,T]\}$ be a general (possibly high-dimensional and non-Markovian) discrete-time stochastic process. For $t \in [1,T]$, let $X_{[t]} \stackrel{\Delta}{=} (X_1, \ldots, X_t)$ be the time t partial trajectory. Let X_t be a D dimensional vector where $D \geq 1$. Let $\{r_t : \mathcal{R}^{D \times t} \to \mathcal{R}^+, t \in [1,T]\}$ be a sequence of general (possibly time-dependent) measurable cost functions mapping trajectories $X_{[t]}$ to non-negative reals. The problem of general optimal stopping is that of computing $\text{OPT} \stackrel{\Delta}{=} \inf_{\tau \in \mathcal{T}} E[r_\tau(X_{[\tau]})]$, with $\mathcal{T}$ the set of all stopping times adapted to the natural filtration $\mathcal{F} = \{\mathcal{F}_t, t \in [1,T]\}$ generated by the process $\{X_t, t \in [1,T]\}$. This formulation fits into the general framework that we introduced in Chapter 1.

Remark. *In the current work we mostly frame our results in terms of the problem* $\inf_{\tau \in \mathcal{T}} E[r_\tau(X_{[\tau]})]$ *instead of the maximization counterpart. This is convenient and natural in light of our approach, which recurses on modified problems with cost functions which are decreasing in the number of recursions, and for which the optimal stopping value decreases to zero. For an alternative view we refer the reader to the parallel work [85] which connects optimal stopping to the minimum cut problem in the theory of network flows. In general our results for the minimization context can be converted to analogous results in the maximization setting through transformations such as* $\sup_{\tau \in \mathcal{T}} E[Z_\tau] = E[\max_{i \in [1,T]} Z_i] - \inf_{\tau \in \mathcal{T}} E[E[\max_{i \in [1,T]} Z_i | \mathcal{F}_\tau] - Z_\tau]$, *and we articulate some explicit guarantees for the maximization setting in both our main results and numerical experiments. Occasionally it will also be convenient to assume that* $\{r_t, t \in [1,T]\}$ *has been normalized, so that* $r_t \in [0,1]$ *for all* $t \in [1,T]$, *and we clearly articulate whenever such an assumption is enforced.*

Additional notation. For $t \in [1,T]$, let $\aleph^t$ denote the set of all D by t matrices (i.e. D rows, t columns) with entries in $\mathcal{R}$, so the time t partial trajectories $\mathbf{X}_{[t]} \in$

$\aleph^t$. Let $Z_t \triangleq r_t(X_{[t]})$, where we write $Z_t(X_{[t]})$ if we wish to make the dependence explicit, and assume that Z_t is integrable for all t. The optimal stopping problem of interest can thus be stated as $\text{OPT} = \inf_{\tau \in \mathcal{T}} E[Z_\tau]$.

2.3.2 Novel expansion representation for OPT

Simple intuition.

We begin by giving the simple intuition behind the expansion. We wish to compute $\text{OPT} = \inf_{\tau \in \mathcal{T}} E[Z_\tau]$. First, it follows from a sample-path argument that $\inf_{\tau \in \mathcal{T}} E[Z_\tau] \geq E[\min_{i \in [1,T]} Z_i]$. We next turn this straightforward bound into an *equality* by compensating with a remainder term, which is characterized by a new optimal stopping problem. To explicitly express this remainder term, we introduce a specific martingale $\{E[\min_{i \in [1,T]} Z_i | \mathcal{F}_t],\ t \in [1,T]\}$, i.e. the Doob martingale of r.v. $\min_{i \in [1,T]} Z_i$, adapted to filtration $\mathcal{F}$. The mean of this martingale is equal to $E[\min_{i \in [1,T]} Z_i]$. By the optional stopping theorem, $E[E[\min_{i \in [1,T]} Z_i | \mathcal{F}_\tau]] = E[\min_{i \in [1,T]} Z_i]$ for any stopping time $\tau \in \mathcal{T}$, we thus have that

$$E[Z_\tau] = E\Big[\min_{i \in [1,T]} Z_i\Big] + E\Big[Z_\tau - E[\min_{i \in [1,T]} Z_i | \mathcal{F}_\tau]\Big].$$

Taking infimum from both sides, we conclude that

$$\text{OPT} = E\Big[\min_{i \in [1,T]} Z_i\Big] + \inf_{\tau \in \mathcal{T}} E\Big[Z_\tau - E[\min_{i \in [1,T]} Z_i | \mathcal{F}_\tau]\Big].$$

For $t \in [1,T]$, let $Z_t^1 \triangleq Z_t$ and $Z_t^2 = Z_t^1 - E[\min_{i \in [1,T]} Z_i^1 | \mathcal{F}_t]$. The above equation becomes

$$\text{OPT} = E\Big[\min_{i \in [1,T]} Z_i^1\Big] + \inf_{\tau \in \mathcal{T}} E[Z_\tau^2].$$

Now, we simply observe that we may recursively repeat this process on the problem $\inf_{\tau \in \mathcal{T}} E[Z_\tau^2]$, and then all subsequent problems. As we will see, this

yields an explicit and rapidly-converging expansion for the optimal value which is amenable to simulation.

Expansion for OPT.

We formalize the above intuition and provide our expansion for OPT. For $k \geq 1$ and $t \in [1, T]$, let $Z_t^{k+1} \stackrel{\Delta}{=} Z_t^k - E[\min_{i\in[1,T]} Z_i^k | \mathcal{F}_t]$. It follows from the basic properties of conditional expectation and non-negativity of $\{Z_t^1, t \in [1, T]\}$ that $Z_t^k \geq 0$ for all $t \in [1, T]$ and $k \geq 1$. We let $\mathbf{Z}$ and $\mathbf{Z}^k$ denote the respective stochastic processes. Then our main result is as follows.

Theorem 2.3.3. $\text{OPT} = \sum_{k=1}^{\infty} E[\min_{t\in[1,T]} Z_t^k]$.

In many ways the statement of Theorem 2.3.3 is quite surprising, as it asserts that **the value of a general path-dependent optimal stopping problem has a representation which looks very much like a closed-form solution**. To make this point clear, let us more explicitly give the first few terms. For $k \geq 1$, let $L_k \stackrel{\Delta}{=} E[\min_{t\in[1,T]} Z_t^k]$. Then L_1, L_2, L_3 are as follows.

$$L_1 = E[\min_{t\in[1,T]} r_t(X_{[t]})] \quad ; \quad L_2 = E\Big[\min_{t\in[1,T]} \Big(r_t(X_{[t]}) - E[\min_{i\in[1,T]} r_i(X_{[i]})|\mathcal{F}_t]\Big)\Big];$$

$$L_3 = E\Big[\min_{t\in[1,T]} \Big(r_t(X_{[t]}) - E[\min_{i\in[1,T]} r_i(X_{[i]})|\mathcal{F}_t] - E\Big[\min_{i\in[1,T]} \Big(r_i(X_{[i]}) - E[\min_{j\in[1,T]} r_j(X_{[j]})|\mathcal{F}_i]\Big)\Big|\mathcal{F}_t\Big]\Big)\Big].$$

The first term, L_1, corresponds to the obvious lower bound. Later terms are the expectations of some elegant and explicit infima with interpretations in terms of certain notions of regret, each of which can be computed by simulation.

2.3.4 Approximation guarantees when truncating the expansion

The power of Theorem 2.3.3 is that it allows for rigorous approximation guarantees when the infinite sum is truncated. Let $E_k \stackrel{\Delta}{=} \sum_{i=1}^{k} L_i$.

Theorem 2.3.5. *Suppose w.p.1* $Z_t \in [0,1]$ *for all* $t \in [1,T]$. *Then for all* $k \geq 1$, $0 \leq \text{OPT} - E_k \leq \frac{1}{k+1}$.

Thus truncating our expansion after k terms yields an absolute error at most $\frac{1}{k+1}$. Theorem 2.3.5 is again quite surprising since this linear rate of convergence is achieved **regardless of the distribution of the underlying stochastic process X**, requiring only that the cost functions are bounded in $[0,1]$. Furthermore, our analysis behind Theorem 2.3.5 yields not only an approximation scheme for OPT, but also a computationally efficient near-optimal stopping policy (albeit one that does *not* take the natural approach of repeatedly computing the *to-go* values). We leave a detailed discussion to the proof of Theorem 2.3.5 in section 2.5. The results regarding the corresponding policy will be presented in the next section.

The boundedness of the cost functions in the statement of Theorem 2.3.5 is not a critical assumption and can be relaxed in several ways, depending on the precise notion of approximation desired. For example, if $Z_t \in [0,U]$ for some $U > 0$, an identical analysis yields a normalized error of $\frac{U}{k+1}$. Alternatively, if $\{Z_t, t \in [1,T]\}$ has sufficiently regular moments, then one can derive analogous results in the unbounded setting, such as the following.

Corollary 1. *Under no assumptions on* $\mathbf{Z}$ *beyond non-negativity and square-integrability,* $\text{OPT} - E_k \leq 3 \times (\frac{E[(Z_T)^2]}{\text{OPT}^2})^{\frac{1}{3}} \times \text{OPT} \times k^{-\frac{1}{3}}$.

Depending on the particular assumptions and desired notion of approximation error (e.g. absolute vs. relative), many such variants of our main results are possible. Furthermore, we suspect the $(\frac{1}{k})^{\frac{1}{3}}$ bound derived in Corollary 1 for the unbounded case is not tight and can be improved, and leave a tighter analysis of how our results change under different assumptions as an interesting open question.

That said, for the setting in which w.p.1 $Z_t \in [0, 1]$ for all $t \in [1, T]$, we now present a lower bound showing that Theorem 2.3.5 is tight up to an absolute multiplicative constant factor of $\frac{1}{4}$. In other words, there is a precise sense in which the rate of convergence of the stated expansion cannot be substantially improved without making some kind of additional assumptions on the optimal stopping problem. We want to emphasize that this worst-case lower bound does not rule out the possibilities that the expansion exhibits a faster convergence rate in certain specific cases, as demonstrated in our numerical experiments. Furthermore, it is a fascinating open question whether fundamentally faster expansions and approaches exist, which we leave for future study.

Lemma 1 (Lower bound)**.** *For any given $k \geq 2$, there exists a 2-period optimal stopping problem with $P(Z_t \in [0, 1]) = 1$ for all $t \in [1, 2]$, yet $\text{OPT} - E_k \geq \frac{1}{4k}$.*

2.3.6 Algorithmic results

We now describe our main algorithm, along with its runtime and sample-complexity analysis. As a natural implication of Theorems 2.3.3 and 2.3.5, the algorithm computes L_i for $i = 1, \ldots, k$ by Monte Carlo simulation for some appropriately chosen truncation level k. To formalize and analyze this straightfor-

ward intuition, we first establish the computational and sampling model used in our analysis. Our model is consistent both with many works in optimal stopping (which often assume the ability to generate a trajectory of the underlying information process conditioned to start from any given state), as well as certain notions of efficient algorithms in the study of generative models for reinforcement learning (in which each of the possibly infinite number of "partial trajectories" $X_{[t]}$ would correspond to a "state" in the mode).

Formal computational and sampling model for algorithm analysis.

Access to samples and data-driven algorithms : We begin by formally defining a subroutine (randomized algorithm) $\mathcal{B}$ which we will refer to as the "base simulator", and which will provide the only means for our algorithms to access information about $\mathbf{X}$. For $t \in [1, T]$ and $\gamma \in \aleph^t$, let $\mathbf{X}(\gamma)$ denote a random matrix distributed as $\mathbf{X}$, conditioned on the event $\{\mathbf{X}_{[t]} = \gamma\}$.

Definition 2.3.1 (Base simulator $\mathcal{B}$)**.** *$\mathcal{B}$ is a randomized algorithm with the following properties. It takes as input $t \in [1, T]$ and $\gamma \in \aleph^t$, and outputs an independent sample of $\mathbf{X}(\gamma)$. $\mathcal{B}(0, \emptyset)$ returns an independent unconditioned sample of $\mathbf{X}$.*

In some cases, generating simulated sample paths can be quite challenging (either computationally or due to lack of data), and there are many interesting questions surrounding how to combine our framework with settings in which generating individual sample paths/evaluating cost functions is very costly. Such questions are generally beyond the scope of this work, and an interesting direction for future work.

Computational model and runtime analysis : Next, we formalize a computational model for how to analyze the run-time of algorithms that use $\mathcal{B}$. Suppose that $\mathcal{B}$ takes at most C units of computational time to terminate and generate one sample. We also suppose that for any $t \in [1, T]$ and $\gamma \in \aleph^t$, $r_t(\gamma)$ can be computed in G units of computational time. Here C and G does not depend on the particular inputs t, γ to $\mathcal{B}$, although they are allowed to depend on T, D, and possibly other parameters. In addition, we suppose that addition, subtraction, multiplication, division, maximum, and minimum of any two numbers can be done in one unit of time, regardless of the values of those numbers. We will ignore all computational costs associated with reading, writing, and storing numbers in memory, as well as inputting numbers to functions. In general our model allows for computation over real numbers, and leave a more refined understanding of bit complexity and related matters for future work.

Main algorithmic results.

We now state the main algorithmic results of the current work. Our results are twofold: 1. computing an ϵ-approximation for the optimal value OPT, and further, 2. providing an $\epsilon-$approximate optimal stopping strategy. We prove that for any given error parameter ϵ, our algorithms can achieve these two goals with high probability in a runtime only polynomial in T, and depending on the dimension (and state-space more generally) only through the cost of simulating individual sample paths, where only polynomial number of such simulations are needed.

Theorem 2.3.7 (Optimal value approximation)**.** *Suppose w.p.1 $Z_t \in [0, 1]$ for all $t \in [1, T]$. Then there exists a randomized algorithm $\mathcal{A}$ which takes as input any $\epsilon, \delta \in$*

$(0,1)$, *and achieves the following. In total computational time at most* $(C+G+1)\times \log(\delta^{-1})\times \exp(100\epsilon^{-2})\times T^{6\epsilon^{-1}}$, *and with only access to randomness at most* $\log(\delta^{-1})\times \exp(100\epsilon^{-2})\times T^{6\epsilon^{-1}}$ *calls to the base simulator* $\mathcal{B}$, *returns a random number* Y *satisfying* $P(|Y-\mathrm{OPT}|\leq\epsilon)\geq 1-\delta$.

Theorem 2.3.8 (Good policy)**.** *Suppose w.p.1* $Z_t\in[0,1]$ *for all* $t\in[1,T]$. *Then for all* $\epsilon\in(0,1)$, *there exists a randomized stopping time* τ_ϵ *s.t.* $E[Z_{\tau_\epsilon}]-\mathrm{OPT}\leq\epsilon$, *and with the following properties. At each time step, the decision of whether to stop (if one has not yet stopped) can be implemented in total computational time at most* $(C+G+1)\exp(100\epsilon^{-2})\times T^{10\epsilon^{-1}}$, *and with only access to randomness at most* $\exp(100\epsilon^{-2})\times T^{10\epsilon^{-1}}$ *calls to the base simulator* $\mathcal{B}$.

Remark. *The above theorems aim at demonstrating the existence of a poly time (in* T *and* d*) approximation algorithm for high-dimensional optimal stopping problems. The constant terms are derived under extremely conservative assumptions (following multiple union bounds) to cover the most pathological cases (with arbitrarily discontinuous distributions/cost functions). In almost all practical cases, the running time of our algorithm is expected to be way better than suggested by the above bounds.*

The rigorous statement of the algorithm $\mathcal{A}$ itself will be given in later sections when we prove the correctness of our algorithmic results. The algorithm is based on using an appropriate truncation of our expansion, with all terms computed by simulation. Interestingly, our policy does *not* proceed by recomputing approximate cost-to-go functions (which could accumulate an amount of error scaling with the time horizon if applied naively), and instead rely on certain stronger path-wise properties (see the proof of Theorem 2.3.5 in Section 2.5).

Algorithmic results in the unbounded maximization setting

Real world optimal stopping tasks are often *maximization* problems with possibly unbounded payoffs. Here we state a variant of our main result which holds in such unbounded maximization settings. Out of space considerations we only state the analogue of Theorem 2.3.7 for approximating the optimal value, but a very similar analysis yields a corresponding efficient policy (in analogy with Theorem 2.3.8). Formally, let's consider the optimal stopping problem: $\widehat{\text{OPT}} \triangleq \sup_{\tau \in \mathcal{T}} E[Z_\tau]$. Instead of requiring the payoff process $\mathbf{Z}$ to be uniformly bounded, we here only assume that the path-wise maximum of $\mathbf{Z}$ has finite first and second moment, i.e. $M_1 \triangleq E[\max_{t\in[1,T]} Z_t] < \infty$ and $M_2 \triangleq E[(\max_{t\in[1,T]} Z_t)^2] < \infty$. What we impose is a significantly weaker condition. Let $\gamma_0 \triangleq \frac{M_2}{(M_1)^2}$ (where we also assume $M_1 > 0$ to exclude uninteresting/pathological cases),

Theorem 2.3.9 (Algorithms for unbounded maximization)**.** *For any arbitrary non negative payoff* $\mathbf{Z}$ *with finite* M_1, M_2*, the following is true. There exists a randomized algorithm* $\hat{\mathcal{A}}$ *which takes as input* M_1, M_2 *and any* $\epsilon, \delta \in (0,1)$*, and achieves the following. In total computational time at most* $(C+G+1)\times\log(\frac{1}{\delta})\times\exp(10^{20}\gamma_0^9\epsilon^{-6})\times T^{10^8\gamma_0^{\frac{9}{2}}\epsilon^{-3}}$ *and with only access to randomness at most* $\log(\frac{1}{\delta})\times\exp(10^{20}\gamma_0^9\epsilon^{-6})\times T^{10^8\gamma_0^{\frac{9}{2}}\epsilon^{-3}}$ *calls to the base simulator* $\mathcal{B}$*, returns a random number* Y *satisfying* $P\left(\frac{\left|Y-\widehat{\text{OPT}}\right|}{\widehat{opt}} \leq \epsilon\right) \geq 1-\delta$.

All bounds presented in this section are worst-case/conservative bounds, and we have tended to choose simplicity of analysis over optimizing of constants for clarity of exposition. The performance on real data should be much better than suggested by these bounds, as we observe in our numerical experiments (see section 2.7). However, as we emphasized earlier, **the significance of these results lies not in their direct practical implication, but rather the sur-**

prising existence of such efficient algorithms for general high-dimensional optimal stopping in the first place. Indeed, Theorems 2.3.3 and 2.3.5 open up the possibility of the design of truly practically competitive algorithms that simultaneously come with strong theoretical performance guarantees, and we view the bounds and algorithms presented here as a first step towards that goal.

2.4 Expansion and Proof of Theorem 2.3.3

Recall that for $k \geq 1$ and $t \in [1, T]$, $Z_t^{k+1} \stackrel{\Delta}{=} Z_t^k - E[\min_{i \in [1,T]} Z_i^k | \mathcal{F}_t]$. We begin by observing that our earlier simple intuition from Section 2.3.2, i.e. recursively applying the optional stopping theorem to the appropriate remainder term, combined with definitions, immediately yields the following.

Lemma 2. *For all $k \geq 1$, $\text{OPT} = \sum_{i=1}^{k} E[\min_{t \in [1,T]} Z_t^i] + \inf_{\tau \in \mathcal{T}} E[Z_\tau^{k+1}]$. In addition, w.p.1 Z_t^k is non-negative and integrable for all $k \geq 1$ and $t \in [1, T]$; and for each $t \in [1, T]$, $\{Z_t^k, k \geq 1\}$ is a monotone decreasing sequence of random variables.*

Proof. Proof of Lemma 2 The fact that $\text{OPT} = \sum_{i=1}^{k} E[\min_{t \in [1,T]} Z_t^i] + \inf_{\tau \in \mathcal{T}} E[Z_\tau^{k+1}]$ for all $k \geq 1$ follows directly from the optional stopping theorem, as showed explicitly in Section 2.3.2. The fact that Z_t^k is non-negative and integrable for all $k \geq 1$ and $t \in [1, T]$ follows from $Z_t^1 = Z_t \geq 0$ and the fact that the martingale $\{E[\min_{i \in [1,T]} Z_i^k | \mathcal{F}_t]\}_{t \in [1,T]}$ is a lower bound on the process $\mathbf{Z}^k$, and is itself non-negative. □

We would be done (at least with the proof of Theorem 2.3.3) if $\lim_{k \to \infty} \inf_{\tau \in \mathcal{T}} E[Z_\tau^{k+1}] = 0$. We now prove that this is indeed the case.

Proof of Theorem 2.3.3: It follows from monotone convergence that $\{\mathbf{Z}^k, k \geq 1\}$ converges a.s., and thus $\{Z_T^{k+1} - Z_T^k, k \geq 1\}$ converges a.s. to 0. By definition, $Z_T^{k+1} = Z_T^k - \min_{i \in T} Z_i^k$, we thus conclude that $\{\min_{i \in [1,T]} Z_i^k, k \geq 1\}$ converges a.s. to 0. We introduce a stopping time $\tau_\infty \stackrel{\Delta}{=} \inf\{t \in [1,T] : \lim_{k \to \infty} Z_t^k = 0\}$.[1] This is a feasible stopping time, adapted to $\mathcal{F}$. By definition, the random variable $Z_{\tau_\infty}^k$ converges a.s. to 0. The fact that $(Z_t^1)_{t \in [1,T]}$ are non-negative and integrable implies that $Z_{\tau_\infty}^1$ is non-negative and integrable. Furthermore $0 \leq Z_{\tau_\infty}^{k+1} \leq Z_{\tau_\infty}^k \leq Z_{\tau_\infty}^1$ for all $k \geq 1$. Combining the above, by Lebesgue's monotone convergence theorem, we conclude that $\lim_{k \to \infty} E[Z_{\tau_\infty}^k] = 0$. Since τ_∞ is a specific stopping time, hence for all $k \geq 1$ we have $E[Z_{\tau_\infty}^k] \geq \inf_{\tau \in \mathcal{T}} E[Z_\tau^k]$. Combining all above complete the proof. □

Remark. *Theorem 2.3.3 confirms the generality of our expansion. It is valid for all optimal stopping problems as long as the underlying stochastic process* $\mathbf{Z}$ *is integrable, without imposing any additional assumptions. We suspect that in the infinite horizon or the continuous setting, the expansion is still valid under mild conditions. This is generally beyond the scope of the current work and we skip any further discussion. We believe that the expansion provides a fundamentally new perspective on the basic structure of optimal stopping (alternative to dynamic programming), which may be of independent interest to researchers in related fields. In the next section, our proof of Theorem 2.3.5 essentially yields an alternative proof of Theorem 2.3.3 under the more restrictive assumptions that* $\mathbf{Z}$ *is bounded.*

[1] If the set is empty, we by convention let $\tau_\infty = T$.

2.5 Rate of convergence and Proof of Theorem 2.3.5

In this section we prove Theorem 2.3.5, Lemma 1 and Corollary 1, our main results regarding rate of convergence. Along the way, we prove a much stronger result, which will later enable us to construct provably good approximation policies.

2.5.1 Proof of Theorem 2.3.5

We first state and prove a path-wise rate of convergence guarantee, essentially that after k iterations of our expansion, the minimum of every sample path is at most $\frac{1}{k}$. This is a much stronger result than that stated in Theorem 2.3.5, which only regards expectations. We will be able to use this pathwise convergence to construct provably accurate and efficient policies.

Lemma 3. *Suppose w.p.1 $Z_t \in [0,1]$ for all $t \in [1,T]$. Then for all $k \geq 1$, w.p.1 $\min_{t\in[1,T]} Z_t^k \leq \frac{1}{k}$.*

Proof : By definitions, measurability, and Lemma 2, for all $k \geq 1$, $Z_T^{k+1} = Z_T - \sum_{i=1}^{k} \min_{t\in[1,T]} Z_T^i \geq 0$ w.p.1. By the monotonicity ensured by Lemma 2, it follows that w.p.1, $k \times \min_{t\in[1,T]} Z_T^k \leq Z_T \leq 1$, and the desired result follows. □

Proof of Theorem 2.3.5 : By Lemma 2, $\text{OPT} = \sum_{i=1}^{k} E[\min_{t\in[1,T]} Z_t^i] + \inf_{\tau\in\mathcal{T}} E[Z_\tau^{k+1}]$. By considering the policy τ which stops the first time $Z_t^{k+1} \leq \frac{1}{k+1}$ and combining with Lemma 3 completes the proof. □

Remark. *Lemma 3 immediately yields a stopping policy: $\tau_k \stackrel{\Delta}{=} \inf\{t \in [1,T] : Z_t^k \leq \frac{1}{k}\}$. It is not difficult to verify that this policy is $\frac{1}{k}$-optimal. We will rigorously prove this*

observation and use it to design efficient good policies in Section 2.6.3.

2.5.2 Lower bound and proof of Lemma 1

This section contains a *bad* instance on which the expansion attains the worst-case convergence rate.

Proof of Lemma 1 : Consider the stopping problem such that $T = 2, D = 1, Z_t = X_t$ for $t \in [1, 2]$, and $P(Z_1 = \frac{1}{n}) = 1$, $P(Z_2 = 1) = \frac{1}{n}, P(Z_2 = 0) = 1 - \frac{1}{n}$. As $\mathbf{Z}$ is a martingale, it follows from optional stopping that $\text{OPT} = \frac{1}{n}$. It then follows from definitions and the basic preservation properties of martingales that $\mathbf{Z}^k$ is a martingale for all $k \geq 1$, and thus $\inf_{\tau \in \mathcal{T}} E[Z_\tau^k] = E[Z_1^k]$ for all $k \geq 1$. A similar argument yields that $Var[Z_1^k] = 0$ for all $k \geq 1$, and thus $Z_1^k = E[Z_1^k]$ for all $k \geq 1$. We now prove by induction that $Z_1^k = \frac{1}{n}(1 - \frac{1}{n})^k$ for all $k \geq 1$, from which the desired result immediately follows. The base case is trivial. Now, suppose the induction holds for all $j \leq k - 1$ for some $k \geq 2$. Using the martingale property and the inductive hypothesis, it follows that $Z_1^{k-1} = \frac{1}{n}(1 - \frac{1}{n})^{k-1}$, and Z_2^{k-1} equals $(1 - \frac{1}{n})^{k-1}$ w.p. $\frac{1}{n}$, and 0 w.p. $1 - \frac{1}{n}$. It follows that Z_1^k equals

$$Z_1^{k-1} - E[\min_{t=1,2} Z_t^{k-1}] \;=\; \frac{1}{n}(1 - \frac{1}{n})^{k-1} - (\frac{1}{n})^2(1 - \frac{1}{n})^{k-1} \;=\; \frac{1}{n}(1 - \frac{1}{n})^k,$$

completing the proof. □

2.5.3 Bounds for unbounded Z and proof of Corollary 1

Proof of Corollary 1: First, we claim that $E[\min_{t \in [1,T]} Z_t^k] \leq \frac{1}{k}\text{OPT}$. Indeed, this follows directly from monotonicity and Lemma 2, which also implies that it suf-

fices to prove that

$$\inf_{\tau\in\mathcal{T}} E[Z_\tau^{k+1}] \le 3 \times (\mathrm{OPT} \times E[Z_T^2])^{\frac{1}{3}} \times \frac{1}{k^{\frac{1}{3}}}.$$

To proceed, let us consider the performance of the following threshold policy. Let $x_k \stackrel{\Delta}{=} (E[X_T^2]\mathrm{OPT}(4k)^{-1})^{\frac{1}{3}}$. Consider the policy which stops at the first time that $Z_t^{k+1} \le x_k$, and simply stops at Z_T^{k+1} if no such t exists in $[1,T]$. Denoting this stopping time by τ_k, then w.p.1 $Z_{\tau_k}^{k+1} \le x_k + I(\min_{t\in[1,T]} Z_t^{k+1} > x_k) Z_T^{k+1}$, which by monotonicity (implying $Z_T^{k+1} \le Z_T$ w.p.1) is at most $x_k + I(\min_{t\in[1,T]} Z_t^{k+1} > x_k) Z_T$. Taking expectations and applying Cauchy-Schwartz and Markov's inequality, we find that $E[Z_{\tau_k}^{k+1}]$ is at most $x_k + (E[(Z_T)^2])^{\frac{1}{2}} \times \left(P(\min_{t\in[1,T]} Z_t^{k+1} > x_k)\right)^{\frac{1}{2}}$, itself at most

$$x_k + (\frac{\mathrm{OPT}k^{-1}E[(Z_T)^2]}{x_k})^{\frac{1}{2}} \quad = \quad (\mathrm{OPT}k^{-1}E[(Z_T)^2])^{\frac{1}{3}} \times (4^{\frac{1}{6}} + 4^{-\frac{1}{3}}).$$

Noting that $(2 + 4^{-\frac{1}{3}}) \le 3$ completes the proof. □

2.6 Simulation analysis, proof of Theorem 2.3.7, 2.3.8

In this section, we complete our algorithmic analysis and prove Theorem 2.3.7, 2.3.8. The algorithms are built on our novel expansion results Theorem 2.3.3, 2.3.5 and Corollary 1, along with an extensive use of Monte Carlo simulation via base simulator $\mathcal{B}$. We start by formalizing and analyzing important subroutines for computing Z_t^k and L_k.

2.6.1 Efficient randomized algorithm for computing Z_t^k

Formal definition of algorithms.

We recursively define a relevant sequence of algorithms $\{\mathcal{B}^k\}_{k\geq 1}$, which takes inputs $(t, \gamma, \epsilon, \delta)$ and returns an (additive) ϵ-approximation to $Z_t^k(\gamma)$ w.p. at least $1-\delta$. For a D by T matrix M, and $t \in [1, T]$, let $M_{[t]}$ denote the submatrix consisting of the first t columns of M. For $\epsilon, \eta \in (0,1)$, let $N(\epsilon, \eta) \stackrel{\Delta}{=} \lceil \frac{1}{2\epsilon^2} \log(\frac{2}{\eta}) \rceil$. Algorithm $\mathcal{B}^1(t, \gamma, \epsilon, \delta)$:

Return $r_t(\gamma)$

Algorithm $\mathcal{B}^{k+1}(t, \gamma, \epsilon, \delta)$:

Create a length-$\lceil 8\log(2\delta^{-1}) \rceil$ vector $\mathbf{A}^m$

For s = 1 to $\lceil 8\log(2\delta^{-1}) \rceil$

Create a length-$N(\frac{\epsilon}{4}, \frac{1}{16})$ vector $\mathbf{A}^0$

For i = 1 to $N(\frac{\epsilon}{4}, \frac{1}{16})$

Generate an ind. call to $\mathcal{B}(t, \gamma)$ and store in D by T matrix $\mathbf{A}^1$

Create a length-T vector $\mathbf{A}^2$

For j = 1 to T

Generate an ind. call to $\mathcal{B}^k(j, \mathbf{A}^1_{[j]}, \frac{\epsilon}{4}, \frac{1}{16N(\frac{\epsilon}{4}, \frac{1}{16})T})$ and store in A_j^2

Compute the minimum value of $\mathbf{A}^2$ and store in A_i^0

Generate an ind. call to $\mathcal{B}^k(t, \gamma, \frac{\epsilon}{2}, \frac{1}{8})$ and store as variable A^3

Compute $A^3 - (N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} A_i^0$ and store in A_s^m

Return the median[2] of $\mathbf{A}^m$

[2]Here we apply a popular technique known as the *median trick*. More detailed discussion, including the rigorous definition of the median and the analysis of this trick was provided in

Formal analysis of $\mathcal{B}^k$.

We now formally analyze $\mathcal{B}^k$, proving in an appropriate sense that it is indeed a "good" algorithm for approximating $Z_t^{k+1}(\gamma)$. We introduce the following additional notation.

$$f_k(\epsilon, \delta) \triangleq \log(2\delta^{-1}) \times 10^{2(k-1)^2} \times \epsilon^{-2(k-1)} \times (T+2)^{k-1} \times (1 + \log(\frac{1}{\epsilon}) + \log(T))^{k-1}.$$

Then we have

Lemma 4. *For all $k \geq 1$, $t \in [1, T]$, $\gamma \in \aleph^t$, $\epsilon, \delta \in (0, 1)$, algorithm $\mathcal{B}^k$ achieves the following when evaluated on $t, \gamma, \epsilon, \delta$: In total computational time at most $(C + G + 1) f_k(\epsilon, \delta)$, and with only access to randomness at most $f_k(\epsilon, \delta)$ calls to the base simulator $\mathcal{B}$, $\mathcal{B}^k$ returns a random number Y satisfying $P(|Y - Z_t^k(\gamma)| > \epsilon) \leq \delta$.*

We leave the proof of Lemma 4 to Technical appendix A.1.

2.6.2 Proof of Theorem 2.3.7

With Lemma 4 in hand, we now complete the proof of our main algorithmic result Theorem 2.3.7. First, let us formally define algorithm $\mathcal{A}$, which uses $\mathcal{B}^k$ and simulation to approximate OPT.

Algorithm $\mathcal{A}(\epsilon, \delta)$:

Technical appendix A.1.

Set $k = \lceil \frac{2}{\epsilon} \rceil, \alpha = \frac{\epsilon}{2}(\lceil \frac{2}{\epsilon} \rceil)^{-1}, \beta = \delta(\lceil \frac{2}{\epsilon} \rceil)^{-1}$, create a length-$k$ vector $\mathbf{L}$

For l = 1: k

Create a length-$N(\frac{\alpha}{2}, \frac{\beta}{2})$ vector $\mathbf{A}^0$

For i = 1 to $N(\frac{\alpha}{2}, \frac{\beta}{2})$

Generate an ind. call to $\mathcal{B}(0, \emptyset)$ and store in D by T matrix $\mathbf{A}^1$

Create a length-T array $\mathbf{A}^2$

For j = 1 to T

Generate an ind. call to $\mathcal{B}'(j, \mathbf{A}^1_{[j]}, \frac{\alpha}{2}, \frac{\beta}{2N(\frac{\alpha}{2}, \frac{\beta}{2})T})$ and store in $\mathbf{A}^2_j$

Compute the minimum value of $\mathbf{A}^2$ and store in $\mathbf{A}^0_i$

Compute $(N(\frac{\alpha}{2}, \frac{\beta}{2}))^{-1} \sum_{i=1}^{N(\frac{\alpha}{2}, \frac{\beta}{2})} \mathbf{A}^0_i$ and store in $\mathbf{L}_l$

Return $\sum_{l=1}^{k} \mathbf{L}_l$

Proof of Theorem 2.3.7 : In light of Lemma 4 and Theorem 2.3.5, by a union bound and triangle inequality, it suffices to individually approximate the first $\lceil \frac{2}{\epsilon} \rceil$ of the L_i, each to within additive error α with probability $1 - \beta$. The compu-

tational cost, divided by $(C + G + 1)$ is at most

$$\begin{aligned}
&\sum_{i=1}^{\lceil \frac{2}{\epsilon} \rceil} f_{i+1}(\alpha, \beta) + \lceil \frac{2}{\epsilon} \rceil + 1 \\
&\leq \lceil \frac{2}{\epsilon} \rceil f_{\lceil \frac{2}{\epsilon} \rceil + 1}(\frac{\epsilon}{2}(\lceil \frac{2}{\epsilon} \rceil)^{-1}, \delta(\lceil \frac{2}{\epsilon} \rceil)^{-1}) + \lceil \frac{2}{\epsilon} \rceil + 1 \\
&\leq 6\epsilon^{-1} f_{\lceil \frac{2}{\epsilon} \rceil + 1}(\frac{\epsilon^2}{6}, \frac{\delta\epsilon}{3}) \\
&\leq 6\epsilon^{-1} \log(6\delta^{-1}\epsilon^{-1}) 10^{2(3\epsilon^{-1})^2} (6\epsilon^{-2})^{2(3\epsilon^{-1})} (T+2)^{3\epsilon^{-1}} (1 + \log(6\epsilon^{-2}) + \log(T))^{3\epsilon^{-1}} \\
&\leq (\log(6) + \log(\epsilon^{-1}) + \log(\delta^{-1})) 10^{18\epsilon^{-2}} 6^{6\epsilon^{-1}+1} \epsilon^{-12\epsilon^{-1}-1} (T+2)^{3\epsilon^{-1}} \\
&\quad \times (1 + \log(6) + 2\log(\epsilon^{-1}) + \log(T))^{3\epsilon^{-1}} \\
&\leq \log(\delta^{-1}) \log(6) \log(\epsilon^{-1}) 10^{18\epsilon^{-2}} 6^{6\epsilon^{-1}+1} \epsilon^{-12\epsilon^{-1}-1} (T+2)^{3\epsilon^{-1}} \\
&\quad \times (\log(6))^{3\epsilon^{-1}} (1 + \log(T))^{3\epsilon^{-1}} (2\log(\epsilon^{-1}))^{3\epsilon^{-1}} \\
&\leq \log(\delta^{-1}) 10^{18\epsilon^{-2}} 6^{9\epsilon^{-1}+2} \epsilon^{-15\epsilon^{-1}-2} 2^{6\epsilon^{-1}} T^{6\epsilon^{-1}} \\
&\leq \log(\delta^{-1}) \times \exp(100\epsilon^{-2}) \times T^{6\epsilon^{-1}}.
\end{aligned}$$

The analysis for the number of calls to the number of calls to the base simulator follows nearly identically, and we omit the details. Combining the above completes the proof. □

2.6.3 Good policies and proof of Theorem 2.3.8

In this section, we discuss how to use our simulation-based approach to implement good approximate stopping strategies, proving Theorem 2.3.8. We begin with the following lemma, relating the value achieved by a single stopping policy across different stopping problems (defined by $\mathbf{Z}^k$).

Lemma 5. *For all (possibly randomized) integer-valued stopping times τ adapted to $\mathcal{F}$ which w.p.1 belonging to $[1, T]$, and all $k \geq 1$, $E[Z_\tau] = E[Z^k_\tau] + \sum_{i=1}^{k-1} E[\min_{t \in [1,T]} Z^i_t]$.*

Proof : We prove only in the case of non-randomized stopping times, as the general setting then follows from a straightforward conditioning argument. We proceed by induction. The base case $k = 1$ follows from definitions. Now, suppose the induction is true for some $k \geq 1$. Then again from definitions and optional stopping,

$$E[Z_\tau^{k+1}] \;=\; E[Z_\tau^k - E[\min_{t\in[1,T]} Z_t^k|\mathcal{F}_\tau]] \;=\; E[Z_\tau^k] - E[\min_{t\in[1,T]} Z_t^k],$$

itself (by induction) equal to $E[Z_\tau] - \sum_{i=1}^k E[\min_{t\in[1,T]} Z_t^i]$, which after rearranging completes the proof. □

Combining Lemma 5 with Lemma 3 and Theorem 2.3.3, we are led to the following corollary.

Corollary 2. *For $k \geq 1$, let τ_k denote the stopping time that stops the first time that $Z_t^k \leq \frac{1}{k}$, where by Lemma 3 such a time exists w.p.1. Then $E[Z_{\tau_k}] - \mathrm{OPT} \leq \frac{1}{k}$.*

Now, we would be done, if not for the fact that we cannot compute Z_t^k exactly in an efficient manner. However, in light of Lemma 4, it is clear how to proceed. In every time period t, we will use simulation to estimate $Z_t^k(X_{[t]})$ (for appropriate k) for the given history $X_{[t]}$ observed so far, and do so with sufficient accuracy and high enough probability to make sure all bounds go through. Let us now make this precise. For any given $\epsilon > 0$, we begin by defining an appropriate (randomized) stopping time τ_ϵ. Namely, τ_ϵ is defined as follows. At time 1, after seeing $X_{[1]}$, make an independent call to $\mathcal{B}^{\lceil 4\epsilon^{-1}\rceil}\left(1, X_{[1]}, \frac{\epsilon}{4}, \frac{\epsilon}{4T}\right)$. If the value returned is at most $\frac{1}{2}\epsilon$, stop. If not, continue. We define the future behavior inductively as follows. Suppose that for some $t \in [1, T-2]$, we have not yet stopped by the end of period t. At time $t+1$, after observing X_{t+1}, make an independent call to $\mathcal{B}^{\lceil 4\epsilon^{-1}\rceil}\left(t+1, X_{[t+1]}, \frac{\epsilon}{4}, \frac{\epsilon}{4T}\right)$. If the value returned is at most $\frac{1}{2}\epsilon$, stop.

If not, continue. Finally, if we have not yet stopped by period T, stop in period T. It is easily verified that for any $\epsilon \in (0,1)$, τ_ϵ is a well-defined, appropriately adapted, randomized stopping time. We now use τ_ϵ to complete the proof of Theorem 2.3.8.

Proof of Theorem 2.3.8 : Let $k_\epsilon \stackrel{\Delta}{=} \lceil \frac{4}{\epsilon} \rceil$. Let $\mathcal{G}_{1,\epsilon}$ denote the event

$$\left\{ \left| \mathcal{B}^{\lceil 4\epsilon^{-1} \rceil}\left(t, X_{[t]}, \frac{\epsilon}{4}, \frac{\epsilon}{4T}\right) - Z_t^{k_\epsilon}(X_{[t]}) \right| \leq \frac{\epsilon}{4} \quad \forall \ t \in [1,T] \right\},$$

and $\mathcal{G}_{2,\epsilon}$ denote the event

$$\left\{ \exists t \in [1,T] \text{ such that } \mathcal{B}^{\lceil 4\epsilon^{-1} \rceil}\left(t, X_{[t]}, \frac{\epsilon}{4}, \frac{\epsilon}{4T}\right) \leq \frac{1}{2}\epsilon \right\},$$

and $\mathcal{G}_{3,\epsilon}$ denote the event $\{Z_{\tau_\epsilon}^{k_\epsilon} \leq \frac{3}{4}\epsilon\}$. Observe that Lemma 3, definitions, and several straightforward union bounds and applications of the triangle inequality ensure that : 1. $P(\mathcal{G}_{1,\epsilon}) \geq 1 - \frac{\epsilon}{4}$; 2.$P(\mathcal{G}_{2,\epsilon}|\mathcal{G}_{1,\epsilon}) = 1$; 3.$P(\mathcal{G}_{3,\epsilon}|\mathcal{G}_{1,\epsilon} \bigcap \mathcal{G}_{2,\epsilon}) = 1$. It follows that $(\mathcal{G}_{3,\epsilon}^c) \leq \frac{\epsilon}{4}$, and thus since by assumption and monotonicity $P(Z_t^{k_\epsilon} \leq 1) = 1$ for all $t \in [1,T]$,

$$\begin{aligned} E[Z_{\tau_\epsilon}^{k_\epsilon}] &= E[Z_{\tau_\epsilon}^{k_\epsilon} I(\mathcal{G}_{3,\epsilon})] + E[Z_{\tau_\epsilon}^{k_\epsilon} I(\mathcal{G}_{3,\epsilon}^c)] \\ &\leq \frac{3}{4}\epsilon + E[I(\mathcal{G}_{3,\epsilon}^c)] \quad \leq \quad \epsilon. \end{aligned}$$

Combining with Lemma 5, Lemma 3, and Theorem 2.3.3, we complete the proof of the first part of the lemma. As for the computational and sampling cost, the algorithm in each time step makes one call to $\mathcal{B}^{\lceil 4\epsilon^{-1} \rceil}$ with parameters $4\epsilon^{-1}$ and $4T\epsilon^{-1}$, and the results follow directly from Lemma 4 along with some straightforward algebra. We omit the details. □

2.7 Numerical examples

We justify our theoretical findings by implementing our algorithms on two classical examples in high-dimensional American option pricing. These examples are considered benchmark and appear in multiple previous works. We report optimal value approximations derived from truncating the expansion at different levels, as well as the performance of an heuristic policy inspired by the expansion. We start this section with a general description of what we do, after which we introduce with details the two concrete examples, and plot all numerical results, respectively.

In both examples, the essential underlying problems are in the maximization setting. We implement the maximization variant of our algorithm as stated in Technical appendix A.2, whose performance is theoretically guaranteed by Theorem 2.3.9. The algorithm as stated requires as input a truncation threshold U_0, which is an enormous number assumed for the purpose of algorithmic analysis, and is treated as infinity in our numerical implementation. As a result, our implementation can be simply described as follows. For a maximization problem $\widehat{\mathrm{OPT}} = \sup_{\tau\in\mathcal{T}} E[Z_\tau]$, we consider $Z_t^1 \triangleq E[\max_{1\le s\le T} Z_s|\mathcal{F}_t] - Z_t,\ t \in [1,T]$. We have $\widehat{\mathrm{OPT}} = \sup_{\tau\in\mathcal{T}} E[Z_\tau] = E[\max_{1\le t\le T} Z_t] - \inf_{\tau\in\mathcal{T}} E[Z_\tau^1]$. Theorem 2.3.3 then promises that an expansion exists for the resulting minimization problem: $\inf_{\tau\in\mathcal{T}} E[Z_\tau^1] = \sum_{k=2}^{\infty} E[\min_{1\le t\le T} Z_t^k] \triangleq \sum_{k=2}^{\infty} L_k$, where $Z_t^k = Z_t^{k-1} - E[\min_{1\le s\le T} Z_s^{k-1}|\mathcal{F}_t]$ for all k,t. Let $L_1 \triangleq E[\max_{1\le t\le T} Z_t]$. We thus have $\widehat{\mathrm{OPT}} = L_1 - \sum_{k=2}^{\infty} L_k$, a "maximization" expansion. Now we can approximate $\widehat{\mathrm{OPT}}$ by truncating this expansion and calculating the first several terms of L_k via Monte Carlo simulation. This algorithm is identical to $\hat{\mathcal{A}}$, with U_0 set to ∞. Truncating the expansion $\widehat{\mathrm{OPT}} = L_1 - \sum_{k=2}^{\infty} L_k$ yields biased-high approximations. Therefore, the outputs of

the algorithm are (with high probability) greater than $\widehat{\text{OPT}}$.

The algorithm is tested on multiple problem instances in both examples. For each different problem instance, we demonstrate the optimal value approximations derived from truncating the expansion after one, two and three terms respectively. We observe in both examples that our approximations converge to the true optimal value at a rate faster than the worst-case rate (i.e. $O(\frac{1}{k})$) of Theorem 2.3.5. The exact choice of number of sample paths used when approximating L_1, L_2 and L_3 will be specified later in the detailed description of the two examples. We haven't put specific effort into optimizing our use of the sampling resources. Certain adjustment of the allocation of the computational/sampling resources may lead to results with similar accuracy but in shorter run time.

Bermudan max calls with D assets

This is a standard benchmark example of high-dimensional American option pricing, appearing in many previous works including [13], [130], [142], [225] and [35]. We first recall the setup: a model with D identically distributed assets is considered where each asset yields dividends with rate δ. All assets are assumed to have the same initial value, i.e. $X_0^d = x_0, \quad d = 1, ..., D$. The risk-neutral dynamic of this asset system is given by

$$dX_t^d = (r - \delta)X_t^d dt + \sigma X_t^d dW_t^d, \qquad d = 1, ..., D,$$

where $W_t^d, d = 1, ..., D$ are independent one-dimensional Brownian motions and r, δ, σ are constants. At one exercising date $t \in \{T_0, ..., T_{\mathcal{J}}\}$, the holder of the option may exercise to receive the payoff

$$h(X_t) = \left(\max(X_t^1, ..., X_t^D) - \kappa \right)^+.$$

We consider in this example $T_j = jT/\mathcal{J}, j = 0, ..., \mathcal{J}$, with $T = 3$ and $\mathcal{J} = 9$. The corresponding price $\mathcal{P}$ of the Bermudan max call is the solution to the following

optimal stopping problem

$$\mathcal{P} \stackrel{\Delta}{=} \sup_{\tau \in \{T_0,\ldots,T_9\}} E[e^{-r\tau} h(X_\tau)].$$

Our algorithm uses Monte Carlo simulation. We approximate the first three terms of our "maximization" expansion

$$L_1 = E[\max_{i\in[0,9]} Z^1_{T_i}] \quad ; \quad L_2 = E[\min_{i\in[0,9]} Z^2_{T_i}] \quad ; \quad L_3 = E[\min_{i\in[0,9]} Z^3_{T_i}],$$

where

$$Z^1_{T_i} = e^{-rT_i} h(X_{T_i}) \quad ; \quad Z^2_{T_i} = E[\max_{j\in[0,9]} e^{-rT_j} h(X_{T_j})|\mathcal{F}_{T_i}] - e^{-rT_i} h(X_{T_i}).$$

$$Z^3_{T_i} = E[\max_{j\in[0,9]} e^{-rT_j} h(X_{T_j})|\mathcal{F}_{T_i}] - e^{-rT_i} h(X_{T_i}) - E\Big[\min_{k\in[0,9]} \Big(E[e^{-rT_j} \max_{j\in[0,9]} h(X_{T_j})|\mathcal{F}_{T_k}] - e^{-rT_k} h(X_{T_k})\Big)\Big|\mathcal{F}_{T_i}\Big].$$

Denote by $\mathcal{Z}^k_{T_i}$ and $\mathcal{L}_j$ the approximations of $Z^k_{T_i}$ and L_j that we obtain for all k, i, j, respectively. $100,000$ sample paths are used for computing $\mathcal{L}_1$; 10000/1000 sample paths are used in the outer/ inner level simulation for computing $\mathcal{L}_2$. For computing $\mathcal{L}_3$, 1000 sample paths are used in the first(outer) level of simulation, 100 sample paths are used in each of the second level of simulation, and 1000 sample paths are used in each of the third (inner) level of simulation. Our algorithm then outputs $\mathcal{L}_1, \mathcal{L}_1 - \mathcal{L}_2$ and $\mathcal{L}_1 - \mathcal{L}_2 - \mathcal{L}_3$, which will be denoted by $\mathcal{E}^1, \mathcal{E}^2$ and $\mathcal{E}^3$ respectively.

We follow the classical choice of model parameters: $\kappa = 100, r = 0.05, \sigma = 0.2, \delta = 0.1$. We test our algorithm with the dimension D taking value in $\{2, 3, 5\}$ and the initial asset price x_0 taking value in $\{90, 100, 110\}$. Table 2.7 below summarizes our results. The standard deviation of the results are derived from independently running the algorithm multiple times. The 95% confidence interval of $\mathcal{P}$ quoted from [13] are included as benchmarks. The relative errors of $\mathcal{E}^1, \mathcal{E}^2$

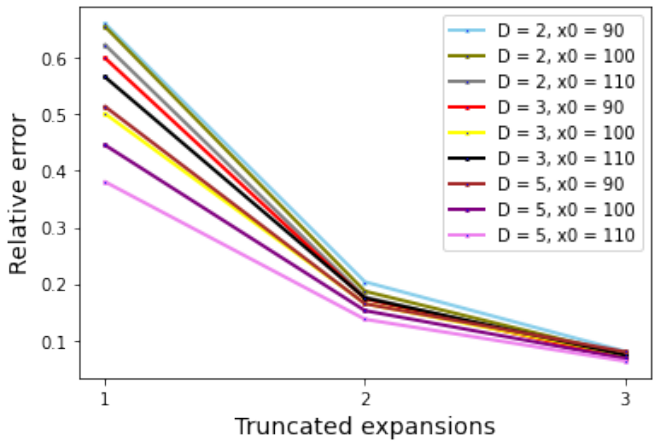

Figure 2.1: Relative errors of $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ for Bermudan max call with different D and x_0

and $\mathcal{E}^2$ in different settings (i.e. choices of D and x_0) are plotted in Figure 2.1. Here we take the middle point of the 95% A & B interval as our benchmark value for $\mathcal{P}$. We find that the truncated expansion $\mathcal{E}_i$ converges faster than the guarantee provided by our theoretical results.

D	x_0	$\mathcal{E}^1$ (SD)	$\mathcal{E}^2$ (SD)	$\mathcal{E}^3$ (SD)	*A&B* Price Interval
2	90	13.38(0.02)	9.70(0.04)	8.71 (0.05)	[8.053, 8.082]
	100	23.02(0.02)	16.51(0.05)	14.95 (0.06)	[13.892, 13.934]
	110	34.61(0.02)	25.10(0.05)	22.80(0.06)	[21.316, 21.359]
3	90	18.04(0.02)	13.24(0.05)	12.04 (0.06)	[11.265, 11.308]
	100	29.28(0.02)	21.97(0.05)	20.09 (0.06)	[18.661, 18.728]
	110	41.43(0.02)	32.16(0.05)	29.66(0.07)	[27.512, 27.663]
5	90	25.17(0.02)	19.37 (0.05)	17.96 (0.06)	[16.602, 16.655]
	100	37.87(0.02)	30.20(0.05)	27.98 (0.06)	[26.109, 26.292]
	110	50.76(0.02)	41.82(0.06)	39.10(0.08)	[36.704, 36.832]

***Table 2.7**: Numerical simulations for pricing Bermudan max calls with parameters $\kappa = 100, r = 0.05, \sigma = 0.2, \delta = 0.1$ and different D and x_0*

A path-dependent financial derivative

This example appeared first in [243] and later in [33]. The basic setup is as

follows: a single asset evolves according to the dynamic

$$dX_t = rX_t dt + \sigma X_t dW_t, \qquad t \geq -100$$

where W_t is a standard Brownian motion and r, δ are constants. Let $\mathbf{X}_{[t]} \stackrel{\Delta}{=} (X_1, ..., X_t)$ denote the entire trajectory up to time t. The financial derivative we consider here allows the holder, on one exercising date $t \in \{0, 1, 2, ...\}$, to exercise and receive the payoff $h_t(\mathbf{X}_{[t]}) = X_t / X_{t-100}$. The price of this financial derivative is then determined by the following infinite horizon optimal stopping problem:

$$\mathcal{P} \stackrel{\Delta}{=} \sup_{\tau \geq 0} E[e^{-r\tau} h_\tau(\mathbf{X}_{[\tau]})] = \sup_{\tau \geq 0} E\Big[e^{-r\tau} \frac{X_\tau}{X_{\tau-100}}\Big].$$

The fact that the payoff at any time depends on the asset value 100 days ago makes this pricing problem path-dependent, and according to [243], essentially high-dimensional (with 100 dim.).

For the sake of numerical implementation, we follow [33] and consider the following finite horizon approximation:

$$\mathcal{P}_T \stackrel{\Delta}{=} \sup_{\tau \in \{0,1,...,T\}} E[e^{-r\tau} h_\tau(\mathbf{X}_{[\tau]})] = \sup_{\tau \in \{0,1,...,T\}} E\Big[e^{-r\tau} \frac{X_\tau}{X_{\tau-100}}\Big].$$

We have $\mathcal{P}_T < \mathcal{P}$ and $\lim_{T\to\infty} \mathcal{P}_T = \mathcal{P}$.

Our algorithm uses Monte Carlo simulation. We approximate the first two terms of our "maximization" expansion

$$L_1 = E[\max_{t \in [0,T]} Z_t^1] \quad ; \quad L_2 = E[\min_{t \in [0,T]} Z_t^2]$$

where

$$Z_t^1 = e^{-rt} h(\mathbf{X}_{[t]}) \quad ; \quad Z_t^2 = E[\max_{s \in [0,T]} e^{-rs} h(\mathbf{X}_{[s]}) | \mathcal{F}_t] - e^{-rt} h(\mathbf{X}_{[t]}).$$

Denote by $\mathcal{Z}_t^k$ and $\mathcal{L}_j$ the approximations of Z_t^k and L_j that we obtain for all k, i, j, respectively. 100000 sample paths are used for computing $\mathcal{L}_1$; 1000/1000

sample paths are used in the outer/ inner level simulation for computing $\mathcal{L}_2$. Our algorithm then outputs $\mathcal{L}_1$ and $\mathcal{L}_1 - \mathcal{L}_2$, which will be denoted by $\mathcal{E}^1$ and $\mathcal{E}^2$ respectively.

As in [243], we assume $r = 0.0004, \sigma = 0.02$. We test our algorithm with different choice of truncation level T as did in [33]. The results are presented in Table 2, along with the numerical reports taken from [33] for comparison. According to [243], 1.282 is a lower bound for $\mathcal{P}$. Recall that in our numerical test, the output values $\mathcal{E}^1$ and $\mathcal{E}^2$ are both upper bounds. We found that, contrary to what [33] stated, $\mathcal{P}_T$ is significantly smaller than 1.282 for $T < 500$, and converges after $T \sim 750$.

T	$\mathcal{E}^1$ (SD)	$\mathcal{E}^2$ (SD)	Results in Becker et al. (2019)
100	1.2525(0.001)	1.2028(0.001)	1.2721
150	1.2961(0.001)	1.2402(0.001)	1.2821
200	1.3250(0.001)	1.2626(0.001)	1.2894
250	1.3450(0.001)	1.2806(0.001)	1.2959
500	1.3909(0.001)	1.3221(0.001)	N/A
750	1.4070(0.001)	1.3412(0.001)	N/A
1000	1.4074(0.001)	1.3432(0.001)	1.3002

Table 2.1: Numerical simulations of our algorithm for pricing the path-dependent derivative of [243] with $r = 0.0004, \sigma = 0.02$ and different T.

2.8 Conclusion

In this work we developed polynomial-time approximation algorithms for the fundamental problem of optimal stopping with high-dimensionality and full

path-dependence. Our simulation-based algorithms come with provable performance guarantees under mild assumptions, with the run time/sample complexity scaling only polynomially in the time horizon T and effectively independent of the dimension D. Our algorithms give the first PRAS analogy (under generative model) in the context of optimal stopping, proving the approximability of this fundamental model. We believe our algorithm sheds light on a class of new algorithms for optimal stopping, and more generally, sequential decision-making/optimal control/RL problems in the high-dimensional regime, that have the potential to efficiently find provably near-optimal policies both theoretically and practically.

Our work leaves many interesting directions for future research.

The design of practically competitive algorithms. In this work, our results and analysis are a proof-of-concept that such a trade-off between accuracy and computational / sample complexity is theoretically possible. Combining our approach with other heuristics (e.g. from ADP and simulation) to improve speed and accuracy, and rigorously comparing to past approaches, will of course be crucial for moving from the proof-of-concept stage to a useful tool for real applications in finance, healthcare and engineering. As an example, we have made no effort to optimize our use of samples, and better-understanding how to allocate samples between the different "levels" of our nested simulations, and/or how to reuse and recombine samples more intelligently, could lead to significant speedups. The same goes for applying e.g. multi-level Monte Carlo, importance sampling and change-of-measure techniques and various "warm-start", preprocessing tricks.

Better theoretical understanding of convergence. We provided several bounds on the rate of convergence of our approach, in various settings. We suspect that in many cases our analysis will be loose, and one may get quite accurate results using only a few terms of the relevant series. It is interesting to understand this phenomena both in the general setting of optimal stopping, as well as for more structured problems such as Robbins' problem ([69]). If one suspected that for any particular instance the expansion was converging more rapidly than suggested by our theoretical results, one could derive tighter upper bounds by simply exhibiting a stopping time τ for which $E[Z_\tau^{k+1}]$ was small, and formalizing such a procedure may also be interesting to consider.

Generalization to stochastic control broadly. We believe that our methodology can be extended to a broad family of stochastic control problems. The first step here would be the extension to multiple stopping, which follows almost directly from our current analysis in light of the well-known (recursive) relation between multiple stopping and optimal stopping ([46]). Indeed, there has been much recent progress on understanding the Bayesian regret for such multiple stopping problems ([15, 71, 249]), and drawing connections to our own work remains an interesting direction for future research. Of course, there is also the broader question of how far our methodology can be extended to general stochastic control problems, while maintaining the relevant notions of tractability. Using reductions similar to those of [47], it seems likely that control problems with few actions, in which one cannot change action too many times (in an appropriate sense), may be a good starting point here.

Lower bounds, randomization, and computational complexity. An interesting

set of questions revolve around proving lower bounds on the computational and sample complexity for the problems studied, e.g. path-dependent optimal stopping. There has been much interesting recent work laying out a theory of computational complexity (with positive and negative results) in the settings of stochastic control and reinforcement learning ([139, 140, 84, 233]), and the pricing of complex financial products ([51, 246, 17, 61]). Better understanding the connection between our approach and those works remains an interesting direction for future research. A key question here centers around the use of randomization and different notions of approximation, as well as questions such as the interaction between computational complexity and sample complexity.

Implications for robust optimal stopping. Our approach may also be helpful in shedding new insight into problems in so-called robust optimal stopping ([30, 209, 135]), as our expansions are general and do not depend (structurally) on the particular distribution under consideration.

Application to problems in operations management, pricing, and mechanism design. Another area where optimal stopping and other such tools have proven useful is in the domain of operations management, mechanism design, and pricing problems ([58, 210, 1, 102, 195]). Extending our results to this setting, remains an interesting direction for future research.

CHAPTER 3

EFFICIENT ALGORITHMS FOR DATA-DRIVEN SEQUENTIAL DECISION-MAKING UNDER THE LIMITED-MOVE CONSTRAINT

3.1 Introduction

Consider a decision-maker (DM) in a system with a randomly evolving state. Throughout the decision-making process, the DM can observe the system's state evolution and decide to take certain actions (e.g. price adjustments, inventory ordering) at a *limited* number of times, yielding a reward that depends on both the actions taken and the trajectory of the state. The DM seeks a dynamic policy that determines, in a real-time fashion, whether to take action and which action to take, so as to maximize the reward.

The sequential decision-making paradigm of this kind is becoming increasingly popular across a variety of industries, because it captures the critical dependence of companies' major operational decisions on certain *exogenous* factors (represented by the system's state), in particular, commodity prices. Indeed, driven by uncontrollable global demand and supply, commodity prices are notoriously unpredictable and volatile (see e.g. [217, 207], or a vivid real-world case: WTI Crude Oil prices). Meanwhile, they can significantly impact companies' profits in many industries, especially those in resource-intensive manufacturing industries such as chemicals, food, automotive, etc. Coping with fluctuating commodity prices to achieve profitability is a crucial task, and the sequential decision-making paradigm is tailor-made for this task, as we illustrate through concrete examples.

Example (Dynamic pricing)**.** *Consider a paint manufacturer who needs to set prices*

for its end products over a fixed period of time, say a year. The paint industry typically suffers from uncertain raw material costs. Indeed, raw materials account for 60% *to* 70% *of the net sales of paints. Moreover, many of these raw materials are petroleum-based, thus the prices are correlated with oil prices which are highly volatile. Cost-plus pricing is a natural strategy to counter such raw material price fluctuation, but the resulting frequent price adjustment is particularly unwelcome, as is generally the case in chemicals markets. Alternatively, manufacturers often "schedule a cadence of price changes once or twice a year in ways that the value chain can absorb" (Bain & Company 2018), which, when executed optimally, can already bring huge profit improvement. When and how should the paint manufacturer make these price changes facing the fluctuating raw material prices? This question fits perfectly into the sequential decision-making paradigm.*

Example (Swing option pricing)**.** *Swing options are widely used as hedging instruments in natural gas, electricity, and other highly volatile commodity markets. Often bundled with base-load forward contracts which specify the purchase of the underlying commodities at a prescribed (i.e. base-load) delivery rate and at a predetermined price, swing options offer certain extra "flexibility-of-delivery": the option holder is endowed with a limited number of opportunities/rights to vary the amount of one-time delivery (see [152], [75] or [247] for further details). The market value of a swing option is equal to the maximum profit the holder can achieve by optimally exercising the rights. Specifically, the option holder needs to determine the best timings, i.e. when to exercise and the corresponding, best amounts of delivery, i.e. how to exercise, in a real-time fashion facing the price uncertainty of the underlying commodity. This problem again fits into the sequential decision-making paradigm.*

Unfortunately, baking commodity prices and/or other relevant exogenous factors into sequential decision-making poses severe computational challenges,

because they are often *high-dimensional* and *path-dependent*. Indeed, it's common in chemicals and food industries that end products require multiple raw materials as input, thus manufacturers need to track the dynamics of all the raw material prices. As a concrete example, 15 to 20 different commodities and purchased intermediates, served as pigments, solvents, resins and various other additives, are used in the manufacturing of one single can of paint. Furthermore, the prices of commodities and commodity-based futures generally exhibit long memory (see [144, 216]), meaning that their future movements are based on their historical trajectories (hence path-dependent) rather than simply their current positions. Sequential decision-making problems associated with such high-dimensional and path-dependent underlying state are well known to suffer from the "curse of dimensionality": any straightforward attempt at dynamic programming (DP) will require a computational time scaling exponentially in the dimension, the time horizon, and/or both. There generally lacks a method to efficiently solve for, or even just approximate (within certain provable precision) the optimal solution as the dimension/the time horizon scale up. Even the simplest "one-move" case (with only one chance to take action, e.g. high-low promotion, exercising American options) is highly nontrivial, which essentially boils down to the celebrated, computationally intractable high-dimensional optimal stopping problem as discussed in a great depth in Chapter 2.

Such fundamental challenge has attracted a considerable attention across different literatures, varying from dynamic pricing and revenue management to stochastic control and computational finance. The majority of the works sidestep the computational difficulty either by imposing strong structural assumptions on the system state and/or the reward functions, such as independent and identically distributed state evolution and linear (in the under-

lying state) reward structures, or through efficient heuristic algorithms such as approximate dynamic programming (ADP) methods that generally lack strong performance guarantee (see literature review for details). While many of these works/methods indeed achieve great empirical successes for problems under certain specific assumptions, the fundamental question still remains unanswered: *for sequential decision-making problems associated with general high-dimensional and path-dependent state and reward structures, does there exist an approach that both* (1) *enjoys a strong performance guarantee, i.e. can make provably near-optimal decisions and* (2) *is in theory efficient, i.e. takes polynomial runtime(in dimension and time horizon)?*

In this work, we propose a new randomized algorithm that addresses the above question. Our algorithm has the following novel features:

- The algorithm admits an elegant trade-off between performance and efficiency through a control variable ϵ. More specifically, for any fixed $\epsilon > 0$, the algorithm can output an ϵ-optimal policy and a $(1-\epsilon)$ approximation of the optimal total reward in a runtime scaling only *polynomially* in the time horizon, and effectively *independent* of the dimension (implicitly depending on the dimension only through certain simulation costs, which will be specified later). The guarantee holds for problems with arbitrary state evolution and reward structure, subject only to the limited-move constraint (limited number of chances to take action), partly providing an affirmative answer to the above question.
- The algorithm is data-driven and model-free. Indeed, it only requires a generative model, i.e. the sample access to the system, rather than any knowledge of the system's actual underlying probability distribution/ state evolution formula or reward functions. Formally, such a generative

model is able to generate independent copies of state trajectories and the corresponding rewards.

- The key building block of the algorithm is the main result of Chapter 2, i.e. Theorem 2.3.3 and Theorem 2.3.7, that for the first time addresses the curse of dimensionality for *optimal stopping* in a general sense. With the observation that any sequential decision-making problem can be framed as recursively solving the so called "one-move" problems (i.e. DM has only one chance to take action, see Proposition 1), we are able to leverage [134] to devise our efficient approximation algorithm.

3.1.1 Main contributions

We make the following contribution.

- **Approximability results and PTAS.**
  We prove that a large class of sequential decision-making problems, in particular those subject to the limited-move constraint, are in theory *efficiently approximable*. For computationally challenging optimization problems, approximability represents an important threshold that distinguishes their hardness, and is central to the study of operations research and theoretical computer science. Recall that the term *polynomial-time-approximation-schemes (PTAS)* refers to such algorithms that for any fixed ϵ, the runtime for achieving $(1-\epsilon)-$approximation is polynomial in the problem size (independent of specific problem instances). Our algorithm is essentially a (randomized) PTAS-analogy, and hence a proof-of-concept of approximability of the unified sequential decision-making problem subject to the

limited-move constraint, which includes dynamic pricing, option pricing and various other real world cases as specific instances, and which allows for arbitrary underlying state evolution and reward structures.

3.1.2 Organization

The rest of this chapter is organized as follows. Following a literature review, in section 3.3, we introduce the model and assumptions. In section 3.4, we build the theoretical foundation of our algorithm. The main algorithms and the main results, i.e. the algorithms' performance and complexity will be presented in section 3.5. Finally, we sketch the key steps of the analysis in section 3.6.

3.2 Literature review

There are a large amount of papers addressing sequential decision-making problems. They come from diverse practical fields, and we have no intention to survey the entire literature. Our focus here will be on the presence of high-dimensional and path-dependent exogenous factors and the computational challenge they impose on solving the associated decision-making problems.

The main application of finding pricing strategy to manage fluctuating prices of multiple raw materials relates our work to the dynamic pricing literature. Recently, there are a stream of works considering dynamic pricing with high-dimensional underlying features (see [12, 155, 220, 96, 166, 154, 24, 191]). Their high-dimensional feature vectors are often used to describe different cus-

tomers/products, and are often assumed to be i.i.d. over time. These works also often assume the demand/willingness-to-pay has some special structures (e.g. linear in the underlying features). The majority of these works are in the setting where certain parameters of the demand model are unknown a priori but can be "learned" over time. Dynamic pricing thus aims at earning (the revenue/profit) while learning (the unknown demand model parameters), i.e. with a contextual-bandit-type structure. Such dynamic pricing and learning problems have received considerable academic attentions in the past decade, see e.g. [53, 141, 88, 115, 14, 65, 166, 107, 192, 214, 118, 50]. Our work differs from the above line of research in the following ways: 1. we do not aim at learning the demand model or the evolution of the underlying state, rather we assume access to the underlying model via a black-box simulator, and solve the decision-making problem directly. 2. Apart from calling the simulator, our algorithm is completely model-free. Namely, it does not rely on any special model structure, allowing for arbitrarily complex demand and non i.i.d. non stationary customer features. The main challenge in our setting is thus the computational issue arising from solving such general high-dimensional stochastic DP, rather than the "learning vs earning" trade-off.

Another related line of research in the dynamic pricing literature investigates the case where demands can depend on past information, such as past price sequences, past sales data etc. i.e. path-dependent. For example, [136, 176, 218, 205, 137, 148, 83, 82, 95, 97, 147, 250, 98, 99] consider the presence of the *reference effect*, taking into account the impact of customer's memory of past prices on their current willingness-to-pay. Other examples include the network effect (demand depending positively on the past sales, see [48, 7, 231, 74, 73, 221, 232]), the scarcity effect (demands depending negatively

on remaining inventory availability; see [245, 22, 103, 189]), and the billboard effect(demands depending positively on remaining inventory availability, see e.g. [255, 2, 182, 76]). Most of the aforementioned works focus on qualitatively/empirically demonstrating how the presence of such path-dependent effects affects the optimal pricing strategy/optimal revenue, providing interesting economic insights. There models are often restrictive/for illustrative purposes, assuming e.g. linear demand structure, simple,independently random social networks etc. Here the most relevant work to ours are [5, 95, 98]. [5] builds and analyzes a model with state-dependent demand. They combine a fluid LP approximation with re-solving technique to come up with a heuristic, periodic pricing algorithm (i.e. satisfying our limited-move constraint of Assumption 1) that is asymptotically optimal. Comparing to their work, our algorithm applies to an even more general setting, enjoying a provable, non asymptotic performance guarantee for essentially all DP, regardless of the specific form of path dependence/demand structure. [95] considers and formulates a promotion optimization problem with reference price effect as a maximum weighted path problem, and shows that the problem is NP-hard when the memory is long. They propose an algorithm whose complexity scales linearly in the time horizon T and the size of the price ladder, but exponential in the memory of the model. They also present an efficient method when the reference effect follows a specific model. Their methods and ours share the similarity that no assumptions are imposed on the structure of the demand functions. The difference is that the complexity of our algorithm depends only polynomially on the time horizon T and the size of the action set even if the memory is $\Omega(T)$. In [98], the authors proposes an algorithm that output a performance-guaranteed high-low type pricing strategy in polynomial time, when the demand follows a special

model, the bounded peak-end model. Comparing to their work, our model-free algorithm applies to problems with more general demand structures. Besides, we essentially have a PTAS in the settings of multiple changes of price, whereas in [98] high-low pricing marks price down only once.

The works studying dynamic pricing with limited price changes are also relevant to this work. In many different settings, even one-price policies (those picking an initial price and sticking to it throughout the selling season) are reported to have provably good performance guarantee (see eg. [123, 122, 87, 196, 52]). [57, 77, 206] study pricing contexts where frequent changes of price is either costly or inapplicable. [80] propose a heuristic algorithm with performance guarantee that adjusts the prices infrequently. [88] proves a bound that concisely links the performance guarantee with the number of price changes. [5, 95, 97], as we discussed above, also assume limited price changes.

Our other application of pricing swing options in commodity markets draws a connection between this work and the literature of option pricing, which is one of the central topics in mathematical finance. Here we mainly focus on computational aspects of the evaluation of swing options and the related optimal multiple stopping problems. Earlier works on swing option pricing often restrict to low-dimensional settings and assume special underlying distributions. ([242, 181, 152] etc.) These approaches, which relies on discretizing the underlying state (i.e. the "tree" model), become computationally intractable in high-dimensional setting (with d, T large). Therefore, many later works have been attempting to develop dimension-reduction techniques to efficiently approximate the corresponding dynamic program. For example, [27, 109, 120, 149] extend the least square Monte Carlo approach of [193]; [138] proposes a method

that reduce the dimension of both the state space and the action space; [46] extends the policy iteration type of method of [175]. [185] uses piecewise linear functions as value function approximations; [199] extends the stochastic mesh methods proposed in [64]; [25] extends the optimal quantization methods that reduce the state space. These approximate dynamic programming (ADP) methods, often derived from existing approaches for approximating optimal (single) stopping problem, are mostly heuristics that lack theoretical performance guarantee. Another main stream of research, including [49, 104, 252, 105, 113, 170], looks at the continuous problem and borrows techniques from PDE. The discussion of these approaches is beyond the scope of this work. In general, they also can't escape the curse of dimensionality while enjoying strong performance guarantee.

Since the seminal work [106] and the follow-up works [225, 142], duality approach, as a fundamentally new recipe for solving complex optimal stopping/ option pricing problems, has become more and more popular in the literature. Such duality approaches seek a "dual representation" of the problem, no longer requiring solving a DP, but boiling down to the search of an optimal martingale. In the specific setting of swing option pricing and optimal multiple stopping, [201] and [229] provide two different dual representations and propose the corresponding simulation algorithms. Based on these dual representations, [78, 47, 43, 44, 11, 23] either extend the idea to more general settings (continuous time/with additional constraints) or develop more efficient simulation algorithms. In fact, the duality approach extends well beyond option pricing/optimal stopping. Examples include the information relaxation techniques (see [68, 66] among others) that generalize the duality-based ideas to deal with general sequential decision-making problems. However, almost all of

these methods still rely on a good approximation of the value function (or its dual counterpart, the optimal martingale), which is usually derived from either the aforementioned ADP methods or pure guess, lacking strong performance guarantee.

Besides, we are aware of other works taking different approaches, such as ideas from robust optimization to sidestep the challenging DP (see e.g. [190, 4, 86]). These approaches achieve various levels of tractability via fundamentally different modelling paradigms, and are generally incomparable to the results presented in this work.

3.3 Model

We consider a decision maker (DM) in a discrete-time system driven by some exogenous stochastic processes. The DM faces an sequential decision-making task over a fixed time horizon. In particular, at each time period, the DM observes the system evolution, either stays passive: doing nothing, or stays active: choosing an action from a certain set, and receives a reward which may depend on both her actions and the underlying stochastic process. The number of active periods can not exceed certain limits. The DM needs to find a policy that dynamically yet optimally allocates the action-taking opportunities and selects the best actions, so as to maximize the expected total reward.

Next we introduce notation and mathematically describe our model.

3.3.1 Formal model description

Let the discrete time, d-dimensional stochastic process $\mathbf{X} \stackrel{\Delta}{=} (X_t)_{t\geq 0}$ be the system's underlying state. To ease notation, we assume in this Chapter that X_t starts in a fixed initial state $x_0 \in \mathcal{R}^d$ at time 0 (which is slightly different from the general model introduced in Chapter 1). Process $\mathbf{X}$ evolves according to some probability law in a fixed time span $[1, T]$. Let $\mathcal{F}_t \stackrel{\Delta}{=} \sigma(X_1, ..., X_t), t \in [0, T]$ denote the associated σ-algebra. We do not impose any structural assumption on how X_t evolves, incorporating e.g. correlations (among the d elements) or non-Markovian dynamics.

At each time period $t \in [0, T]$, based on the available information, i.e. the state trajectory $(X_0, ..., X_t)$, the DM needs to decide between (1) staying passive, and (2) choosing an action from a set. We unify notation by letting the DM pick an action $a_t \in \{p^*\} \cup \mathcal{A}_t$, where p^* is the "passive action" and $\mathcal{A}_t$ is the (active) action set. Once the action is chosen, a per-period reward will be generated. We allow the reward to depend on both the sequence of the actions taken and the trajectory of the state, i.e. $r_t(a_0, a_1, ..., a_t, X_0, ..., X_t)$ [1] where r_t is a deterministic function[2]. To ease notation, we always (when causing no ambiguity) suppress the dependence on X_t and write $r_t(a_0, a_1, ..., a_t)$ instead. The reader should be aware that $r_t(a_0, a_1, ..., a_t)$ is a $\mathcal{F}_t-$measurable random variable.

Denote by $\pi = \{\pi_0, ..., \pi_T\}$ a (possibly randomized) dynamic policy, where each π_t is a mapping from the set of all possible trajectories of $(X_0, ..., X_t)$ to the set of all possible probability distributions over extended action set $\{p^*\} \cup \mathcal{A}_t$. The DM's task is to find a policy to maximize the expected total reward (with the expectation taken implicitly over $(X_t)_{1\leq t\leq T}$ and the possible randomness of

[1] For notational convenience, we restrict $a_0 \in \mathcal{A}_0$ and assume $r_0(a_0, x_0) = 0$ for all a_0.

[2] This is without loss of generality since any extra randomness can be taken care of by replacing the original reward by its (conditional) expectation.

the policy π):

$$\text{OPT} \triangleq \sup_{\pi} E\Big[\sum_{t=1}^{T} r_t(\pi_0, ..., \pi_t)\Big].$$

We omit the discussion on the existence of the optimal (or ϵ-optimal) policy. Our model can be reformulated as an Markov Decision Process (MDP) by considering an enlarged state space, and the results regarding the existence of optimal policies for MDPs can be found in classical textbooks, e.g. [219].

3.3.2 Examples

Our model incorporates various real-world sequential decision-making applications as specific instances. We discuss two concrete examples, dynamic pricing and swing option pricing.

- **Dynamic pricing**. Consider a paint manufacturer who sets prices for a single product, one type of paint, over a fixed period of time to maximize the profit. Paints typically require multiple raw materials as input, whose prices are highly volatile and path-dependent. In this case, the high-dimensional state X_t captures the prices of all raw materials, and other demand-relevant economic indicators. The active action set $\mathcal{A}_t$ consists of prices the manufacturer may set for the product. The passive action p^* translates to *no price change*. At each time t, the manufacturer either picks a new price from $\mathcal{A}_t$, or chooses p^* and sticks with the time$-(t-1)-$price. Orders are taken when the business is profitable, i.e. when her price choice p_t is greater than the unit cost c_t. The one-period demand $\mathcal{D}_t$ is then generated and fulfilled, and the manufacturer collects the one-period profit $(p_t - c_t)^+\mathcal{D}_t$. In paint manufacturing, due to the existing complex inven-

tory management strategies and possibly long lead times on the supply side, current unit costs often rely on past prices of raw materials, i.e. $c_t \triangleq c_t(X_0, ...X_t)$. Meanwhile, paints' one-period demands can depend on, in addition to current prices, certain indicators in multiple past periods, such as the number/sizes/duration of construction projects in the area launched/announced in the past few months, i.e. $\mathcal{D}_t \triangleq \mathcal{D}_t(p_t, X_0, ...X_t)$. Therefore, the paint manufacturer's one-period profit is actually a function of the trajectory of the commodity prices and the economic indicators, thus is incorporated within the general reward form $r_t(a_0, ..., a_t, X_0, ..., X_t)$ in our model.

- **Swing option pricing**. Consider the evaluation of an electricity swing option. The swing option consists of an underlying forward contract and a limited number of swing rights. The forward contract specifies an obligation to purchase electricity at a "baseload" daily rate of m MWh, and at a predetermined price p over a fixed time horizon. A swing right, whenever exercised, relaxes the restriction on the purchasing amount to an interval $[m^-, m^+]$. The price of the swing option is determined by ruling out arbitrage opportunities, which consists of buying the option and then exercising the swing rights at the best times with the best purchasing amount. To compute the option price, one needs to find the optimal option-exercising strategy. In this case, the state X_t characterizes the market price of electricity. X_t needs not be the price itself, rather it can represent the high-dimensional underlying process that drives a multi-factor electricity price model, $p_t(X_t)$. The active action set $\mathcal{A}_t$ consists of all possible purchasing amount allowed by the swing right, i.e. $[m^-, m^+]$. The passive action p^* simply refers to not exercising at the current time. The reward $r_t(a_t)$ cap-

tures the extra payoff relative to the forward contract: if $a_t = p^*, r_t(a_t) = 0$, otherwise $r_t(a_t) = (p - p_t(X_t))(a_t - m)$.

3.3.3 Assumptions

Motivated by the above real-world examples and also in consideration of algorithmic analysis, we impose several assumptions on our model.

Assumption 1 (Limited moves)**.** *The number of the (active) action-taking periods cannot exceed a constant $K > 0$. Equivalently, $\sum_{t=1}^{T} 1_{\{\pi_t \neq p^*\}} \leq K$.*

The limited-move constraint holds in a number of applications. In dynamic pricing, it translates to a common business rule, namely *limited price change*. Historically, avoiding menu cost was the main consideration behind limited price change. Nowadays, as more and more businesses move towards digitization, the major reason for imposing limited price change becomes to protect companies' brands and market share, and to prevent customers from turning too strategic. Indeed, in such industries as chemicals manufacturing, companies typically fear volumes loss from frequent price increases. Also, supermarket retailers limit the number of price changes in hope of *"preserve the image of their store and not to train customers to be deal seekers"*([95]). Besides, dynamic pricing with limited price changes are also common in some specific scenarios, such as end-of-season clearance selling. In option pricing, the limited-move constraint restricts the number of option rights. Indeed, the valuation of the most widely used American and Bermudan options can be cast as sequential decision-making subject to a one-move constraint.

Assumption 2 (The generative model: system simulator)**.** *We assume the access to a system simulator, denoted by $\mathcal{S}$, which takes as input any $t \in [0, T]$, any action sequence $a_0, ..., a_t$ and any trajectory $(x_0, .., x_t)$, and outputs in C units of computational time:*

1. *(when $t < T$) an independent sample of X_{t+1}, conditional on $X_i = x_i, i \in [0, t]$*
2. *the reward $r_t(a_0, ..., a_t, x_0, ..., x_t)$*

Our second assumption specifies how we access the underlying model, namely, through simulation. In typical real-world applications, the ability to simulate the system's state and reward translates to the ability to predict future commodity prices, to predict demand level, and/or to simulate costs, which is exactly what various commercial software can provide. Indeed, the growing availability of all kinds of data and the companies' increasing eager to digitize their businesses largely boost the use of such commercial software in various decision-making scenarios.

One key advantage of using such simulator is that it frees us from imposing any restrictive modelling assumptions. For example, our simulation-based algorithm allows the existence of and can successfully deal with certain "shocks" e.g. jumps in commodity prices and sudden surges in demand, which are realistic yet often imposing severe challenges on model-based approaches to sequential decision-making.

From the perspective of algorithm design, encapsulating the underlying model and allowing access only via such a system simulator makes our algorithm essentially a meta-algorithm which is easily customized for a variety of sequential decision-making tasks (simply by plugging in the corresponding system simulators).

We assume that calling the simulator once, either for sampling the underlying state or for evaluating the reward requires C units of computational time. Here we (implicitly) require that C depends on dimension d and time horizon T only polynomially (if not linearly). The assumption captures the majority of the practical scenarios where sampling and reward evaluation are not too costly. However there do exist applications with difficult-to-simulate distributions and/or reward functions. To combine our framework with such settings is a potentially interesting direction for future research and is beyond the scope of the current work.

Finally we present some technical conditions. These conditions are not critical for our algorithm to work, rather they serve for the purpose of more clearly illustrating the main algorithmic analysis results.

We force the action set to be finite. In dynamic pricing, this assumption translates to price being chosen from a discrete set, which is very common. For example, many retailers prefer having price ending in 99 cents. Also, *discrete price ladder* is one of the well-established business rules in the context of promotion planning. In general, by discretizing any continuous action set, we can always achieve a discrete action set. Similar performance and computational efficiency results as presented in this work will hold under minimal extra continuity constraints.

In most real-world applications, rewards are naturally non-negative. For example, manufacturers only take orders when profitable (the current price exceeds the unit cost). Even in cases where negative reward is in theory possible, non-negativity can sometimes be assumed without changing the problem's structure/the optimal policy. For example, in swing option pricing, a profit-maximizing agent will never exercise a right to achieve negative payoff. Hence

it is without loss of generality to remove all "bad" actions from the action set, ensuring non-negative rewards.

We impose mild regularity constraints on the variability of the total reward. More specifically, we require the total reward (under an arbitrary policy and on an arbitrary state trajectory) to (*i*) be bounded away from infinity, and (*ii*) deviate not too far from the optimal value, controlled by a constant multiplier α. In the concrete context of dynamic pricing over a selling season, this constraint asks the maximum expected profit to be proportional to the largest possible profit. Intuitively, the constraint holds if (1) the profit does not exceed certain limit even in the best times (i.e. on the best state trajectories with booming demands and low raw material prices) and (2) the profit under the optimal pricing policy behaves reasonably (e.g. at least 10% of the profit in the best times) in normal times (i.e. on most of the state trajectories). While (2) is often satisfied in reality, (1) also makes sense, largely because practical limits such as manufacturers' limited workforce productivity sets a natural limit on the total profit. In theory, to ensure the constraint holds, one can impose a maximum cap on the total reward and sets the constant α accordingly. Such operation will barely affect the problem's structure provided that the cap is large enough, thus is tolerable since we are ultimately interested in approximations rather than exact optimal solutions. These intuitive arguments can be made rigorous with minimal additional assumptions on the probability distribution of the reward, and we omit the discussion here.

Formally, the technical conditions required in our algorithmic analysis are summarized as follows:

Assumption 3. *1. (Finite action set). $\mathcal{A}_t$ is a static, discrete set, with size $|\mathcal{A}_t| = M$.*

2. *(Non-negativity).* $r_t(a_0, ..., a_t) \geq 0$ *a.s. for any* $t \in [1, T]$ *and any* $(a_s)_{0 \leq s \leq t}$.

3. *(Boundedness).* $\sup_{a_0,...,a_T,x_0,...,x_T} \sum_{t=1}^{T} r_t(a_0, ..., a_t, x_0, ..., x_t) < \infty$. *Furthermore, there exists a constant* $\alpha \in (0, 1)$ *such that for any actions* $(a_t)_{0 \leq t \leq T}$, $\text{OPT} \geq \alpha \sum_{t=1}^{T} r_t(a_0, ..., a_t)$ *a.s.*

3.3.4 Additional notation

Let $[t]$ denote the sequence $(0, 1, ..., t)$. Accordingly, $z_{[t]} \stackrel{\Delta}{=} (z_0, ..., z_t)$. Similarly we let $[s, t]$ and $z_{[s,t]}$ denote $(s, s + 1, ..., t)$ and $(z_s, ..., z_t)$, respectively. By convention, $[-1]$ denotes the empty set $\emptyset$. Let $N(\epsilon, \delta) \stackrel{\Delta}{=} \lceil \frac{1}{2}\epsilon^{-2} \log(\frac{2}{\delta}) \rceil$ be a constant that is frequently used in our analysis. Also define recursively ${}^{k}U(y, z) \stackrel{\Delta}{=} y^{{}^{k-1}U(y,z)}$ for $k \geq 1$, with ${}^{0}U(y, z) \stackrel{\Delta}{=} z$ for any $y, z > 0$. These constants will appear in our main algorithmic analysis results.

3.4 Theory

We first formulate the sequential decision-making problem as a set of recursive equations.

3.4.1 Optimality equations

Let $\mathcal{J}_t^k(a_{[t-1]}, x_{[t]})$ denote the *value-to-go function* at time t, conditional on the state trajectory $x_0, ..., x_t$ and the action sequence $a_0, ..., a_{t-1}$, and with a remaining k opportunities to take active actions. To ease notation, we suppress the depen-

dence on the underlying state X_t, writing $J_t^k(a_{[t-1]})$ instead. We now establish a set of associated recursive equations that connect the functions $(\mathcal{J}_t^k)_{t\geq 0,k\geq 1}$. For all[3] $0 \leq t \leq T$ and any action sequence $a_0, ..., a_{t-1}$,

$$\begin{aligned}\mathcal{J}_t^k(a_{[t-1]}) &= \max_{a\in\mathcal{A}_t} \sup_{t+1\leq\tau\leq T} E\Big[\sum_{s=t}^{\tau-1} r_s(a_{[s]}) + \mathcal{J}_\tau^{k-1}(a_{[\tau-1]})\Big|\mathcal{F}_t\Big] \quad k \geq 2, \qquad (3.1)\\ &\qquad s.t. \;\; a_t = a \;\text{ and }\; a_s = p^* \text{ for } s \in [t+1, \tau-1].\\ \mathcal{J}_t^1(a_{[t-1]}) &= \max_{a\in\mathcal{A}_t} E\Big[\sum_{s=t}^{T} r_s(a_{[s]})\Big|\mathcal{F}_t\Big], \qquad (3.2)\\ &\qquad s.t. \;\; a_t = a \;\text{ and }\; a_s = p^* \text{ for } s \in [t+1, T].\end{aligned}$$

Here in equation (3.1), τ is a stopping time adapted to the filtration $(\mathcal{F}_t)_{1\leq t\leq T}$. In words, the value-to-go at time t with k active-action-taking opportunities is achieved by taking the best possible active action at time t, staying passive afterwards until the best stopping time τ, and making the best decisions subject to $k-1$ active-action-taking opportunities starting from time τ. Formally we have

Proposition 1. *The class of functions $(\mathcal{J}_t^k)_{t\geq 0,k\geq 1}$ statisfying the equations (3.1) and (3.2) are the value-to-go functions of the sequential decision-making problem. Specifically, $\mathcal{J}_0^{K+1} = \text{OPT}$.*

A rigorous proof of Proposition 1 is given in technical appendix B.1. We briefly discuss its implications. The set of equations (3.1) are similar to the standard dynamic programming(DP) Bellman equations in their recursive natures. However, as a key distinction, the Bellman recursion is with respect to *time*, while the set of equations (3.1) are recursive in the *remaining number of active action-taking opportunities.* On the one hand, such formulation can drastically reduce the recursion depth required for achieving optimality, especially in the limited-move setting with $K \ll T$ (a depth-K recursion effectively suf-

[3]For notational convenience, we assume $\mathcal{J}_t^k = 0$ for $t \geq T+1$

fices whereas DP requires depth-T). On the other hand, each recursion of equation (3.1) contains an optimal stopping problem, which is itself computationally challenging in high dimension. Efficiently solving the sequential decision-making problem via equations (3.1) and (3.2) would be in theory possible if there exists a computationally efficient approach to each corresponding optimal stopping problem.

Earlier in Chapter 2 we propose such an efficient approach to general optimal stopping problems that enjoys provably near-optimal performance guarantee with runtime scaling only polynomially in time horizon T, effectively independent of dimension d. The approach is a key building block in the main algorithms of this work. We next briefly review the main theoretical results of Chapter 2, and state a slightly more general algorithmic result required by the current setting.

3.4.2 A new approach to optimal stopping

Notation and model setup.

In view of equation (3.1), we recall Chapter 2 to set up a general formulation of optimal stopping with respect to the underlying state X_t. Let $Z_t \triangleq g_t(X_{[t]})$ be our target stochastic process, where $g_t : \mathcal{R}^d \to \mathcal{R}^+$ is an arbitrary deterministic and non-negative cost function. Let $\mathcal{T}$ be the set of stopping times with respect to $(\mathcal{F}_t)_{1\le t\le T}$. An optimal stopping problem is formally defined as $Z^* \triangleq \inf_{\tau\in\mathcal{T}} E[Z_\tau]$. Then the main results of Chapter 2, are given by the following.

For $t \in [1, T]$, let $Z_t^1 \triangleq Z_t$. For $k \ge 1$ and $t \in [1, T]$, recursively define $Z_t^{k+1} \triangleq Z_t^k - E[\min_{i\in[1,T]} Z_i^k|\mathcal{F}_t]$. Let $L_k \triangleq E[\min_{t\in[1,T]} Z_t^k]$.

Lemma 6 (Theorem 2.3.3 of Chapter 2)**.** $Z^* = \sum_{k=1}^{\infty} L_k$.

Suppose further that w.p.1. $Z_t \in [0, U]$ for some constant $U \geq 0$ and $t \in [1, T]$. Then

Lemma 7 (Theorem 2.3.5 of Chapter 2)**.** $0 \leq Z^* - \sum_{k=1}^{n} L_s \leq \frac{U}{n+1}$, *for all* $n \geq 1$.

In light of Lemma 6 and Lemma 7, in Chapter 2 we develop algorithms for approximating the optimal value Z^* and finding the (near-)optimal stopping policy. The approach utilizes Monte-Carlo methods to compute $L_1, L_2, ..., L_k$ for some k large enough, each with high precision. To account for the runtime and sample complexity of the algorithms, we first establish a formal sampling and computational model, analogous to the generative model of Chapter 2, but slightly weaker.

Assumption 4. *There exists a simulator that takes as input* $\epsilon, \delta \in (0, 1)$, $t \in [T]$ *and any trajectory* $(x_1, ...x_t)$, *and achieves the follows.*

- *(Sampling). It can (when* $t < T$*) output i.i.d. samples of* X_{t+1} *conditional on* $X_{[t]} = x_{[t]}$ *each taking at most C units of time.*
- *(Cost evaluation). It can output a random number Y satisfying* $P(|Y - g_t(X_{[t]})| > \epsilon U) < \delta$ *in a runtime at most* $h_1(\epsilon, \delta)$, *and requiring the number of independent samples at most* $h_2(\epsilon, \delta)$, *where* h_1 *and* h_2 *are some deterministic non-negative functions decreasing in both* ϵ *and* δ.

The simulator model allows for *inexact* evaluation of the costs, effectively extending the setting of Theorem 2.3.7 in Chapter 2 to incorporate a broader class of scenarios including that of equation (3.1).

We next present the analysis of two algorithms $\mathcal{W}_a$ and $\mathcal{W}_b$, as variants of the algorithms of Theorem 2.3.7 under Assumption 4, that can (with high probability) 1. compute an ϵ-approximation for the optimal value Z^*, and 2. provide an $\epsilon-$optimal stopping policy. The runtime and sampling complexity of the two algorithms both scale polynomially in the time horizon T, and depending on the dimension d only implicitly through C, h_1 and h_2. More concretely,

Lemma 8 ($\mathcal{W}_a$: value approximation)**.** *There exists an algorithm, denoted by $\mathcal{W}_a$, that takes as input $\epsilon, \delta \in (0,1)$, and achieves the following. It returns a random number Y satisfying $P(|Y - Z^*| > \epsilon U) < \delta$ in total computational time at most*

$$\Big(C \times H_2(\epsilon,\delta,T) + H_1(\epsilon,\delta,T)\Big) \times \exp(200\epsilon^{-2}) T^{6\epsilon^{-1}} (1 + \log(\frac{1}{\delta}))^{6\epsilon^{-1}}$$

and requiring the number of independent samples from the simulator at most

$$H_2(\epsilon,\delta,T) \times \exp(200\epsilon^{-2}) T^{6\epsilon^{-1}} (1 + \log(\frac{1}{\delta}))^{6\epsilon^{-1}}$$

with the constants $H_i(\epsilon,\delta,T) \stackrel{\Delta}{=} h_i(4^{-6\epsilon^{-1}}, \delta \exp(-200\epsilon^{-2}) T^{-6\epsilon^{-1}} (1 + \log(\frac{1}{\delta}))^{-3\epsilon^{-1}})$ $\quad i \in \{1,2\}$.

Lemma 9 ($\mathcal{W}_b$: good policy)**.** *There exists an algorithm, denoted by $\mathcal{W}_b$, that for all $\epsilon \in (0,1)$, outputs a randomized stopping time τ_ϵ s.t. $E[Z_{\tau_\epsilon}] - Z^* \leq \epsilon U$, and with the following properties. At each time step, the decision of whether to stop (if one has not yet stopped) can be implemented in total computational time at most*

$$\exp(200\epsilon^{-2}) T^{6\epsilon^{-1}} \times \Big(C \times h_2(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2}) T^{-6\epsilon^{-1}}) + h_1(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2}) T^{-6\epsilon^{-1}})\Big)$$

and with the number of samples required from the simulator at most

$$\exp(200\epsilon^{-2}) T^{6\epsilon^{-1}} \times h_2(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2}) T^{-6\epsilon^{-1}}).$$

The pseudo-code of algorithms $\mathcal{W}_a$ and $\mathcal{W}_b$ will be given in the technical appendix B.1.1. Here we sketch the main steps: $\mathcal{W}_a$ truncates the expansion $Z^* =$

$\sum_{i=1}^{\infty} L_i$ at k and approximates $L_1, ..., L_k$ separately with Monte-Carlo simulation via the simulator. $\mathcal{W}_b$ repeatedly calls $\mathcal{W}_a$, compares the obtained value-to-go approximation with the cost-at-stop, and determines whether to stop. The proof of Lemma 8 and 9 combines the result of Lemma 7 with some classical tools in probability theory(e.g. concentration inequalities such as Azuma-Hoeffding). We also leave them to the technical appendix B.1.1.

3.5 Algorithm

We now present our main sequential decision-making algorithms. Following the ideas proposed in section 3.4, our algorithms use the optimal stopping solvers $\mathcal{W}_a$ and $\mathcal{W}_b$ to recursively solve equations (3.1) and (3.2). We begin by describing a set of subroutines for approximating value-to-go functions $\mathcal{J}_t^k$.

3.5.1 Subroutines for computing $\mathcal{J}_t^k$

The subroutines, denoted by $(\mathcal{Q}^k)_{k\geq 1}$, take as input two control parameters $\epsilon, \delta \in (0, 1)$, historical action sequence $a_0, a_1, ..., a_{t-1}$ and state trajectory $x_0, x_1, ..., x_t$, and output high probability ($w.p.\ 1-\delta$) near optimal ($\epsilon-$optimal) approximations of the value-to-go $\mathcal{J}_t^k(a_{[t-1]}, x_{[t]})$. For the ease of analysis, we assume that these subroutines know parameters α and M (defined in Assumption 3), which is not required when used in practice.

Subroutine $\mathcal{Q}^1$ (for computing $\mathcal{J}_t^1$)

Inputs: control parameters ϵ, δ, history $a_{[t-1]}, x_{[t]}$

for $a \in \mathcal{A}_t$ **do**

fix a policy $\pi^a \stackrel{\Delta}{=} (a_{[t-1]}, a, p^*, ..., p^*)$,

sample and compute $N(\alpha\epsilon, \frac{\delta}{M})$ *i.i.d.* copies of $\sum_{s=t}^{T} r_s(\pi^a_{[s]}, X_{[s]})$ using $\mathcal{S}$ and store their average as $\mathbf{Y}^a$

end for

$a^* \leftarrow \arg\max_{a \in \mathcal{A}_t} \mathbf{Y}^a$

Outputs: a^* and $\mathbf{Y}^{a^*}$

Subroutine $Q^{\mathbf{k}}$ (for computing $\mathcal{J}_t^k$)

Inputs: control parameters ϵ, δ, history $a_{[t-1]}, x_{[t]}$

for $a \in \mathcal{A}_t$ **do**

fix a policy $\pi^a \stackrel{\Delta}{=} (a_{[t-1]}, a, p^*, ..., p^*)$,

Call $\mathcal{W}_a$ with parameters $\alpha\epsilon$ and δ/M to solve

$$\sup_{\tau \in [t,T]} E\left[\sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}, X_{[s]}) + \mathcal{J}_\tau^{k-1}(\pi^a_{[\tau-1]}, X_{[\tau]}) \middle| X_{[t]} = x_{[t]} \right]$$

with $(\mathcal{J}_j^{k-1})_{j \geq 1}$ approximated by Q^{k-1}

store the output as $\mathbf{Y^a}$

end for

$a^* \leftarrow \arg\max_{a \in \mathcal{A}_t} \mathbf{Y}^a$

Outputs: a^* and $\mathbf{Y}^{a^*}$

Remark. *Q^k makes calls to $\mathcal{W}_a$ to solve the optimal stopping problem in equation (3.1). $\mathcal{W}_a$'s access to the underlying model is through simulation. More specifically, $\mathcal{W}_a$ needs a simulator that can sample from the underlying state and evaluate the "reward-at-stop", as specified in Assumption 4. The sampling of the underlying state X_t can be done by calling simulator $\mathcal{S}$ by Assumption 2. Evaluating the "reward-at-stop",*

however, requires the knowledge of $(r_j)_{j\geq 1}$ *and* $\mathcal{J}_j^{k-1}$*. The former can again be obtained by calling simulator* $\mathcal{S}$ *(see Assumption 2). The latter is only accessible through running* Q^{k-1}*. In that way, subroutines* $(Q^k)_{k\geq 1}$ *are linked up in a recursive manner. We will provide a formal performance and runtime analysis of subroutines* Q^k *in later sections.*

3.5.2 Main algorithms

Our first main algorithm, denoted by Q_a, computes an approximation of OPT with desired performance guarantee. It is achieved by calling Q^{K+1} with appropriate inputs.

Sketch of Algorithm $Q_{\mathbf{a}}$ (for computing OPT)

Call Q^{K+1} with control parameters (ϵ, δ)

Output: The value-to-go approximation returned by Q^{K+1}

Our second algorithm, denoted by Q_b, can compute a good decision-making policy subject to at most K active action-taking opportunities. The horizon T is divided into $K+1$ epochs. The algorithm computes and takes an active action at the starting point of an epoch, and remains passive while solving an optimal stopping problem to determine when to end the current epoch and start a new one.

Sketch of Algorithm $Q_{\mathbf{b}}$ (for computing a decision-making policy)

---

for $k = 0 : K$ **do**

at the beginning of the k-th epoch, $t_k \leftarrow$ the current time, $x_{[t_k-1]} \leftarrow$ the state trajectory, $a_{[t_k-1]} \leftarrow$ the historical action sequence

call $\mathcal{Q}^{K+1-k}$ with control parameters $(\frac{\epsilon}{8(K+1)}, \frac{M\alpha\epsilon}{8(K+1)} \wedge 1)$ to get an action a_k

fix a policy $\pi^k \stackrel{\Delta}{=} (a_{[t_k-1]}, a_k, p^*, ..., p^*)$

call $\mathcal{W}_b$ with control parameter $\frac{\alpha\epsilon}{2(K+1)}$ to solve

$$\sup_{\tau \in [t_k+1, T]} E[\sum_{s=t_k}^{\tau-1} r_s(\pi^k_{[s]}, X_{[s]}) + \mathcal{J}_\tau^{K+1-k}(\pi^k_{[\tau-1]}, X_{[\tau]}) | X_{[t_k]} = x_{[t_k]}]$$

with $(\mathcal{J}_j^{K+1-k})_{j \geq 0}$ approximated by $\mathcal{Q}^{K+1-k}$

end the epoch when $\mathcal{W}_b$ outputs **STOP**

if the current period == T OR $k == K$

Output π^k **break**

end if

end for

3.5.3 Main results: performance and complexity

We present our main results: the performance and complexity of the algorithms $\mathcal{Q}_a$ and $\mathcal{Q}_b$. Our first theorem establishes that algorithm $\mathcal{Q}_a$ efficiently computes a provably near-optimal approximation of the optimal value.

Theorem 3.5.4. *Under Assumptions 1, 2 and 3, and for any $\epsilon, \delta \in (0, 1)$, algorithm $\mathcal{Q}_a$ can output a random number Y satisfying $P(|Y - \mathrm{OPT}| > \epsilon \times \mathrm{OPT}) \leq \delta$. Furthermore, the above can be achieved in total computational time at most*

$$C \times (10TM^2)^{KU(10^6, \frac{1}{\alpha\epsilon})} \left(1 + \log(\frac{1}{\delta})\right)^{KU(10^6, \frac{1}{\alpha\epsilon})}$$

and with number of samples required from the simulator $\mathcal{S}$ at most

$$(10TM^2)^{kU(10^6, \frac{1}{\alpha\epsilon})} \left(1 + \log(\frac{1}{\delta})\right)^{kU(10^6, \frac{1}{\alpha\epsilon})}$$

Our second theorem establishes that the algorithm efficiently computes a decision-making policy subject to the limited-move constraint (Assumption 1), with strong theoretical performance guarantee.

Theorem 3.5.5. *Under Assumptions 1, 2 and 3, and for any $\epsilon \in (0,1)$, algorithm $\mathcal{Q}_b$ can compute a randomized decision-making policy π subject to at most K active action-taking opportunities, such that $|E[\sum_{t=1}^{T} r_t(\pi_{[t]})] - \mathrm{OPT}| \leq \epsilon \times \mathrm{OPT}$. Furthermore, at each time, π outputs its decision in total computational time at most $2C(100TM)^{k+1}U\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)$ and with number of samples required from simulator $\mathcal{S}$ at most $2(100TM)^{k+1}U\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)$*

Remark. *At this point, our algorithmic analyses serve more as a proof-of-concept that there does exist a PTAS (analogy) for such general sequential decision-making problems. The bounds we present here are no way near being tight. We leave improving these bounds to future works. Different from DP-based approaches, the runtime of our algorithm scales polynomially in the time horizon T, the size of the action set M, and depends on the dimension d only implicitly through simulation costs. However, the dependence on K, the number of active action-taking periods can be worse than being exponentially. These observations suggest that our algorithm is most effective for problems with* (1) *relatively long time horizon,* (2) *high-dimensional and complex underlying state, and* (3) *costly active actions limited to a small number of action-taking opportunities. Decision-making problems with these features arise from many real-world tasks other than dynamic pricing and option pricing, ranging from personal/public healthcare settings (e.g. when to perform organ transplantation; when to shut down/reopen during a pandemic etc.) to public policy making (when/how to change interest rates etc.), where algorithms like ours are of great potential.*

3.6 Analysis

In this section we outline the proof of Theorem 3.5.4 and Theorem 3.5.5. In subsection 3.6.1, we present the analysis of the subroutines $(Q^k)_{k\geq 1}$, from which Theorem 3.5.4 follows directly. We then apply the results to the analysis of Q_b, proving the performance guarantee of its output policy in subsection 3.6.2 and the computational/sampling complexity bounds in subsection 3.6.3. We complete the proof of our main results in subsection 3.6.4.

3.6.1 Analysis of subroutines $(Q^k)_{k\geq 1}$

Algorithm Q^k approximates the value functions $(\mathcal{J}_t^k)_{t\in[0,T]}$ of a given online decision-making problem via Monte Carlo simulation. We now provide the formal analysis, accounting for the computational/sampling cost that Q^k takes to achieve a desired approximation precision.

Lemma 10. *For any $\epsilon, \delta \in (0,1)$, action sequence $a_{[t-1]}$ and state trajectory $x_{[t]}$, algorithm Q^{k+1} achieves the following. In total computational time at most*

$$C \times (10TM^2)^{kU(10^6, \frac{1}{\alpha\epsilon})}\left(1 + \log(\frac{1}{\delta})\right)^{kU(10^6, \frac{1}{\alpha\epsilon})}$$

and number of samples required from simulator $\mathcal{S}$ at most

$$(10TM^2)^{kU(10^6, \frac{1}{\alpha\epsilon})}\left(1 + \log(\frac{1}{\delta})\right)^{kU(10^6, \frac{1}{\alpha\epsilon})},$$

return a random number Y satisfying $P(|Y - \mathcal{J}_t^{k+1}(a_{[t-1]}, x_{[t]})| > \epsilon \times \mathrm{OPT}) < \delta$.

Q^{k+1} is recursively defined, calling Q^k multiple times, with the optimal stopping subroutine $\mathcal{W}_a$ being the key building block. In order to certify that the

output of Q^{k+1} does achieve the desired approximation precision and to account for the computational and sampling costs, we shall use induction, certain concentration bounds and Lemma 8, the algorithm analysis of $\mathcal{W}_a$. The full proof of Lemma 10 is given in Appendix B.2.

3.6.2 Performance guarantee of Q_b

We show in this subsection that the policy returned by Q_b does have a good performance.

Lemma 11. *For any given time period t, state trajectory $x_{[t]}$, action sequence $a_{[t-1]}$ and the number of remaining active action-taking opportunities k, algorithm Q_b can compute a randomized decision-making policy π with $\pi_{[t-1]} = a_{[t-1]}$, and taking at most k active actions in $[t+1, T]$, such that*

$$\left| E\Big[\sum_{s=t}^{T} r_s(\pi_{[s]}) \Big| \mathcal{F}_t \Big] - \mathcal{J}_t^{k+1}(a_{[t-1]}, x_{[t]}) \right| \leq \frac{k+1}{K+1} \epsilon \times \mathrm{OPT}.$$

We use induction, combined with the analysis of $\mathcal{W}_b$ and Q^k (Lemma 9 and 10) to prove the above result. The proof is via fairly standard analytical technique, with the use of several union bounds and concentration bounds. We again leave it to the appendix B.2.

3.6.3 Computational and sampling complexity analysis of Q_b

We establish the computational and sampling analysis of Q_b in this subsection. We show that, at each time, Q_b takes a runtime scaling polynomially in both the time horizon T and the size of the action set M to compute an output action.

Lemma 12. *For any given time period t, state trajectory $x_{[t]}$, action sequence $a_{[t-1]}$ and the number of remaining active action-taking opportunities K, algorithm $\mathcal{Q}_b$ outputs an action in computational time at most $2C(100TM)^{K+1}U\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)$, and with number of independent samples required from simulator $\mathcal{S}$ at most $2(100TM)^{K+1}U\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)$.*

Recall that at each time, $\mathcal{Q}_b$ makes one call to $\mathcal{W}_b$ to determine whether to take active action, and (possibly) one call to $\mathcal{Q}^K$ to pick a new action. Therefore the computational and sampling costs of $\mathcal{Q}_b$ can be bounded by that of $\mathcal{W}_b$ and $\mathcal{Q}^K$, which are the contents of Lemma 9 and 10. We leave the full proof to appendix B.2.

3.6.4 Proof of the main results

With Lemma 10,11 and 12, we complete the proof of our main results, Theorem 3.5.4 and 3.5.5.

Proof of Theorem 3.5.4. Recall that $\mathcal{Q}_a$ simply makes one call to $\mathcal{Q}^{K+1}$ with control parameters (ϵ, δ). Therefore, the result follows immediately from Lemma 10. □

Proof of Theorem 3.5.5. The fact that the output policy is a good policy, i.e.

$$\left| E\left[\sum_{t=1}^{T} r_t(\pi_{[t]}) \right] - \mathrm{OPT} \right| \leq \epsilon \times \mathrm{OPT}$$

is an immediate corollary of Lemma 11. The computational and sampling costs of algorithm $\mathcal{Q}_b$ follow directly from Lemma 12. Combining the above then completes our proof. □

3.7 Conclusion

In this work, we present a new methodology for high-dimensional online decision-making problems subject to the limited-move constraint. In contrast to most past approaches in the literature, our algorithms can trade-off between the desired approximation precision and the runtime/sample complexity. Indeed, for any given control parameter ϵ, our algorithm can obtain an ϵ-approximation of the optimal value, as well as an ϵ-optimal decision-making policy, taking a runtime scaling polynomially in the time horizon T, the size of the action set M, and depending on the dimension (and state space more generally) only through the cost of simulating underlying state/reward. The "data-driven" simulation nature of the algorithms also free us from imposing restrictive assumptions on the underlying model. Indeed, the algorithms work in total generality, allowing for arbitrary high-dimensional and full path-dependent underlying state dynamics and reward structures.

CHAPTER 4

ONLINE MAX WEIGHT BIPARTITE INDEPENDENT SET WITH HIGH-DIMENSIONAL UNSTRUCTURED FEATURES: EFFICIENT SIMULATION IMPLIES EFFICIENT ALGORITHMS

4.1 Introduction

The growing availability of data and the ever-improving machine learning capabilities offer the unique opportunity to incorporate "data-driven" insights into online decision-making. Indeed, powerful models capture relevant features from the (often) highly complicated underlying environment, cast them as data streams, track their history and predict their future movements. Such models facilitate more realistic and more accurate decision-making.

On the other hand, many important decision-making problems possess natural and fundamental *combinatorial optimization* formulations. Typically, combinatorial constraints such as packing, matching etc. characterize various dependence structures between decisions that are common in reality. Indeed, the connection between (offline) decision-making and combinatorial optimization has long been established and well studied in the academic literature.

It is thus well-motivated to formulate and study sequential decision-making problems within a framework where (1) there exists auxiliary, decision-relevant information (often in the form of randomly evolving feature vectors/data streams) and an associated predictive model; and (2) the decisions are subject to certain combinatorial constraints. Such problems are not classical combinatorial optimization problems due to its sequential nature and the existence of the predictive models. Yet they also can not be categorized as typical re-

inforcement learning problems, because the decisions are themselves "path-dependent," subject to certain combinatorial constraints.

Recently, a line of relevant works emerged in the literature (e.g. [248, 258]). These works aim generally at designing efficient algorithms to incorporate the predictive power of machine learning into online decision-making, and derive interesting results in many different settings. However, the existing works typically either adopt the classical *competitive ratio* analysis (i.e. taking a fundamentally non-probabilistic/non-Bayesian perspective), or rely crucially on certain independence assumptions of the underlying features/data streams that reduce the problem complexity (see literature review). In the regime where the underlying features/data streams are high-dimensional and non i.i.d./path-dependent and where the dependence structure between decisions are non-trivial, the problem is typically intractable due to curse of dimensionality issues. In such general settings, the following questions remain open.

Question 1. *Does there exist efficient algorithms that can accurately approximate the performance of the optimal decision-making policies (as a benchmark against which other policies/algorithms will be measured) ?*

Question 2. *Does there exist efficient algorithms that can find provably near-optimal decision-making policies?*

In this work, we make an attempt to address the above questions. Our focus is on the classic *maximum weight independent set* problem in combinatorial optimization. In its online version, a fixed graph is given and known a priori. Each node of the graph is associated with some randomly evolving features (possibly high-dimensional and with path-dependent dynamics), revealed sequentially. One has to decide immediately whether to pick a node upon the revelation of

its features, and the PICK decision will incur a feature-dependent reward. One has to ensure the picked nodes form an independent set, while aiming at maximizing the expected cumulative rewards.

As our main contribution, we prove that this problem is efficiently computable in the *bipartite* and *bounded-degree* setting. More precisely, we devise an algorithm for approximating the performance of the optimal decision-making policy (i.e. the optimal value) that is essentially analogous to a *polynomial time approximation scheme* (PTAS). Namely, for any fixed error tolerance ϵ, δ, our approach guarantees to find an ϵ-approximation of the optimal value with probability at least $1 - \delta$ in a runtime *linear* in the time horizon (equivalently, the number of nodes of the graph), and effectively *independent* of the underlying dimension (implicitly depending on the dimension only through certain simulation costs, which will be specified later). The guarantee holds for arbitrary dynamics of the underlying features and reward structure, providing an affirmative answer to QUESTION 1 in the setting of biparitite bounded-degree online maximum weight independent set.

Our algorithm is (Monte-Carlo) simulation-based. Indeed, it harnesses the full potential of the predictive model, treating it as a "black-box" simulator that repeatedly generate independent trajectories (predictions) of the features, without requiring explicit knowledge of e.g. the dynamic equation or the probability distribution. Technically, our algorithm is intimately related to certain parallel/distributed algorithms (see literature review). However, the existing distributed algorithms generally come from a more combinatorial setting, and are not applicable to our problem which possess a probabilistic structure (see literature review). The key theoretical novelty of this work resides in the combined use of the simulator and a greedy-type technique first introduced in [134]

for tackling high-dimensional optimal stopping, which clears the computational obstacle of approximating a *maximal* objective in a large graph with probabilistic structure. We suspect this general simulation-based technique can be extended beyond the specific setting of online maximum weight independent set/optimal stopping, to the online versions of more combinatorial optimization problems.

Besides, our algorithm exhibits an interesting correlation decay property. More precisely, to achieve a fixed approximation precision, the computation at each node in the graph only ever uses the information of its neighbouring nodes within a constant distance. We precisely characterize this constant, which is determined completely by the max degree of the graph and the fixed error tolerance, and is independent of the total number of nodes in the graph. Therefore, our approach is able to find near-optimal solutions with high probability in a *decentralized* way. This aligns with many existing works, e.g. [125], which report the *correlation decay* property of the *(offline)* maximum weight independent set problem with random weights.

4.1.1 Organization

The rest of this chapter is organized as follows. Following a literature review, in Section 4.3, we define our problem, briefly introduce the required preliminaries (on network flow theory) and (informally) summarize the main result. We sketch our algorithmic idea in Section 4.4. The main algorithms and their analysis is presented in Section 4.5.

4.2 Related literature

Our work is related to several areas of research.

Online algorithms and competitive analysis: The majority of the work on online algorithms is within the competitive ratio analysis framework. Competitive analysis takes a pessimistic perspective on the underlying uncertainty, preparing for the worst to happen, and rewarding algorithms with minimal loss in such circumstances. This framework is successfully applied in the analysis of various problems/algorithms, such as online paging/caching ([236, 119]); online bipartite matching (e.g. [162, 200, 161, 117, 168], only to name a few); the secretary problem/prophet inequality (e.g. [128, 145, 1, 102, 101, 21, 171] only to name a few) and many others. We refer the interested readers to the book [59] for more details. A paper of relevance to our own is [132], which studies online maximum weight independent set in various stochastic models and analyzes the algorithms within the competitive analysis framework. Certain new competitive models are proposed and analyzed in various concrete settings, with an effort to incorporate stochastic information into the design of online algorithms (see, e.g. [127, 202, 203, 167]). We want to stress that, in our work, instead of introducing the optimal offline solutions as benchmarks, we measure all algorithms against the optimal online algorithm. Here *optimal* is in the sense of maximizing the expected performance under the predictive model. Therefore, our results are not directly comparable to results derived within the competitive analysis framework.

Online algorithms and regret analysis: For many concrete problems, the analysis of online algorithms takes a *regret*-based approach. Examples include online learning (see e.g. the book [143]), bandit problems (see e.g. the book [70])

etc. Traditionally, the regret analysis framework is only applied to online problems with a learning/repeated-game structure, often without any combinatorial constraint on decisions.[1] There does exist works on certain online decision-making problems subject to some specific combinatorial constraints on the decisions, that adopt the regret analysis framework (such as [15, 248, 16, 72]). However, these results (regret bounds) rely vitally on the independence assumption of the underlying stochastic process. As a comparison, we allow the underlying stochastic process to have arbitrary, path-dependent dynamics. Indeed, to the best of our knowledge, it seems that the existing literature contains very few if any small-regret or no-regret results in such a framework.

Combinatorial optimization and local/parallel computing: There is an intimate connection between our simulation approximation technique and the algorithmic ideas developed by many works in the local/parallel computing literature. We apply a reduction of an online bipartite maximum weight independent set to a massive offline bipartite maximum weight independent set (and further to a massive max flow problem), and follow the algorithmic idea in [94] of "eliminating length l augmenting paths" to solve the resulting max flow problem. A similar idea in the *unweighted* case first appeared in the celebrated 1973 paper by Hopcroft and Karp [146] on bipartite min vertex cover/ max matching. There is since a line of work that builds on the Hopcroft-Karp framework for *locally* approximately computing (unweighted) max matching/ min vertex cover (see [208, 194, 197, 114]). There is a major distinction between our setting and theirs. Indeed, although the original graph has bounded degree, in our offline max-flow reduction, the (expanded, as we call it) graph may contain nodes with arbitrarily large degree. Furthermore, this graph assigns probabilities and weights (both can be arbitrarily unbalanced) to its massive number of

[1] The *combinatorial bandit* problem actually don't impose combinatorial structure on decisions.

edges. Our simulation model access such a graph naturally. The algorithms in the previously mentioned papers, on the other hand, typically rely on combinatorial methods to "eliminate the augmenting paths". Such techniques are fundamentally inapplicable/inefficient in our setting. [116] talks briefly about possible extensions of these techniques to the weighted case, but no rigorous algorithm and complexity analysis is provided. We believe that our model and techniques are by nature different from this line of works. It's indeed an interesting question to explore further the connection between our algorithm and the local computing algorithms within multiple local computational models, which we leave as an open problem. We refer the readers to the survey [188] or [226] for more details if interested.

Besides the literature on local computing algorithms, our core technique, the greedy iterative flow pushing procedure (Algorithm 2) as a direct extension of [134], is also generally related to certain parallel/distributed ideas in different settings (e.g. constant approximations for non-bipartite min vertex cover) such as [169, 177, 178], and may find its origin in [212]. In general, we are aware of the existence of various relevant but distinct results on related problems of (weighted) maximum matching/ min vertex cover in multiple different computational models (e.g. MPC, PRAM, distriubted LOCAL). These results are not directly comparable to ours.

Correlation decay: For certain combinatorial optimization problems (e.g. maximum (weighted) matching, maximum (weighted) independent set etc.), correlation decay was established and used in the design of efficient approximation algorithms in trees and sparse, locally tree-like graphs (see [8, 10, 9, 126, 251, 124]). Specifically, [125] provides an efficient approximation algorithm for maximum (weighted) independent set with random weights in a graph of small

degree. Our results indicate the existence of correlation decay, and hence efficient local algorithms when the problem is online (in the bipartite and bounded-degree graph).

Optimal stopping and control: The idea of using a simulator to conduct an iterative greedy flow-pushing procedure was first used in [134] on the problem of optimal stopping (see Chapter 2). Indeed, from a combinatorial optimization perspective, the problem of optimal stopping can be viewed as a max-flow min-cut instance in a massive tree network. Unlike in a tree network where the greedy algorithm directly outputs a max flow, we need to add an extra layer (Dinitz algorithm) in the computation of max flow in the current paper since a blocking flow is not necessarily maximum in general. Nontheless, the technique of [134] leads to an efficient procedure for approximately computing blocking flows, which is the core building block of the main algorithm of this work. The method introduced in [134] is in spirit related to many other techniques in stochastic control such as information relaxation (e.g. [68, 67] etc.). The connection between our algorithm and these other relevant works is also an interesting question to explore further, which we leave as an open problem.

Quantum-inspired algorithms and the simulator: In our framework, the concept of a simulator of the underlying data streams naturally comes from the machine-learning predictive tools widely used in reality. In quantum computing, there is a recently emerging line of work on *quantum-inspired classical algorithms* (e.g. [241, 89, 18, 129] among many others) that uses a "simulator" in a similar way as we do in our setting to derive efficient algorithms. More specifically, such quantum-inspired algorithms assume access to the problem instance (often in a matrix form) through certain ability to draw samples from the matrix-specified distribution, resembling the simulator in our setting. They typically

enjoy a significant speedup against previous classical algorithms. These results (and ours) suggest that simulation-based algorithms can be a promising new direction for efficiently solving hard problems in many different settings.

4.3 Model, preliminaries and main result

4.3.1 Problem setup

Suppose there is an undirected, bounded-degree bipartite graph $G = (V, E)$. We assume $V = \{1, 2, ..., T\}$ for notational simplicity. Consider a decision maker (DM) and a stochastic process $\mathbf{x} \stackrel{\Delta}{=} (\mathbf{x}_i)_{i\geq 1}$ with $\mathbf{x}_i \in \mathcal{R}^d$ for some $d \geq 1$. The *online maximum weight independent set* problem (ONLINE MWIS) is defined as follows. Suppose the DM maintains a vertex set $\mathcal{I}$ that is initially empty. At each time $i \in [1, T]$, the DM observes the stochastic process's current value $\mathbf{x}_i = x_i$, and then decides whether to include vertex i in set $\mathcal{I}$. A decision of including i results in a reward of $c_i(x_1, ..., x_i)$, where $(x_j)_{i\in[1,i]}$ is the historical trajectory of the process, and $(c_i)_{i\geq 1}$ are arbitrary real-valued functions that take value in $[0, 1]$. As the sole constraint imposed on this vertex-taking process, set I is required to be an *independent set* of graph G throughout time $[1, T]$. Namely, for any pair of nodes (i, j) with $i < j$ and $(i, j) \in E$, if i was included in set I, then j becomes unavailable. The DM's goal is to find the optimal policy under which the expected total reward is maximized. We denote by OPT the maximum expected total reward.

4.3.2 Max-flow formulation

It turns out that there exists a reduction of ONLINE MWIS to MAX FLOW, which we present in this section. To begin with, let's introduce some notation.

Notation. Let $\mathcal{L} \subset V$ and $\mathcal{R} \triangleq V/\mathcal{L}$ denote the two parts of graph G, respectively. Let Δ denote the maximum degree in G. We assume that the stochastic process only takes value in a finite set.[2] More precisely, for $i \in [1, T]$, let $\mathcal{H}_i$ denote the finite set of all possible trajectories of $(\mathbf{x}_j)_{j\in[1,i]}$. $\mathcal{H}_i$ consists of $d \times i$ matrices. Let $\mathcal{F}_i$ denote the generated σ-field. For each $\omega \in \mathcal{H}_i$, we denote by $p(\omega) \triangleq \mathbb{P}(\mathbf{x}_{[1,i]} = \omega)$ its probability. By basic probability laws, we must have $p(\omega) = \sum_{\gamma\in\mathcal{H}_T:\omega\subset\gamma} p(\gamma)$, where we slightly abuse the notation $\omega \subset \gamma$ to mean that ω is consistent with the first i columns of γ, or equivalently, ω is part of the history of γ. We shall omit the subscript in the reward functions, writing $c(\omega)$ instead of $c_i(\omega)$ for all $\omega \in \mathcal{H}_i$ and $i \in [1, T]$, assuming causing no confusion.

We now state the MAX FLOW reduction of ONLINE MWIS.

Expanded graph. Let's define a directed graph $G^{\mathbf{x}}$ as follows. There exits a source node s and a sink node t. For each $\omega \in \mathcal{H}_i, i \in [1, T]$, there exists an associated node, denoted slightly abusively by ω. For each $\omega_1 \in \mathcal{H}_i$ with $i \in \mathcal{L}$ and $\omega_2 \in \mathcal{H}_j$ with $j \in \mathcal{R}$, there exists an arc in $G^{\mathbf{x}}$ pointing from source node s to ω_1 and an arc pointing from ω_2 to sink node t. There exists an arc between the two nodes ω_1 and ω_2, pointing from ω_1, if and only if (1). $(i, j) \in E$; and (2). $\omega_1 \subset \omega_2$ (when $i < j$) or vice versa (when $i > j$). We denote by $V^{\mathbf{x}}$ the set of nodes, i.e. $V^{\mathbf{x}} = \cup_{i\in[1,T]}\mathcal{H}_i \bigcup \{s, t\}$, and $E^{\mathbf{x}}$ the set of arcs. We call $G^{\mathbf{x}}$ the *expanded graph* associated with graph G and stochastic process $\mathbf{x}$.

Proposition 2. *The longest s-t path in $G^{\mathbf{x}}$ is with length* 3.

[2]We impose this assumption so that we can utilize certain combinatorial tools without running into problems. We suspect the removal of this assumption won't affect our final results, and we leave the discussion of the infinite case as a future direction.

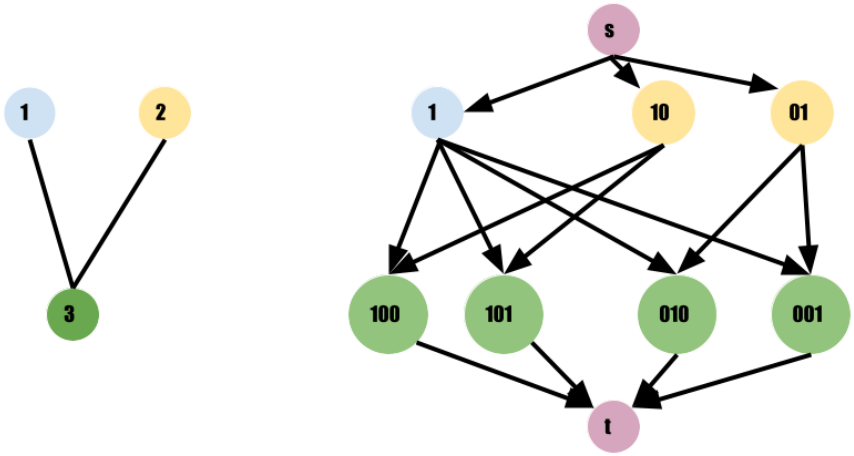

Figure 4.1: An example of the expanded graph: original graph is bipartite $G(\{1,2,3\},\{(1,3),(2,3)\})$ with underlying uncertainty given by a binary tree.

Expanded flow network. We introduce a set of capacities on $G^{\mathbf{x}}$. Arc e of the form (s,ω) or (ω,t) have capacity $c^{\mathbf{x}}(e) \stackrel{\Delta}{=} c(\omega)\times p(\omega)$. All other arcs have capacity ∞. We call the resulting flow network the *expanded flow network* associated with graph G and stochastic process $\mathbf{x}$, denoted by $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$, by convention. We denote by OPT′ the max flow value in network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$.

Lemma 13 (MAX FLOW reduction). MAX FLOW *instance* $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ *is a valid reduction of the* ONLINE MWIS *associated with* $G, \mathbf{x}$ *and* c. *We have*

$$\text{OPT} + \text{OPT}' = E\Big[\sum_{i=1}^{T} c(\mathbf{x}_{[1,i]})\Big].$$

We leave the proof of Proposition 2 and Lemma 13 to Technical appendix

C.1. Here any direct attempt at utilizing textbook max-flow algorithms to solve $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ will lead to complete computational failure, since the problem size is by definition dependent on the state space of the stochastic process $\mathbf{x}$, which can be uncontrollably large. Instead, we combine the classical results in network flow theory with novel techniques developed in the context of optimal stopping and control (see Chapter 2) to come up with implementable algorithms with desirable performance and complexity guarantee.

4.3.3 Network flow preliminaries

In this section, we review some classical network flow results. Before delving into a formal discussion of existing results, let's first introduce several necessary terminologies – both standard ones commonly used in the literature/textbooks and unconventional ones tailored to suit our needs in this work.

Terminologies. For any flow network with noncyclic arc set A and capacities u, we shall add reverse arcs and form an arc set A_r: if $(i, j) \in A$ then both $(i, j), (j, i) \in A_r$, with additional capacities $u(j, i) = 0$. We call the new graph with arc set A_r *residual graph* (unconventional definition, independent of any specific flow). The *residual network* associated with a flow f is defined on the residual graph with the set of capacities $u_f : u_f(i, j) = u(i, j) - f(i, j), u_f(j, i) = f(i, j)$ for any arc $(i, j) \in A$ (and thus $(j, i) \in A_r$). An arc is called *saturated* by f if $u(i, j) - f(i, j) = 0$. By our definition, all 0 capacity, or saturated arcs are kept in the residual network. Flows in residual networks can use a pair of mutually reverse arcs simultaneously. For any such flow f, its *net flow* is defined to only push flow $f(i, j) - f(j, i)$ on arc (i, j) (when $f(i, j) > f(j, i)$). A flow f is *regular* if its net flow is itself. We call a flow *blocking* in some network if every s-t path

contains a saturated arc. We call a path *disconnected* if one of the arcs on the path has 0 capacity. The *level* of a node i, denoted by $d(i)$, is the length of the shortest connected s-i path. An arc (i, j) is *admissible* if it's connected and $d(j) = d(i) + 1$. The *level subnetwork* is defined on the residual graph, characterized by the following set of capacities: for all $(i, j) \in A$ such that (i, j) is admissible, the capacity is $u(i, j)$; for all other arcs the capacity is 0. The *level residual subnetwork* associated with a flow f is the level subnetwork of the residual network of f. We call a flow *l-disconnecting* if $d(t) > l$ in the residual network of f.

We start by introducing the celebrated *Dinitz's algorithm.* This algorithm iteratively computes blocking flows in the level subnetwork of residual networks, keeps increasing the distance between the source and the sink, and terminates at optimality when the sink becomes unreachable from the source.

Dinitz's algorithm. *The algorithm maintains a feasible flow f in network $\mathcal{N}$, initialized as a zero flow, and repeats the following until $d(t) = \infty$ in the residual network.*

1. *Find a blocking flow f' in the level residual subnetwork of f.*
2. *Update $f \leftarrow f + f'$.*

In a Lemma, we state the fact that Dinitz's algorithm strictly increases $d(t)$ between iterations.

Lemma 14. *Suppose a flow f' is blocking in the level residual subnetwork of some flow f. Then the level of sink $d(t)$ is increased by at least one in the residual network $\mathcal{N}_{f+f_0}$ associated with flow* NET $(f + f_0)$.

Corollary 3. *Dinitz algorithm terminates at optimality in at most n iterations, where n is the number of nodes in the network.*

Next we introduce a result of [94], that provides an optimality guarantee for $l-$disconnecting flows in a shallow network.

Lemma 15. *Consider a network of depth r and with a maximum s-t flow value M^*. Suppose a flow f is $l-$disconnecting for some $l \geq r$, then it holds that*

$$M^* \leq \left(\frac{l+1+r}{l+1-r}\right) M.$$

We refer the reader to a proof of Lemma 14 and Corollary 3 in textbooks such as [253]. We point the reader to the paper [94] for a proof of Lemma 15.

4.3.4 Main results

We informally state the main result of this work. We present an efficient algorithm for computing the value of MAX FLOW instance $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$, which then gives an approximation of OPT, the optimal value of ONLINE MWIS instance $\{G, \mathbf{x}, c\}$. To achieve a high-probability $(1-\delta)$ high-accuracy $(1-\epsilon)$ solution, the algorithm only runs in a time *linear* in the number of nodes T, and effectively independent of the dimension d of the underlying process $\mathbf{x}$. We have

Theorem 4.3.5 (Informal)**.** *Suppose* $\text{OPT} = O(T)$. *Then there exists a randomized algorithm that can output a number y satisfying $|y - \text{OPT}| < \epsilon\text{OPT}$ with probability at least $1-\delta$, in a running time TM with M a constant depending only on ϵ, δ, Δ.*

Remark (Simulation-based)**.** *The algorithm in Theorem 4.3.5 takes as a build-in subroutine a simulator of the underlying stochastic process $\mathbf{x}$, and relies on repeated calls to the simulator for independent (conditional) trajectories as the means of accessing $\mathbf{x}$. The sampling cost is implicitly included in constant M. The rigorous definition of such a simulator will be given in Section 4.5.*

Remark (Locality)**.** *The algorithm exhibits an interesting correlation decay property, and is intimately related to the local computation model. At a high level, the algorithm conducts computation at each node in the expanded graph* $G^{\mathbf{x}}$*. To achieve an* (ϵ, δ)*−approximation of* OPT*, the algorithm only ever uses the information from the neighbourhood of each node within a given constant distance. Our analysis reveals that this constant depends only on* ϵ, δ *and* Δ*, independent of the size of the graph* T*. Therefore, the algorithm possesses an interesting property that its computation at each node is fundamentally local.*

4.4 Algorithms: the ideal version

On a very high level, the approach we take to solving our specific MAX FLOW instance $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ is to follow Dinitz's algorithm. Namely we compute blocking flows for a finite number of iterations, and terminate with provable optimality guarantee promised by Lemma 15. A key challenge here is that the blocking flows are intractable via any existing, standard network flow algorithm, which typically sequentially searches for and then augments the shortest s-t paths. We take an alternative approach that, roughly speaking, works on all shortest paths simultaneously. In this section, we sketch the main steps of our approach. Our demonstration of the algorithm here will be "ideal", that ignores any physical limit, allowing the "computer" to take an infinite amount of steps and store an infinite amount of information. Such "ideal algorithms" will be converted into implementable algorithms with rigorous performance and runtime analysis in later sections.

As discussed in the previous section, we slightly modify the network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ by adding reverse arcs to $E^{\mathbf{x}}$, and assigning capacities 0 to these addi-

tional arcs. We still denote the resulting network by $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. We also slightly modify the underlying graph G, by adding two vertices, respectively -1 and -2. Vertex -1 is adjacent to, and only to all vertices in $\mathcal{L}$, and vertex -2 is adjacent to, and only to all vertices in $\mathcal{R}$. They correspond to source s and sink t in $G^{\mathbf{x}}$. We still denote the resulting graph by G. A (directed) path in $G^{\mathbf{x}}$ is called *regular* if source s only appears as the starting node of the path (if it appears) and sink t only appears as the ending node of the path (if it appears). Similarly, a path in G is called regular if vertex -1 only appears as the starting vertex of the path (if it appears) and vertex -2 only appears as the ending vertex of the path (if it appears). To simplify our notation, we let $\mathcal{H}_{-1} \triangleq \{s\}$ and $\mathcal{H}_{-2} \triangleq \{t\}$.

4.4.1 Max flow in path subnetwork and Algorithm 1′

In this section, we define a specific subgraph of $G^{\mathbf{x}}$, the *path subgraph*, and introduce an algorithm which solves for the maximum flow in networks with this specific underlying graph structure. The algorithm will serve as a key subroutine for our main blocking flow computation. Let's begin with a quick observation which connects the (regular) paths in graph $G^{\mathbf{x}}$ with the (regular) paths in the original graph G.

Lemma 16. *There is a natural surjective function ϕ mapping from the regular (directed) paths in graph $G^{\mathbf{x}}$ to the regular paths in graph G (with directions).*

Proof. Consider an arbitrary regular directed path $\mathcal{P} = (\omega_1, ..., \omega_k)$ in $G^{\mathbf{x}}$. By definition of $G^{\mathbf{x}}$, there exists $i_1, ..., i_k \in [1, T] \cup \{-1, -2\}$ such that $\omega_j \in \mathcal{H}_{i_j}$ for all $j \in [1, k]$. The fact that there exists arcs between consecutive nodes in the path, along with the definition of $G^{\mathbf{x}}$, imply that $(i_j, i_{j+1}) \in E$ for all $j \in [1, k-1]$. There-

fore $(i_1, ..., i_k)$ forms a (undirected) path in G. We assume a direction that points from i_1 to i_k, and denote the directed path by P. The fact that $\mathcal{P}$ is regular implies that P is regular. Then the function $\phi(\mathcal{P}) = P$ proves what we need. □

Now suppose $P = (i_1, ..., i_k)(k \geq 2)$ is an arbitrary regular path in graph G, with an assigned direction pointing from vertex i_1 towards vertex i_k.

Definition 4.4.1 (Path sample)**.** *A P-sample is a directed path $\mathcal{P}$ in graph $G^{\mathbf{x}}$ that belongs to the preimage of P under function ϕ, i.e. $\phi(\mathcal{P}) = P$.*

Definition 4.4.2 (Path subgraph)**.** *Given two nodes $\omega_1 \in \mathcal{H}_{i_1}, \omega_k \in \mathcal{H}_{i_k}$, the (P, ω_1, ω_k)-subgraph is a subgraph of $G^{\mathbf{x}}$ that consists of all P-samples starting at node ω_1 and ending at node ω_k.*

Definition 4.4.3 (Path subnetwork)**.** *Given a set of arbitrary capacities u on graph $G^{\mathbf{x}}$ and the (P, ω_1, ω_k)-subgraph W, we call $\{W, u, \omega_1, \omega_k\}$ a $(P, \omega_1, \omega_k, u)$-subnetwork.*

To unify notation, all flows in $(P, \omega_1, \omega_k, u)$-subnetwork are defined on the entire graph $G^{\mathbf{x}}$, more formally, $f : E^{\mathbf{x}} \to \mathcal{R}$, assuming 0 wherever unspecified.[3] We now present our ideal algorithm that solves for maximum flows in path subnetworks.

Algorithm 1′ (max flow in path subnetworks)

Input: (P, ω_1, ω_k)-subgraph with $P = (i_1, ..., i_k)$, capacities u

initialize scalar $M = 0$, flow vector $f = \mathbf{0}$

if $(k == 2)$

return $M = u(\omega_1, \omega_k)$; $f(\omega_1, \omega_k) = u(\omega_1, \omega_k)$, $f = 0$ elsewhere

[3] A flow in some path subnetworks may not be a flow in the entire network as the conservation rule is violated at the sub-source and sub-sink.

else compute $j := \arg\min\{i_2, ..., i_{k-1}\}$ and store subpaths

$P_1 \leftarrow (i_1, ..., i_j), P_2 \leftarrow (i_j, ..., i_k)$

for all $\omega \in \mathcal{H}_j : \omega_1, \omega_k \subset \omega$ **do**

call Algorithm 1′ with inputs $(P_1, \omega_1, \omega, u)$ and $(P_2, \omega, \omega_k, u)$

store outputs as $(M_{1,\omega}, f_{1,\omega})$ and $(M_{2,\omega}, f_{2,\omega})$ respectively

compute $M_\omega \leftarrow \min(M_{1,\omega}, M_{2,\omega})$ and $f_\omega \leftarrow f_{1,\omega}\frac{M_\omega}{M_{1,\omega}} + f_{2,\omega}\frac{M_\omega}{M_{2,\omega}}$

update $M \leftarrow M + M_\omega; \quad f \leftarrow f + f_\omega$

end for

end if

Output: M and f

Proposition 3 (Optimality of Algorithm 1′)**.** *Algorithm 1′ with input* (P, s, t, u) *returns the maximum flow value and the (regular) maximum flow, thereby solving the* MAX FLOW *problem in the* (P, s, t, u)*-subnetwork.*

The performance guarantee of Algorithm 1′ depends on a specific decomposability property of path subnetworks, which enables us to break them into smaller path subnetworks and solve for optimality with recursion. A detailed proof of Proposition 3 will be left to Technical appendix C.2.

4.4.2 Blocking flow and Algorithm 2′

In this section, we present a procedure that leverages Algorithm 1′ to accomplish Dinic's blocking flow computation. More precisely, the procedure maintains a regular, feasible flow throughout. In each iteration, it calls Algorithm 1′ on all *shortest s-t* path subnetworks, pushes a weighted sum of the outputting

max flows, and updates the capacities accordingly. After infinite iterations it returns a flow that saturates all shortest s-t paths, hence blocking in the level subnetwork.

We now state this procedure as *Algorithm* 2′. Recall that Δ is the max degree of vertices $[1, T]$ in graph G.

Algorithm 2′ (blocking flow in level residual subnetwork)

Input: l-disconnecting flow f_0 for l odd

initialize flow $f' = \mathbf{0}$ in residual network $\mathcal{N}_{f_0} = \{G^{\mathbf{x}}, c^{\mathbf{x}}_{f_0}, s, t\}$, value $M' = 0$

store in $\mathcal{M}_{l+2}$ all regular paths: $-1 \to -2$ in G, with length $(l+2)$;

for iteration $i = 1 : \infty$

for $P \in \mathcal{M}_{l+2}$

call Algorithm 1′ with input (P, s, t, u) and store outputs as M_P, f_P

update $f' \leftarrow f' + \frac{1}{\Delta^l} f_P$

end for

update $u \leftarrow c^{\mathbf{x}}_{f_0} - f'$

end for

Output: f'

Recall that we say a flow l-disconnecting, if the length of the shortest connected s-t path in its residual network is greater than l. Algorithm 2′ can augment any l-disconnecting flow such that it becomes $(l+2)$-disconnecting, thereby increasing $d(t)$ by 2. We state this property as Proposition 4 and defer its proof to Technical appendix C.3.

Proposition 4 (Guarantee for Algorithm 2′)**.** *Suppose Algorithm 2′ takes as input*

an l-disconnecting flow f_0 and outputs a flow f'. Then $f_0 + f'$ is $(l+2)$-disconnecting.

4.4.3 Max flow and Algorithm 3′

With Algorithm 1′ and Algorithm 2′ in hand, we now provide Algorithm 3′ that solves the MAX FLOW instance $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. It's validity follows directly from Dinic's algorithm, Proposition 4 and Corollary 3. We omit the proof.

Algorithm 3′ (main algorithm)

Input: flow network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$

initialize flow vector $f = \mathbf{0}$

for odd $l = 1, 3, ...$

call Algorithm 2′ with input f and store the output as f'

update $f \leftarrow \text{NET}(f + f')$

end for

Output: f

4.5 Algorithms: the implementable version, and their analysis

Algorithm 1′, Algorithm 2′ and the main Algorithm 3′ are all unrealistic in terms of runtime. More specifically, in Algorithm 1′, the size of the for loop is on the order of the size of the state space of the stochastic process $\mathbf{x}$, which can be tremendously large. In algorithm 2′, the iteration repeats for an infinite amount of times. In algorithm 3′, although the iteration count is finite before it termi-

nates (by Corollary 3), it is on the order of $O(T)$, which will lead to an undesirable runtime complexity (exponential in T). To remedy these shortcomings, we propose modified versions of the three algorithms, so that they become realistic and implementable. We also present rigorous runtime and performance analysis for these modified algorithms.

To deal with the possibly tremendous state space of the underlying stochastic process $\mathbf{x}$, we introduce a natural *simulation model*, which has the power to generate independent (conditional) trajectories of $\mathbf{x}_t$. Such a model is the *generative model* as introduced in Chapter 1.

simulation model. *Assume access to a simulator $\mathcal{S}$ of the underlying stochastic process $\mathbf{x}$. More precisely, $\mathcal{S}$ can take as input any $i \in [0, T]$ and any historical sequence $x_{[1,i]}$, and output* (1) *independent copies of the trajectory $x_{[1,T]}$, conditional on $\mathbf{x}_j = x_j, j \in [1, i]$; (output unconditional trajectories when $i = 0$) and* (2) *reward $c_i(x_1, ..., x_i)$. Either task can be done at a computational cost of at most C.*

To link network flow problems with the simulation model, we introduce a probabilistic representation for flows and capacities on graph $G^{\mathbf{x}}$, which simply rescales the original flows and capacities with a certain set of probabilities defined for each arc of the graph. The resulting "flows" and "capacities" will have a random-variable-type interpretation.

Probabilistic representation for flows and capacities on $G^{\mathbf{x}}$. Taking an arbitrary arc $(\omega_i, \omega_j) \in G^{\mathbf{x}}$ with $\omega_i \in \mathcal{H}_i$ and $\omega_j \in \mathcal{H}_j$, we assign probability $p(\omega_i, \omega_j) \stackrel{\Delta}{=} \min(p(\omega_i), p(\omega_j)) = p(\omega_{i \vee j})$ to the arc, where we recall that for any $\omega \in \mathcal{H}_k$, $p(\omega) = P(\mathbf{x}_{[1,k]} = \omega)$, and $p(s) = p(t) = 1$ by convention. We call each (ω_i, ω_j) an *arc sample* of edge (i, j) if $\omega_{i \wedge j} \subset \omega_{i \vee j}$. Notice that here we essentially

build a probability space for each $(i, j) \in E$, with state space $\mathcal{H}_{i\vee j}$, σ-fields $\mathcal{F}_{i\vee j}$ and probability assignment $p(\omega_i, \omega_j)$ for each arc sample (ω_i, ω_j). As a result, any real-valued function defined on the vector space of all arc samples of (i, j) can thus be viewed as a random variable on the associated probability space. Now with the observation that flows and capacities on expanded graph $G^{\mathbf{x}}$ can be thought of as functions on all arc samples of all edges, we can essentially view each of them as a collection of random variables defined on the edges in graph G. More precisely, for any flow f in the network and any $(i, j) \in E$, the induced random variable is given by $\mathbf{z}^{i,j}(\omega_i, \omega_j) \stackrel{\Delta}{=} f(\omega_i, \omega_j)/p(\omega_i, \omega_j)$, for any arc sample (ω_i, ω_j). By such definition, $\mathbf{z}^{i,j}$ is an $\mathcal{F}_{i\vee j}$-measurable random variable. Similarly, for any set of capacities u, the induced random variable is given by $\mathbf{w}^{i,j}(\omega_i, \omega_j) \stackrel{\Delta}{=} u(\omega_i, \omega_j)/p(\omega_i, \omega_j)$.

Certain network flow concepts have probabilistic interpretations under this new representation.

Lemma 17. *Suppose* $\mathbf{z} \stackrel{\Delta}{=} \{\mathbf{z}^{i,j},\ (i, j) \in E\}$ *represents a feasible flow associated to capacities* $\mathbf{w} \stackrel{\Delta}{=} \{\mathbf{w}^{i,j},\ (i, j) \in E\}$, *then* $\mathbf{z}^{i,j} \leq \mathbf{w}^{i,j},\ w.p.1.$

Lemma 18. *Suppose* $\mathbf{z}$ *represents a feasible flow with respect to capacities* $\mathbf{w}$, *then the residual capacities are given, for any* $(i, j) \in E$, *by* $\mathbf{w}_z^{i,j} = \mathbf{w}^{i,j} - \mathbf{z}^{i,j}$ $w.p.1$ *if either* $i \in \mathcal{L}, j \in \mathcal{R}$, *or* $i = -1$, *or* $j = -2$; $\mathbf{w}_z^{i,j} = \mathbf{z}^{j,i}$ $w.p.1$ *for all other cases.*

We next state an important property of the above probabilistic representation, that in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$, all feasible flows and their associated (finite) residual capacities become bounded random variables taking value in $[0, 1]$ under the probabilistic representation. We leave the proof to Techinical appendix C.4.

Lemma 19 (boundedness of flow and residual capacity)**.** *Suppose* f *is a feasible flow in* $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. *Let* $\mathbf{z}$ *be the induced collection of random variables. Then*

*w.p.*1, $\mathbf{z}^{i,j} \in [0,1]$. *Furthermore, if we denote by* $\mathbf{w}$ *the collection of random variable representing the* f*-residual capacities. Then for any* $(i,j) \in E$, $\mathbf{w}^{i,j} = \infty$ *if* $i \in \mathcal{L}$ *and* $j \in \mathcal{R}$, *and* $\mathbf{w}^{i,j} \in [0,1]$ *w.p.*1 *otherwise.*

Remark. *Since arc samples are really determined by only one of their end nodes (the one appearing later in time/bearing more information), rigorously speaking any associated random variable should be a function only of that node. For example,* $\mathbf{z}^{i,j}(\omega_i, \omega_j)$ *should be* $\mathbf{z}^{i,j}(\omega_i)$ *if* $\omega_j \subset \omega_i$, *and* $\mathbf{z}^{i,j}(\omega_j)$ *if the other way round. We introduce the notion arc sample and writing random variable realizations as* $\mathbf{z}^{i,j}(\omega_i, \omega_j)$ *to avoid confusion. We will use the two notations interchangeably in later sections. Whenever* $\mathbf{z}^{i,j}(\omega_i, \omega_j)$ *is used, we usually omit the superscript and write* $\mathbf{z}(\omega_i, \omega_j)$ *instead, which simplifies the notation.*

From now on, we will by default use this random variable representation whenever mentioning flows, capacities and residual networks (unless otherwise specified).

4.5.1 Simulation, $\epsilon-$optimal max flow in path subnetworks and Algorithm 1

In this section, we present a modified, more realistic version of Algorithm 1′. In particular, we observe that with the probabilistic representation, the max flow value in path subnetworks is simply a nested expectation. We then make use of simulator $\mathcal{S}$ and classical Monte Carlo approaches to come up with an approximation algorithm.

We first introduce random variables that represent max flows and their values in path subnetworks. Formally, for a given regular path $P = (i_1, ..., i_k) \in G$,

we define random variable $\mathbf{y}_P$ as follows. For each $(P, \omega_1, \omega_k, u)$-subnetwork with maximum flow value M^*, $\mathbf{y}_P(\omega_1 \vee \omega_k)$[4] $\stackrel{\Delta}{=} M^*/p(\omega_1, \omega_k)$. By such definition, $\mathbf{y}_P$ is $\mathcal{F}_{i_1 \vee i_k}$-measurable. Particularly, $\mathbf{y}_P$ is deterministic when P is an $-1 \to -2$ path in G (associated with s-t paths in $G^{\mathbf{x}}$). In fact, in such cases $\mathbf{y}_P = M^*$.

We now restate the key ideas of Algorithm 1′ using the probabilistic flow/capacity representation.

Lemma 20. *Consider a path of a single edge $(i, j) \in G$. It holds true that $\mathbf{y}_{(i,j)} = \mathbf{w}^{i,j}$, w.p.1. with $\mathbf{w}$ the capacities.*

Lemma 21. *Consider an arbitrary path subnetwork with underlying path $P = (i_1, ..., i_k), k \geq 3$. Suppose $i_1, i_k < \min(i_2, ..., i_{k-1})$, and j is such that $i_j = \min(i_2, ..., i_{k-1})$. Suppose $P_1 = (i_1, ..., i_j)$ and $P_2 = (i_j, ..., i_k)$. Then*

$$\mathbf{y}_P = E[\min(\mathbf{y}_{P_1}, \mathbf{y}_{P_2}) | \mathcal{F}_{i_1 \vee i_k}] \quad w.p.1.$$

We defer the proof of Lemma 20 and Lemma 21 to Technical appendix C.2. By recursively applying the two lemmas, we derive a nested conditional expectation representation for the value of the maximum flow in any path network. Our main approach in Algorithm 1, the implementable version of Algorithm 1′, is to approximate this nested expectation representation by Monte Carlo methods with independent samples drawn from simulator $\mathcal{S}$.

Algorithm 1 has two parts. Part a takes as input an arbitrary regular path $P = (i_1, ..., i_k) \in G$ with $i_1, i_k < \min(i_2, ..., i_{k-1})$, and a node $\omega \in \mathcal{H}_{i_1 \vee i_k}$, and outputs an approximation of the random variable realization $\mathbf{y}_P(\omega)$. Part b takes as input the same P, along with an arc sample (ω_q, ω_{q+1}) of some edge $(i_q, i_{q+1}) \in P$, and returns the value on that edge in the max flow. We let $\mathbf{z}_P$ denote the max flow.

[4] we slightly abuse the notation $\vee$ to mean the following here: $\omega_1 \vee \omega_k = \omega_1$ if $\omega_k \subset \omega_1$, otherwise $\omega_1 \vee \omega_k = \omega_k$

Then the output is an approximation of $\mathbf{z}_P(\omega_q, \omega_{q+1})$.

Algorithm 1 accesses the expanded network through a capacity evaluation subroutine $\mathcal{W}$ that outputs the capacity of any input arc sample. Later when introducing other algorithms, Algorithm 1 will be treated as a subroutine to solve max flow problems on path subnetworks with different capacities, hence associated, different $\mathcal{W}s$ are required. We here don't explicitly spell out subroutine $\mathcal{W}$ (it naturally builds on our simulator $\mathcal{S}$ and will be specified in different contexts). Instead, we only want the reader to bear in mind that $\mathcal{W}$ can take as input any arc sample, and output the associated capacities on the arc.

Algorithm 1a (approximate max flow value in path subnetworks)

Input: regular path $P = (i_1, ..., i_k)$; node $\omega \in \mathcal{H}_{i_1 \vee i_k}$; subroutine $\mathcal{W}$

initialize scalar $y = 0$

if $(k == 2)$

call $\mathcal{W}$ on arc sample $(\omega[1, i_1], \omega)$ (if $i_1 < i_2$) or $(\omega, \omega[1, i_2])$ (if $i_2 < i_1$)

return the output

else compute $j : i_j = \min\{i_2, ..., i_{k-1}\}$ and store

$P_1 \leftarrow (i_1, ..., i_j), P_2 \leftarrow (i_j, ..., i_k)$

for $i = 1 : N$ **do**

call simulator $\mathcal{S}$ for an independent $\omega_j^i \in \mathcal{H}_j$ conditional on ω

call Algorithm 1a with inputs (P_1, ω_j^i) and (P_2, ω_j^i)

store outputs as $(y_{P_1}^i)$ and $(y_{P_2}^i)$ respectively

store $y^i \leftarrow \min(y_{P_1}^i, y_{P_2}^i)$

end for

do $y = \text{AVERAGE}(y^1, ..., y^N)$

end if

Output: y

Algorithm 1b (approximate max flow in path subnetworks)

Input: regular path $P = (i_1, ..., i_k)$, arc sample $e = (\omega_q, \omega_{q+1})$ of (i_q, i_{q+1});
subroutine $\mathcal{W}$

initialize scalar $z = 0$

if $(k == 2)$ **call** $\mathcal{W}$ on e with output w and store $z \leftarrow w$

else compute $j : i_j = \min\{i_2, ..., i_{k-1}\}$ and store

$$P_1 \leftarrow (i_1, ..., i_j), P_2 \leftarrow (i_j, ..., i_k)\ ,\ \omega_j \leftarrow (\omega_q \vee \omega_{q+1})[1, i_j]$$

call Algorithm 1a with inputs (P_1, ω_j) and (P_2, ω_j)

store outputs as y_{P_1} and y_{P_2} respectively

call Algorithm 1b with inputs $(P_i, e), i \in \{1, 2\}$ (whichever $i : e \in P_i$)

store outputs as z'

do $z \leftarrow z' \min(y_{P_1}, y_{P_2})/y_{P_i}$

end if

Output: z

Proposition 5 (Guarantee of Algorithm 1)**.** *Assume capacity evaluation subroutine $\mathcal{W}$ satisfies the following: for any edge $(i, j) \in E$, at a computational cost $\nu_1(\epsilon, \delta)$ and with number of calls to simulator $\mathcal{S}$ at most $\nu_2(\epsilon, \delta)$, $\mathcal{W}$ can output a random variable w such that $|w - \mathbf{w}^{i,j}| < \epsilon$ with probability at least $1 - \delta$, where $\mathbf{w}$ is the random variable of the true capacity. For any regular path $P = (i_1, ..., i_k) \in G$ with length $k - 1$, and any $(i_q, i_{q+1}) \in P$*

- *Algorithm* 1*a can achieve the following. With a runtime at most*

$$(1 + \log(\frac{1}{\delta})) \times (\frac{1}{\epsilon})^{3(k-2)} \times 100^{(k-2)^2} \times (C + \nu_1(\epsilon 16^{-(k-2)}, \delta \wedge \epsilon 16^{-(k-2)})),$$

and number of calls to simulator $\mathcal{S}$ at most

$$(1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon})^{3(k-2)}\times 100^{(k-2)^2}\times\nu_2(\epsilon 16^{-(k-2)},\delta\wedge\epsilon 16^{-(k-2)}),$$

it outputs a number y, such that $|y-\mathbf{y}_P|<\epsilon$ with probability at least $1-\delta$.

- *Algorithm 1b can achieve the following. With a runtime at most*

$$(1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon\delta})^{3(k-1)}\times 100^{(k-1)^2}\times\left(C+\nu_1(\frac{\epsilon\delta}{4^{2(k-2)}},\frac{\epsilon\delta}{4^{2(k-2)}})\right)$$

and number of calls to simulator $\mathcal{S}$ at most

$$(1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon\delta})^{3(k-1)}\times 100^{(k-1)^2}\times\nu_2(\frac{\epsilon\delta}{4^{2(k-2)}},\frac{\epsilon\delta}{4^{2(k-2)}}),$$

it outputs a random variable z, such that $|z-\mathbf{z}_P^{i_q,i_{q+1}}|<\epsilon$ with probability at least $1-\delta$.

Remark. *The reason we allow capacity evaluation to be inexact is that in later applications, Algorithm 1 will be taken as a subroutine and called repeatedly. In particular, capacities on arcs are typically computed using the outputs from the algorithm in previous iterations, which introduces inexactness and randomness. The probability statement of Algorithm 1b is with respect to* (1) *the intrinsic randomness introduced by the use of simulator $\mathcal{S}$ and* (2) *the probability space $(\mathcal{H}_{i_q\vee i_{q+1}},\mathcal{F}_{i_q\vee i_{q+1}},\mathbb{P})$.*

The proof of this proposition involves the use of several concentration inequalities and union bounds, along with careful counting of the runtime and the number of samples required. We leave it to Technical appendix C.4.

4.5.2 Truncation, approximate blocking flows and Algorithm 2

In this section, we present a modified, realistic version of Algorithm 2′. The new Algorithm 2 only repeats the iterations for a finite number of times before terminating. We show that this truncated version of Algorithm 2′ already achieves

desired approximation guarantee, provided that the number of iterations is sufficiently large.

Algorithm 2 works in the residual network of some given flow $\mathbf{z}_0$. We assume that we have access to its residual network $\mathcal{N}_{\mathbf{z}_0}$ through the following black-box scheme: given any edge $e \in E$, we can call a capacity evaluation subroutine $\mathcal{W}$ to output a random variable that is a high probability approximation of the residual capacity random variable $\mathbf{w}_{\mathbf{z}_0}$. Later we will see that in all applications of Algorithm 2, subroutine $\mathcal{W}$ can be built from repeatedly calling simulator $\mathcal{S}$ and Algorithm 2 itself, and hence we are not requiring any additional sampling power other than simulator $\mathcal{S}$. Now we formally present Algorithm 2.

Algorithm 2 (approximate blocking flow)

Input: arc sample $e = (\omega_i, \omega_j)$ of edge (i, j);

parameters: iteration count k, depth l

initialize $z \leftarrow 0$

if $(k > 0)$

call Algorithm 2 with inputs $(e, k-1, l)$ and store outputs as z

for P : $-1 \rightarrow -2$ regular path; with length $(l+2)$; $(i, j) \in P$

call Algorithm 1b on (P, e) and **Subroutine** $\mathcal{W}_{k-1}$

store outputs as z_P

end for

do $z \leftarrow z + \Delta^{-l}\,\mathrm{SUM}(z_P)$ (recall Δ is the max degree)

end if

Output: z

Subroutine $\mathcal{W}_k$ (capacity evaluation)

Input: arc sample e;

parameters: error tolerance parameter η; subroutine $\mathcal{W}$

initialize scalar $z_e = 0$

call $\mathcal{W}$ on e and store the output as w_e^0

do $w_e^0 \leftarrow w_e^0 \, 1_{\{w_e^0 > \eta\}}$

if $k > 0$

call Algorithm 2 with inputs (e, k, l) and store outputs as z_e

end if

Output: $w_e^0 - z_e$

Before stating the performance guarantee of Algorithm 2, we first generalize the concept of *l-disconnecting*. We call a flow $\mathbf{z}$ (η, l)-*disconnecting*, if for any regular $-1 \to -2$ path $P \in G$ with length $\leq l$, it holds that $\min_{(i,j)\in P} \mathbf{w}_{\mathbf{z}}^{i,j} \leq \eta, \ w.p.1$, where $\mathbf{w}_{\mathbf{z}}$ is the $\mathbf{z}$-residual capacity. Stated in the common network flow language, every s-t path Q in $G^{\mathbf{x}}$ with length $\leq l$ contains an arc e of capacity less than $\epsilon \times p(e)$ in the associated residual network of flow $\mathbf{z}$,

Proposition 6 (Performance guarantee of Algorithm 2)**.** *Suppose for any given edge* $(i, j) \in E$*, at a computational cost at most* $\mu_1(\epsilon, \delta)$ *and with number of calls to simulator* $\mathcal{S}$ *at most* $\mu_2(\epsilon, \delta)$*, subroutine* $\mathcal{W}$ *can return a random variable* w*, such that* $|w - \mathbf{w}_{\mathbf{z}_0}^{i,j}| \leq \epsilon$ *with probability at least* $1 - \delta$*, where* $\mathbf{w}_{\mathbf{z}_0}$ *is the residual capacity of an* (η, l)*-disconnecting flow* $\mathbf{z}_0$*. Then Algorithm* 2 *can achieve the follows for any edge* $(i, j) \in G$*. At a computational cost at most*

$$\left(\frac{(100\Delta)^l}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^l}} \times \left(C + \mu_1\left((\epsilon\delta(100\Delta)^{-l})^{4^{\eta^{-1}\Delta^l}}, (\epsilon\delta(100\Delta)^{-l})^{4^{\eta^{-1}\Delta^l}}\right)\right)$$

and with number of calls to simulator $\mathcal{S}$ at most

$$\left(\frac{(100\Delta)^l}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^l}} \times \mu_2\left((\epsilon\delta(100\Delta)^{-l})^{4^{\eta^{-1}\Delta^l}}, (\epsilon\delta(100\Delta)^{-l})^{4^{\eta^{-1}\Delta^l}}\right),$$

it can output a random variable z such that $|z - \mathbf{z}^{i,j}| < \epsilon$ with probability at least $1 - \delta$, where $\mathbf{z}$ is a flow satisfying that NET $(\mathbf{z}_0 + \mathbf{z})$ *is $(\eta, l+2)$-disconnecting.*

We leave the proof of this proposition to the Technical Appendix C.4.

4.5.3 L-disconnecting flows and Algorithm 3

The main idea of Algorithm 3 is to repeatedly call Algorithm 2 to find new flows that increase the (η)-level of sink, just as in Dinic's Algorithm. However, different from Dinic's Algorithm which iterates for $O(T)$ phases until s and t are disconnected in the current residual network, Algorithm 3 terminates after L phases for some large, predetermined constant L independent of T. For any given arc, it then outputs an (η, L)-disconnecting flow with high precision and in high probability. We now present Algorithm 3.

Algorithm 3 (approximate L-disconnecting flow)

Input: arc $e = (\omega_1, \omega_2)$;

parameters: truncation level L (odd); error tolerance η

initialize scalar $z = 0$

if $(L > 1)$

call Algorithm 3 on $(e, L-2, \eta)$ and store output as z_0

call Algorithm 2 with $(\eta^{-1}\Delta^{L-2}, L-2)$ and **Subroutine** $\mathcal{W}^{L-2}$

on arcs e and $e' \stackrel{\Delta}{=} (\omega_2, \omega_1)$ and store outputs as z_1 and z_2

do $z \leftarrow z_0 + z_1 - z_2$

end if

Output: z

Subroutine $\mathcal{W}^l$ (capacity evaluation)

Input: arc sample $e = (\omega_1, \omega_2)$

initialize: scalar $z \leftarrow 0$

if (e is a non reverse arc)

call Algorithm 3 on (e, l, η) and store output as z_e

call simulator $\mathcal{S}$ for $c^{\mathbf{x}}(e)$ and store output as w_e

do $z \leftarrow w_e - z_e$

else if (e is a reverse arc)

call Algorithm 3 on $(e' \stackrel{\Delta}{=} (\omega_2, \omega_1), l, \eta)$ and store output in z

end if

Output: z

Proposition 7 (Guarantee of Algorithm 3)**.** *For any edge* $(i, j) \in E$*, at a computational cost at most*

$$C\left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta L}}$$

and number of calls to simulator $\mathcal{S}$ *at most*

$$\left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta L}},$$

Algorithm 3 can output a random variable z *on* (i, j) *satisfying* $|z - \mathbf{z}_L^{i,j}(e)| < \epsilon$ *with probability at least* $1 - \delta$*, where* $\mathbf{z}_L$ *is an* (η, L)*-disconnecting flow in* $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$.

We leave the proof to Technical appendix C.4.4.

4.5.4 Approximate max flow value and Algorithm 4

Building on Algorithm 3 and Monte Carlo simulation, Algorithm 4 returns a high probability optimal value approximation of the MAX FLOW instance $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$, thereby approximately solving ONLINE MWIS.

Algorithm 4 (main approximation algorithm)

Input:

initialize scalar $y = 0$

for $i \in \mathcal{L}$

for $j = 1 : N$

call simulator $\mathcal{S}$ for an arc (s, ω_i^j) with $\omega_i^j \in \mathcal{H}_i$ and reward $c(\omega_i^j)$

call Algorithm 3 on (s, ω_i^j) with L and store output as $z_{i,j}$

end for

do $y_i \leftarrow \text{AVERAGE}_j(z_{i,j})$

end for

do $y \leftarrow \text{SUM}_i(y_i)$

Output: y

Proposition 8 (Guarantee of Algorithm 4)**.** *At a computational cost at most*

$$CT\left(\frac{(100\Delta)^{30\epsilon^{-1}}}{\epsilon\delta}\right)^{4^{300\epsilon^{-2}\Delta^{30\epsilon^{-1}}}}$$

and with number of calls to simulator $\mathcal{S}$ at most

$$T\left(\frac{(100\Delta)^{30\epsilon^{-1}}}{\epsilon\delta}\right)^{4^{300\epsilon^{-2}\Delta^{30\epsilon^{-1}}}},$$

Algorithm 4 *can output a number y satisfying $|y - \text{OPT}'| < T\epsilon$ with probability at least $1 - \delta$.*

We defer the proof of Proposition 8 to Technical appendix C.5.

4.5.5 Main results

With all algorithm defined in the previous sections, we now present our main results.

Theorem 4.5.6 (Formal version of Theorem 4.3.5)**.** *Suppose* OPT $> \alpha T$ *for some* $\alpha \in (0, 1)$, *then there exists an algorithm that can achieve the following. At a computational cost at most*

$$CT\Big(\frac{(100\Delta)^{30(\alpha\epsilon)^{-1}}}{\alpha\epsilon\delta}\Big)^{4^{300(\alpha\epsilon)^{-2}\Delta^{30(\alpha\epsilon)^{-1}}}}$$

and with number of calls made to the simulator $\mathcal{S}$ *at most*

$$T\Big(\frac{(100\Delta)^{30(\alpha\epsilon)^{-1}}}{\alpha\epsilon\delta}\Big)^{4^{300(\alpha\epsilon)^{-2}\Delta^{30(\alpha\epsilon)^{-1}}}}$$

The algorithm outputs a random number y *such that with probability at least* $1 - \delta$,

$$\left|\frac{y - \text{OPT}}{\text{OPT}}\right| < \epsilon$$

The proof of Theorem 4.3.5 and the statement of the algorithms will be deferred to Technical appendix C.6.

Observation. We observe that, our algorithmic analysis exhibits certain correlation decay property of the online stochastic maximum weight independent set problem for bipartite and bounded degree graphs. More precisely, Algorithm 4 finds an ϵ-optimal flow of the MAX FLOW instance $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ associated to the ONLINE MWIS instance $\{G, c, \mathbf{x}\}$. Consider a node $\omega \in \mathcal{L}^{\mathbf{x}}$ such that $\omega \in \mathcal{H}_i$. The computation of the flow on arc (s, ω) consists of calling Algorithm 3 for $O(\frac{1}{\epsilon})$ rounds. Each round of Algorithm 3 directly uses the information of nodes as far

as $O(\frac{1}{\epsilon})$ away from ω. Each round of Algorithm 3 calls Algorithm 2 for $O(\frac{1}{\epsilon})\Delta^{O(\frac{1}{\epsilon})}$ iterations. Each iteration of Algorithm 2 increase the distance of correlation by at most $O(\frac{1}{\epsilon})$. Combining all above, by the end Algorithm 4, the computation at node ω only ever uses the weights of nodes at most a distance of $O(\frac{1}{\epsilon^3})\Delta^{O(\frac{1}{\epsilon})}$ away. Now suppose additionally that the underlying model is *i.i.d.* with some arbitrary distribution. Then we immediately conclude that the (multiplicative) $\epsilon-$optimal flow on arc (s, ω), as a realization of the associated random variable on the edge $(-1, i)$, is *independent* of the majority of the underlying graph, i.e. those parts out of the $O(\frac{1}{\epsilon^3})\Delta^{O(\frac{1}{\epsilon})}-$ neighbourhood of vertex i. In other words, the MAX FLOW reduction of ONLINE MWIS on bipartite bounded degree graph exhibits correlation decay when the underlying stochastic process is some i.i.d. sequence.

Remark. *Theorem 4.3.5 guarantees an efficient algorithm for approximating the optimal value of* ONLINE MWIS*, one may wonder whether the results directly imply an efficient algorithm for the approximate optimal policy. The problem is technically equivalent to converting a near-optimal max flow to a min cut. Due to the specific structure of our flow network, such a conversion is not directly applicable. We leave the efficient or even local computation of near-optimal decision-making policies as an interesting open problem.*

4.6 Conclusion

In this work, we study the online stochastic maximum weight independent set problem in bipartite and bounded-degree graphs. Our model is the first of its kind that both allows for high-dimensional and path-dependent underlying driving process and possesses a nontrivial combinatorial structure. We

propose a randomized algorithm for approximating the expected total weights under the optimal policy. For any given control parameter ϵ, our algorithm can obtain an ϵ-approximation of the optimal value, as well as an ϵ-optimal decision-making policy, taking a runtime scaling linearly in the time horizon/ the number of nodes T, and depending on the dimension (and the underlying process more generally) only through the access to the Monte-Carlo simulator. Furthermore, our algorithm exhibits certain correlation decay/locality property, that the computation at each node in the graph only ever uses the information in a neighbouring area whose size is a constant, independent of the rest of the graph.

APPENDIX A

CHAPTER 2 OF APPENDIX

This appendix contains proofs of some theoretical Lemmas, a detailed algorithmic analysis for the unbounded maximization setting, and the proof of Theorem 2.3.9 as presented in Chapter 2.

A.1 Proof of Lemma 4

A.1.1 Auxiliary Lemmas and proofs

Before proceeding, let's state several facts that will be used here and also repeatedly in later proofs. First, we recall a standard result from probability theory used often to prove concentration for estimators.

Lemma 22 (Hoeffding's inequality)**.** *Suppose that for some $n \geq 1$ and $U > 0$, $\{X_i, i \in [1, n]\}$ are i.i.d., and $P(X_1 \in [0, U]) = 1$. Then $P\left(\left|n^{-1}\sum_{i=1}^{n} X_i - E[X_1]\right| \geq \eta\right) \leq 2\exp\left(-\frac{2\eta^2 n}{U^2}\right)$.*

We omit the proof.

We next recall a "smart trick" — a direct corollary of the Hoeffding's inequality, that is commonly used in speeding up algorithms for estimating a number.

Lemma 23 (Median Trick)**.** *Suppose we have an algorithm A that can estimate a target number in correct range of ϵ with probability $3/4$. Consider a new algorithm A' constructed as follows: Repeat the algorithm A for $m = \lceil 8\log(2\delta^{-1})\rceil$ times and take the*

median of the m answers [1]. *Then A′ is able to estimate the target number in correct range of* ϵ *with probability* $1-\delta$.

Proof of Lemma 23 : Suppose the m outputs are $q_1, ..., q_m$, and the target number is x. Let's define $Z_i \stackrel{\Delta}{=} \mathbf{1}(|q_i - x| < \epsilon)$ for all $i \in [1, m]$. Let's without loss of generality assume $q_1 \leq q_2 \leq ... \leq q_m$. Notice that for any $j \in [1, \lfloor \frac{m+1}{2} \rfloor]$, the interval $[q_j, q_{j+\lfloor \frac{m}{2} \rfloor}]$ must contain the median of the sequence. Also, the event $\sum_{i=1}^{m} Z_i > \frac{m}{2}$ implies that there exists at least $\lceil \frac{m+1}{2} \rceil$ numbers in $\{q_1, .., q_m\}$ that satisfy $|q_i - x| < \epsilon$. Let the smallest among these number be q_u and the largest be q_v. Then $v - u \geq \lfloor \frac{m}{2} \rfloor$. Using the previous observation, we have median$(q_1, ..., q_m) \in [q_u, q_v]$, and therefore $|\text{median}(q_1, ..., q_m) - x| < \epsilon$. Taking the contraposition of the above reasoning, we conclude that $|\text{median}(q_1, ..., q_m) - x| \geq \epsilon$ implies $\sum_{i=1}^{m} Z_i \leq \frac{m}{2}$. Combining the above with an application of the Hoeffding's Lemma 22, we have

$$P(\Big|\text{median}(q_1, ..., q_m) - x\Big| \geq \epsilon) \leq P\Big(\frac{1}{m}\sum_{i=1}^{m} Z_i \leq \frac{1}{2}\Big) \leq 2\exp\Big(-\frac{m}{8}\Big) \leq \delta,$$

thus completing the proof. □

Finally, recall that f_k is defined by

$$f_k(\epsilon, \delta) = \log(2\delta^{-1}) \times 10^{2(k-1)^2} \times \epsilon^{-2(k-1)} \times (T+2)^{k-1} \times (1 + \log(\frac{1}{\epsilon}) + \log(T))^{k-1}.$$

We need the following auxiliary lemma, which demonstrates that f_k satisfies certain recursions corresponding to our algorithm's performance.

Lemma 24. *For all* $\epsilon, \delta \in (0, 1)$ *and* $k \geq 1$.

$$f_{k+1}(\epsilon, \delta) \geq 8\lceil \log(2\delta^{-1}) \rceil \times (N(\frac{\epsilon}{4}, \frac{1}{16}) + 1) \times (T+2) \times f_k(\frac{\epsilon}{4}, \frac{1}{16N(\frac{\epsilon}{4}, \frac{1}{16})T})$$

[1] Here by default we define the median of an ordered sequence $x_1 \leq ... \leq x_n$ as $\frac{1}{2}(x_{\lfloor \frac{n+1}{2} \rfloor} + x_{\lfloor \frac{n}{2} \rfloor + 1})$ to incorporate the case with even n

Proof of Lemma 24: We have that

$$
\begin{aligned}
& 8\lceil \log(2\delta^{-1})\rceil \times (N(\frac{\epsilon}{4}, \frac{1}{16}) + 1) \times (T+2) \times f_k(\frac{\epsilon}{4}, \frac{1}{16N(\frac{\epsilon}{4}, \frac{1}{16})T}) \\
\leq \; & 8\lceil \log(2\delta^{-1})\rceil \times (32\log(32)\epsilon^{-2} + 1) \times (T+2) \times f_k(\frac{\epsilon}{4}, \frac{\epsilon^2}{512\log(32)T}) \\
\leq \; & 8\lceil \log(2\delta^{-1})\rceil \times (32\log(32)\epsilon^{-2} + 1) \times (T+2) \times \log(1024\log(32)T\epsilon^{-2}) \\
& \quad \times 10^{2(k-1)^2} \times 4^{2(k-1)} \times \epsilon^{-2(k-1)} \times (T+2)^{k-1} \times \left(1 + \log(\frac{4}{\epsilon}) + \log(T)\right)^{k-1} \\
\leq \; & \log(2\delta^{-1}) \times (T+2)^k \times \epsilon^{-2k} \times \left(1 + \log(\frac{1}{\epsilon}) + \log(T)\right)^k \times 10^{2(k-1)^2} \times 4^{2(k-1)} \times 10^4 \times 4^{k-1} \\
\leq \; & \log(2\delta^{-1}) \times 10^{2k^2} \times (T+2)^k \times \epsilon^{-2k} \times \left(1 + \log(\frac{1}{\epsilon}) + \log(T)\right)^k,
\end{aligned}
$$

completing the proof. □

A.1.2 Proof of Lemma 4

With Lemmas 22, 23 and 24 in hand, we now prove Lemma 4.

Proof of Lemma 4: We proceed by induction. The base case $k = 1$ is trivial. Now suppose the induction is true for some $k \geq 1$. We first show that $\mathcal{B}^{k+1}$ satisfies the desired high probability error bounds. By Lemma 23, it suffices that in each of the $\lceil 8\log(2\delta^{-1})\rceil$ outer loops, the algorithm $\mathcal{B}^{k+1}$ compute and store a number within the desired range of error with probability at least 3/4. Let $\{X_i, i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]\}$ be an i.i.d. sequence of r.v.s, each distributed as $\min_{i\in[1,T]} Z_i^k(\mathbf{X}(\gamma)_{[i]})$, where the same realization of $\mathbf{X}(\gamma)$ is used for all $i \in [1, T]$. Then it follows from our inductive hypothesis, the Lipschitz property of the min function, a union bound over all $i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]$ and $j \in [1, T]$, and some straightforward algebra, that we may construct $\{X_i, i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]\}$ and $\{A_i^0, i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]\}$ on a common probability space s.t. with probability at least $1 - \frac{1}{16}$, $|X_i - A_i^0| < \frac{1}{16}$ for all

$i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]$. Applying Lemma 22 to $\{X_i, i \in [1, N(\frac{\epsilon}{4}, \frac{1}{16})]\}$, with parameters $\eta = \frac{\epsilon}{4}, U = 1, n = N(\frac{\epsilon}{4}, \frac{1}{16})$, we conclude (after some straightforward algebra) that on the same probability space,

$$P\left(\left|(N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} X_i - E[X_1]\right| < \frac{\epsilon}{4}\right) \geq 1 - \frac{1}{16}.$$

Here we apply Lemma 22 with $U = 1$, since $X_i \in [0, 1]$ for all $i \geq 1$. Noting that the event $\left\{|X_i - A_i^0| < \frac{\epsilon}{4} \text{ for all i}\right\}$ implies the event

$$\left\{\left|(N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} A_i^0 - (N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} X_i\right| < \frac{\epsilon}{4}\right\},$$

we may combine the above with a union bound and the triangle inequality to conclude that on the same probability space (and hence in general),

$$P\left(\left|(N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} A_i^0 - E[X_1]\right| < \frac{\epsilon}{2}\right) \geq 1 - \frac{1}{8}.$$

As the inductive hypothesis ensures that $P(\left|A^3 - Z_t^k(\gamma)\right| > \frac{\epsilon}{2}) \leq \frac{1}{8}$, we may again apply a union bound and the triangle inequality, along with the definition of $Z_t^k(\gamma)$ and X_1, to conclude that

$$P\left(\left|A^3 - (N(\frac{\epsilon}{4}, \frac{1}{16}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{1}{16})} A_i^0 - Z_t^{k+1}(\gamma)\right| > \epsilon\right) \leq \frac{3}{4} \quad \text{as desired.} \tag{A.1}$$

We next prove that $\mathcal{B}^{k+1}$ satisfies the desired computational and sample complexity bounds. In each of the $\lceil 8 \log(2\delta^{-1}) \rceil$ outer loop, the only access to randomness for $\mathcal{B}^{k+1}$ is through its $N(\frac{\epsilon}{4}, \frac{1}{16})$ direct calls to $\mathcal{B}(t, \gamma)$ (whose output is each time stored in $\mathbf{A}^1$), its $N(\frac{\epsilon}{4}, \frac{1}{16})T$ calls to $\mathcal{B}^k(j, A^1_{[j]}, \frac{\epsilon}{4}, \frac{1}{16N(\frac{\epsilon}{4}, \frac{1}{16})T})$, and its one final call to $\mathcal{B}^k(t, \gamma, \frac{\epsilon}{2}, \frac{1}{8})$ (whose output is stored in A^3). It thus follows from the inductive hypothesis, and several easily verified monotonicities of N and f_k, that the

number of calls to the base simulator made by $\mathcal{B}^{k+1}(t,\gamma,\epsilon,\delta)$ is at most

$$\begin{aligned} & 8\times\lceil\log(2\delta^{-1})\rceil\times\left(N(\frac{\epsilon}{4},\frac{1}{16})+N(\frac{\epsilon}{4},\frac{1}{16})\times T\times f_k(\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T})+f_k(\frac{\epsilon}{2},\frac{1}{8})\right) \\ \leq\quad & 8\times\lceil\log(2\delta^{-1})\rceil\times(N(\frac{\epsilon}{4},\frac{1}{16})+1)\times(T+2)\times f_k(\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T}). \end{aligned}$$

We next focus on computational costs. For each $s\in[1,\lceil 8\log(2\delta^{-1})\rceil]$ In each of the $N(\frac{\epsilon}{4},\frac{1}{16})$ iterations (indexed by i), first one direct call is made to $\mathcal{B}(t,\gamma)$ (at computational cost C); then T calls are made to $\mathcal{B}^k(j,A^1_{[j]},\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T})$ (for different values of j), each at computational cost at most $(C+G+1)\times f_k(\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T})$; then the minimum of a length-T array is computed (at computational cost T). One additional call is then made to $\mathcal{B}^k(t,\gamma,\frac{\epsilon}{2},\frac{1}{8})$, at computational cost at most $(C+G+1)\times f_k(\frac{\epsilon}{2},\frac{1}{8})$; and finally the average of $N(\frac{\epsilon}{4},\frac{1}{16})$ is computed and subtracted from A^3, at computational cost $N(\frac{\epsilon}{4},\frac{1}{16})+1$. It thus follows from the inductive hypothesis, and several easily verified monotonicities of N and f_k, that the computational cost of $\mathcal{B}^{k+1}(t,\gamma,\epsilon,\delta)$ is at most

$$\begin{aligned} & 8\times\lceil\log(2\delta^{-1})\rceil\times\Big(N(\frac{\epsilon}{4},\frac{1}{16})C+N(\frac{\epsilon}{4},\frac{1}{16})T(C+G+1)f_k(\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T}) \\ & +\ T+(C+G+1)f_k(\frac{\epsilon}{2},\frac{1}{8})+N(\frac{\epsilon}{4},\frac{1}{16})+1\Big) \\ & \quad\leq\quad 8\times\lceil\log(2\delta^{-1})\rceil\times(C+G+1)\times(N(\frac{\epsilon}{4},\frac{1}{16})+1)\times(T+2)\times f_k(\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T}). \end{aligned}$$

By Lemma 24 and some straightforward algebra, we get the desired computational complexity bound for $\mathcal{B}^{k+1}$. Combining all above, by induction we complete the proof. □

A.2 Unbounded maximization and proof of Theorem 2.3.9

In this section we show how to combine our previous algorithms with an appropriate transformation and careful truncation argument to prove Theorem 2.3.9.

At a high level, we proceed as follows: 1. truncating the payoff $\mathbf{Z}$ at an appropriate large number, 2. normalizing and converting to a minimization problem, and 3. computing an appropriate number of terms in our expansion. It turns out that, as long as the squared coefficient of variation of $\max_{t\in[1,T]} Z_t$ is of a reasonable size, all errors can be appropriately controlled if the truncation and the number of terms are carefully chosen as functions of the error tolerance ϵ and this squared coefficient of variation.

A.2.1 Auxiliary truncation lemma and proof

Before formally describing the relevant algorithm, we prove several results laying the groundwork for our truncation-based approach. For any $U > 0$, define $Z^1_{U,t} \triangleq U^{-1} \times \min(U, Z^1_t)$; $Z^{1,-}_{U,t} \triangleq 1 - Z^1_{U,t}$; and for $k \geq 1, Z^{k+1,-}_{U,t} \triangleq Z^{k,-}_{U,t} - E[\min_{i\in[1,T]} Z^{k,-}_{U,i}|\mathcal{F}_t]$; let $L^-_{U,k} \triangleq E[\min_{i\in[1,T]} Z^{k,-}_{U,i}]$ and $E^-_{U,k} \triangleq \sum_{i=1}^k L^-_{U,i}$. Recall that $M_1 = E[\max_{t\in[1,T]} Z^1_t]$, $M_2 = E[(\max_{t\in[1,T]} Z^1_t)^2]$ and $\gamma_0 = \frac{M_2}{(M_1)^2}$. Then we have

Proposition 9. *For all $U > 0, k \geq 1$ and $z \in \mathcal{R}$*

$$\left|\widehat{\text{OPT}} - U \times (1 - E^-_{U,k} + z)\right| \leq 7 \times \gamma_0^{\frac{3}{2}} \times \left(\frac{U}{M_1} \times ((k+1)^{-1} + |z|) + (\frac{U}{M_1})^{-\frac{1}{2}}\right) \times \widehat{\text{OPT}}.$$

The proof of Proposition 9 requires some auxiliary lemmas. First, we bound the error introduced by our truncation.

Lemma 25. *For all $U > 0$,*

$$0 \leq \widehat{\text{OPT}} - U \times \sup_{\tau\in\mathcal{T}} E[Z^1_{U,\tau}] \leq (M_2)^{\frac{1}{2}} \times (\frac{M_1}{U})^{\frac{1}{2}}.$$

Proof : Non-negativity follows from the fact that w.p.1 $Z^1_t \geq U \times Z^1_{U,t}$ for all $t \in [1, T]$. To prove the other direction, let τ_* denote an optimal stopping time for

the problem $\sup_{\tau \in \mathcal{T}} E[Z^1_\tau]$, where existence follows from [90]. Then by a straightforward coupling and rescaling,

$$\begin{aligned}
E[Z^1_{\tau_*}] - U \times E[Z^1_{U,\tau_*}] &\leq E[Z^1_{\tau_*} I(Z^1_{\tau_*} > U)] \\
&\leq E[Z^1_{\tau_*} I(\max_{t \in [1,T]} Z^1_t > U)] \\
&\leq E[(\max_{t \in [1,T]} Z^1_t) \times I(\max_{t \in [1,T]} Z^1_t > U)] \\
&\leq (M_2)^{\frac{1}{2}} \times \left(P(\max_{t \in [1,T]} Z^1_t > U) \right)^{\frac{1}{2}} \quad \text{by Cauchy-Schwarz} \\
&\leq (M_2)^{\frac{1}{2}} \times (\frac{M_1}{U})^{\frac{1}{2}} \quad \text{by Markov's inequality,}
\end{aligned}$$

completing the proof. □

Next, we prove that if γ_0 is not too large, then $\frac{\widehat{\text{OPT}}}{M_1}$ cannot be too small.

Lemma 26. $\widehat{\text{OPT}} \geq \frac{4}{27} \times \gamma_0^{-1} \times M_1$.

Proof : Recall the celebrated Paley-Zygmund inequality, i.e. the fact that for any $\delta \in (0,1)$ and non-negative r.v. X,

$$P(X > \delta E[X]) \geq (1-\delta)^2 \times \frac{(E[X])^2}{E[X^2]}. \tag{A.2}$$

Now, for $\delta \in (0,1)$, consider the stopping time τ_δ which stops the first time that that $Z^1_t \geq \delta \times M_1$, and stops at time T if no such time exists in $[1,T]$. Then by non-negativity and (A.2),

$$E[Z^1_{\tau_\delta}] \quad \geq \quad \delta \times (1-\delta)^2 \times \frac{(M_1)^3}{M_2}.$$

Optimizing over δ (a straightforward exercise in calculus) then completes the proof. □

By combining Lemmas 25 - 26, we are led to the following corollary.

Corollary 4. *For all $U > 0$,*

$$0 \leq \widehat{\mathrm{OPT}} - U \times \sup_{\tau \in \mathcal{T}} E[Z^1_{U,\tau}] \leq \frac{27}{4} \times (\gamma_0)^{\frac{3}{2}} \times (\frac{M_1}{U})^{\frac{1}{2}} \times \widehat{\mathrm{OPT}}.$$

Proof : It follows from Lemma 26 that

$$\begin{aligned} \frac{(M_2)^{\frac{1}{2}}}{\widehat{\mathrm{OPT}}} &\leq \frac{27}{4} \times \frac{(M_2)^{\frac{3}{2}}}{(M_1)^3} \\ &= \frac{27}{4} \times (\gamma_0)^{\frac{3}{2}}. \end{aligned}$$

Combining with Lemma 25 completes the proof. □

We now complete the proof of Proposition 9.

Proof of Proposition 9 : For all $U > 0$,

$$\sup_{\tau \in \mathcal{T}} E[Z^1_{U,\tau}] = 1 - \inf_{\tau \in \mathcal{T}} E[Z^{1,-}_{U,\tau}]; \tag{A.3}$$

and Theorem 2.3.5 implies that for all $U > 0$ and $k \geq 1$,

$$1 - E^-_{U,k} - \frac{1}{k+1} \leq \sup_{\tau \in \mathcal{T}} E[Z^1_{U,\tau}] \leq 1 - E^-_{U,k}. \tag{A.4}$$

It then follows from Corollary 4 and the triangle inequality that

$$\left| \widehat{\mathrm{OPT}} - U \times (1 - E^-_{U,k} + z) \right| \leq \frac{27}{4} \times (\gamma_0)^{\frac{3}{2}} \times (\frac{M_1}{U})^{\frac{1}{2}} \times \widehat{\mathrm{OPT}} + U \times (|z| + (k+1)^{-1}).$$

Furthermore,

$$\begin{aligned} \frac{U}{\widehat{\mathrm{OPT}}} &= \frac{U}{M_1} \times \frac{M_1}{\widehat{\mathrm{OPT}}} \\ &\leq \frac{U}{M_1} \times \frac{M_1}{\frac{4}{27} \times \frac{(M_1)^3}{M_2}} = \frac{27}{4} \times \gamma_0 \times \frac{U}{M_1}. \end{aligned}$$

Combining the above with the triangle inequality, the fact that $\gamma_0 \geq 1$ by Jensen's inequality, and some straightforward algebra completes the proof. □

A.2.2 Description of algorithms

We now describe several relevant algorithms. In all cases, these will be (very) slight variations of the previously defined algorithms $\mathcal{B}^k(t,\gamma,\epsilon,\delta)$ and $\hat{\mathcal{B}}^k(\epsilon,\delta)$, essentially identical except there is now an extra parameter U input to the algorithm, which causes the algorithms to perform all calculations as if r_t were instead equal to $r'_t = 1 - U^{-1} \times \min(U, r_t)$. First, we define the appropriate "base case" algorithm.

Algorithm $\mathcal{B}^{1,-}(t,\gamma,\epsilon,\delta,U)$:

Return $1 - U^{-1} \times \min\left(U, r_t(\gamma)\right)$

For $k \geq 1$, the definition of $\mathcal{B}^{k+1,-}(t,\gamma,\epsilon,\delta,U)$ is nearly identical to that of $\mathcal{B}^{k+1}(t,\gamma,\epsilon,\delta)$, the only difference being that in each inner loop, the call to $\mathcal{B}^k(j,\mathbf{A}^1_{[j]},\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T})$ is replaced by a call to $\mathcal{B}^{k,-}(j,\mathbf{A}^1_{[j]},\frac{\epsilon}{4},\frac{1}{16N(\frac{\epsilon}{4},\frac{1}{16})T},U)$; and the call to $\mathcal{B}^k(t,\gamma,\frac{\epsilon}{2},\frac{1}{8})$ is replaced by a call to $\mathcal{B}^{k,-}(t,\gamma,\frac{\epsilon}{2},\frac{1}{8},U)$.

Then we have the following auxiliary algorithmic result, completely analogous to Lemma 4.

Lemma 27. *Under only the assumption that $Z_t \geq 0$ for all $t \in [1,T]$ w.p.1, the following is true. For all $k \geq 1$, $t \in [1,T]$, $\gamma \in \aleph^t$, $\epsilon,\delta \in (0,1)$, $U > 0$, the randomized algorithm $\mathcal{B}^{k,-}$ achieves the following when evaluated on $t,\gamma,\epsilon,\delta,U$. In total computational time at most $4(C+G+1)f_k(\epsilon,\delta)$, and with only access to randomness at most $f_k(\epsilon,\delta)$ calls to the base simulator $\mathcal{B}$, returns a random number X satisfying $P(|X - Z^{k,-}_{U,t}| \leq \epsilon) \geq 1-\delta$.*

The proof is essentially identically to the proof of Lemma 4, and we omit the details. In fact, Lemma 27 follows from Lemma 4 applied to the alternative minimization problem in which one uses the alternate cost functions $r'_t = 1 - U^{-1} \times \min(U, r_t)$. Next, we formally state the algorithm $\hat{\mathcal{A}}$, which will implement the appropriate truncation and calls to $\mathcal{B}^{k,-}$ (with appropriate parameters) to yield the approximation guaranteed in Theorem 2.3.9.

Algorithm $\hat{\mathcal{A}}(\epsilon, \delta, M_1, M_2)$:

Compute $\frac{M_2}{(M_1)^2}$ and store in γ_0, compute $10^4 \times \gamma_0^3 \times \epsilon^{-2} \times M_1$ and store in U_0

Set $k = \lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil, \alpha = ((\frac{U_0}{M_1})^{\frac{3}{2}} + 1)^{-2}, \beta = \delta \times ((\frac{U_0}{M_1})^{\frac{3}{2}} + 1)^{-1}$

Generate a length-k vector $\mathbf{L}$

For l = 1 to k

Create a length-$N(\frac{\alpha}{2}, \frac{\beta}{2})$ vector $\mathbf{A}^0$

For i = 1 to $N(\frac{\alpha}{2}, \frac{\beta}{2})$

Generate an ind. call to $\mathcal{B}(0, \emptyset)$ and store in D by T matrix $\mathbf{A}^1$

Create a length-T array $\mathbf{A}^2$

For j = 1 to T

Generate an ind. call to $\mathcal{B}^{l,-}(j, \mathbf{A}^1_{[j]}, \frac{\alpha}{2}, \frac{\beta}{2N(\frac{\alpha}{2}, \frac{\beta}{2})T}, U_0)$

Store in $\mathbf{A}^2_j$

Compute the minimum value of $\mathbf{A}^2$ and store in $\mathbf{A}^0_i$

Compute $(N(\frac{\alpha}{2}, \frac{\beta}{2}))^{-1} \sum_{i=1}^{N(\frac{\alpha}{2}, \frac{\beta}{2})} \mathbf{A}^0_i$ and store in $\mathbf{L}_l$

Store $\sum_{l=1}^{k} \mathbf{L}_l$ as Y

Return $U_0 \times (1 - Y)$

With Algorithm $\hat{\mathcal{A}}$ defined and Theorem 9 in hand, the proof of Theorem

2.3.9 follows similarly to the proof of Theorem 2.3.5, albeit with the additional complication of needing to verify the appropriateness of the truncation etc.

A.2.3 Proof of Theorem 2.3.9

Proof of Theorem 2.3.9 : Recall that $U_0 = 10^4 \times \gamma_0^3 \times \epsilon^{-2} \times M_1$ and $k = \lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil$. It then follows from the definition of $\hat{\mathcal{A}}$, and Theorem 2.3.7 that with probability at least $1 - \delta$, the sum $Y = \sum_{l=1}^{k} \mathbf{L}_l$ that $\hat{\mathcal{A}}$ computes must satisfies $|Y - E^{-}_{U_0,k}| \leq (\frac{U_0}{M_1})^{-\frac{3}{2}}$. Thus by Proposition 9, to prove the first part of the theorem (i.e. that the algorithm returns a value with the stated guarantees) it suffices to prove that $7 \times \gamma_0^{\frac{3}{2}} \times \left(\frac{U_0}{M_1} \times (2 \times (\frac{U_0}{M_1})^{-\frac{3}{2}}) + (\frac{U_0}{M_1})^{-\frac{1}{2}} \right) \leq \epsilon$, equivalently that $21\gamma_0^{\frac{3}{2}} (\frac{U_0}{M_1})^{-\frac{1}{2}} \leq \epsilon$. Since

$$21\gamma_0^{\frac{3}{2}} (\frac{U_0}{M_1})^{-\frac{1}{2}} \quad = \quad 21 \times \gamma_0^{\frac{3}{2}} \times \left(10^4 \times \gamma_0^3 \times \epsilon^{-2} \right)^{-\frac{1}{2}} \quad < \quad \epsilon,$$

combining the above completes the proof.

Next, let us prove the second part of the theorem regarding the runtime analysis. Recall that

$$f_k(\epsilon, \delta) = \log(\frac{2}{\delta}) \times 10^{2(k-1)^2} \times \epsilon^{-2(k-1)} \times (T+2)^{k-1} \times (1 + \log(\frac{1}{\epsilon}) + \log(T))^{k-1}.$$

Carefully accounting for all operations performed by $\hat{\mathcal{A}}$, and applying Lemma 27 and the monotonicities of various functions, we find that the computational

cost divided by $C + G + 1$ is at most

$$
\begin{aligned}
& 16 + \lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil \times 4 \times f_{\lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil + 1}\Big(((\frac{U_0}{M_1})^{\frac{3}{2}} + 1)^{-2}, \delta \times ((\frac{U_0}{M_1})^{\frac{3}{2}} + 1)^{-1} \Big) + \lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil \\
\leq \quad & 10^2 (\frac{U_0}{M_1})^{\frac{3}{2}} f_{\lceil (\frac{U_0}{M_1})^{\frac{3}{2}} \rceil + 1}(\frac{1}{4}(\frac{U_0}{M_1})^{-3}, \frac{1}{4}\delta(\frac{U_0}{M_1})^{-\frac{3}{2}}) \\
\leq \quad & 10^2 \times (\frac{U_0}{M_1})^{\frac{3}{2}} \times \Big(\log(8) + \log(\frac{1}{\delta}) + \frac{3}{2}\log(\frac{U_0}{M_1}) \Big) \times 10^{8(\frac{U_0}{M_1})^3} \times (4(\frac{U_0}{M_1})^3)^{4(\frac{U_0}{M_1})^{\frac{3}{2}}} \\
& \times \quad (T+2)^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \times \Big(1 + \log(4) + 3\log(\frac{U_0}{M_1}) + \log(T) \Big)^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \\
\leq \quad & \log(\frac{1}{\delta}) \times 10^{25(\frac{U_0}{M_1})^3} \times T^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \times \Big(3 + 3\log(\frac{U_0}{M_1}) + \log(T) \Big)^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \\
\leq \quad & \log(\frac{1}{\delta}) \times 10^{26(\frac{U_0}{M_1})^3} \times T^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \times \Big(1 + \log(\frac{U_0}{M_1}) + \log(T) \Big)^{2(\frac{U_0}{M_1})^{\frac{3}{2}}} \\
\leq \quad & \log(\frac{1}{\delta}) \times 10^{\left(10^{14}\gamma_0^9 \epsilon^{-6}\right)} \times T^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \\
& \times \quad \Big(1 + 4\log(10) + 3\log(\gamma_0) + 2\log(\frac{1}{\epsilon}) + \log(T) \Big)^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \\
\leq \quad & \log(\frac{1}{\delta}) \times 10^{\left(10^{15}\gamma_0^9 \epsilon^{-6}\right)} \times T^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \times \Big(1 + \log(\gamma_0) + \log(\frac{1}{\epsilon}) + \log(T) \Big)^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \\
\leq \quad & \log(\frac{1}{\delta}) \times \exp\left(10^{16}\gamma_0^9 \epsilon^{-6}\right) \times T^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \times \Big(1 + \log(\gamma_0) + \log(\frac{1}{\epsilon}) + \log(T) \Big)^{\left(10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}\right)} \\
\leq \quad & \log(\frac{1}{\delta}) \times \exp\left(10^{17}\gamma_0^9 \epsilon^{-6}\right) \times T^{10^8 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \times \Big(1 + \log(\gamma_0) + \log(\frac{1}{\epsilon}) \Big)^{10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \\
\leq \quad & \log(\frac{1}{\delta}) \times \exp\left(10^{18}\gamma_0^9 \epsilon^{-6}\right) \times T^{10^8 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \times \Big(1 + \log(\gamma_0) \Big)^{10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \\
\leq \quad & \log(\frac{1}{\delta}) \times \exp\left(10^{19}\gamma_0^9 \epsilon^{-6}\right) \times T^{10^8 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \times (2\gamma_0)^{10^7 \gamma_0^{\frac{9}{2}} \epsilon^{-3}} \\
\leq \quad & \log(\frac{1}{\delta}) \times \exp\left(10^{20}\gamma_0^9 \epsilon^{-6}\right) \times T^{10^8 \gamma_0^{\frac{9}{2}} \epsilon^{-3}}.
\end{aligned}
$$

The analysis for the number of calls to the base simulator follows nearly identically, and we omit the details. Combining the above completes the proof. □

APPENDIX B

CHAPTER 3 OF APPENDIX

We provide here technical details of the proofs and analysis of Chapter 3. Recall Hoeffding's inequality.

Theorem (Hoeffding's inequality)**.** *Suppose that for some $n \geq 1$ and $N > 0$, $\{X_i, i \in [1, n]\}$ are i.i.d., and $P(X_1 \in [0, N]) = 1$. Then*

$$P\Big(\Big|n^{-1}\sum_{i=1}^{n} X_i - E[X_1]\Big| \geq \eta\Big) \leq 2\exp\Big(-\frac{2\eta^2 n}{N^2}\Big).$$

B.1 The optimality equations and Proposition 1

This section is devoted to proving Proposition 1.

Proof of Proposition 1. We prove the proposition by induction. The case $k = 1$ trivially holds true. Suppose for some $k \geq 1$ the statement is true. Namely, the value functions $(\mathcal{J}_t^j)_{1\leq j\leq k, t\geq 0}$ solve the equations (3.1). Let's introduce a policy $\pi^a \triangleq (a_{[t-1]}, a, p^*, ..., p^*)$ and denote by $\pi^{a,s,k}$ the policy with the following property: $\pi^{a,s,k}_{[s-1]} = \pi^a_{[s-1]}$, and $\pi^{a,s,k}$ only take k active actions in $[s, T]$. Applying the inductive hypothesis, we may rewrite equation (3.1) associated with $\mathcal{J}_t^{k+1}$ as follows

$$\begin{aligned}
\mathcal{J}_t^{k+1}(a_{[t-1]}) &= \max_{a\in\mathcal{A}_t} \sup_{t+1\leq\tau\leq T} E\Big[\sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \sup_{\pi^{a,\tau,k}} E\Big[\sum_{s=\tau}^{T} r_s(\pi^{a,\tau,k}_{[s]})\Big|\mathcal{F}_\tau\Big]\Big|\mathcal{F}_t\Big] \\
&= \max_{a\in\mathcal{A}_t} \sup_{t+1\leq\tau\leq T} \sup_{\pi^{a,\tau,k-1}} E\Big[\sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \sum_{s=\tau}^{T} r_s(\pi^{a,\tau,k}_{[s]})\Big|\mathcal{F}_t\Big] \\
&= \sup_{\pi} E\Big[\sum_{s=t}^{T} r_s(\pi_{[s]})\Big|\mathcal{F}_t\Big] \qquad s.t.\ \ \pi_{[t-1]} = a_{[t-1]},\ \sum_{j=t+1}^{T} 1_{\{\pi_j\neq p^*\}} \leq k\ \ a.s.
\end{aligned}$$

where the second equality follows from the tower rule. Notice that the last equality indicates that $\mathcal{J}_t^{k+1}$, as the solution to equation (3.1) is the $(k+1)$-st value function. Combining with the induction hypothesis then completes the proof. □

B.1.1 Optimal stopping and proof of Lemma 8 and 9

This subsection is devoted to the proof of Lemma 8 and Lemma 9 in section 3.4. Exploiting Lemma 6 and 7, algorithm $\mathcal{W}_a$ approximates OPT by recursively computing $(Z_t^k)_{t\in[T]}$. Occasionally we shall make the dependence of $(Z_t^k)_{t\in[T]}$ on the underlying state $(X_t)_{t\in[T]}$ explicit by spelling out $Z_t^k(x_{[t]})$, where $X_{[t]} = x_{[t]}$ is the trajectory.

Now we define a sequence of auxiliary algorithms $(\mathcal{B}^k)_{k\geq 1}$, that outputs "good" approximation of $Z_t^k(x_{[t]})$. We start with the case $k = 1$, which simply calls the simulator of Assumption 4 with appropriate input parameters:

Algorithm $\mathcal{B}^1$ (for computing $Z_t^1(x_{[t]})$)

Input: $t, x_{[t]}, \epsilon, \delta$

Output: *cost evaluation* with parameters $\epsilon, \delta, t, x_{[t]}$

For $k \geq 1$, we define $\mathcal{B}^{k+1}$ inductively as follows:

Algorithm $\mathcal{B}^{k+1}$ (for computing $Z_t^{k+1}(x_{[t]})$)

Input: $t, x_{[t]}, \epsilon, \delta$

for $i = 1 : N(\frac{\epsilon}{4}, \frac{\delta}{4})$ **do**

generate an ind. sample of $X_{[t+1,T]}$ conditional on $X_{[t]} = x_{[t]}$

store in $\mathbf{A}^1 \leftarrow X_{[T]}$

for $j = 1 : T$ **do**

generate an ind. call to $\mathcal{B}^k(j, \mathbf{A}^1_{[j]}, \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T})$ and store in $\mathbf{A}^2[j]$

end for

$\mathbf{A}^3[i] \leftarrow \min_{j \in [T]} \mathbf{A}^2[j]$

end for

generate an ind. call to $\mathcal{B}^k(t, x_{[t]}, \frac{\epsilon}{2}, \frac{\delta}{2})$ and store in $\mathbf{A}^4$

Output: $\mathbf{A}^4 - (N(\frac{\epsilon}{4}, \frac{\delta}{4}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{4}, \frac{\delta}{4})} \mathbf{A}^3[i]$

We now formally analyze $\mathcal{B}^k$. Let's first introduce the following additional notation. For any $k \geq 1$, let the auxiliary functions f_k and q_k be defined as

$$\begin{aligned} f_k(\epsilon, \delta, T) &\triangleq 10^{2(k-1)^2} \times \epsilon^{-2(k-1)} \times (T+2)^{k-1} \times (1 + \log(\frac{1}{\delta}) + \log(\frac{1}{\epsilon}) + \log(T))^{k-1} \\ q_k(\epsilon, \delta, T) &\triangleq \delta \times (f_k(\epsilon, \delta, T))^{-1}. \end{aligned}$$

We will need the following auxiliary lemma, which demonstrates that functions f_k and q_k satisfies certain recursive bounds. The proof is almost identical to that of Lemma 24 in Appendix A, and we omit here.

Lemma 28. *For all $\epsilon, \delta \in (0, 1)$ and $k \geq 1$,*

$$\begin{aligned} f_{k+1}(\epsilon, \delta, T) &\geq (N(\frac{\epsilon}{4}, \frac{\delta}{4}) + 1) \times (T+2) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T), \\ q_{k+1}(\epsilon, \delta, T) &\leq q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T). \end{aligned}$$

With Lemma 28 in hand, we now prove the following lemma, which certifies that $(\mathcal{B}^k)_{k \geq 1}$ is indeed a set of "good" approximation algorithms.

Lemma 29. *For all $k \geq 1$, $t \in [1, T]$, $x_{[t]} \in \mathcal{R}^{d \times t}$, $\epsilon, \delta \in (0, 1)$, algorithm $\mathcal{B}^k$ achieves the following. In total computational time at most*

$$f_k(\epsilon, \delta, T) \times (C \times h_2(\frac{\epsilon}{4^k}, q_k(\epsilon, \delta, T)) + h_1(\frac{\epsilon}{4^k}, q_k(\epsilon, \delta, T)) + 1)$$

and with number of samples required from the simulator at most

$$f_k(\epsilon, \delta, T) \times h_2(\frac{\epsilon}{4^k}, q_k(\epsilon, \delta, T)),$$

returns a (possibly random) number X satisfying $P(|X - Z_t^k(x_{[t]})| > \epsilon U) \leq \delta$.

Proof of Lemma 29. We first show that $\mathcal{B}^k$ has the desired output. We proceed by induction. The base case is immediate by the definition of the simulator in Assumption 4. Now suppose for some $k \geq 1$, the statement is true. We prove it also holds in case $k + 1$. Indeed, for each $i \in [N(\epsilon/4, \delta/4)]$, T calls are made to $\mathcal{B}^k$ and the outputs are stored as $\mathbf{A}^2$. By induction hypothesis, we have

$$P\left(\left|\mathbf{A}^2[j] - Z_j^k(x_{[t]}, X_{[t+1,j]})\right| > \frac{\epsilon}{4}U\right) < \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}.$$

Applying a union bound and we have

$$P\left(\left|\mathbf{A}^3[i] - \min_{j \in [T]} Z_j^k\right| > \frac{\epsilon}{4}U\right) < \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})}.$$

With two more union bounds the desired result follows directly.

Next we prove the runtime and sample complexity bounds. The base case $k = 1$ is trivial. Suppose the induction is true for some $k \geq 1$. We set to prove case $k + 1$. The only access to randomness for $\mathcal{B}^{k+1}$ is through its $N(\frac{\epsilon}{4}, \frac{\delta}{4})$ direct calls to the simulator (whose output is each time stored in $\mathbf{A}^1$), its $N(\frac{\epsilon}{4}, \frac{\delta}{4})T$ calls to $\mathcal{B}^k(j, \mathbf{A}^1_{[j]}, \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T})$, and its one final call to $\mathcal{B}^k(t, x_{[t]}, \frac{\epsilon}{2}, \frac{\delta}{2})$ (whose output is stored in $\mathbf{A}^3$). It thus follows from the induction hypothesis and the monotonicity of several functions N,f_k, q_k, h_1 and h_2, that the number of calls to the simulator

made by $\mathcal{B}^{k+1}(t, x_{[t]}, \epsilon, \delta)$ is at most

$$\begin{aligned} & N(\frac{\epsilon}{4}, \frac{\delta}{4}) + N(\frac{\epsilon}{4}, \frac{\delta}{4}) \times T \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T) \times h_2(\frac{\epsilon}{4 \times 4^k}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) \\ & \quad + f_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T) \times h_2(\frac{\epsilon}{2 \times 4^k}, q_k(\frac{\epsilon}{2}, \frac{\delta}{2}), T) \\ \leq & \quad N(\frac{\epsilon}{4}, \frac{\delta}{4}) \times (T+2) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T) \times h_2(\frac{\epsilon}{4 \times 4^k}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) \\ \leq & \quad f_{k+1}(\epsilon, \delta, T) \times h_2(\frac{\epsilon}{4^{k+1}}, q_{k+1}(\epsilon, \delta, T)). \end{aligned} \tag{B.1}$$

We next focus on computational costs. In each of the $N(\frac{\epsilon}{4}, \frac{\delta}{4})$ iterations of the outer for loop (indexed by i), first one direct call is made to the simulator at computational cost C; then T calls are made to $\mathcal{B}^k(j, \mathbf{A}^1_{[j]}, \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T})$ (for different values of j), each at computational cost at most

$$\Big(C \times h_2(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + h_1(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + 1\Big) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T);$$

then the minimum of a length-T array is computed (at computational cost T). One additional call is then made to $\mathcal{B}^k(t, x_{[t]}, \frac{\epsilon}{2}, \frac{\delta}{2})$, at computational cost at most

$$\Big(C \times h_2(\frac{\epsilon}{2 \times 4^k}, q_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T)) + h_1(\frac{\epsilon}{2 \times 4^k}, q_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T)) + 1\Big) \times f_k(\frac{\epsilon}{2}, \frac{\delta}{2});$$

and finally the average of $N(\frac{\epsilon}{4}, \frac{\delta}{4})$ is computed and subtracted from $\mathbf{A}^3$, at computational cost $N(\frac{\epsilon}{4}, \frac{\delta}{4}) + 1$. It thus follows from the inductive hypothesis, and several easily verified monotonicities of N, f_k, q_k and h_1, h_2, that the computa-

tional cost of $\mathcal{B}^{k+1}(t, x_{[t]}, \epsilon, \delta)$ is at most

$$
\begin{aligned}
& T \times N(\frac{\epsilon}{4}, \frac{\delta}{4}) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T) \\
& \quad \times \left(C \times h_2(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + h_1(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + 1\right) \\
& \quad + \ N(\frac{\epsilon}{4}, \frac{\delta}{4})C + T + N(\frac{\epsilon}{4}, \frac{\delta}{4}) + 1 \\
& \quad + \left(C \times h_2(\frac{\epsilon}{2 \times 4^k}, q_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T)) + h_1(\frac{\epsilon}{2 \times 4^k}, q_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T)) + 1\right) \times f_k(\frac{\epsilon}{2}, \frac{\delta}{2}, T) \\
\leq \quad & \left(C \times h_2(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + h_1(\frac{\epsilon}{4^{k+1}}, q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T)) + 1\right) \\
& \quad \times (N(\frac{\epsilon}{4}, \frac{\delta}{4}) + 1) \times (T + 2) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T). \qquad \text{(B.2)}
\end{aligned}
$$

Combining the above calculation with Lemma 24, we prove that the induction is true in case $k + 1$, thus completing the proof. □

With Lemma 29 in hand, we now complete the proof of Lemma 8. We first formally define algorithm $\mathcal{W}_a$, which combines $\mathcal{B}^k$ and Lemma 6 and 7 that gives a good approximation of OPT:

Algorithm $\mathcal{W}_a$

Input: ϵ, δ

set $G \triangleq \lceil \frac{2}{\epsilon} \rceil$

for $i = 1 : G$ **do**

generate a call to $\hat{\mathcal{B}}^i(\frac{\epsilon}{2G}, \frac{\delta}{G})$ and store in $\mathbf{A}^0[i]$

end for

Output: $\sum_{i=1}^{G} \mathbf{A}^0[i]$

---

with $\hat{\mathcal{B}}^k$ a modified version of $\mathcal{B}^k$ that approximates $E[\min_{t \in [T]} Z_t^k]$:

Algorithm $\hat{\mathcal{B}}^k$ (for computing $E[\min_{t\in[T]} Z_t^k]$)

Input: ϵ, δ:

for $i = 1 : N(\frac{\epsilon}{2}, \frac{\delta}{2})$ **do**

call the simulator to generate $(x_1, ..., x_T)$ and store in $\mathbf{A}^1$

for $j = 1 : T$ **do**

generate an ind. call to $\mathcal{B}^k(j, \mathbf{A}^1_{[j]}, \frac{\epsilon}{2}, \frac{\delta}{2N(\frac{\epsilon}{2}, \frac{\delta}{2})T})$ and store in $\mathbf{A}^2[j]$

end for

compute the minimum value of $\mathbf{A}^2$ and store in $\mathbf{A}^0[i]$

end for

Output: $(N(\frac{\epsilon}{2}, \frac{\delta}{2}))^{-1} \sum_{i=1}^{N(\frac{\epsilon}{2}, \frac{\delta}{2})} \mathbf{A}^0[i]$

Proof of Lemma 8. By arguments almost identical to the proof of Lemma 29, we conclude that $\hat{\mathcal{B}}^k(\epsilon, \delta)$ in runtime at most $(C \times h_2(\frac{\epsilon}{4^{k+1}}, q_{k+1}(\epsilon, \delta, T)) + h_1(\frac{\epsilon}{4^{k+1}}, q_{k+1}(\epsilon, \delta, T)) + 1) \times f_{k+1}(\epsilon, \delta, T)$ and calls to the simulator at most $2 \times h_2(\frac{\epsilon}{4^{k+1}}, q_{k+1}(\epsilon, \delta, T)) \times f_{k+1}(\epsilon, \delta, T)$, output a number X satisfies $P(|X - E[\min_{t\in[T]} Z_t^k]| > \epsilon U) < \delta$. We omit the details. Now let's analyze algorithm $\mathcal{W}_a$. We first prove algorithm $\mathcal{W}_a$, as defined, can output a number satisfying the desired probability bounds. Indeed, this follows directly from Lemma 7, the above analysis of $\hat{\mathcal{B}}^k$ and a union bound. It remains to bound the computational and sample complexity. $\mathcal{W}_a(\epsilon, \delta)$ makes G calls to $\hat{\mathcal{B}}^i(\frac{\epsilon}{2G}, \frac{\delta}{G}), i \in [G]$. Therefore, the algorithm's

runtime is bounded by

$$\begin{aligned}
&\sum_{i=1}^{G}\Big(C\times h_2(\frac{\epsilon}{4^{i+1}},q_{i+1}(\epsilon,\delta,T))+h_1(\frac{\epsilon}{4^{i+1}},q_{i+1}(\epsilon,\delta,T))+1\Big)f_{i+1}(\epsilon,\delta,T)+G+1\\
&\leq \Big(C\times h_2(\frac{\epsilon}{4^{G+1}},q_{G+1}(\epsilon,\delta,T))+h_1(\frac{\epsilon}{4^{G+1}},q_{G+1}(\epsilon,\delta,T))+1\Big)\times G\times f_{G+1}(\frac{\epsilon^2}{6},\frac{\delta\epsilon}{3},T)+G+1\\
&\leq 6\Big(C\times h_2(\frac{\epsilon}{4^{G+1}},q_{G+1}(\epsilon,\delta,T))+h_1(\frac{\epsilon}{4^{G+1}},q_{G+1}(\epsilon,\delta,T))+1\Big)\times\epsilon^{-1}\times f_{G+1}(\frac{\epsilon^2}{6},\frac{\delta\epsilon}{3},T)\\
&\leq \Big(C\times H_2(\epsilon,\delta,T)+H_1(\epsilon,\delta,T)\Big)\times\epsilon^{-1}\times f_{G+1}(\frac{\epsilon^2}{6},\frac{\delta\epsilon}{3},T)\\
&\leq \Big(C\times H_2(\epsilon,\delta,T)+H_1(\epsilon,\delta,T)\Big)\times\exp(200\epsilon^{-2})T^{6\epsilon^{-1}}(1+\log(\frac{1}{\delta}))^{6\epsilon^{-1}},
\end{aligned}$$

where the last inequality follows from the definition of function f_k and several straightforward bounds, and we omit the details. The analysis of the sample complexity is nearly identical and we also omit the details. Combining the above completes the proof. □

We continue to prove Lemma 9. We state without proof the following results, which are respectively Lemma 2 and Corollary 3 in [134], and which show how to find a good stopping policy.

Lemma 30. *For all $k \geq 1$, $\min_{t\in[T]} Z_t^k \leq \frac{U}{k}$. a.s.*

Lemma 31. *For $k \geq 1$, let τ_k be the stopping time that stops the first time that $Z_t^k \leq \frac{U}{k}$, where such a time exists w.p.1 by Lemma 30. Then $|E[Z_{\tau_k}] - OPT| \leq \frac{U}{k}$.*

Combining Lemma 29 and 31, we complete the proof of Lemma 9. Let's first formally define algorithm $\mathcal{W}_b$.

Algorithm $\mathcal{W}_b$

Input: ϵ

for $t = 1 : T$ **do**

observe $X_t = x_t$, make one call to $\mathcal{B}^{\lceil 4\epsilon^{-1}\rceil}(t, x_{[t]}, \frac{\epsilon}{4}, \frac{\epsilon}{4T})$ and save the output as **I**

if I $< \frac{\epsilon}{2}U$

Output: STOP **break**

end if

end for

Output: STOP

Proof of Lemma 9. The fact that $\mathcal{W}_b$, as defined, outputs stopping strategies with desired performance guarantee follows from Lemma 7 and 31, and several straightforward union bounds. The argument is in fact identical to the proof of Corollary 3 in [134] and we omit the technical details here. Next we show the runtime and sampling complexity bounds. Notice that at each time t, one call is made to $\mathcal{B}^{\lceil 4\epsilon^{-1}\rceil}$ with parameters $\frac{\epsilon}{4}, \frac{\epsilon}{4T}$. By directly applying Lemma 29, we conclude that the computational cost for algorithm $\mathcal{W}_b$ to output the decision in each time period is at most

$$
\begin{aligned}
& f_{\lceil 4\epsilon^{-1}\rceil}(\frac{\epsilon}{4}, \frac{\epsilon}{4T}, T) \times \left(C \times h_2(\frac{\epsilon}{4^{1+\lceil 4\epsilon^{-1}\rceil}}, q_{\lceil 4\epsilon^{-1}\rceil}(\frac{\epsilon}{4}, \frac{\epsilon}{4T}, T)) + h_1(\frac{\epsilon}{4^{1+\lceil 4\epsilon^{-1}\rceil}}, q_{\lceil 4\epsilon^{-1}\rceil}(\frac{\epsilon}{4}, \frac{\epsilon}{4T}, T)) + 1 \right) \\
\leq \quad & \exp(200\epsilon^{-2}) \times T^{6\epsilon^{-1}} \\
& \times \left(C \times h_2(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}) + h_1(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}) \right)
\end{aligned}
$$

The number of samples required from the simulator can be accounted in a similar manner. We thus complete the proof. □

B.2 Main algorithms and proof of Lemma 10, 11 and 12

This section is devoted to the detailed analysis of our main algorithms in section 3.5. Specifically, we analyze algorithms $(Q^k)_{k\geq 1}$, which is the content of Lemma 10; we prove algorithm Q_b's performance guarantee, which is the content of Lemma 11; and we provide the computational and sampling complexity analysis, which is the content of Lemma 12.

Additional notation. For notational simplicity, we introduce the following functions

$$\begin{aligned} g^k(\epsilon,\delta) &\triangleq (10TM^2)^{kU(10^6,\frac{1}{\epsilon})}\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,\frac{1}{\epsilon})} \\ f^k(\epsilon,\delta) &\triangleq C(10TM^2)^{kU(10^6,\frac{1}{\epsilon})}\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,\frac{1}{\epsilon})}. \end{aligned}$$

Notice that here we omit other arguments of f^k and g^k, such as T and M, assuming they are fixed. These auxiliary functions satisfy certain properties such as a specific recursive inequality and monotonicity, which can greatly simplify our algorithmic analysis. We state them as the following lemma. The proof is quite tedious and is deferred to the last section.

Lemma 32. *Functions $g^k(\epsilon,\delta)$ and $f^k(\epsilon,\delta)$ are decreasing in ϵ and δ. They satisfy*

$$\begin{aligned} g^0(\epsilon,\delta) &\geq 2MT\epsilon^{-2}\log(2M\delta^{-1}) \\ f^0(\epsilon,\delta) &\geq 3CMT\epsilon^{-2}\log(2M\delta^{-1}). \end{aligned}$$

Furthermore, they satisfy the recursion

$$\begin{aligned} g^{k+1}(\epsilon,\delta) \quad &\geq \quad 2g^k\Big(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\delta M^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\Big) \\ &\qquad \times \exp(200\epsilon^{-2})MT^{6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\ f^{k+1}(\epsilon,\delta) \quad &\geq \quad 2\Big(C\times g^k(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\delta M^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}) \\ &\qquad + f^k(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\delta M^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}})\Big) \\ &\qquad \times \exp(200\epsilon^{-2})MT^{6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \end{aligned}$$

In view of the boundedness Assumption 3, we also introduce the maximum cap on total rewards: $\mathcal{L} \stackrel{\Delta}{=} \sup_{a_{[T]},x_{[T]}} \sum_{t=1}^{T} r_t(a_{[t]}, x_{[t]})$. Combining with Assumption 3, we then have $\text{OPT} \geq \alpha\mathcal{L}$. The cap $\mathcal{L}$ does not appear in the final complexity bound, rather it serves as an "intermediate" in the derivation of our results. It can be arbitrarily large.

With Lemma 32 and the maximum cap $\mathcal{L}$, we now complete the proof of Lemma 10.

Proof. Proof of Lemma 10 Let us first show that the output of algorithm Q^k does satisfy the probability error bounds. We proceed by induction. In the base case $k = 1$ with the past action sequence $a_0, ..., a_{t-1}$ and the state trajectory $x_0, ..., x_t$, for each $a \in \mathcal{A}_t$, fix a policy $\pi^a \stackrel{\Delta}{=} (a_{[t-1]}, a, p^*, ..., p^*)$. Let W_a be a *r.v.* that is distributed as $\sum_{s=t}^{T} r_s(\pi^a_{[s]})$, conditional on $X_{[t]} = x_{[t]}$. Q^1 calls the simulator $\mathcal{S}$ to generate $N(\alpha\epsilon, \frac{\delta}{M})$ *i.i.d.* samples of W_a and stores their average as $\mathbf{Y}^a$. By the definition of the maximum cap $\mathcal{L}$, We have $0 \leq W_a \leq \mathcal{L}$ for all $a \in \mathcal{A}_t$ and $\alpha\mathcal{L} \leq \text{OPT} \leq \mathcal{L}$. Combining the above with Lemma 22, we get

$$P(|\mathbf{Y}^a - E[W_a]| \geq \epsilon \times \text{OPT}) \leq \frac{\delta}{M} \text{ for all } a \in \mathcal{A}_t.$$

By definition, $\mathcal{J}_t^1 = \max_{a\in\mathcal{A}_t} E[W_a]$. Applying a union bound, we get

$$P(|\max_{a\in\mathcal{A}_t} \mathbf{Y}^a - \mathcal{J}_t^1(a_{[t-1]}, x_{[t]})| \geq \epsilon \times \text{OPT}) \leq \delta,$$

which completes the proof for the base case. Now suppose the induction is true for some $k \geq 1$, we prove it's also true in the case $k+1$. For $a \in \mathcal{A}_t$, let $\mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) \stackrel{\Delta}{=} \sup_{\tau\geq t} E[\sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \mathcal{J}_\tau^k(\pi^a_{[\tau-1]})\big|X_{[t]} = x_{[t]}]$. Namely, $\mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]})$ is the optimal value achieved by choosing action a in the current period, subject to k active action-taking opportunities in $[t+1, T]$, given the past state trajectory $X_{[t]} = x_{[t]}$ and action sequence $a_{[t-1]}$. Specifically, $\mathcal{J}_t^{k+1}(a_{[t-1]}, x_{[t]}) = \max_{a\in\mathcal{A}_t} \mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]})$. The algorithm calls the optimal stopping subroutine $\mathcal{W}_a$ to solve for $\mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]})$ for each a. By Lemma 8 and the fact that $\mathcal{L}$ is the maximum cap, we conclude that $\mathcal{W}_a$, when evaluated on $(\alpha\epsilon, \delta/M)$, returns a random number $\mathbf{Y}^a$ satisfying $P(\left|\mathbf{Y}^a - \mathcal{J}_t^{k+1,a}\right| > \alpha\epsilon\mathcal{L}) < \frac{\delta}{M}$. Combining the above with the assumption that $\text{OPT} \geq \alpha\mathcal{L}$ and a union bound we then have

$$P(|\max_{a\in\mathcal{A}_t} \mathbf{Y}^a - \mathcal{J}_t^{k+1}(a_{[t-1]}, x_{[t]})| > \epsilon\text{OPT}) < \delta.$$

Hence the statement holds true in case $k+1$. By induction we complete the proof.

Next we prove that $\mathcal{Q}^{k+1}$ satisfies the desired computational and sample complexity bounds. By our definition of g^k and f^k, it suffices to prove that $\mathcal{Q}^{k+1}$ achieves the desired performance at a computational cost $f^k(\alpha\epsilon, \delta)$ and the number of calls to the simulator $g^k(\alpha\epsilon, \delta)$. Again we proceed by induction. In the base case $k = 1$, the samples required by $\mathcal{Q}^1$ with input parameters ϵ, δ (and regardless of the state trajectory and the action sequence) is through its (at most) $4N(\alpha\epsilon, \delta/M) \times T \times M$ direct calls to $\mathcal{S}$. Thus, the number of independent samples generated is at most $2MT(\alpha\epsilon)^{-2}\log(2M\delta^{-1})$, which is bounded by $g^0(\alpha\epsilon, \delta)$ by Lemma 32. The accounting of computational costs are as follows. In each round

of the for loop,at most $TN(\alpha\epsilon, \delta/M)$ samples are generated; at most $TN(\alpha\epsilon, \delta/M)$ random rewards are generated, both through simulator $\mathcal{S}$. Then the sum of a T-vector is calculated, the average of $N(\alpha\epsilon, \delta/M)$ numbers is computed and finally the maximum of M numbers is computed. Thus the computational cost of Q^0 when called on $\epsilon, \delta, \alpha, M$ is at most

$$\begin{aligned} & \Big(2CTN(\alpha\epsilon, \delta/M) + T + N(\alpha\epsilon, \delta/M)\Big) \times M + M \\ \leq \quad & 3CMT(\alpha\epsilon)^{-2} \log(2M\delta^{-1}). \end{aligned}$$

By Lemma 32 we have that the above is further bounded by $f^0(\alpha\epsilon, \delta)$. Combining the above with the definition of f^0 and g^0, we conclude that the base case is true.

Suppose the induction is true for some $k \geq 1$, we prove it in the case $k+1$. Notice that Q^{k+1} makes repeated calls to $\mathcal{W}_a$. Let's carefully account for the computational and sampling costs of calling $\mathcal{W}_a$ within Q^{k+1}. With state trajectory $x_0, ..., x_t$ and action sequence $a_0, ..., a_{t-1}$, the specific optimal stopping problem here has the following reward function $\sum_{s=t}^{j-1} r_s(\pi^a_{[s]}) + \mathcal{J}^k_j(\pi^a_{[j-1]})$ for each $j \in [t, T]$ and some $a \in \mathcal{A}_t$, with $\pi^a = (a_{[t-1]}, a, p^*, ..., p^*)$ as defined above. Let $Z_j \stackrel{\Delta}{=} \sum_{s=t}^{j-1} r_s(\pi^a_{[s]}) + \mathcal{J}^k_j(\pi^a_{[j-1]})$. Recall that Z_j is $\mathcal{F}_j$-measurable. We specify *the computational and sampling cost to evaluate Z_j to a desired precision with high probability*, which is key to the analysis of $\mathcal{W}_a$ and will appear in the ultimate complexity bounds of Q^{k+1}. We show that, one can get a number $\tilde{Z}$ such that $P(|\tilde{Z} - Z_j| \geq \eta_1 \mathcal{L}) \leq \eta_2$ at computational cost at most $2f^k(\alpha\eta_1, \eta_2)$ and number of samples required from simulator $\mathcal{S}$ at most $g^k(\alpha\eta_1, \eta_2)$. Indeed, Z_j consists of two parts. The first part $\sum_{s=t}^{j-1} r_s(\pi^a_{[s]})$ can be exactly evaluated using simulator $\mathcal{S}$ at computational cost at most CT, according to Assumption 2. We deal with the second part. The induction hypothesis and the fact that $\text{OPT} \leq \mathcal{L}$ imply that one can get a number $\tilde{J}$ such that $P(|\tilde{J} - \mathcal{J}^k_j| \geq \eta_1 \mathcal{L}) \leq \eta_2$ at computa-

tional costs at most $f^k(\alpha\eta_1, \eta_2)$ and number of samples required from simulator $\mathcal{S}$ at most $g^k(\alpha\eta_1, \eta_2)$. Combining the above two parts, we conclude that one can get a number $\tilde{Z}$ such that $P(|\tilde{Z} - Z_j| \geq \eta_1 L) \leq \eta_2$ at computational cost at most $h_1(\eta_1, \eta_2) = CT + f^k(\alpha\eta_1, \eta_2) \leq 2f^k(\alpha\eta_1, \eta_2)$ and number of samples required from simulator $\mathcal{S}$ at most $h_2(\eta_1, \eta_2) = g^k(\alpha\eta_1, \eta_2)$.

Now let's study the sampling cost. In each round of the for loop, Q^{k+1}'s only access to simulator $\mathcal{S}$ is through its call to $\mathcal{W}_a$, evaluated on $\alpha\epsilon$ and δ/M. Combining the previous discussion with Lemma 8, we have that the total number of samples required by Q^{k+1} is at most

$$
\begin{aligned}
& H_2(\alpha\epsilon, \delta/M, T) \times \exp(200\alpha^{-2}\epsilon^{-2}) T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} M \\
= \quad & h_2(4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}}) \\
& \quad \times \exp(200\alpha^{-2}\epsilon^{-2}) M T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} \\
= \quad & g^k\Big(\alpha 4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}}\Big) \\
& \quad \times \exp(200\alpha^{-2}\epsilon^{-2}) M T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} \\
\leq \quad & g^k\Big(\alpha\epsilon 4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}}\Big) \\
& \quad \times \exp(200\alpha^{-2}\epsilon^{-2}) M T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} \\
\leq \quad & g^{k+1}(\alpha\epsilon, \delta) \qquad \text{(B.3)}
\end{aligned}
$$

where the last inequality follows from Lemma 32.

We next focus on computational costs. In each iteration of the for loop, one call is made to $\mathcal{W}_a$ evaluated on $\alpha\epsilon, \delta/M$. Combining the previous discussion

with Lemma 8, we have the total computational cost is at most

$$
\begin{aligned}
&\Big(C \times H_2(\alpha\epsilon, \delta/M, T) + H_1(\alpha\epsilon, \delta/M, T)\Big) \times \exp(200\alpha^{-2}\epsilon^{-2}) T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} M \\
= \quad & \Big(C \times h_2(4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}}) \\
& \quad + h_1(4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}})\Big) \\
& \quad \times \exp(200\alpha^{-2}\epsilon^{-2}) M T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} \\
\leq \quad & \Big(C \times g^k(\alpha\epsilon 4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}}) \\
& \quad + 2 f^k(\alpha\epsilon 4^{-6\alpha^{-1}\epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2}\epsilon^{-2}) T^{-6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{-3\alpha^{-1}\epsilon^{-1}})\Big) \\
& \quad \times \exp(200\alpha^{-2}\epsilon^{-2}) M T^{6\alpha^{-1}\epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1}\epsilon^{-1}} \\
\leq \quad & f^{k+1}(\alpha\epsilon, \delta), \qquad \text{(B.4)}
\end{aligned}
$$

where again, the last inequality follows from Lemma 32. Combining bounds (B.3) and (B.4) with the definition of f^k and g^k then completes the proof. □

Proof of Lemma 11. We proceed by induction. In the base case $k = 1$, the algorithm calls Q^1 with $(\frac{\epsilon}{8(K+1)}, \frac{M\alpha\epsilon}{8(K+1)} \wedge 1)$, which outputs an action $a^{\mathbf{B}} \in \mathcal{A}_t$, where we spell out $\mathbf{B}$ as a random variable defined in $[M]$ (due to the randomized nature of Q^1). The algorithm then fixes $a^{\mathbf{B}}$ as its choice for the current period, and takes p^* for the rest of the periods. (since $k = 1$ means no more active action is allowed in later periods.) The expected total reward is $E[\sum_{s=t}^{T} r_s(\pi^{\mathbf{B}}_{[s]} | X_{[t]} = x_{[t]}]$, which we denote by $\bar{\mathcal{J}}_t^{1,\mathbf{B}}(a_{[t-1]}, x_{[t]})$ (here $\pi^{\mathbf{B}} = (a_{[t-1]}, a^{\mathbf{B}}, p^*, ..., p^*)$, as defined previously). Notice that the expectation is taken not only over the state evolution but also over $\mathbf{B}$, so that $\bar{\mathcal{J}}_t^{1,\mathbf{B}}(a_{[t-1]}, x_{[t]})$ is a constant measuring the expected performance of algorithm Q_b. We denote by $\mathcal{J}_t^{1,\mathbf{B}}(a_{[t-1]}, x_{[t]})$ the expected performance of algorithm Q_b conditional on the realization $\mathbf{B}$. We have that $\bar{\mathcal{J}}_t^{1,\mathbf{B}} = E^{\mathbf{B}}[\mathcal{J}_t^{1,\mathbf{B}}]$, where the expectation is taken over random variable $\mathbf{B}$. Suppose the true optimal action at this time point is a^*. And the corresponding policy π^* is defined as

$(a_{[t-1]}, a^*, p^*, ..., p^*)$. By definition we have

$$E\Big[\sum_{s=t}^{T} r_s(\pi^*_{[s]})\Big| X_{[t]} = x_{[t]}\Big] = \mathcal{J}_t^1(a_{[t-1]}, x_{[t]}).$$

We bound the gap between $\bar{\mathcal{J}}_t^{1,\mathbf{B}}$ and $\mathcal{J}_t^1$. For notational simplicity, we hide the dependence on $X_{[t]} = x_{[t]}$ in the following probability statements. Recall that for each $a \in \mathcal{A}_t$, algorithm $\mathcal{Q}^1$ computes a number $\mathbf{Y}^a$ that satisfies (with the given control parameters here)

$$P\Big(\Big|\mathbf{Y}^a - \mathcal{J}_t^{1,a}\Big| > \frac{\alpha\epsilon}{8(K+1)}\mathcal{L}\Big) \le \frac{\alpha\epsilon}{8(K+1)}. \tag{B.5}$$

For the fixed actions a^* or $a^{\mathbf{B}}$, we use $\mathbf{Y}^*$ and $\mathbf{Y}^{\mathbf{B}}$ to denote the corresponding output approximate value from $\mathcal{Q}^1$. We thus have

$$\begin{aligned}
P\Big(\mathcal{J}_t^1 - \mathcal{J}_t^{1,\mathbf{B}} > \frac{\alpha\epsilon}{4(K+1)}\mathcal{L}\Big) &= P\Big(\mathcal{J}_t^1 - Y^* + Y^* - Y^{\mathbf{B}} + Y^{\mathbf{B}} - \mathcal{J}_t^{1,\mathbf{B}} > \frac{\alpha\epsilon}{4(K+1)}\mathcal{L}\Big) \\
&\le P\Big(\mathcal{J}_t^1 - Y^* + Y^{\mathbf{B}} - \mathcal{J}_t^{1,\mathbf{B}} > \frac{\alpha\epsilon}{4(K+1)}\mathcal{L}\Big) \\
&\le P\Big(|\mathcal{J}_t^1 - Y^*| > \frac{\alpha\epsilon}{8(K+1)}\mathcal{L}\Big) + P\Big(|Y^{\mathbf{B}} - \mathcal{J}_t^{1,\mathbf{B}}| > \frac{\alpha\epsilon}{8(K+1)}\mathcal{L}\Big) \\
&\le \frac{\alpha\epsilon}{8(K+1)} + \frac{\alpha\epsilon}{8(K+1)} = \frac{\alpha\epsilon}{4(K+1)},
\end{aligned}$$

where the first inequality follows from the defining property of the algorithm-selected action $a^{\mathbf{B}}$ that $Y^{\mathbf{B}} \ge \mathbf{Y}^a$, $a \in \mathcal{A}_t$, the second inequality follows from a union bound and the third inequality follows from (B.5). With the above probability bound, we may further bound the expected gap $E[\mathcal{J}_t^1 - \mathcal{J}_t^{1,\mathbf{B}}]$ (where the expectation is taken over random variable $\mathbf{B}$) by

$$\begin{aligned}
E\Big[\mathcal{J}_t^1 - \mathcal{J}_t^{1,\mathbf{B}}\Big] &\le P\Big(\mathcal{J}_t^1 - \mathcal{J}_t^{1,\mathbf{B}} > \frac{\alpha\epsilon}{4(K+1)}\mathcal{L}\Big) \times \mathcal{L} + P\Big(\mathcal{J}_t^1 - \mathcal{J}_t^{1,\mathbf{B}} \le \frac{\alpha\epsilon}{4(K+1)}\mathcal{L}\Big) \times \frac{\alpha\epsilon}{4(K+1)}\mathcal{L} \\
&\le \frac{\alpha\epsilon}{4(K+1)}\mathcal{L} + \frac{\alpha\epsilon}{4(K+1)}\mathcal{L} \\
&= \frac{\epsilon}{2(K+1)}\alpha\mathcal{L} < \frac{\epsilon}{(K+1)}\mathrm{OPT},
\end{aligned}$$

where in the first inequality we use the definition of the maximum cap $\mathcal{L}$ which yields $w.p.1.\ 0 < \mathcal{J}_t^{1,\mathbf{B}} \le \mathcal{J}_t^1 < \mathcal{L}$ and in the second inequality we use Assumption

3 that $\alpha\mathcal{L} \leq \text{OPT}$. Since $\bar{\mathcal{J}}_t^{1,\mathbf{B}} = E[\mathcal{J}_t^{1,\mathbf{B}}]$, we thus conclude that the base case is true.

Now suppose for some $k \geq 1$, the conclusion holds. We prove it also holds in the case $k+1$. Recall the process of algorithm $\mathcal{Q}_b$. It first makes a call to $\mathcal{Q}^{k+1}$ to compute an action to take. While proceeding with the passive action p^* afterwards, the algorithm makes a call to $\mathcal{W}_b$ at every time to decide whether to start a new epoch. When $\mathcal{W}_b$ outputs STOP, the algorithm will call $\mathcal{Q}^k$ to compute another action, and the budget of action change decreases by one, becoming k. Before formally analyzing the algorithm, let's first introduce several notation.

As before, we let $a^{\mathbf{B}}$ be the output action from $\mathcal{Q}^{k+1}$. Notice that $\mathbf{B}$ is a random variable due to the randomized nature of algorithm $\mathcal{Q}^{k+1}$. $\pi^{\mathbf{B}} = (a_{[t-1]}, a^{\mathbf{B}}, p^*, ..., p^*)$ is the corresponding policy. We let $\mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) \triangleq \sup_\tau E[\sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \mathcal{J}_\tau^k(\pi^a_{[\tau-1]})|X_{[t]} = x_{[t]}]$ denote the largest total expected reward given that the choice of action in the current period is a. Let $\bar{\mathcal{J}}_t^{k+1,\mathbf{B}}(a_{[t-1]}, x_{[t]}) \triangleq E[\mathcal{J}_t^{k+1,\mathbf{B}}(a_{[t-1]}, x_{[t]})]$ where the expectation is taken over $\mathbf{B}$. In words, $\bar{\mathcal{J}}_t^{k+1,\mathbf{B}}$ is the largest total expected reward given that the current choice of action follows the output of $\mathcal{Q}_b$. We let the output stopping time returned by $\mathcal{W}_b$ be $\tau_{\mathbf{B}}$. We denote by $\tilde{\mathcal{J}}_t^{k+1,\mathbf{B}}(a_{[t-1]}, x_{[t]}) \triangleq E[\sum_{s=t}^{\tau_{\mathbf{B}}-1} r_s(\pi^{\mathbf{B}}_{[s]}) + \mathcal{J}_{\tau_{\mathbf{B}}}^k(\pi^{\mathbf{B}}_{[\tau-1]})|X_{[t]} = x_{[t]}]$ the expected total reward following the current choice $\mathbf{B}$ from $\mathcal{Q}^{k+1}$ and the stopping time $\tau_{\mathbf{B}}$ computed from $\mathcal{Q}_b$, and following the true optimal policy afterwards. By the randomized nature of $\mathcal{W}_b$, $\tau_{\mathbf{B}}$ is random even conditional on $\mathbf{B}$. The expectation in the definition of $\tilde{\mathcal{J}}$ is taken over $X_t, \mathbf{B}$ and $\tau_{\mathbf{B}}$. Let $\hat{\mathcal{J}}_t^{k+1}(a_{[t-1]}, x_{[t]})$ denote the expected total reward following the policy returned by $\mathcal{Q}_b$. By the definition of $\mathbf{B}$ and $\tau_{\mathbf{B}}$, $(\hat{\mathcal{J}}_t^k)_{k\geq 1}$ satisfy the recursion: $\hat{\mathcal{J}}_t^{k+1}(a_{[t-1]}, x_{[t]}) = E[\sum_{s=t}^{\tau_{\mathbf{B}}-1} r_s(\pi^{\mathbf{B}}_{[s]}) + \hat{\mathcal{J}}_{\tau_{\mathbf{B}}}^k(\pi^{\mathbf{B}}_{[\tau_{\mathbf{B}}-1]})|X_{[t]} = x_{[t]}]$.

For notational simplicity, in the proof we use $\mathcal{J}_t^{k+1}, \tilde{\mathcal{J}}_t^{k+1,\mathbf{B}}, \bar{\mathcal{J}}_t^{k+1,\mathbf{B}}$ and $\hat{\mathcal{J}}_t^{k+1}$ instead of spelling out the dependence on $a_{[t-1]}$ and $x_{[t]}$, assuming causing no ambiguity. Our intended goal is to show

$$\mathcal{J}_t^{k+1} - \hat{\mathcal{J}}_t^{k+1} \leq \frac{k+1}{K+1}\epsilon \times \text{OPT}. \tag{B.6}$$

We rewrite the above as

$$\mathcal{J}_t^{k+1} - \bar{\mathcal{J}}_t^{k+1,\mathbf{B}} + \bar{\mathcal{J}}_t^{k+1,\mathbf{B}} - \tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} + \tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} - \hat{\mathcal{J}}_t^{k+1} \leq \frac{k1}{K+1}\epsilon \times \text{OPT}.$$

To prove inequality (B.6), it suffices to show

$$\mathcal{J}_t^{k+1} - \bar{\mathcal{J}}_t^{k+1,\mathbf{B}} \quad \leq \quad \frac{1}{2(K+1)}\epsilon \times \text{OPT} \tag{B.7}$$

$$|\bar{\mathcal{J}}_t^{k+1,\mathbf{B}} - \tilde{\mathcal{J}}_t^{k+1,\mathbf{B}}| \quad \leq \quad \frac{1}{2(K+1)}\epsilon \times \text{OPT} \tag{B.8}$$

$$|\tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} - \hat{\mathcal{J}}_t^{k+1}| \quad \leq \quad \frac{k}{K+1}\epsilon \times \text{OPT} \tag{B.9}$$

We next prove the above bounds.

- **Proof of bound (B.7).** Recall that for each $a \in \mathcal{A}_t$, $\mathcal{Q}^{k+1}$ calls $\mathcal{W}_a$ with control parameters $(\frac{\alpha\epsilon}{8(K+1)}, \frac{\alpha\epsilon}{8(K+1)})$, which computes a number $\mathbf{Y}^a$ satisfying

  $$P\Big(|\mathbf{Y}^a - \mathcal{J}_t^{k+1,a}| > \frac{\alpha\epsilon}{8(K+1)}\mathcal{L}\Big) \leq \frac{\alpha\epsilon}{8(K+1)}.$$

  Notice that the above inequality is identical to bound (B.5). The rest of the proof is also exactly identical to that in the base case (right after bound (B.5)), and we omit here. As a result, we conclude that $\mathcal{J}_t^{k+1} - \bar{\mathcal{J}}_t^{k+1,\mathbf{B}} \leq \frac{1}{2(K+1)}\epsilon \times \alpha \times \mathcal{L} \leq \frac{1}{2(K+1)}\epsilon \times \text{OPT}$, completing the proof of bound (B.7).

- **Proof of bound (B.8).** Recall that conditional on $a^{\mathbf{B}} = a$, $\mathcal{W}_b$ is called to solve the optimal stopping problem

  $$\mathcal{J}_t^{k+1,a} = \sup_{\tau} E\Big[\sum_{s=t}^{\tau-1} r_s(\pi_{[s]}^a) + \mathcal{J}_\tau^k \Big| X_{[t]} = x_{[t]}\Big]$$

with control parameter $\frac{\alpha\epsilon}{2(K+1)}$. By Lemma 9, the output stopping time $\tau_{\mathbf{B}}$ satisfies

$$\mathcal{J}_t^{k+1,a} - E\Big[\sum_{s=t}^{\tau_{\mathbf{B}}-1} r_s(\pi^a_{[s]}) + \mathcal{J}^k_{\tau_{\mathbf{B}}}(\pi^a_{[\tau_{\mathbf{B}}-1]})\Big| X_{[t]} = x_{[t]}, a^{\mathbf{B}} = a\Big] \leq \frac{\epsilon}{2(K+1)}\alpha\mathcal{L} \leq \frac{\epsilon}{2(K+1)}\text{OPT}.$$

Taking expectation over $\mathbf{B}$, we thus conclude that $\tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} - \tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} \leq \frac{\epsilon}{2(K+1)}\text{OPT}$, completing the proof of bound (B.8).

- **Proof of bound (B.9).** By definition, we have

$$\begin{aligned}
& |\tilde{\mathcal{J}}_t^{k+1,\mathbf{B}} - \hat{\mathcal{J}}_t^{k+1}| \\
= \quad & \Big|E\Big[\sum_{s=t}^{\tau_{\mathbf{B}}-1} r_s(\pi^{\mathbf{B}}_{[s]}) + \hat{\mathcal{J}}^k_{\tau_{\mathbf{B}}}(\pi^{\mathbf{B}}_{[\tau_{\mathbf{B}}-1]})\Big| X_{[t]} = x_{[t]}\Big] - E\Big[\sum_{s=t}^{\tau_{\mathbf{B}}-1} r_s(\pi^{\mathbf{B}}_{[s]}) + \mathcal{J}^k_{\tau_{\mathbf{B}}}(\pi^{\mathbf{B}}_{[\tau_{\mathbf{B}}-1]})\Big| X_{[t]} = x_{[t]}\Big]\Big| \\
= \quad & \Big|E[\hat{\mathcal{J}}^k_{\tau_{\mathbf{B}}} - \mathcal{J}^k_{\tau_{\mathbf{B}}}\Big| X_{[t]} = x_{[t]}]\Big| \\
\leq \quad & E[\,\Big|\hat{\mathcal{J}}^k_{\tau_{\mathbf{B}}} - \mathcal{J}^k_{\tau_{\mathbf{B}}}\Big|\,\Big| X_{[t]} = x_{[t]}] \\
\leq \quad & \frac{k}{K+1}\epsilon \times \text{OPT},
\end{aligned}$$

completing the proof of bound (B.9). Here the last inequality follows from our induction hypothesis.

Combining bounds (B.7), (B.8) and (B.9) proves inequality (B.6) for case $k+1$, which then completes our induction and proves the lemma. □

Proof. Proof of Lemma 12 We begin with the sampling cost. At each period in the $(K-k)$-th epoch, first one call is made to $\mathcal{W}_b$, with control parameter

$\zeta \stackrel{\Delta}{=} \frac{\alpha\epsilon}{2(K+1)}$, whose sampling cost is at most

$$
\begin{aligned}
& \exp(400\zeta^{-2})T^{6\zeta^{-1}} \times h_2(4^{-6\zeta^{-1}}, e^{-400\zeta^{-2}}T^{-6\zeta^{-1}}) \\
& \qquad\qquad \text{(by Lemma 9)} \\
\leq\ & \exp(400\zeta^{-2})T^{6\zeta^{-1}} \times 2g_k(\frac{\alpha}{2}4^{-6\zeta^{-1}}, \frac{1}{2}e^{-400\zeta^{-2}}T^{-6\zeta^{-1}}) \\
& \qquad\qquad \text{(by arguments in the proof of Lemma 10)} \\
=\ & \exp(400\zeta^{-2}) \times T^{6\zeta^{-1}} \times 2(10TM^2)^{kU\left(10^6, \frac{2}{\alpha}4^{6\zeta^{-1}}\right)} \\
& \qquad \times\left(1 + \log(2) + 400\zeta^{-2} + 6\zeta^{-1}\log(T)\right)^{kU\left(10^6, \frac{2}{\alpha}4^{6\zeta^{-1}}\right)} \\
\leq\ & \exp(400\zeta^{-2}) \times T^{6\zeta^{-1}} \times (10TM^2)^{kU\left(10^6, \frac{2}{\alpha}4^{6\zeta^{-1}}\right)} \\
& \qquad \times(10^3\zeta^{-2} \times T)^{kU\left(10^6, \frac{2}{\alpha}4^{6\zeta^{-1}}\right)} \\
\leq\ & \exp(400\zeta^{-2}) \times T^{6\zeta^{-1}} \times (10TM^2)^{kU\left(10^6, 4^{8\zeta^{-1}}\right)} \\
& \qquad \times(10^3\zeta^{-2} \times T)^{kU\left(10^6, 4^{8\zeta^{-1}}\right)} \\
\leq\ & 10^{2\times^{k+1}U\left(10^6, \zeta^{-1}\right)} \times T^{k+1U\left(10^6, \zeta^{-1}\right)} \times M^{k+1U\left(10^6, \zeta^{-1}\right)} \\
& \qquad\qquad \text{(by bounds (B.10) and (B.11) in the proof of Lemma 32)} \\
=\ & (100TM)^{k+1U\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)}.
\end{aligned}
$$

If the output from $\mathcal{W}_b$ is STOP, Q_b then makes a call to Q^{k-1} with control parameters $(\frac{\epsilon}{8(K+1)}, \frac{M\alpha\epsilon}{8(K+1)} \wedge 1)$ (or equivalently $(\frac{\zeta}{4\alpha}, \frac{\zeta M}{4} \wedge 1)$), whose sampling cost is at most

$$
\begin{aligned}
& (10TM^2)^{k-1U\left(10^6, 4\zeta^{-1}\right)} \times (1 + \log(4) + \log(\zeta^{-1})))^{k-1U\left(10^6, 4\zeta^{-1}\right)} \\
& \qquad\qquad \text{(by Lemma 10)} \\
\leq\ & (10TM^2)^{k-1U\left(10^6, 4\zeta^{-1}\right)} \times (2\zeta^{-1})^{k-1U\left(10^6, 4\zeta^{-1}\right)} \\
\leq\ & (10TM^2)^{k-1U\left(10^6, 4^{8\zeta^{-1}}\right)} \times (2\zeta^{-1})^{k-1U\left(10^6, 4^{8\zeta^{-1}}\right)} \\
\leq\ & (10TM^2)^{kU\left(10^6, \zeta^{-1}\right)} \qquad \text{(by bounds (B.10) and (B.11))} \\
\leq\ & (10TM^2)^{kU\left(10^6, \frac{2(K+1)}{\alpha\epsilon}\right)}.
\end{aligned}
$$

In any case, Q_b will take at most $2(100TM)^{k+1}{}^{U\left(10^6,\frac{2(K+1)}{\alpha\epsilon}\right)}$ number of samples from simulator $\int$ to compute an output.

We can account for the computational cost in an almost identical manner, and we here omit the technical details. Combining the above then prove the lemma. □

B.3 Proofs of auxiliary results

This section consists of detailed proofs Lemma 32.

Proof of Lemma 32. Recall that

$$\begin{aligned} g^k(\epsilon,\delta) &= (10TM^2)^{k}{}^{U(10^6,\frac{1}{\epsilon})}\left(1+\log\left(\frac{1}{\delta}\right)\right)^{k}{}^{U(10^6,\frac{1}{\epsilon})} \\ f^k(\epsilon,\delta) &= C(10TM^2)^{k}{}^{U(10^6,\frac{1}{\epsilon})}\left(1+\log\left(\frac{1}{\delta}\right)\right)^{k}{}^{U(10^6,\frac{1}{\epsilon})}. \end{aligned}$$

The fact that both g^k and f^k are monotone in ϵ and δ is straightforward and we omit the proof.

Next let's show the bounds on g^0 and f^0. Indeed, $g^0(\epsilon,\delta) = 10^{\epsilon^{-1}}(TM^2)^{\epsilon^{-1}}(1+\log(\frac{1}{\delta}))^{\epsilon^{-1}} \geq 10^{\epsilon^{-1}}TM^2\log(\frac{e}{\delta}) \geq 10^{\epsilon^{-1}}TM\log(\frac{eM}{\delta}) \geq 2\epsilon^{-2}TM\log(\frac{2M}{\delta})$. Similarly, $f^0(\epsilon,\delta) = C10^{\epsilon^{-1}}(TM^2)^{\epsilon^{-1}}(1+\log(\frac{1}{\delta}))^{\epsilon^{-1}} \geq C10^{\epsilon^{-1}}TM\log(\frac{eM}{\delta}) \geq 3CMT\epsilon^{-2}\log(\frac{2M}{\delta})$.

Finally, let's prove that g^k and f^k satisfy the recursive inequalities. Let's first state and prove several properties of ${}^kU$.

$$\begin{aligned} {}^kU(10^6,10^{\frac{6}{\epsilon}}) &\geq 2\times{}^kU(10^6,4^{\frac{8}{\epsilon}})+\frac{6}{\epsilon} && \text{(B.10)} \\ 10^{{}^kU(10^6,10^{\frac{6}{\epsilon}})} &\geq 10^{{}^kU(10^6,4^{\frac{8}{\epsilon}})}\times\left(400\epsilon^{-2}\right)^{{}^kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2}) && \text{(B.11)} \end{aligned}$$

The above can be shown via induction. Indeed, in the base case $k=0$, inequality (B.10) becomes $10^{6\epsilon^{-1}} \geq 2\times4^{8\epsilon^{-1}}+6\epsilon^{-1}$ which can be easily verified for all $\epsilon\in(0,1)$.

Inequality (B.11) becomes

$$10^{10^{\frac{6}{\epsilon}}} \geq 10^{4^{\frac{8}{\epsilon}}} \times (400\epsilon^{-2})^{4^{\frac{8}{\epsilon}}} \times \exp(200\epsilon^{-2}).$$

Similarly, it can be verified directly that the above holds for all $\epsilon \in (0, 1)$. Now suppose inequalities (B.10) and (B.11) hold true for some $k \geq 0$, we prove it also holds in case $k + 1$. Indeed,

$$\begin{aligned}
{}^{k+1}U(10^6, 10^{\frac{6}{\epsilon}}) &= 10^{6\times {}^{k}U(10^6, 10^{\frac{6}{\epsilon}})} \\
&\geq 10^{12\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})+6\times\frac{6}{\epsilon}} \\
&= \left({}^{k+1}U(10^6, 4^{\frac{8}{\epsilon}})\right)^2 \times 10^{\frac{36}{\epsilon}} \\
&\geq 2\times {}^{k+1}U(10^6, 4^{\frac{8}{\epsilon}}) + \frac{6}{\epsilon}.
\end{aligned}$$

$$\begin{aligned}
10^{{}^{k+1}U(10^6, 10^{\frac{6}{\epsilon}})} &= \exp(\log(10) \times 10^{6\times {}^{k}U(10^6, 10^{\frac{6}{\epsilon}})}) \\
&\geq \exp\left(\log(10) \times 10^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})} \times (400\epsilon^{-2})^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})} \times \exp(1200\epsilon^{-2})\right) \\
&\geq \exp\left(\log(10) \times \left(10^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})} + (400\epsilon^{-2})^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})} + \exp(1200\epsilon^{-2})\right)\right) \\
&= 10^{10^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})}} \times 10^{\left(400\epsilon^{-2}\right)^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})}} \times 10^{\exp(1200\epsilon^{-2})} \\
&\geq 10^{10^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})}} \times (400\epsilon^{-2})^{10^{6\times {}^{k}U(10^6, 4^{\frac{8}{\epsilon}})}} \times \exp(200\epsilon^{-2}) \\
&= 10^{{}^{k+1}U(10^6, 4^{\frac{8}{\epsilon}})} \times (400\epsilon^{-2})^{{}^{k+1}U(10^6, 4^{\frac{8}{\epsilon}})} \times \exp(200\epsilon^{-2}).
\end{aligned}$$

Inequalities (B.10) and (B.11) thus follow from induction. With these bounds in

hand, we can now prove the recursive bounds on g^k and f^k. For g^k we have

$$
\begin{aligned}
g^{k+1}(\epsilon,\delta) &= (10TM^2)^{k+1U(10^6,\frac{1}{\epsilon})}\Big(1+\log(\frac{1}{\delta})\Big)^{k+1U(10^6,\frac{1}{\epsilon})} \\
&= (10TM^2)^{kU(10^6,10^{\frac{6}{\epsilon}})}\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,10^{\frac{6}{\epsilon}})} \\
&\geq 10^{kU(10^6,4^{\frac{8}{\epsilon}})}\times(400\epsilon^{-2})^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2}) \\
&\quad \times(TM^2)^{2\times kU(10^6,4^{\frac{8}{\epsilon}})+\frac{6}{\epsilon}}\times\Big(1+\log(\frac{1}{\delta})\Big)^{2\times kU(10^6,4^{\frac{8}{\epsilon}})+6\epsilon^{-1}} \\
&\geq (10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times T^{kU(10^6,4^{\frac{8}{\epsilon}})}\times M^{2\times kU(10^6,4^{\frac{8}{\epsilon}})}\times(400\epsilon^{-2})^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\Big(1+\log(\frac{1}{\delta})\Big)^{2\times kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2})\times T^{6\epsilon^{-1}}\times M^{6\epsilon^{-1}}\times(1+\log(\frac{1}{\delta}))^{6\epsilon^{-1}} \\
&\geq (10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times T^{kU(10^6,4^{\frac{8}{\epsilon}})}\times M^{1+kU(10^6,4^{\frac{8}{\epsilon}})}\times(400\epsilon^{-2})^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2})\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 2(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\Big((1+\log(\frac{1}{\delta}))\times T\times M\times 200\epsilon^{-2}\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 2(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\Big(6\epsilon^{-1}(1+\log(\frac{1}{\delta}))+6\epsilon^{-1}M+6\epsilon^{-1}T+200\epsilon^{-2}\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 2(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\Big(3\epsilon^{-1}(1+\log(\frac{M}{\delta}))+\log(2)+6\epsilon^{-1}\log(T)+200\epsilon^{-2}\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&= 2(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\Big(1+\log\Big(2\frac{M}{\delta}T^{6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{3\epsilon^{-1}}\exp(200\epsilon^{-2})\Big)\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&= 2g^k\Big(4^{-8\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big) \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 2g^k\Big(\frac{\epsilon}{2}4^{-6\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big) \\
&\quad \times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}}.
\end{aligned}
$$

For f^k, similarly we have

$$
\begin{aligned}
f^{k+1}(\epsilon,\delta) &= C(10TM^2)^{k+1U(10^6,\frac{1}{\epsilon})}\Big(1+\log(\frac{1}{\delta})\Big)^{k+1U(10^6,\frac{1}{\epsilon})} \\
&\geq C\times(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times T^{kU(10^6,4^{\frac{8}{\epsilon}})}\times M^{1+kU(10^6,4^{\frac{8}{\epsilon}})}\times(400\epsilon^{-2})^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad\times\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2})\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 4C\times(10TM^2)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times T^{kU(10^6,4^{\frac{8}{\epsilon}})}\times M^{1+kU(10^6,4^{\frac{8}{\epsilon}})}\times(200\epsilon^{-2})^{kU(10^6,4^{\frac{8}{\epsilon}})} \\
&\quad\times\Big(1+\log(\frac{1}{\delta})\Big)^{kU(10^6,4^{\frac{8}{\epsilon}})}\times\exp(200\epsilon^{-2})\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 4C\times g^k\Big(4^{-8\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big) \\
&\quad\times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&= 2\Big(C\times g^k\Big(4^{-8\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big) \\
&\qquad+f^k\Big(4^{-8\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big)\Big) \\
&\quad\times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}} \\
&\geq 2\Big(C\times g^k\Big(\frac{\epsilon}{2}4^{-6\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big) \\
&\qquad+f^k\Big(\frac{\epsilon}{2}4^{-6\epsilon^{-1}},\frac{1}{2}\delta M^{-1}T^{-6\epsilon^{-1}}(1+\log(\frac{M}{\delta}))^{-3\epsilon^{-1}}\exp(-200\epsilon^{-2})\Big)\Big) \\
&\quad\times\exp(200\epsilon^{-2})\times M\times T^{6\epsilon^{-1}}\times(1+\log(\frac{M}{\delta}))^{6\epsilon^{-1}}.
\end{aligned}
$$

Combining the above then completes the proof. □

APPENDIX C

CHAPTER 4 OF APPENDIX

The chapter contains all technical proofs of Chapter 4.

C.1 Max-flow reduction and proof of Lemma 13

Proof of Proposition 2. We prove by construction. We create a subset $\mathcal{L}^{\mathbf{x}} \subset V^{\mathbf{x}}$ for all nodes $\omega_i \in \mathcal{H}_i$ with $i \in \mathcal{L}$. We also create a subset $\mathcal{R}^{\mathbf{x}} \subset V^{\mathbf{x}}$ for all nodes $\omega_j \in \mathcal{H}_j$ with $j \in \mathcal{R}$. By definition, the only existing arcs in $E^{\mathbf{x}}$ either points from s to $\mathcal{L}^{\mathbf{x}}$, or from $\mathcal{L}^{\mathbf{x}}$ to $\mathcal{R}^{\mathbf{x}}$, or from $\mathcal{R}^{\mathbf{x}}$ to t. Hence any s-t path in $G^{\mathbf{x}}$ is of length 3. Therefore the graph is of depth 3. □

Proof of Lemma 13. Since our ONLINE MWIS problem is finite-sized with bounded total reward, its optimal value must exist and the deterministic optimal policy is well-defined (possibly non unique). We omit further discussion here. To prove ONLINE MWIS can be reduced to MAX FLOW, we first show that the ONLINE MWIS associated with $G, \mathbf{x}$ and c can be reduced to (offline) MWIS. We write down the integer programming (IP) formulation of ONLINE MWIS.

$$
\begin{aligned}
\text{(ONLINE MWIS IP)} \qquad \max \quad & \sum_{\omega \in \mathcal{H}_T} \sum_{i=1}^{T} p(\omega) \times y_i(\omega[1,i]) \times c(\omega[1,i]) \\
s.t. \quad & y_i(\omega[1,i]) + y_j(\omega[1,j]) \leq 1, \quad \forall (i,j) \in E, \ \forall\, \omega \in \mathcal{H}_T \\
& y_i(\omega[1,i]) \in \{0,1\}. \quad \forall i \in V, \ \forall\, \omega \in \mathcal{H}_T
\end{aligned}
$$

Any feasible solution y to the above IP corresponds to an admissible policy π_y for ONLINE MWIS. The mapping is simply: $y_i(\omega) = 1 \rightarrow \pi_y(\omega) =$ PICK i and $y_i(\omega) = 0 \rightarrow \pi_y(\omega) =$ SKIP i. The solution and the policy incur the same objective

value. Furthermore, the mapping is invertible /one on one, hence solving ONLINE MWIS is equivalent to solving ONLINE MWIS IP.

Now consider an offline MWIS instance with graph $G^{\mathbf{x}}/\{s,t\}$ and rewards $u(\omega) \triangleq c(\omega) \times p(\omega)$ for each $\omega \in G^{\mathbf{x}}/\{s,t\}$. By Proposition 2, $G^{\mathbf{x}}/\{s,t\}$ is bipartite. We claim that ONLINE MWIS IP as presented above solves this offline MWIS instance. Indeed, any feasible solution y to ONLINE MWIS IP corresponds to a feasible solution to the offline MWIS instance. The one-on-one mapping is as follows: $y_i(\omega) = 1 \rightarrow$ PICK ω and $y_i(\omega) = 0 \rightarrow$ SKIP ω. Again, the two solutions incur the same objective values. Hence, solving ONLINE MWIS with $G, \mathbf{x}, c$ is equivalent to solving MWIS with $G^{\mathbf{x}}/\{s,t\}$ and u.

The reduction from (offline bipartite) MWIS with $G^{\mathbf{x}}/\{s,t\}$ and u to MAX FLOW $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ is classical. In words, the complement of the min cut in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ forms the max weight independent set in $G^{\mathbf{x}}/\{s,t\}$ with capacity u. We omit a detailed discussion, instead referring the reader to standard textbooks such as [211]. Hence we complete the proof. □

C.2 Path subnetwork and proof of Proposition 3

We denote by $f^*_{P,u,a,b}$ the maximum flow in (P,a,b,u)-subnetwork, and $M^*_{P,u,a,b}$ its flow value. We prove a stronger result, which incorporates Proposition 3 as its direct corollary.

Proposition 10. *Suppose $P = (i_1, ..., i_k)$ with $k \geq 3$ and $i_1, i_k < \min\{i_2, ..., i_{k-1}\}$. Let $j : i_j = \min\{i_2, ..., i_{k-1}\}$. Denote by $P_1 \triangleq (i_1, ..., i_j)$ and $P_2 \triangleq (i_j, ..., i_k)$ the two segments of P. Then the maximum flow value in the $(P, \omega_1, \omega_k, u)$-subnetwork satisfies the following.*

$$M^*_{P,u,\omega_1,\omega_k} = \sum_{\omega \in \mathcal{H}_{i_j} : \omega_1, \omega_k \subset \omega} \min\left(M^*_{P_1,u,\omega_1,\omega}, M^*_{P_2,u,\omega,\omega_k}\right).$$

The max flow in the $(P, \omega_1, \omega_k, u)$-subnetwork satisfies the following.

$$f^*_{P,u,\omega_1,\omega_k} = \sum_{\omega \in \mathcal{H}_{i_j} : \omega_1, \omega_k \subset \omega} \left(f^*_{P_1,u,\omega_1,\omega} \times \frac{M^*_{P,u,\omega_1,\omega_k}}{M^*_{P_1,u,\omega_1,\omega}} + f^*_{P_2,u,\omega,\omega_k} \times \frac{M^*_{P,u,\omega_1,\omega_k}}{M^*_{P_2,u,\omega,\omega_k}} \right)$$

Proof. We first prove the recursion holds for M^*. Consider any two $\omega, \omega' \in \mathcal{H}_{i_j}$ such that $\omega_1, \omega_k \subset \omega, \omega'$. For any P-sample Q_1 conditional on $\omega \in Q_1$ and any P-sample Q_2 conditional on $\omega' \in Q_2$, we observe that Q_1 and Q_2 meet only at source s and sink t. Such structure motivates us to consider partitioning the original $(P, \omega_{i_1}, \omega_{i_k}, u)$-subnetwork into smaller subnetworks. More precisely, we take all $(P, \omega_{i_1}, \omega_{i_k})$-samples that contain ω to form a subgraph G_ω. Then the original path subgraph equals to $\cup_{\omega \in \mathcal{H}_{i_j} : \omega_1, \omega_k \subset \omega} G_\omega$, and for any $\omega, \omega', G_\omega \cap G_{\omega'} = \{s, t\}$. Therefore, the max flow value in $(P, \omega_1, \omega_k, u)$-subnetwork equals the sum of the max flow values in these subnetworks G_ω with capacity u. Now let's zoom in on G_ω with a fixed ω. All s-t paths in G_ω pass through ω, and that G_ω can be further decomposed into two subnetworks, $(P_1, \omega_1, \omega, u)$-subnetwork and $(P_2, \omega, \omega_k, u)$-subnetwork. Hence ω is the bottleneck, and the max flow value in $\{G_\omega, s, t, u\}$ is the minimum between the max flow value into and out of ω, which are the max flow value in $(P_1, \omega_1, \omega, u)$-subnetwork and $(P_2, \omega, \omega_k, u)$-subnetwork, respectively. Combining all the above, we prove the recursion for M^*, the max flow value, is valid.

We next prove the recursion holds for f^*. Indeed, by our previous argument, the flows on different G_ω are irrelevant because they share no common nodes other than s, t. For each ω, suppose we have the max flow $f^*_{P_1,\omega_1,\omega,u}$ and $f^*_{P_2,\omega,\omega_k,u}$ in the two corresponding subnetworks. We keep the flow unchanged in the subnetwork with smaller max flow values. We scale the flow by the ratio between the two max flow values in the other subnetwork. We then piece the two flows together to get a flow on G_ω. We can see that, the resulting flow in $\{G_\omega, u\}$ (1) is

non negative, because we piece together two non negative flows and scale one of them by a positive constant; (2) is feasible, because we only ever decreases the flow on any arc; (3) remains a flow, because the flow is a combination of two flows, conservation at all nodes except ω still hold. As for ω, our rescaling makes the flow into and out of ω balanced, so the conservation rule also holds; and (4) is optimal, because by the previous argument, the max flow value in $\{G_\omega, u\}$ is the smaller of the two max flow values of $\{P_1, \omega_1, \omega, u\}$-subnetwork and $\{P_2, \omega, \omega_k, u\}$-subnetwork, which is achieved by our proposed flow.

Combining all the above, we have proved Proposition 10. □

With Proposition 10, we complete our proof of Proposition 3.

Proof of Proposition 3. We prove a stronger result. For any $k \geq 2$ and any $P = (i_1, ..., .i_k) \in G$ that are regular and satisfying $i_1, i_k < \min(i_2, ..., i_{k-1})$, Algorithm 1′ can output the max flow and its value in any $(P, \omega_1, \omega_k, u)$-subnetwork. We argue by induction on k. In the base case $k = 2$, the path subnetwork is simply an arc, and Algorithm 1′ trivially returns the true max flow and its value (both equal to the capacity). Now suppose for all path of length $\leq k-1$, Algorithm 1′ can output the correct max flow and value for path networks. Then case k follows directly from Proposition 10. By induction, we conclude that the statement holds true. The statement of Proposition 3 is the above statement restricted to (P, s, t, u)-subnetworks. Therefore we complete the proof. □

C.3 Blocking flow and proof of Proposition 4

We first state and prove two intermediate results, Lemma 33 and Lemma 34.

Lemma 33. *Algorithm 2′ maintains a regular, feasible flow in $\mathcal{N}_{f_0}$.*

Proof of Lemma 33. We first prove the flow f' remains regular. It suffices to show that for any arc (ω_i, ω_j) and its reverse arc (ω_j, ω_i), either $f'(\omega_i, \omega_j) = 0$ or $f'(\omega_j, \omega_i) = 0$. First, if either of the two nodes is s or t, then the statement holds true because by definition, f' adds a linear combination of regular flows (as output of algorithm 1′) in each iteration, none of which ever uses arcs of the form (t, ω) or (ω, s). Now suppose both ω_i and ω_j are not source s and sink t. We argue by contradiction. If at some point during the iteration, f' uses both arcs, i.e. $f'(\omega_j, \omega_i) > 0$ and $f'(\omega_i, \omega_j) > 0$, then by the updating rule of f' and the definition of flow, there must exist two connected s-t paths P_1 and P_2, both of length $(l + 2)$, such that $(\omega_i, \omega_j) \in P_1$ and $(\omega_j, \omega_i) \in P_2$. Since the update only increases the flows/decreases the capacities on arcs, we conclude that P_1 and P_2 must have been connected at the beginning of the algorithm, namely in residual network $\mathcal{N}_{f_0}$. Since f_0 is l-connecting for an odd l, the length of the shortest s-t path is $(l + 2)$ in $\mathcal{N}_{f_0}$. Hence both P_1 and P_2 are shortest s-t path, which implies $d(\omega_i) = d(\omega_j) + 1$ and $d(\omega_j) = d(\omega_i) + 1$ simultaneously, a contradiction. Therefore, f' is always regular.

Next we prove it's also feasible. Let's focus on an arbitrary arc (ω_i, ω_j), and show that $f'(\omega_i, \omega_j)$ never exceeds the capacity $c^{\mathbf{x}}_{f_0}(\omega_i, \omega_j)$. In a fixed iteration round, the amount of flow that is been pushed through (ω_i, ω_j) is $\frac{1}{\Delta^l} \sum_{P \in \mathcal{M}_{l+2}} f_P(\omega_i, \omega_j)$. Since f_P is the outputting flow of Algorithm 1′ with input $\{P, s, t, u\}$, it must hold that $f_P(\omega_i, \omega_j) < u(\omega_i, \omega_j)$ for all P. Also, the number of $P \in \mathcal{M}_{l+2}$ such that $f_P(\omega_i, \omega_j) \neq 0$ is at most Δ^l (because Δ is the max degree and the free vertices on the path is at most l), we conclude that $\frac{1}{\Delta^l} \sum_{P \in \mathcal{M}_{l+2}} f_P(\omega_i, \omega_j) < u(\omega_i, \omega_j)$, and the flow remains feasible between iterations. Combining the above completes the proof. □

Lemma 34. *Algorithm 2′ outputs a flow that saturates all s-t paths with length* $(l+2)$ *in* $\mathcal{N}_{f_0}$.

Proof of Lemma 34. Since Algorithm 2′ only increases flows on all arcs, and the flow is always regular and feasible in $\mathcal{N}_{f_0}$(by Lemma 33), we conclude by monotone convergence theorem that Algorithm 2′ ultimately returns a flow f' that is regular and feasible in $\mathcal{N}_{f_0}$, and should be invariant under one iteration of Algorithm 2′. Now suppose by contradiction, that there exists an s-t path $\mathcal{P} \in \mathcal{N}_{f_0}$ that are with length $(l+2)$ and not saturated by flow f'. Let $u = c^{\mathbf{x}}_{f_0} - f'$, then the capacities u are strictly positive on path $\mathcal{P}$. Consider the corresponding path in $G : P = \phi(\mathcal{P})$. The path subnetwork associated with (P, s, t, u) must have strictly positive max flow value M_P, because a positive amount of flow can be pushed through path $\mathcal{P}$. Hence after one iteration, the flow value of f' is increased by at least $\frac{1}{\Delta^l} M_P$, a strictly positive number, which contradicts the fact that f' is the limiting flow. Therefore f' must saturate all length $(l+2)$ s-t paths in residual graph $\mathcal{N}_{f_0}$. □

Using Lemma 33 and Lemma 34, we prove Proposition 3.

Proof of Proposition 3. If f_0 is already $(l+2)$-disconnecting, then algorithm 2′ outputs a zero flow, and the statement follows. Otherwise, there exists length $(l+2)$ s-t paths in $\mathcal{N}_{f_0}$. By Proposition 14, it suffices to prove that f' is blocking in the level subnetwork of $\mathcal{N}_{f_0}$. By definition, all connected s-t paths in the level subnetwork are shortest paths between s and t, which are of length $(l+2)$. By Lemma 34, f' saturates all such paths in $\mathcal{N}_{f_0}$, thereby saturating all s-t paths in its level subnetwork, completing the proof. □

C.4 Implementable algorithms and analysis

C.4.1 Boundedness of rescaling and proof of Lemma 19

Proof of Lemma 19. We first prove $\mathbf{z}^{i,j}$ is bounded in $[0,1]$. Indeed, there are four cases. In case one and case two, either $i = -1$ or $j = -2$. The associated arc samples e are of the form (s,ω) or (ω,t). In such cases, $\mathbf{z}^{i,j}(e) = f(e)/p(\omega)$. By definition and boundedness constraints on reward c, we have $0 \leq f(e) \leq c^{\mathbf{x}}(e) = c(\omega)\times p(\omega) \leq p(\omega)$. Combining the above, we have $\mathbf{z}^{i,j}(e) \leq 1$ for all e, completing the proof. In case three, $i \in \mathcal{R}$ and $j \in \mathcal{L}$, (or equivalently $i = -2, j = -1$) the capacities are 0 on these reverse arcs in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. Hence for any arc sample e, $0 \leq \mathbf{z}^{i,j}(e) \leq c^{\mathbf{x}}(e) = 0$, completing the proof. Finally, we consider case four $i \in \mathcal{L}$ and $j \in \mathcal{R}$. In such cases, capacities are ∞ and we use flow conservation to prove the desired result. Indeed, without loss of generality we assume $i > j$. For any arc sample $e = (\omega_i, \omega_j)$, we focus on node ω_i. There is one arc carries positive flow to ω_i, that is (s,ω_i), and possibly multiple arcs, including (ω_i,ω_j), transport flows away from ω_i. Therefore, by flow conservation, it must be the case that $f(s,\omega_i) \geq f(\omega_i,\omega_j)$. As a result, we conclude that $\mathbf{z}^{i,j}(\omega_i,\omega_j) \leq \mathbf{z}^{-1,i}(s,\omega_i) \leq 1$, where the last inequality follows from case one. Combining the above cases, we conclude that random variables $\mathbf{z}^{i,j} \in [0,1],\ w.p.1$ for any $(i,j) \in E$.

We proceed to show that the induced residual capacities also satisfy the desired boundedness property. Again we consider four cases. In case one and two, either $i = -1$ or $j = -2$, arc sample e is of the form (s,ω) or (ω,t). In such cases, by definition, we have $\mathbf{w}^{i,j}(e) = c^{\mathbf{x}}(e)/p(\omega) - \mathbf{z}^{i,j}(e) = c(\omega) - \mathbf{z}^{i,j}(e)$. By the fact that both $c(\omega)$ and $\mathbf{z}^{i,j}(e)$ are in $[0,1]$ and that $c(\omega) \geq \mathbf{z}^{i,j}(e)$ (feasibility), we conclude that $\mathbf{w}^{i,j}(e) \in [0,1]$. In case three, we consider all reverse arcs. For any such arc

sample e, we denote by e' its reverse arc sample. Then by definition, we have $\mathbf{w}^{i,j}(e) = \mathbf{z}^{j,i}(e')$. With the previous argument we thus conclude $\mathbf{w}^{i,j}(e) \in [0, 1]$. Finally, in case four we have $i \in \mathcal{L}$ and $j \in \mathcal{R}$, any arc sample e in such case has capacity $c^{\mathbf{x}}(e) = \infty$, therefore subtracting any feasible flow and rescaling by any finite constant won't change the capacity, namely $\mathbf{w}^{i,j}(e) = \infty$. Combining all above, we prove the Lemma. □

C.4.2 Approximate max flow in path networks and proof of Proposition 5

Proof of Lemma 20. By definition, a path subnetwork defined by a single arc (ω_i, ω_j) is itself. Hence the max flow through the network equals the capacity on the arc. □

Proof of Lemma 21. Since both i_1 and i_k are smaller than i_j, we conclude that both $\mathbf{y}_{P_1}$ and $\mathbf{y}_{P_2}$ are $\mathcal{F}_{i_j}$-measurable random variables. Hence by definition, for any $\omega_1 \in \mathcal{H}_{i_1}$ and $\omega_k \in \mathcal{H}_{i_k}$ such that $\omega_1 \subset \omega_k$ or $\omega_k \subset \omega_1$, and any $\omega \in \mathcal{H}_{i_j}$ such that $\omega_1, \omega_k \subset \omega$, the following holds true: $\min(\mathbf{y}_{P_1}(\omega), \mathbf{y}_{P_2}(\omega)) = \min\left(M^*_{P_1,u,\omega_1,\omega}, M^*_{P_2,u,\omega,\omega_k}\right)/p(\omega)$, where we recall that M^* denotes the max value of

flow in the associated path subnetwork. Therefore

$$
\begin{aligned}
&E\Big[\min(\mathbf{y}_{P_1}, \mathbf{y}_{P_2})\Big|\mathcal{F}_{i_1\vee i_k}\Big](\omega_1 \vee \omega_k) \\
&= \sum_{\omega\in\mathcal{H}_{i_j}:\omega_{i_1\vee i_k}\subset\omega} \min(\mathbf{y}_{P_1}(\omega), \mathbf{y}_{P_2}(\omega))P(\mathbf{x}_{i_j} = \omega\Big|\mathbf{x}_{i_1\vee i_k} = \omega_1 \vee \omega_k) \\
&= \sum_{\omega\in\mathcal{H}_{i_j}:\omega_1,\omega_k\subset\omega} \min(\mathbf{y}_{P_1}(\omega), \mathbf{y}_{P_2}(\omega))\frac{p(\omega)}{\min(p(\omega_1), p(\omega_k))}. \\
&= \frac{1}{\min(p(\omega_1), p(\omega_k))} \sum_{\omega\in\mathcal{H}_{i_j}:\omega_1,\omega_k\subset\omega} \min\left(M^*_{P_1,u,\omega_1,\omega}, M^*_{P_2,u,\omega,\omega_k}\right) \\
&= \frac{1}{\min(p(\omega_1), p(\omega_k))} M^*_{P,u,\omega_1,\omega_k} = \mathbf{y}_P(\omega_1 \vee \omega_k),
\end{aligned}
$$

where the second to last equality follows from the validity of Algorithm 1′ and Proposition 3. □

Proof of Proposition 5. We first prove that the desired approximation guarantee can be achieved by Algorithm 1. For part *a*, the algorithm calls itself with inputs (P_1, ω^i_j) and (P_2, ω^i_j), and outputs $y^i_{P_1}$ and $y^i_{P_2}$, where $(\omega^i_j)_{i\in[1,N]}$ are N independent samples in $\mathcal{H}_{i_j}$ drawn from $\mathcal{S}$ conditional on $\mathbf{x}_{i_1\vee i_k} = \omega$. We pick the appropriate parameters such that for each i, $y^i_{P_1}$ and $y^i_{P_2}$ are $\frac{\epsilon}{16}$-approximations of $\mathbf{y}_{P_1}(\omega^i_j)$ and $\mathbf{y}_{P_2}(\omega^i_j)$ with probability at least $1 - \epsilon/16$, respectively. We pick parameter $N = 8\epsilon^{-2}\log(4/\delta)$. Now using Lemma 21, we have that

$$
\begin{aligned}
\left|\frac{1}{N}\sum_{i=1}^N \min(y^i_{P_1}, y^i_{P_2}) - \mathbf{y}_P(\omega)\right| &= \left|\frac{1}{N}\sum_{i=1}^N \min(y^i_{P_1}, y^i_{P_2}) - E[\min(\mathbf{y}_{P_1}, \mathbf{y}_{P_2})\Big|\mathbf{x}_{i_1\vee i_k} = \omega]\right| \\
&\le \frac{1}{N}\sum_{i=1}^N \left|\min(y^i_{P_1}, y^i_{P_2}) - \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j))\right| \qquad \text{(C.1)} \\
&\quad + \left|\frac{1}{N}\sum_{i=1}^N \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j)) - E[\min(\mathbf{y}_{P_1}, \mathbf{y}_{P_2})\Big|\mathbf{x}_{i_1\vee i_k} = \omega]\right| \qquad \text{(C.2)}
\end{aligned}
$$

Let $(\mathcal{U}_i)_{i=1,..,N}$ denote N i.i.d. random variables taking binary values: $\mathcal{U}_i = 1$ *w.p.* $\epsilon/8$; $= \epsilon/8$ *w.p.* $(1 - \epsilon/8)$. For each i, by our guarantee on $y^i_{P_1}, y^i_{P_2}$ and a union bound, we have that $\left|\min(y^i_{P_1}, y^i_{P_2}) - \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j))\right|$ is with

probability $1 - \frac{\epsilon}{8}$ bounded by $\epsilon/8$. Combining with the fact that $\left|\min(y^i_{P_1}, y^i_{P_2}) - \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j))\right|$ never exceeds 1 by Lemma 19, we thus conclude that $\left|\min(y^i_{P_1}, y^i_{P_2}) - \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2})(\omega^i_j)\right|$ is stochastically dominated by $\mathcal{U}_i$, For $k \neq i$, $y^k_{P_1}, y^k_{P_2}$ and $\mathbf{y}_{P_1}(\omega^k_j), \mathbf{y}_{P_2}(\omega^k_j)$ are independent of the the corresponding random variables indexed by i. Hence by the basic property of stochastic ordering, we can bound (C.1) by

$$\begin{aligned}
\mathbb{P}\Big(\frac{1}{N}\sum_{i=1}^{N}\Big|\min(y^i_{P_1}, y^i_{P_2}) - \min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j))\Big| > \frac{\epsilon}{2}\Big) &\leq \mathbb{P}\Big(\frac{1}{N}\sum_{i=1}^{N}\mathcal{U}_i > \frac{\epsilon}{2}\Big) \\
&\leq \mathbb{P}\Big(\Big|\frac{1}{N}\sum_{i=1}^{N}\mathcal{U}_i - E[\mathcal{U}]\Big| > \frac{\epsilon}{2} - E[\mathcal{U}]\Big) \\
&\leq \mathbb{P}\Big(\Big|\frac{1}{N}\sum_{i=1}^{N}\mathcal{U}_i - E[\mathcal{U}]\Big| > \frac{\epsilon}{2} - \frac{\epsilon}{4}\Big) \leq 2\exp\Big(-2(\frac{\epsilon}{4})^2 N\Big) \leq \frac{\delta}{2}.
\end{aligned}$$

where the second to last inequality follows from Hoeffding's inequality (Lemma 22).

We proceed to bound (C.2) by Lemma 22 as

$$\mathbb{P}\Big(\Big|\frac{1}{N}\sum_{i=1}^{N}\min(\mathbf{y}_{P_1}(\omega^i_j), \mathbf{y}_{P_2}(\omega^i_j)) - E[\min(\mathbf{y}_{P_1}, \mathbf{y}_{P_2})|\mathbf{x}_{i_1 \vee i_k} = \omega]\Big| > \frac{\epsilon}{2}\Big) \leq 2\exp\Big(-2(\frac{\epsilon}{2})^2 N\Big) \leq \frac{\delta}{2}.$$

Combining the two bounds with another union bound, we conclude that $\mathbb{P}(|y - \mathbf{y}_P(\omega)| > \epsilon) \leq \delta$, confirming that the high probability approximation guarantee is achieved by Algorithm 1a.

We proceed to prove the performance guarantee is achieved by part b of the algorithm. For any given $(i_q, i_{q+1}) \in P$, algorithm 1b calls Algorithm 1a with inputs (P_1, ω_j) and (P_2, ω_j) and outputs y_{P_1} and y_{P_2}. Each with probability at least $1 - \delta/8$, y_{P_1} and y_{P_2} are $\epsilon\delta/16$-approximations of $\mathbf{y}_{P_1}$ and $\mathbf{y}_{P_2}$. Algorithm 1b then makes a call to itself with inputs (P_i, e), and outputs a z' that is with probability

$1-\delta/2$, an $\epsilon/2$-approximation of $\mathbf{z}_{P_i}^{i_q,i_{q+1}}$. We thus have

$$\begin{aligned}
\left|z'\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}-\mathbf{z}_P^{i_q,i_{q+1}}\right| &= \left|z'\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\frac{\min(\mathbf{y}_{P_1},\mathbf{y}_{P_1})}{\mathbf{y}_{P_i}}\right| \\
&\leq \left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|\times\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}+\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\times\left|\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}-\frac{\min(\mathbf{y}_{P_1},\mathbf{y}_{P_2})}{\mathbf{y}_{P_i}}\right| \\
&\leq \left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|+\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\left|\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}-\frac{\min(\mathbf{y}_{P_1},\mathbf{y}_{P_2})}{\mathbf{y}_{P_i}}\right| \\
&\leq \left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|+\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\left(\left|\frac{\min(y_{P_1},y_{P_2})}{\mathbf{y}_{P_i}}-\frac{\min(\mathbf{y}_{P_1},\mathbf{y}_{P_2})}{\mathbf{y}_{P_i}}\right|+\left|\frac{\min(y_{P_1},y_{P_2})}{\mathbf{y}_{P_i}}-\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}\right|\right) \\
&= \left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|+\frac{\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)}{\mathbf{y}_{P_i}}\left(\left|\min(y_{P_1},y_{P_2})-\min(\mathbf{y}_{P_1},\mathbf{y}_{P_2})\right|+\frac{\min(y_{P_1},y_{P_2})}{y_{P_i}}\left|y_{P_2}-\mathbf{y}_{P_2}\right|\right) \\
&\leq \left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|+2\frac{\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)}{\mathbf{y}_{P_i}}\max\left(\left|y_{P_2}-\mathbf{y}_{P_2}\right|,\left|y_{P_1}-\mathbf{y}_{P_1}\right|\right)
\end{aligned}$$

We have $\mathbb{P}(\left|z'-\mathbf{z}_{P_i}^{i_q,i_{q+1}}(e)\right|\geq\epsilon/2)\leq\delta/2$. By a union bound, $\mathbb{P}(\max(\left|y_{P_2}-\mathbf{y}_{P_2}\right|,\left|y_{P_1}-\mathbf{y}_{P_1}\right|)\geq\delta\epsilon/16)\leq\delta/4$. We observe that $E[\mathbf{z}_{P_i}^{i_q,i_{q+1}}|\mathcal{F}_{i_j}]=\mathbf{y}_{P_i}$, because the arc samples of (i_q,i_{q+1}) form a valid cut of the P_i-subnetwork, and the flow value $\mathbf{y}_{P_i}$ must equal the total flow value through the cut, which is the conditional expectation. Now with a Markov inequality, we may conclude that

$$\mathbb{P}\left(\frac{\mathbf{z}_{P_i}^{i_q,i_{q+1}}}{\mathbf{y}_{P_i}}\geq\frac{4}{\delta}\middle|\mathcal{F}_{i_j}\right)\leq\frac{\delta}{4}\frac{E[\mathbf{z}_{P_i}^{i_q,i_{q+1}}|\mathcal{F}_{i_j}]}{\mathbf{y}_{P_i}}=\frac{\delta}{4}.$$

Combining the above with a union bound we finally conclude that $\mathbb{P}(\left|z-\mathbf{z}_P^{i_q,i_{q+1}}\right|>\epsilon)\leq\delta$, confirming that the high probability approximation guarantee is achieved by Algorithm 1*b*.

Next let's account for the computational and sampling complexity of Algorithm 1. We analyze part *a* first. Let's denote by $R_a(k,\epsilon,\delta)$ and $S_a(k,\epsilon,\delta)$ the upper bounds on the runtime of and number of simulation calls required by Algorithm 1*a* with any input path of length $(k-1)$, and with output precision requirement (ϵ,δ). In the base case i.e. $k=2$, one call is made to $\mathcal{W}$, with precision requirement (ϵ,δ). By assumption, we need a runtime of $\nu_1(\epsilon,\delta)$ and number of samples from $\mathcal{S}$ $\nu_2(\epsilon,\delta)$. In case $k\geq 3$, Algorithm 1*a* finds the minimum of $\{i_2,...,i_{k-1}\}$ in a

runtime $k-2$. It then calls simulator $\mathcal{S}$ $N = 8\epsilon^{-2}\log(4/\delta)$ times for independent ω_j^i, in a runtime CN. Then N calls are made to Algorithm 1a, each with inputs $(P_1, \omega_j^i), i = 1, ..., N$ and $(P_2, \omega_j^i), i = 1, ..., N$, and each with precision requirement $(\epsilon/16, \epsilon/16)$, each taking runtime at most $R_a(k_1, \epsilon/16, \epsilon/16)$ and $R_a(k_2, \epsilon/16, \epsilon/16)$ where k_i is the length of P_i, satisfying $k_1 + k_2 = k + 1$. Finally N minimums are computed and the average of N numbers are computed, at a computational cost of $2N$. Combining the above, the total runtime is at most

$$k + CN + N\Big(R_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + R_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big) + 2N.$$

We proceed to account for the sampling complexity. Algorithm 1a makes N direct calls to simulator $\mathcal{S}$. It then makes N calls to itself with inputs P_1, ω_j^i and P_2, ω_j^i, each with precision requirement $(\epsilon/16, \epsilon/16)$. Hence each needs number of samples at most $S_a(k_1, \epsilon/16, \epsilon/16)$ and $S_a(k_2, \epsilon/16, \epsilon/16)$. Combining the above, the total sampling complexity is at most

$$N + N\Big(S_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + S_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big).$$

We set functions

$$R_a(k, \epsilon, \delta) \triangleq (1 + \log(\frac{1}{\delta})) \times (\frac{1}{\epsilon})^{3(k-2)} \times 100^{(k-2)^2} \times (C + \nu_1(\epsilon 16^{-(k-2)}, \delta \wedge \epsilon 16^{-(k-2)})),$$

and

$$S_a(k, \epsilon, \delta) \triangleq (1 + \log(\frac{1}{\delta})) \times (\frac{1}{\epsilon})^{3(k-2)} \times 100^{(k-2)^2} \times \nu_2(\epsilon 16^{-(k-2)}, \delta \wedge \epsilon 16^{-(k-2)}).$$

They satisfies that $R_a(2, \epsilon, \delta) \geq \nu_1(\epsilon, \delta)$ and $S_a(2, \epsilon, \delta) \geq \nu_2(\epsilon, \delta)$. By Lemma 37, they also satisfy the recursive inequalities that

$$R_a(k, \epsilon, \delta) \geq k + CN + N\Big(R_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + R_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big) + 2N$$

and

$$S_a(k, \epsilon, \delta) \geq N + N\Big(S_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + S_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big).$$

Hence R_a and S_a are valid upper bounds on the runtime and sample complexity of Algorithm 1a.

Finally we analyze the runtime and sample complexity of Algorithm 1b. Let's denote by $R_b(k,\epsilon,\delta), S_b(k,\epsilon,\delta)$ the upper bound on runtime of and number of simulation calls required by Algorithm 1b with any input path of length $(k-1)$,any edge on the path, and with output precision requirement (ϵ,δ). In the base case $k=2$, one call is made to $\mathcal{W}$, with precision requirement (ϵ,δ). By assumption, we need a runtime of $\nu_1(\epsilon,\delta)$ and number of samples from $\mathcal{S}$ $\nu_2(\epsilon,\delta)$. In case $k \geq 3$, Algorithm 1b first computes the minimum of a $(k-2)$-vector at a computational cost $k-2$. It then makes two calls to Algorithm 1a with input P_1 and P_2, each with a precision requirement $(\epsilon\delta/16, \delta/8)$, at computational times at most $R_a(k_1, \epsilon\delta/16, \delta/8)$ and $R_a(k_2, \epsilon\delta/16, \delta/8)$, respectively. Next, it makes one call to itself on P_i with a precision requirement $(\epsilon/2, \delta/2)$ at a computational cost $R_b(k_i, \epsilon/2, \delta/2)$. Simple algebraic operations are performed at the end of the algorithm, at a computational cost bounded by 3. To sum up, the total runtime is at most

$$k + R_a(k_1, \epsilon\delta/16, \delta/8) + R_a(k_2, \epsilon\delta/16, \delta/8) + R_b(k_i, \epsilon/2, \delta/2) + 3.$$

We proceed to account for the sampling complexity. Algorithm 1b calls simulator $\mathcal{S}$ during its two calls to Algorithm 1a, at a sampling complexity at most $S_a(k_1, \epsilon\delta/16, \delta/8)$ and $S_a(k_2, \epsilon\delta/16, \delta/8)$, respectively. It then calls itself, which requires a sampling complexity at most $S_b(k_i, \epsilon/2, \delta/2)$. Combining all the above, the total number of samples required is at most

$$S_a(k_1, \epsilon\delta/16, \delta/8) + S_a(k_2, \epsilon\delta/16, \delta/8) + S_b(k_i, \epsilon/2, \delta/2).$$

We set functions as

$$R_b(k,\epsilon,\delta) \triangleq (1+\log(\frac{1}{\delta})) \times (\frac{1}{\epsilon\delta})^{3(k-1)} \times 100^{(k-1)^2} \times \left(C + \nu_1(\frac{\epsilon\delta}{4^{2(k-2)}}, \frac{\epsilon\delta}{4^{2(k-2)}})\right)$$

and

$$S_b(k,\epsilon,\delta) \triangleq (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon\delta})^{3(k-1)}\times 100^{(k-1)^2}\times \nu_2(\frac{\epsilon\delta}{4^{2(k-2)}},\frac{\epsilon\delta}{4^{2(k-2)}})$$

They satisfies that $R_b(2,\epsilon,\delta)\geq\nu_1(\epsilon,\delta)$ and $S_b(2,\epsilon,\delta)\geq\nu_2(\epsilon,\delta)$. By Lemma 38, they also satisfy the recursive inequalities

$$R_b(k,\epsilon,\delta) \geq k+R_a(k_1,\epsilon\delta/16,\delta/8)+R_a(k_1,\epsilon\delta/16,\delta/8)+R_b(k_i,\epsilon/2,\delta/2)+3$$

and

$$S_b(k,\epsilon,\delta) \geq S_a(k_1,\epsilon\delta/16,\delta/8)+S_a(k_2,\epsilon\delta/16,\delta/8)+S_b(k_i,\epsilon/2,\delta/2).$$

Hence R_b and S_b are valid upper bounds on the runtime and sample complexity of Algorithm 1*b*.

Combining all above, we complete our proof. □

C.4.3 Approximate blocking flow and proof of Proposition 6

We start by providing several results related to Algorithm 2′ before moving on to deal with Algorithm 2. We first state and prove a corollary of Lemma 19, that when running Algorithm 2′, all intermediate flows and (finite) capacities are bounded in [0, 1].

Corollary 5. *Suppose Algorithm* 2′ *starts in the residual network of a feasible flow* $\mathbf{z}_0$*, then after* k *rounds of iterations for any* $k\geq 1$*, both the current flow and the current (finite) capacities on any arc are bounded in* $[0,1]$

Proof. By Lemma 19, at the beginning of Algorithm 2′, the finite residual capacities are all bounded in [0, 1]. In later iterations, the finite capacities only decrease

and remains non-negative, hence in $[0, 1]$. Therefore, the feasibility constraint requires that all flows found are in $[0, 1]$ on arcs of finite capacities. Now it remains to show that on arcs with infinite capacities, the flow never exceeds 1 during the process of Algorithm 2′. We argue by contradiction. Suppose after some iteration l, the current flow on an arc e of infinite capacity is more than 1. By Lemma 33, by time l the algorithm finds a flow $\mathbf{z}'_l$ that is feasible in residual network $\mathcal{N}_{\mathbf{z}_0}$. Therefore the net flow of $\mathbf{z}_0 + \mathbf{z}'_l$ is a feasible flow in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. However, Lemma 33 also asserts that $\mathbf{z}'_l$ is regular. The fact that $\mathbf{z}'_l$ uses the arc e with infinite capacity implies that the reverse arc is never ever used during the process of Algorithm 2′. Hence the net flow of $\mathbf{z}_0 + \mathbf{z}'_l$ on arc e is at least $\mathbf{z}'_l(e)$, which exceeds 1 by assumption. But by Lemma 19, a feasible flow in network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ never exceeds 1 on any arc. The contradiction thus proves our result. □

We now extend certain concepts in networks introduced earlier to their ϵ-*approximate* versions. To accommodate the probabilistic interpretation of flows and capacities that we adopt, we only make these extensions in networks defined on residual graph $G^{\mathbf{x}}$. These concepts can actually be defined for more general networks. Given a set of capacities $\mathbf{w}$ on the residual graph $G^{\mathbf{x}}$, we say an arc sample e is ϵ-*saturated* by a flow $\mathbf{z}$ if $\mathbf{w}(e) - \mathbf{z}(e) \leq \epsilon$. By treating certain less than ϵ quantities as 0 in a similar manner, we can effectively extend all other terminologies introduced before to their ϵ-approximate versions. We omit the details. With these concepts, we can generalize Dinic's algorithm to its ϵ-approximate version, which can output an approximation of the max flow. Now, we state the extension of Lemma 14. It will be of use in the proof of Prosposition 6.

Corollary 6. *Suppose a flow* $\mathbf{z}$ *is* ϵ*-blocking in the* ϵ*-level subnetwork of the residual network of flow* $\mathbf{z}_0$*. Then the* ϵ*-level of sink* $d_\epsilon(t)$ *is increased by at least one in the*

residual network $\mathcal{N}_{\mathbf{z}+\mathbf{z}_0}$ *associated with flow* NET $(\mathbf{z}+\mathbf{z}_0)$.

Proof of Corollary 6. The proof is identical to that of Lemma 14. □

We next state and prove a lemma, that the flow found by Algorithm 2′, when truncated at iteration K with K large enough, can augment any (η, l)-disconnecting flow to become an $(\eta, l+2)$-disconnecting flow. For any network $\mathcal{N}$, we introduce its *η-thresholded subnetwork*, which has capacity 0 on all arcs with capacity less than η in $\mathcal{N}$.

Lemma 35. *Suppose Algorithm 2′ starts in the η-thresholded residual network of an (η, l)-disconnecting flow $\mathbf{z}_0$, performs its iteration for $K \triangleq \eta^{-1} l \Delta^l$ rounds and outputs a flow $\mathbf{z}_K$. Then* NET $(\mathbf{z}_0+\mathbf{z}_K)$ *is an $(\eta, l+2)$-disconnecting flow.*

Proof of Lemma 35. If all s-t paths with length $\leq (l+2)$ are η-disconnected in $\mathcal{N}_{\mathbf{z}_0}$, the residual network of $\mathbf{z}_0$, Algorithm 2′ will output 0 flow, and flow $\mathbf{z}_0$ is already $(\eta, l+2)$-disconnecting and the statement follows. Otherwise, there exists an s-t path of length $(l+2)$ with capacities greater than η on all arcs. In such cases, we prove that all length $(l+2)$ s-t paths become η-disconnected after running for enough iterations. We denote the capacities in the j-th iteration of Algorithm 2′ by $\mathbf{w}^j$. For any regular $-1 \to -2$ path $P \in G$, we denote the P-subnetwork maximum flow that Algorithm 1′ outputs in the j-th iteration of Algorithm 2′ by $\mathbf{z}_P^j$. We consider any length $(l+2)$ s-t path $Q = (s, \omega_1, ..., \omega_{l+1}, t) \in \mathcal{N}_{\mathbf{z}_0}$ whose capacities are all greater than η. To simplify notation, suppose $\phi(Q) = P$.

We prove that $\min_{e \in Q} \mathbf{w}^K(e) \leq \eta$. By definition of maximum flow, in iteration $j \geq 1$, flow $\mathbf{z}_P^j$ saturates at least one arc on path Q, i.e. there exists an arc $e \in Q$ such that $\mathbf{w}^j(e) = \mathbf{z}_P^j(e) < \infty$. By the updating rule of Algorithm 2′, $\mathbf{w}^{j+1}(e) \geq \mathbf{w}^j(e) - \frac{1}{\Delta^l}\mathbf{z}_P^j(e) = (1 - \frac{1}{\Delta^l})\mathbf{w}^j(e)$. Therefore, after each iteration of

Algorithm 2′, there exists an arc whose (finite) capacity is decreased by a factor of $(1 - \frac{1}{\Delta^l})$. Since there are $(l + 1)/2$ different arcs with finite capacities on Q, we conclude by pigeonhole principle that after K iterations, there must exist one arc whose capacity is decreased by a factor of $(1 - \frac{1}{\Delta^l})^{K/l}$. By Corollary 5, all finite capacities are bounded in $[0, 1]$ during the process of Algorithm 2′. Therefore, after K iterations, there must exist one arc whose capacity is at most $(1 - \frac{1}{\Delta^l})^{K/l} = (1 - \frac{1}{\Delta^l})^{\frac{\Delta^l}{\eta}} \leq e^{-\frac{1}{\eta}} \leq \eta$, confirming the assertion.

We next prove the lemma. Since we have shown that flow $\mathbf{z}_K$ η-saturates all length $(l + 2)$ s-t path Q in the η-thresholded residual network of $\mathbf{z}_0$, it must be feasible and η-saturates all s-t path of length $(l + 2)$ in the residual network of $\mathbf{z}_0$. By definition, $\mathbf{z}_K$ is an η-blocking flow in the η-level subnetwork of the residual network of $\mathbf{z}_0$. Combining the above with Corollary 6, we conclude that in the residual network of $\mathbf{z}_0 + \mathbf{z}_K$, the length of the shortest η-connected s-t path is increased by at least one. Therefore $\mathbf{z}_0 + \mathbf{z}_K$ is $(\eta, l + 2)$-disconnecting, proving the lemma. □

With Corollary 5 and Lemma 35, we proceed to give the proof of Proposition 6.

Proof of Proposition 6. We first show that Algorithm 2 can indeed output the high probability approximation of an $(\eta, l + 2)$-disconnecting flow. We set iteration count $k \leftarrow \eta^{-1}\Delta^l$. By Lemma 35, were all the outputs from Algorithm 1 and $\mathcal{W}$ accurate, the k-th iteration of Algorithm 2 can return a flow $\mathbf{z}_k$ such that $\mathbf{z}_0 + \mathbf{z}_k$ is $(\eta, l + 2)$-disconnecting. We denote the "ideal" outputs as of Lemma 35 after the i-th iteration by $\mathbf{z}_i, i = 1, ..., k$ and the real outputs by $z_i, i = 1, ..., k$. It suffices to show that for any $(i, j) \in E$, $|z_k^{i,j} - \mathbf{z}_k^{i,j}| < \epsilon$ with probability at least $1 - \delta$. For given $(i, j) \in E$, let $\mathcal{M}_{l+2}^{i,j}$ denote the set of length $(l+2)$, regular $-1 \to -2$ path that

contains edge (i, j). Algorithm 2 first makes one call to itself with input $(k-1, e, l)$, returning z', which satisfies $|z' - \mathbf{z}_{k-1}^{i,j}| < \epsilon/2$ with probability at least $1-\delta/2$. It then calls Algorithm 1b with input (P, e) and capacity evaluation subroutine $\mathcal{W}^{k-1}$, for all $P \in \mathcal{M}_{l+2}^{i,j}$, with outputs z_P required to satisfy $|z_P^{(i,j)} - \mathbf{z}_{P,k-1}^{i,j}| < \epsilon/2$, each with probability at least $(1 - \frac{\delta}{2\Delta^l})$, where $\mathbf{z}_{P,k-1}$ is the maximum flow in P-subnetwork with capacities $\mathbf{w}_{k-1}$. Then it outputs $z_k^{i,j} = z' + \Delta^{-l} \operatorname{SUM}(z_P)$ as the approximation of $\mathbf{z}_k^{i,j} = \mathbf{z}_{k-1}^{i,j} + \Delta^{-l} \operatorname{SUM}(\mathbf{z}_{P,k-1}^{i,j})$. We have that

$$\begin{aligned} \left|z_k^{i,j} - \mathbf{z}_k^{i,j}\right| &= \left|z' + \Delta^{-l} \operatorname{SUM}(z_P) - \mathbf{z}_{k-1}^{i,j} - \Delta^{-l} \operatorname{SUM}(\mathbf{z}_{P,k-1}^{i,j})\right| \\ &\leq \left|z' - \mathbf{z}_{k-1}^{i,j}\right| + \frac{1}{\Delta^l} \sum_{P \in \mathcal{M}_{l+2}^{i,j}} \left|z_P - \mathbf{z}_{P,k-1}^{i,j}\right| \end{aligned}$$

With a union bound on P, we get that with probability at least $1 - \delta/2$, $|z_P - \mathbf{z}_{P,k-1}^{i,j}| \leq \epsilon/2$ for all $P \in \mathcal{M}_{l+2}^{i,j}$. Since G is bounded with max degree Δ, we have that $|\mathcal{M}_{l+2}^{i,j}| \leq \Delta^l$. Hence we have with probability at least $1 - \delta/2$, $\frac{1}{\Delta^l} \sum_{P \in \mathcal{M}_{l+2}^{i,j}} \left|z_P - \mathbf{z}_{P,k-1}^{i,j}\right| \leq \epsilon/2$. With another union bound, we get the desired result.

We next account for the computational and sampling complexity of Algorithm 2. Let's denote by $R_2(\eta, k, l, \epsilon, \delta)$ and $S_2(\eta, k, l, \epsilon, \delta)$ the upper bounds of runtime and sampling complexity for Algorithm 2, with any input parameter η, k, l and output precision requirement ϵ, δ. In the base case $k = 0$, the algorithm output 0 flow at 0 computational cost and sampling cost. In case k with $k > 0$, one call is made to Algorithm 2 with parameters $\eta, k-1, l$, and with precision requirement $(\epsilon/2, \delta/2)$, at a computational cost $R_2(\eta, k-1, l, \epsilon/2, \delta/2)$. Then at most Δ^l calls are made to Algorithm 1b on length $l+2$ paths with capacity evaluation subroutine $\mathcal{W}_{k-1}$, each with output precision requirement $(\epsilon/2, \frac{\delta}{2\Delta^l})$. For any $(i, j) \in E$, capacity evaluation subroutine $\mathcal{W}_{k-1}$ can output an (ϵ, δ)-approximation of the capacities $\mathbf{w}_{k-1}^{i,j}$ with one call to $\mathcal{W}$ with precision requirement $(\epsilon/2, \delta/2)$ at a computational cost $\mu_1(\epsilon/2, \delta/2)$, and one another call to Algorithm 2 with parameters

$(\eta, k-1, l)$ and with precision requirement $(\epsilon/2, \delta/2)$, at a computational cost $R_2(\eta, k-1, l, \epsilon/2, \delta/2)$. The total computational cost for $\mathcal{W}_{k-1}$ to return an (ϵ, δ)-approximation of $\mathbf{w}_{k-1}^{i,j}$ is thus bounded by $\mu_1(\epsilon/2, \delta/2) + R_2(\eta, k-1, l, \epsilon/2, \delta/2)$. By Proposition 5, we can bound the runtime required for Algorithm 2's each call to Algorithm 1*b* by

$$(1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \times \left(C + \mu_1(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + R_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\right)$$

Finally, average of Δ^l numbers is computed at a computational cost of Δ^l. Combining the above, we conclude that the total runtime cost is at most

$$R_2(\eta, k-1, l, \epsilon/2, \delta/2) + \Delta^l + \Delta^l \times (1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \times \left(C + \mu_1(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + R_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\right).$$

We proceed to account for the sampling cost. The one call to Algorithm 2 on parameters $(\eta, k-1, l)$ and precision level $(\epsilon/2, \delta/2)$ requires number of samples from simulator $\mathcal{S}$ at most $S_2(\eta, k-1, l, \epsilon/2, \delta/2)$. Each of the at most Δ^l calls to Algorithm 1*b* on length $l+2$ paths with capacity evaluation subroutine $\mathcal{W}_{k-1}$, with precision level $(\epsilon/2, \frac{\delta}{2\Delta^l})$ require number of samples from simulator $\mathcal{S}$ at most

$$(1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \times \left(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + S_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\right)$$

The total sampling cost is at most

$$S_2(\eta, k-1, l, \epsilon/2, \delta/2) + \Delta^l \times (1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \times \left(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + S_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\right).$$

We set functions R_2 and S_2 as

$$R_2(\eta,k,l,\epsilon,\delta) \triangleq \left(\frac{(100\Delta)^l}{\epsilon\delta}\right)^{4^k} \times \left(C+\mu_1\left((\epsilon\delta(100\Delta)^{-l})^{4^k},(\epsilon\delta(100\Delta)^{-l})^{4^k}\right)\right)$$

and

$$S_2(\eta,k,l,\epsilon,\delta) \triangleq \left(\frac{(100\Delta)^l}{\epsilon\delta}\right)^{4^k} \times \mu_2\left((\epsilon\delta(100\Delta)^{-l})^{4^k},(\epsilon\delta(100\Delta)^{-l})^{4^k}\right)$$

By Lemma 39, they satisfy the recursions

$$\begin{aligned} R_2(\eta,k,l,\epsilon,\delta) \quad \geq \quad & R_2(\eta,k-1,l,\epsilon/2,\delta/2)+\Delta^l \\ & + \Delta^l \times (1+\log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \\ & \times\Big(C+\mu_1(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})+R_2(\eta,k-1,l,\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big). \end{aligned}$$

and

$$\begin{aligned} S_2(\eta,k,l,\epsilon,\delta) \quad \geq \quad & R_2(\eta,k-1,l,\epsilon/2,\delta/2) \\ & + \Delta^l \times (1+\log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \\ & \times\Big(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})+S_2(\eta,k-1,l,\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big). \end{aligned}$$

Finally we plug in the choice of k as specified by Algorithm 2 : $k = \eta^{-1}\Delta^l$ and complete the proof. □

C.4.4 L-disconnecting flows and proof of Proposition 7

Proof of Proposition 7. We first show that Algorithm 3 achieves the desired approximation guarantee. Let's argue by induction. In the base case $L = 1$, both the true flow and the output of Algorithm 3 are 0 flow, therefore the algorithm is exact. Now suppose for input $L-2$, Algorithm 3 can achieve the desired approximation guarantee. Suppose with $\eta, L-2$ as input, Algorithm 3 outputs a high probability approximation of flow $\mathbf{z}_{L-2}$ on each edge, where $\mathbf{z}_{L-2}$ is

$(\eta, L-2)$-disconnecting in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. We consider case L. We first show that capacity evaluation subroutine $\mathcal{W}^{L-2}$ can achieve the desired guarantee. Indeed, the evaluation of $c^{\mathbf{x}}$ is exact, and the evaluation of flow $\mathbf{z}_{L-2}$ is guaranteed by inductive hypothesis, hence for both reverse and non-reverse arcs, $\mathcal{W}^{L-2}$ can output high probability approximation of the residual capacities of flow $\mathbf{z}_{L-2}$. Let's analyze Algorithm 3. Algorithm 3 first makes one call to itself with parameters $(L-2, \eta)$, and outputs a random variable z that satisfies $\mathbb{P}(|z-\mathbf{z}^{i,j}_{L-2}| > \epsilon/3) \leq 1-\delta/3$. It then makes two calls to Algorithm 2 with parameters $(\eta^{-1}\Delta^{L-2}, L-2)$ and capacity evaluation subroutine $\mathcal{W}^{L-2}$. By Proposition 6, the outputs z_1 and z_2 can be made to satisfy $|z_1 - \mathbf{z}^{i,j}| < \epsilon/3$ and $|z_2 - \mathbf{z}^{j,i}| < \epsilon/3$, both with probability at least $1-\delta/3$, with flow $\mathbf{z}$ in the residual network of $\mathbf{z}_{L-2}$ satisfying that $\text{NET}\,(\mathbf{z}+\mathbf{z}_{L-2})$ is (η, L)-disconnecting. We denote $\mathbf{z}_L \stackrel{\Delta}{=} \text{NET}\,(\mathbf{z}+\mathbf{z}_{L-2})$. Then by a union bound, we conclude that

$$\mathbb{P}\left(\left|z_0 + z_1 - z_2 - \mathbf{z}_L\right| > \epsilon\right) < \delta.$$

We next account for the runtime and sampling complexity of Algorithm 3. Let's denote by $R_3(\eta, L, \epsilon, \delta)$ and $S_3(\eta, L, \epsilon, \delta)$ the upper bounds on runtime and number of calls to simulator made by Algorithm 3 to output an (ϵ, δ)-approximation of the (η, L)-disconnecting flow $\mathbf{z}_L$. We start by analyzing subroutine $\mathcal{W}^l$. It makes one call to simulator S and one call to Algorithm 3 with input parameters (l, η). To output an (ϵ, δ)-approximation of the residual capacity of flow $\mathbf{z}_l$, it takes a computational cost $C + R_3(\eta, l, \epsilon, \delta)$, and with number of calls to simulator $\mathcal{S}$ at most $1 + S_3(\eta, l, \epsilon, \delta)$. With the analysis of W^l, we now proceed to analyze Algorithm 3. It first makes one call to itself with parameters $(L-2, \eta)$ and precision requirement $(\epsilon/3, \delta/3)$, at a computational cost $R_3(\eta, L-2, \epsilon/3, \delta/3)$. It then makes two calls to Algorithm 2, with input parameters $(\eta^{-1}\Delta^{L-2}, L-2)$ and capacity evaluation subroutine $\mathcal{W}^{L-2}$, both with pre-

cision requirement $(\epsilon/3, \delta/3)$. By Proposition 6, each call incur a computational cost of $R_2(\eta, \eta^{-1}\Delta^{L-2}, L-2, \epsilon/3, \delta/3)$. Combining the above, the total runtime can be bounded by

$$R_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \times \left(2C + R_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right).$$

Similarly, the total number of calls to simulator can be bounded by

$$S_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \times \left(1 + S_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right).$$

We set R_3 and S_3 as follows.

$$R_3(\eta, L, \epsilon, \delta) \triangleq C\left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta^L}}$$

and

$$S_3(\eta, L, \epsilon, \delta) \triangleq \left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta^L}}$$

Then by Lemma 40, we have that

$$R_3(\eta, L, \epsilon, \delta) \geq R_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \times \left(2C + R_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right)$$

and

$$S_3(\eta, L, \epsilon, \delta) \geq S_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \times \left(1 + S_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right).$$

Hence R_3 and S_3 are valid upper bounds on the computational and sampling complexity of Algorithm 3.

Combining all the above, we complete our proof. □

C.5 Approximate max flow and proof of Proposition 8

We first prove that an (η, l)-disconnecting flow in network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ is sufficiently close to its optimal max flow. We proceed by steps. First, we state a Lemma showing that we can manipulate an (η, l)-disconnecting flow to get an l-disconnecting flow in a network sufficiently close to $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$.

Lemma 36. *Suppose a flow* $\mathbf{z}$ *of value* y *is* (η, l)*-disconnecting in* $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$*. Then there exists a feasible flow* $\mathbf{z}'$ *of value* y' *and a set of capacities* c'*, such that* $\mathbf{z}'$ *is* l*-disconnecting in* $\{G^{\mathbf{x}}, c', s, t\}$*, and that* $y' > y - \Delta T\eta/2$*. Furthermore, the max flow in two networks are different by at most* $(\Delta + 1)T\eta$*.*

Proof. By definition, a flow $\mathbf{z}$ is (η, l)-disconnecting if for any length $\leq l$, $-1 \to -2$ path $P \in G$, $\min_{(i,j)\in P} \mathbf{w}_{\mathbf{z}}^{i,j} \leq \eta \;\; w.p.1$ where $\mathbf{w}_{\mathbf{z}}$ is the associated residual capacity. We define a new flow $\mathbf{z}'$ as follows. For any (i, j) such that $i \in \mathcal{L}$ and $j \in \mathcal{R}$, we have $\mathbf{z}'^{i,j} \stackrel{\Delta}{=} \max(\mathbf{z} - \eta, 0) \;\; w.p.1$. On other edges, namely those containing -1 or -2, the flow values are uniquely determined by conservation law. Hence $\mathbf{z}'$ is a well defined flow. We first show that flows $\mathbf{z}$ and $\mathbf{z}'$ are close to each other. More precisely, their flow values y and y' satisfy $y' > y - \Delta T\eta/2$.

Indeed, in network $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$, arcs that point from $\mathcal{L}^{\mathbf{x}}$ to $\mathcal{R}^{\mathbf{x}}$ form a valid cut that separates s from t. Hence by network flow theory, the flow value in $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$ equals the sum of flows on arcs belonging to the cut. Under our probabilistic representation, the flow on all arcs that are samples of a fixed edge $(i, j) \in E$ is given by $E[\mathbf{z}^{i,j} - \mathbf{z}'^{i,j}]$, which is at most η. Therefore, the total difference between the two flows on all middle arcs that point from $\mathcal{L}^{\mathbf{x}}$ to $\mathcal{R}^{\mathbf{x}}$ is bounded by

$$\sum_{(i,j)\in E: i\in\mathcal{L}, j\in\mathcal{R}} E[\mathbf{z}^{i,j} - \mathbf{z}'^{i,j}] \leq \sum_{(i,j)\in E: i\in\mathcal{L}, j\in\mathcal{R}} \eta \leq \frac{\Delta T}{2}\eta$$

where the last inequality follows from the fact that one of $\mathcal{L}$ and $\mathcal{R}$ must contain less than $T/2$ vertices, each of which is on at most Δ different edges from $\mathcal{L}$ to $\mathcal{R}$.

Now let's define a new set of capacity c', so that $\mathbf{z}'$ is l-disconnecting in $\{G^{\mathbf{x}}, c', s, t\}$. Let $\mathbf{w}'$ be the associated set of random variables under the probabilistic representation, and $\mathbf{w}$ be the set of random variables of the original capacity $c^{\mathbf{x}}$. For all edges $(i, j) \in G$ with either $i = -1, j \in \mathcal{L}$ or $j = -2, i \in \mathcal{R}$, we define $\mathbf{w}'^{i,j} \stackrel{\Delta}{=} \max(\mathbf{w}^{i,j} - (\Delta + 1)\eta, \mathbf{z}'^{i,j})$ $w.p.1$. and for all other edges $(i, j) \in E$ with $i \in \mathcal{L}$ and $j \in \mathcal{R}$, the capacities remain infinity $\mathbf{w}'^{i,j} \stackrel{\Delta}{=} \infty$. Flow $\mathbf{z}'$ is feasible in the network with capacities $\mathbf{w}'$, because by definition of $\mathbf{w}'$, $\mathbf{z}'^{i,j} \leq \mathbf{w}'^{i,j}$ $w.p.1$ for all edges from -1 or to -2.

We prove that flow $\mathbf{z}'$ is l-disconnecting in the network with capacities $\mathbf{w}'$. Indeed, it suffices to show that on any length $\leq l$, $-1 \to -2$ regular path $P \in G$, $\min_{(i,j)\in P} \mathbf{w}'^{i,j}_{\mathbf{z}'} = 0$ $w.p.1$. By assumption, $\min_{(i,j)\in P} \mathbf{w}^{i,j}_{\mathbf{z}} \leq \eta$ $w.p.1$. There are two cases. In case one, the path-wise minimum is achieved on a reverse arc e of edge (i, j) with $i \in \mathcal{R}$ and $j \in \mathcal{L}$. Suppose e' is its pairing arc. Then we have that $\mathbf{w}^{i,j}_{\mathbf{z}}(e) = \mathbf{z}^{j,i}(e') \leq \eta$. Combining with the definition of $\mathbf{z}'$, we conclude that $w'^{i,j}_{\mathbf{z}'}(e) = \mathbf{z}'^{j,i}(e') = 0$. In case two, the path-wise minimum is achieved on an arc e of the form (s, ω_i) or (ω_j, t). We then have that $\mathbf{w}^{i,j}(e) - \mathbf{z}^{i,j}(e) \leq \eta$. We prove that for any such arc e, the difference between $\mathbf{z}'^{i,j}(e)$ and $\mathbf{z}^{i,j}(e)$ is at most $\Delta\eta$. Without loss of generality, let's consider $e = (s, \omega)$ with $\omega \in \mathcal{H}_i$. By the structure of $G^{\mathbf{x}}$, the flow entering node ω can only uses arc samples of $(i, j) \in E$ for $j \in \mathcal{R}$ to leave ω, with at most Δ such edges in E. By our previous reasoning, the difference of flows of $\mathbf{z}$ and $\mathbf{z}'$ on arc samples of any one fixed edge (i, j) is at most η. Combining the above observation, we conclude that the difference of flows between $\mathbf{z}$ and $\mathbf{z}'$ on all arcs linking ω to $\mathcal{R}$ is at most $\Delta\eta$. Thus by flow conservation law, we conclude that the difference of flow of $\mathbf{z}$ and $\mathbf{z}'$ on arc (s, ω) is at most $\Delta\eta$.

However, $\mathbf{w}'$ is reduced by $(\Delta + 1)\eta$ (if possible). Hence if an arc is ϵ-saturated by flow $\mathbf{z}$ with capacities $\mathbf{w}$, it must be saturated by flow $\mathbf{z}'$ with capacity $\mathbf{w}'$. Since $\mathbf{z}$ is (η, l)-disconnecting with capacities $\mathbf{w}$, we finally conclude that $\mathbf{z}'$ is l-disconnecting with capacities $\mathbf{w}'$.

Finally, we prove that max flow in the network with capacities $\mathbf{w}$ is sufficiently close to that in the network with capacities $\mathbf{w}'$. Indeed, by max-flow min-cut theorem, the difference between the max-flow in the two networks equals the difference between the min-cut in the two networks, which is further bounded by the total difference of edge capacities in the two networks. We have argued that, the total difference of the capacities on all arcs that are samples of some fixed edge (i, j) (of the form $(-1, j)$ or $(i, -2)$) is $E[\mathbf{w}^{i,j} - \mathbf{w}'^{i,j}] \leq (\Delta + 1)\eta$. There are T such edges, and any arc in the network with finite capacity must be a sample of one of these edges. We then conclude that the difference between the max flow values in the two networks is at most $T(\Delta + 1)\eta$. □

Combining Lemma 36 with Proposition 2 and Lemma 15, we directly derive a bound on the difference between an (η, l)-disconnecting flow and the true max flow, whose proof we omit.

Corollary 7. *Suppose flow* $\mathbf{z}$ *with value* M *is* (η, l)*-disconnecting in network* $\{G^{\mathbf{x}}, c^{\mathbf{x}}, s, t\}$. *Then*

$$M \geq \left(\frac{l-2}{l+4}\right)\left(\text{OPT}' - (\Delta + 1)T\eta\right) - \frac{1}{2}T\Delta\eta.$$

With Corollary 7, we set to prove Proposition 8.

Proof of Proposition 8. We first show that Algorithm 4 can achieve the promised guarantee, outputting a $T\epsilon$-approximation of OPT′ with high probability. We set

$L = \frac{24}{\epsilon}$. and $N = 128\epsilon^{-2}\delta^{-2}\log(\frac{8}{\epsilon\delta})$. For each $i \in \mathcal{L}$, Algorithm 4 makes N independent calls to simulator $\mathcal{S}$, followed with N calls to Algorithm 3, outputting N numbers $w_{i,j}$. We require that with probability at least $1 - \frac{\epsilon\delta}{32}$, $|w_i - \mathbf{z}_L| < \frac{\epsilon\delta}{32}$, where $\mathbf{z}_L$ is an $(\epsilon/12, L)$-disconnecting flow. The algorithm then takes average of the N numbers and derive $y_i = \text{AVERAGE}(w_{i,j})$. It outputs y as the sum of y_i. The total value of flow $\mathbf{z}_L$ on arc samples of $(-1, i)$ is $\mathbf{y}_i = E[\mathbf{z}_L^{-1,i}]$, and the total value of $\mathbf{z}_L$ is $\mathbf{y} = \sum_{i\in\mathcal{L}} \mathbf{y}_i$. We reason that

$$\begin{aligned}\left|y_i - \mathbf{y}_i\right| &= \left|\frac{1}{N}\sum_{j=1}^{N} w_{i,j} - E[\mathbf{z}_L^{-1,i}]\right| \\ &\leq \frac{1}{N}\sum_{j=1}^{N}\left|w_{i,j} - \mathbf{z}_L^{-1,i}(s,\omega_i^j)\right| + \left|\frac{1}{N}\sum_{j=1}^{N}\mathbf{z}_L^{-1,i}(s,\omega_i^j) - E[\mathbf{z}_L^{-1,i}]\right|\end{aligned}$$

Let $(\mathcal{U}_j, j = 1, ..., N)$ be N i.i.d. random variables with the distribution: $\mathcal{U} = \frac{\epsilon\delta}{32}$ $w.p$ $1 - \frac{\epsilon\delta}{32}$; and $\mathcal{U} = 1$ $w.p.$ $\frac{\epsilon\delta}{32}$. Then $(\left|w_{i,j} - \mathbf{z}_L^{-1,i}(s,\omega_i^j)\right|, j = 1, ..., N)$ are N independent random variables that are stochastically dominated by $\mathcal{U}$. Hence we have that

$$\begin{aligned}&\mathbb{P}\Big(\frac{1}{N}\sum_{j=1}^{N}\left|w_{i,j} - \mathbf{z}_L^{-1,i}(s,\omega_i^j)\right| > \frac{\epsilon\delta}{8}\Big) \leq \mathbb{P}\Big(\frac{1}{N}\sum_{j=1}^{N}\mathcal{U}_j > \frac{\epsilon\delta}{8}\Big) \\ &= \mathbb{P}\Big(\frac{1}{N}\sum_{j=1}^{N}\mathcal{U}_j - E[\mathcal{U}] > \frac{\epsilon\delta}{8} - E[\mathcal{U}]\Big) \\ &\leq \mathbb{P}\Big(\Big|\frac{1}{N}\sum_{j=1}^{N}\mathcal{U}_j - E[\mathcal{U}]\Big| > \frac{\epsilon\delta}{16}\Big) \\ &\leq 2\exp\Big(-2(\frac{\epsilon\delta}{16})^2 N\Big) \leq \frac{\epsilon\delta}{8} \qquad \text{(by Lemma 22 Hoeffding's inequality)}\end{aligned}$$

By Hoeffding's Lemma, we also have

$$\mathbb{P}\Big(\Big|\frac{1}{N}\sum_{j=1}^{N}\mathbf{z}_L^{-1,i}(s,\omega_i^j) - E[\mathbf{z}_L^{-1,i}]\Big| > \frac{\epsilon\delta}{8}\Big) \leq 2\exp\Big(-2(\frac{\epsilon\delta}{8})^2 N\Big) \leq \frac{\epsilon\delta}{8}$$

Combining the above with a union bound, we conclude that $\mathbb{P}(|y_i - \mathbf{y}_i| > \frac{\epsilon\delta}{4}) < \frac{\epsilon\delta}{4}$. Using the fact that $\mathbf{y}_i$ and $y_i \in [0, 1]$ hence $|\mathbf{y}_i - y_i| \in [0, 1]$, we conclude that

$$E[|y_i - \mathbf{y}_i|] \leq \mathbb{P}(|y_i - \mathbf{y}_i| > \frac{\epsilon\delta}{4}) + (1 - \mathbb{P}(|y_i - \mathbf{y}_i| > \frac{\epsilon\delta}{4})) \times \frac{\epsilon\delta}{4} \leq \frac{\epsilon\delta}{4} + \frac{\epsilon\delta}{4} = \frac{\epsilon\delta}{2}$$

and that

$$E[|\mathbf{y} - y|] = E\Big[\Big|\sum_{i\in\mathcal{L}}(y_i - \mathbf{y}_i)\Big|\Big] \le \sum_{i\in\mathcal{L}} E[|y_i - \mathbf{y}_i|] \le \frac{T\epsilon\delta}{2}.$$

We apply a Markov inequality and get

$$\mathbb{P}(|\mathbf{y} - y| > \frac{1}{2}T\epsilon) \le \frac{2E[|\mathbf{y} - y|]}{T\epsilon} \le \delta.$$

On the other hand, since $\mathbf{z}_L$ is an $(\epsilon/12, L)$-disconnecting flow, by Lemma 36, we have that

$$\mathbf{y} \ge \Big(\frac{L-2}{L+4}\Big)(\text{OPT}' - (\Delta+1)T\frac{\epsilon}{12\Delta}) - \frac{1}{2}T\Delta\frac{\epsilon}{12\Delta} \ge \text{OPT}' - T(\frac{6}{L} + \frac{\epsilon}{4}) = \text{OPT}' - \frac{1}{2}T\epsilon.$$

Combining the above, we conclude that $\mathbb{P}(|\text{OPT}' - y| > T\epsilon) \le \delta$, hence the output of Algorithm 4 achieves the approximation guarantee.

We next account for the computational and sampling complexity. For each $i \in \mathcal{L}$, Algorithm 4 makes N calls to simulator S, each at a computational cost C, followed by N calls to Algorithm 3, with truncation level $L = \frac{24}{\epsilon}$ and error tolerance $\eta = \frac{\epsilon}{12}$, and with precision requirement $(\epsilon\delta/32, \epsilon\delta/32)$. By Proposition 7, each of these calls incur computational cost at most

$$C\Big(1024 \times \frac{(100\Delta)^{24\epsilon^{-1}}}{\epsilon^2\delta^2}\Big)^{4^{288\epsilon^{-2}\Delta^{24\epsilon^{-1}}}}.$$

Then the average of N numbers is computed at a computational cost of N. Finally the sum of at most T numbers is computed, at a computational cost of T. Combining the above, the total computational cost is at most

$$\begin{aligned} & CTN + TNC\Big(1024 \times \frac{(100\Delta)^{24\epsilon^{-1}}}{\epsilon^2\delta^2}\Big)^{4^{288\epsilon^{-2}\Delta^{24\epsilon^{-1}}}} + NT + T \\ \le \quad & CT\Big(\frac{(100\Delta)^{30\epsilon^{-1}}}{\epsilon\delta}\Big)^{4^{300\epsilon^{-2}\Delta^{30\epsilon^{-1}}}} \end{aligned}$$

Similarly, we can account for the total number of simulator calls made by Algorithm 4 to achieve the required precision, which is at most

$$T\Big(\frac{(100\Delta)^{30\epsilon^{-1}}}{\epsilon\delta}\Big)^{4^{300\epsilon^{-2}\Delta^{30\epsilon^{-1}}}}.$$

Combining the above completes the proof. □

C.6 Main results and proof of Theorem 4.3.5

Proof of Theorem 4.3.5. By Lemma 13, we only need to approximate OPT′ and $E[\sum_{i=1}^{T} c(\mathbf{x}_{[1,i]})]$ in order to derive an approximation of OPT. By Proposition 8, OPT′ can be approximated by Algorithm 4. On the other hand, $E[\sum_{i=1}^{T} c(\mathbf{x}_{[1,i]})]$ can be directly estimated by vanilla monte carlo methods via simulator $\mathcal{S}$. The approximation guarantee and computational complexity follows from Proposition 8 and Lemma 22, and we omit here. □

C.7 Technical tools and proofs

Theorem (Hoeffding's inequality)**.** *Suppose that for some $n \geq 1$, $\{X_i, i \in [1,n]\}$ are i.i.d., and $P(X_1 \in [0,1]) = 1$. Then $\mathbb{P}\Big(\Big|n^{-1}\sum_{i=1}^{n} X_i - E[X_1]\Big| \geq \eta\Big) \leq 2\exp\Big(-2\eta^2 n\Big)$.*

Lemma 37. *The recursion holds true for $k \geq 3, \epsilon, \delta \in (0,1)$*

$$R_a(k,\epsilon,\delta) \;\geq\; k + CN + N\Big(R_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + R_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big) + 2N$$

and

$$S_a(k,\epsilon,\delta) \;\geq\; N + N\Big(S_a(k_1, \frac{\epsilon}{16}, \frac{\epsilon}{16}) + S_a(k_2, \frac{\epsilon}{16}, \frac{\epsilon}{16})\Big).$$

where $N = 8\epsilon^{-2}\log(4/\delta)$ and $k_1 + k_2 = k + 1$.

Proof. We recall that functions R_a and S_a are given by

$$R_a(k,\epsilon,\delta) \;=\; (1 + \log(\frac{1}{\delta})) \times (\frac{1}{\epsilon})^{3(k-2)} \times 100^{(k-2)^2} \times (C + \nu_1(\epsilon 16^{-(k-2)}, \delta \wedge \epsilon 16^{-(k-2)})),$$

and

$$S_a(k,\epsilon,\delta) \;=\; (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon})^{3(k-2)}\times 100^{(k-2)^2}\times\nu_2(\epsilon 16^{-(k-2)},\delta\wedge\epsilon 16^{-(k-2)}).$$

We reason as follows.

$$\begin{aligned}
&k+CN+N\Big(R_a(k_1,\frac{\epsilon}{16},\frac{\epsilon}{16})+R_a(k_2,\frac{\epsilon}{16},\frac{\epsilon}{16})\Big)+2N\\
&\quad\le 2\times N\times R_a(k-1,\frac{\epsilon}{16},\frac{\epsilon}{16})+k+C\times N\\
&\quad= 2\times\frac{8}{\epsilon^2}\times\log(\frac{4}{\delta})\times(1+\log(\frac{16}{\epsilon}))\times(\frac{16}{\epsilon})^{3k-9}\times 100^{(k-3)^2}\times\Big(C+\nu_1(\epsilon 16^{-(k-2)},\frac{\epsilon}{16}\wedge\epsilon 16^{-(k-2)})\Big)\\
&\qquad\qquad +k+C\times\frac{8}{\epsilon^2}\times\log(\frac{4}{\delta})\\
&\quad\le (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon})^{3k-7}\times\log(\frac{1}{\epsilon})\times 64\times 16^{3k-9}\times 100^{(k-3)^2}\times(C+\nu_1(\epsilon 16^{-(k-2)},\epsilon 16^{-(k-2)}))\\
&\qquad\qquad +k+C\times\frac{8}{\epsilon^2}\times(1+\log(\frac{1}{\delta}))\\
&\quad\le (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon})^{3k-6}\times(64^{2k-5}+1)\times 100^{(k-3)^2}\times(C+\nu_1(\epsilon 16^{-(k-2)},\epsilon 16^{-(k-2)}))\\
&\quad\le (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon})^{3(k-2)}\times 100^{(k-2)^2}\times(C+\nu_1(\epsilon 16^{-(k-2)},\delta\wedge\epsilon 16^{-(k-2)}))\\
&\quad= R_a(k,\epsilon,\delta)
\end{aligned}$$

The proof for S_a is nearly identical and we omit here. □

Lemma 38. *The recursion holds true for $k\ge 3,\epsilon,\delta\in(0,1)$*

$$R_b(k,\epsilon,\delta) \;\ge\; k+R_a(k_1,\epsilon\delta/16,\delta/8)+R_a(k_2,\epsilon\delta/16,\delta/8)+R_b(k_i,\epsilon/2,\delta/2)+3$$

and

$$S_b(k,\epsilon,\delta) \;\ge\; S_a(k_1,\epsilon\delta/16,\delta/8)+S_a(k_2,\epsilon\delta/16,\delta/8)+S_b(k_i,\epsilon/2,\delta/2).$$

Proof. We recall that functions R_a and S_a are given by

$$R_b(k,\epsilon,\delta) \;=\; (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon\delta})^{3(k-1)}\times 100^{(k-1)^2}\times\Big(C+\nu_1(\frac{\epsilon\delta}{4^{2(k-2)}},\frac{\epsilon\delta}{4^{2(k-2)}})\Big),$$

and

$$S_b(k,\epsilon,\delta) \;=\; (1+\log(\frac{1}{\delta}))\times(\frac{1}{\epsilon\delta})^{3(k-1)}\times 100^{(k-1)^2}\times\nu_2(\frac{\epsilon\delta}{4^{2(k-2)}},\frac{\epsilon\delta}{4^{2(k-2)}})$$

We reason as follows.

$$
\begin{aligned}
& k + R_a(k_1, \frac{\epsilon\delta}{16}, \frac{\delta}{8}) + R_a(k_2, \frac{\epsilon\delta}{16}, \frac{\delta}{8}) + R_b(k_i, \frac{\epsilon}{2}, \frac{\delta}{2}) + 3 \\
&\leq\ k + 3 + 2R_a(k-1, \frac{\epsilon\delta}{16}, \frac{\delta}{8}) + R_b(k-1, \frac{\epsilon}{2}, \frac{\delta}{2}) \\
&\leq\ 2(k+3) + 2R_a(k-1, \frac{\epsilon\delta}{16}, \frac{\delta}{8}) + 2R_a(k-2, \frac{\epsilon\delta}{4\times 16}, \frac{\delta}{2\times 8}) + R_b(k-2, \frac{\epsilon}{4}, \frac{\delta}{4}) \\
&\leq\ (k-2)(k+3) + 2\sum_{i=1}^{k-2} R_a(k-i, \frac{\epsilon\delta}{16\times 4^{i-1}}, \frac{\delta}{8\times 2^{i-1}}) + R_b(2, \frac{\epsilon}{2^{k-2}}, \frac{\delta}{2^{k-2}}) \\
&\leq\ (k-2)(k+3) + 2\sum_{i=1}^{k-2} R_a(k-i, \frac{\epsilon\delta}{16\times 4^{i-1}}, \frac{\delta}{8\times 2^{i-1}}) + \nu_1(\frac{\epsilon}{2^{k-2}}, \frac{\delta}{2^{k-2}}) \\
&=\ (k-2)(k+3) + \nu_1(\frac{\epsilon}{2^{k-2}}, \frac{\delta}{2^{k-2}}) \\
&\qquad + 2\sum_{i=1}^{k-2} (1 + \log(\frac{2^{i+2}}{\delta})) \times (\frac{4^{i+1}}{\epsilon\delta})^{3(k-i-2)} \times 100^{(k-i-2)^2} \times \Big(C + \nu_1(\frac{\epsilon\delta}{4^{2k-i-3}}, \frac{\epsilon\delta}{4^{2k-i-3}})\Big) \\
&\leq\ (1 + \log(\frac{1}{\delta})) \times (\frac{1}{\epsilon\delta})^{3(k-1)} \times 100^{(k-1)^2} \times \Big(C + \nu_1(\frac{\epsilon\delta}{4^{2(k-2)}}, \frac{\epsilon\delta}{4^{2(k-2)}})\Big) \\
&=\ R_b(k, \epsilon, \delta).
\end{aligned}
$$

The proof for S_b is nearly identical and we omit here. □

Lemma 39. *The following recursion holds true for $k, l \geq 1, \eta, \epsilon, \delta \in (0,1)$.*

$$
\begin{aligned}
R_2(\eta, k, l, \epsilon, \delta) \quad \geq \quad & R_2(\eta, k-1, l, \epsilon/2, \delta/2) + \Delta^l \\
& + \ \Delta^l \times (1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \\
& \qquad \times \Big(C + \mu_1(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + R_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big).
\end{aligned}
$$

and

$$
\begin{aligned}
S_2(\eta, k, l, \epsilon, \delta) \quad \geq \quad & S_2(\eta, k-1, l, \epsilon/2, \delta/2) \\
& + \ \Delta^l \times (1 + \log(\frac{2\Delta^l}{\delta})) \times (\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)} \times 100^{3(l+2)^2} \\
& \qquad \times \Big(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}) + S_2(\eta, k-1, l, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l}, \frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big).
\end{aligned}
$$

Proof. Recall that

$$
R_2(\eta, k, l, \epsilon, \delta) \ = \ \Big(\frac{(100\Delta)^l}{\epsilon\delta}\Big)^{4^k} \Big(C + \mu_1\Big((\epsilon\delta(100\Delta)^{-l})^{4^k}, (\epsilon\delta(100\Delta)^{-l})^{4^k}\Big)
$$

and

$$S_2(\eta,k,l,\epsilon,\delta) \;=\; \Big(\frac{(100\Delta)^l}{\epsilon\delta}\Big)^{4^k}\times\mu_2\Big((\epsilon\delta(100\Delta)^{-l})^{4^k},(\epsilon\delta(100\Delta)^{-l})^{4^k}\Big).$$

We reason as follows.

$$\begin{aligned}
&S_2(\eta,k-1,l,\epsilon/2,\delta/2)\\
&+\;\Delta^l\times(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)}\times 100^{3(l+2)^2}\\
&\qquad\times\Big(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})+S_2(\eta,k-1,l,\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big)\\
&\le\;2(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)+1}\times 100^{3(l+2)^2}\\
&\qquad\times\Big(\mu_2(\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})+S_2(\eta,k-1,l,\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\Big)\\
&\le\;4(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)+1}\times 100^{3(l+2)^2}\times S_2(\eta,k-1,l,\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l},\frac{\epsilon\delta}{4^{2(l+2)}\Delta^l})\\
&=\;4(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)+1}\times 100^{3(l+2)^2}\\
&\qquad\times\Big(\frac{(100\Delta)^l\times 16^2\times(16\Delta)^{2l}}{\epsilon^2\delta^2}\Big)^{4^{k-1}}\\
&\qquad\times\;\mu_2\Big(\Big(\frac{\epsilon^2\delta^2}{(100\Delta)^l\times 16^2\times(16\Delta)^{2l}}\Big)^{4^{k-1}},\Big(\frac{\epsilon^2\delta^2}{(100\Delta)^l\times 16^2\times(16\Delta)^{2l}}\Big)^{4^{k-1}}\Big)\\
&\le\;4(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)+1}\times 100^{3(l+2)^2}\\
&\qquad\times\Big(\frac{(100\Delta)^l\times 16^2\times(16\Delta)^{2l}}{\epsilon^2\delta^2}\Big)^{4^{k-1}}\times\mu_2\Big(\Big(\frac{\epsilon^2\delta^2}{(100\Delta)^{4l}}\Big)^{4^{k-1}},\Big(\frac{\epsilon^2\delta^2}{(100\Delta)^{4l}}\Big)^{4^{k-1}}\Big)\\
&\le\;4(1+\log(\frac{2\Delta^l}{\delta}))\times(\frac{4\Delta^l}{\epsilon\delta})^{3(l+2)+1}\times 100^{3(l+2)^2}\\
&\qquad\times\Big(\frac{(100\Delta)^l\times 16^2\times(16\Delta)^{2l}}{\epsilon^2\delta^2}\Big)^{4^{k-1}}\times\mu_2\Big(\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k},\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k}\Big)\\
&\le\;(\frac{1}{\epsilon\delta})^{6l}\times\Delta^{3(l+2)^2}\times 100^{4(l+2)^2}\\
&\qquad\times\Big(\frac{(100\Delta)^{3l}}{\epsilon^2\delta^2}\Big)^{4^{k-1}}\times\mu_2\Big(\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k},\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k}\Big)\\
&\le\;\Big(\frac{(100\Delta)^l}{\epsilon\delta}\Big)^{6l}\times\Big(\frac{(100\Delta)^l}{\epsilon\delta}\Big)^{3\times 4^{k-1}}\times\mu_2\Big(\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k},\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k}\Big)\\
&\le\;\Big(\frac{(100\Delta)^l}{\epsilon\delta}\Big)^{\times 4^k}\times\mu_2\Big(\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k},\Big(\frac{\epsilon\delta}{(100\Delta)^l}\Big)^{4^k}\Big)\qquad(\text{since } k\gg l)\\
&=\;S_2(\eta,k,l,\epsilon,\delta).
\end{aligned}$$

The proof for R_2 is nearly identical and we omit here. □

Lemma 40. *The following recursion holds true for $L \geq 3, \eta, \epsilon, \delta \in (0,1)$.*

$$\begin{aligned} R_3(\eta, L, \epsilon, \delta) &\geq R_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \\ &\times \left(2C + R_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right) \end{aligned}$$

and

$$\begin{aligned} S_3(\eta, L, \epsilon, \delta) &\geq S_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \\ &\times \left(1 + S_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right). \end{aligned}$$

Proof. Recall that

$$R_3(\eta, L, \epsilon, \delta) = C\left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta^L}}$$

and

$$S_3(\eta, L, \epsilon, \delta) = \left(\frac{(100\Delta)^L}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta^L}}$$

We reason as follow.

$$
\begin{aligned}
& S_3(\eta, L-2, \frac{\epsilon}{3}, \frac{\delta}{3}) + 2 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \\
& \qquad \times \left(1 + S_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right)\right) \\
\leq \quad & 3 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \\
& \qquad \times \; S_3\left(\eta, L-2, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}, (9^{-1}\epsilon\delta(100\Delta)^{-(L-2)})^{4^{\eta^{-1}\Delta^{L-2}}}\right) \\
= \quad & 3 \times \left(\frac{9(100\Delta)^{L-2}}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}}} \\
& \qquad \times \left((100\Delta)^{L-2}(\frac{9(100\Delta)^{L-2}}{\epsilon\delta})^{2\times 4^{\eta^{-1}\Delta^{L-2}}}\right)^{4^{\eta^{-1}(L-2)\Delta^{L-2}}} \\
\leq \quad & 3 \times \left(\frac{1}{\epsilon\delta}\right)^{4^{\eta^{-1}\Delta^{L-2}} + 2\times 4^{\eta^{-1}\Delta^{L-2}} \times 4^{\eta^{-1}(L-2)\Delta^{L-2}}} \\
& \qquad \times \; (9(100\Delta)^{L-2})^{4^{\eta^{-1}\Delta^{L-2}} + 4^{\eta^{-1}(L-2)\Delta^{L-2}} + 2\times 4^{\eta^{-1}\Delta^{L-2}} \times 4^{\eta^{-1}(L-2)\Delta^{L-2}}} \\
\leq \quad & \left(\frac{1}{\epsilon\delta}\right)^{4^{\eta^{-1}L\Delta^L}} \times ((100\Delta)^L)^{4^{\eta^{-1}L\Delta^L}} \\
= \quad & S_3(\eta, L, \epsilon, \delta).
\end{aligned}
$$

The proof for R_3 is nearly identical and we omit here. □

BIBLIOGRAPHY

[1] Melika Abolhassani, Soheil Ehsani, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Robert Kleinberg, and Brendan Lucier. Beating 1-1/e for ordered prophets. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 61–71, 2017.

[2] Dale D Achabal, Shelby McIntyre, and Stephen A Smith. Maximizing profits from periodic department store promotions. *Journal of Retailing*, 66(4):383, 1990.

[3] Yves Achdou and Olivier Pironneau. *Computational methods for option pricing*, volume 30. Siam, 2005.

[4] Elodie Adida and Georgia Perakis. A robust optimization approach to dynamic pricing and inventory control with no backorders. *Mathematical Programming*, 107(1-2):97–129, 2006.

[5] Hyun-Soo Ahn, Christopher Ryan, Joline Uichanco, and Mengzhenyu Zhang. Certainty equivalent pricing under sales-dependent and inventory-dependent demand. *Available at SSRN 3502478*, 2019.

[6] Se-Ryoong Ahn, Hyeong-Ohk Bae, Hyeng-Keun Koo, and Ki-Jung Lee. A survey on american options: old approaches and new trends. *Bulletin of the Korean Mathematical Society*, 48(4):791–812, 2011.

[7] Amir Ajorlou, Ali Jadbabaie, and Ali Kakhbod. Dynamic pricing in social networks: The word-of-mouth effect. *Management Science*, 64(2):971–979, 2016.

[8] David Aldous. Asymptotics in the random assignment problem. *Probability Theory and Related Fields*, 93(4):507–534, 1992.

[9] David Aldous and J Michael Steele. The objective method: probabilistic combinatorial optimization and local weak convergence. In *Probability on discrete structures*, pages 1–72. Springer, 2004.

[10] David J Aldous. The ζ (2) limit in the random assignment problem. *Random Structures & Algorithms*, 18(4):381–418, 2001.

[11] Nikolay Aleksandrov and BM Hambly. A dual approach to multiple exercise option problems under constraints. *Mathematical methods of operations research*, 71(3):503–533, 2010.

[12] Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Repeated contextual auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, pages 622–630, 2014.

[13] Leif Andersen and Mark Broadie. Primal-dual simulation algorithm for pricing multidimensional american options. *Management Science*, 50(9):1222–1234, 2004.

[14] Victor F Araman and Rene Caldentey. Dynamic pricing for nonperishable products with demand learning. *Operations research*, 57(5):1169–1188, 2009.

[15] Alessandro Arlotto and Itai Gurvich. Uniformly bounded regret in the multisecretary problem. *Stochastic Systems*, 9(3):231–260, 2019.

[16] Alessandro Arlotto and Xinchang Xie. Logarithmic regret in the dynamic and stochastic knapsack problem with equal rewards. *Stochastic Systems*, 2020.

[17] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products.

[18] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd. Quantum-inspired algorithms in practice. *arXiv preprint arXiv:1905.10415*, 2019.

[19] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

[20] Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

[21] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. 2007.

[22] Subramanian Balachander, Yan Liu, and Axel Stock. An empirical analysis of scarcity strategies in the automobile industry. *Management Science*, 55(10):1623–1637, 2009.

[23] Sven Balder, Antje Mahayni, and John Schoenmakers. Primal-dual linear monte carlo algorithm for multiple stopping - an application to flexible caps. *Quantitative Finance*, 13(7):1003–1013, 2013.

[24] Gah-Yi Ban and N Bora Keskin. Personalized dynamic pricing with machine learning. *Available at SSRN 2972985*, 2018.

[25] Olivier Bardou, Sandrine Bouthemy, and Gilles Pages. Optimal quantization for the pricing of swing options. *Applied Mathematical Finance*, 16(2):183–217, 2009.

[26] Jérôme Barraquand and Didier Martineau. Numerical valuation of high dimensional multivariate american securities. *Journal of financial and quantitative analysis*, pages 383–405, 1995.

[27] Christophe Barrera-Esteve, Florent Bergeret, Charles Dossal, Emmanuel Gobet, Asma Meziou, Remi Munos, and Damien Reboul-Salze. Numerical methods for the pricing of swing options: a stochastic control approach. *Methodology and computing in applied probability*, 8(4):517–540, 2006.

[28] Kengy Barty, Pierre Girardeau, Cyrille Strugarek, and Jean-Sébastien Roy. Application of kernel-based stochastic gradient algorithms to option pricing. *Monte Carlo Methods and Applications*, 14(2):99–127, 2008.

[29] Christian Bayer, Martin Redmann, and John Schoenmakers. Dynamic programming for optimal stopping via pseudo-regression. *arXiv preprint arXiv:1808.04725*, 2018.

[30] Erhan Bayraktar and Song Yao. On the robust optimal stopping problem. *SIAM Journal on Control and Optimization*, 52(5):3135–3175, 2014.

[31] Christian Beck, Arnulf Jentzen, and Thomas Kruse. Nonlinear monte carlo methods with polynomial runtime for high-dimensional iterated nested expectations. *arXiv preprint arXiv:2009.13989*, 2020.

[32] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 20:74, 2019.

[33] Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Timo Welti. Solving high-dimensional optimal stopping problems using deep learning. *arXiv preprint arXiv:1908.01602*, 2019.

[34] Denis Belomestny. Pricing bermudan options by nonparametric regression: optimal rates of convergence for lower estimates. *Finance and Stochastics*, 15(4):655–683, 2011.

[35] Denis Belomestny, Christian Bender, and John Schoenmakers. True upper bounds for bermudan products via non-nested monte carlo. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 19(1):53–71, 2009.

[36] Denis Belomestny et al. On the rates of convergence of simulation-based optimization algorithms for optimal stopping problems. *The Annals of Applied Probability*, 21(1):215–239, 2011.

[37] Denis Belomestny, Roland Hildebrand, and John Schoenmakers. Optimal stopping via pathwise dual empirical maximisation. *Applied Mathematics & Optimization*, 79(3):715–741, 2019.

[38] Denis Belomestny, Maxim Kaledin, and J Schoenmakers. Semi-tractability of optimal stopping problems via a weighted stochastic mesh algorithm. *arXiv preprint arXiv:1906.09431*, 2019.

[39] Denis Belomestny and Grigori N Milstein. Monte carlo evaluation of american options using consumption processes. *International Journal of theoretical and applied finance*, 9(04):455–481, 2006.

[40] Denis Belomestny and John Schoenmakers. *Advanced Simulation-Based Methods for Optimal Stopping and Control: With Applications in Finance*. Springer, 2018.

[41] Denis Belomestny, John Schoenmakers, and Fabian Dickmann. Multilevel dual approach for pricing american style derivatives. *Finance and Stochastics*, 17(4):717–742, 2013.

[42] Denis Belomestny, John Schoenmakers, Vladimir Spokoiny, and Yuri Tavyrikov. Optimal stopping via deeply boosted backward regression. *arXiv preprint arXiv:1808.02341*, 2018.

[43] Christian Bender. Dual pricing of multi-exercise options under volume constraints. *Finance and Stochastics*, 15(1):1–26, 2011.

[44] Christian Bender. Primal and dual pricing of multiple exercise options in continuous time. *SIAM Journal on Financial Mathematics*, 2(1):562–586, 2011.

[45] Christian Bender, Christian Gärtner, and Nikolaus Schweizer. Pathwise dynamic programming. *Mathematics of Operations Research*, 43(3):965–995, 2018.

[46] Christian Bender and John Schoenmakers. An iterative method for multiple stopping: convergence and stability. *Advances in applied probability*, 38(3):729–749, 2006.

[47] Christian Bender, John Schoenmakers, and Jianing Zhang. Dual representations for general multiple stopping problems. *Mathematical Finance*, 25(2):339–370, 2015.

[48] Bernard Bensaid and Jean-Philippe Lesne. Dynamic monopoly pricing with network externalities. *International Journal of Industrial Organization*, 14(6):837–855, 1996.

[49] Fred Espen Benth, Jukka Lempa, and Trygve Kastberg Nilssen. On the optimal exercise of swing options in electricity markets. *Journal of Energy Markets*, 4(4):3–28, 2012.

[50] Dimitris Bertsimas and Georgia Perakis. Dynamic pricing: A learning approach. In *Mathematical and computational models for congestion charging*, pages 45–79. Springer, 2006.

[51] Dimitris Bertsimas and Ioana Popescu. On the relation between option and stock prices: a convex optimization approach. *Operations Research*, 50(2):358–374, 2002.

[52] Omar Besbes, Adam N Elmachtoub, and Yunjie Sun. Static pricing: Universal guarantees for reusable resources. *arXiv preprint arXiv:1905.00731*, 2019.

[53] Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57(6):1407–1420, 2009.

[54] Christopher Beveridge, Mark S Joshi, and Robert Tang. Practical policy iteration: generic methods for obtaining rapid and tight bounds for bermudan exotic derivatives using monte carlo simulation. *Available at SSRN 1331904*, 2009.

[55] Sérgio C Bezerra, Alberto Ohashi, Francesco Russo, and Francys de Souza. Discrete-type approximations for non-markovian optimal stopping problems: Part ii. *Methodology and Computing in Applied Probability*, pages 1–35, 2018.

[56] Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference On Learning Theory*, pages 1691–1692. PMLR, 2018.

[57] Gabriel R Bitran and Susana V Mondschein. Periodic pricing of seasonal products in retailing. *Management science*, 43(1):64–79, 1997.

[58] Jose Blanchet, Guillermo Gallego, and Vineet Goyal. A markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905, 2016.

[59] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[60] Bruno Bouchard and Xavier Warin. Monte-carlo valuation of american options: facts and new algorithms to improve existing methods. In *Numerical methods in finance*, pages 215–255. Springer, 2012.

[61] Mark Braverman and Kanika Pasricha. The computational hardness of pricing compound options. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 103–104, 2014.

[62] Mark Broadie and Menghui Cao. Improved lower and upper bound algorithms for pricing american options by simulation. *Quantitative Finance*, 8(8):845–861, 2008.

[63] Mark Broadie and Paul Glasserman. Pricing american-style securities using simulation. *Journal of economic dynamics and control*, 21(8-9):1323–1352, 1997.

[64] Mark Broadie, Paul Glasserman, et al. A stochastic mesh method for pricing high-dimensional american options. *Journal of Computational Finance*, 7:35–72, 2004.

[65] Josef Broder and Paat Rusmevichientong. Dynamic pricing under a general parametric choice model. *Operations Research*, 60(4):965–980, 2012.

[66] David B Brown and James E Smith. Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Science*, 57(10):1752–1770, 2011.

[67] David B Brown and James E Smith. Information relaxations, duality, and convex stochastic dynamic programs. *Operations Research*, 62(6):1394–1415, 2014.

[68] David B Brown, James E Smith, and Peng Sun. Information relaxations and duality in stochastic dynamic programs. *Operations research*, 58(4-part-1):785–801, 2010.

[69] F Thomas Bruss. What is known about robbins' problem? *Journal of applied probability*, 42(1):108–120, 2005.

[70] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[71] Pornpawee Bumpensanti and He Wang. A re-solving heuristic with uniformly bounded loss for network revenue management. *arXiv preprint arXiv:1802.06192*, 2018.

[72] Pornpawee Bumpensanti and He Wang. A re-solving heuristic with uniformly bounded loss for network revenue management. *Management Science*, 2020.

[73] Luis MB Cabral, David J Salant, and Glenn A Woroch. Monopoly pricing with network externalities. *International Journal of Industrial Organization*, 17(2):199–214, 1999.

[74] Ozan Candogan, Kostas Bimpikis, and Asuman Ozdaglar. Optimal pricing in the presence of local network effects. In *International Workshop on Internet and Network Economics*, pages 118–132. Springer, 2010.

[75] René Carmona and Michael Ludkovski. Swing options. *Encyclopedia of Quantitative Finance*, 2010.

[76] Felipe Caro and Jeremie Gallien. Clearance pricing optimization for a fast-fashion retailer. *Operations Research*, 60(6):1404–1422, 2012.

[77] Sabri Celik, Alp Muharremoglu, and Sergei Savin. Revenue management with costly price adjustments. *Operations research*, 57(5):1206–1219, 2009.

[78] Shyam S Chandramouli and Martin B Haugh. A unified approach to multiple stopping and duality. *Operations Research Letters*, 40(4):258–264, 2012.

[79] Nan Chen and Paul Glasserman. Additive and multiplicative duals for american option pricing. *Finance and Stochastics*, 11(2):153–179, 2007.

[80] Qi Chen, Stefanus Jasin, and Izak Duenyas. Real-time dynamic pricing with minimal and flexible price adjustment. *Management Science*, 62(8):2437–2455, 2015.

[81] Shuhang Chen, Adithya M Devraj, Ana Bušić, and Sean Meyn. Zap q-learning for´ optimal stopping. In *2020 American Control Conference (ACC)*, pages 3920–3925. IEEE, 2020.

[82] Xin Chen, Peng Hu, and Zhenyu Hu. Efficient algorithms for the dynamic pricing problem with reference price effect. *Management Science*, 63(12):4389–4408, 2016.

[83] Xin Chen, Peng Hu, Stephen Shum, and Yuhan Zhang. Dynamic stochastic inventory management with reference price effects. *Operations Research*, 64(6):1529–1536, 2016.

[84] Yichen Chen and Mengdi Wang. Lower bound on the computational complexity of discounted markov decision problems. *arXiv preprint arXiv:1705.07312*, 2017.

[85] Yilun Chen and David Goldberg. Bridging optimal stopping and max-flow min-cut: a theory of duality. 2020.

[86] Yiwei Chen and Vivek F Farias. Robust dynamic pricing with strategic customers. *Mathematics of Operations Research*, 43(4):1119–1142, 2018.

[87] Yiwei Chen, Vivek F Farias, and Nikolaos K Trichakis. On the efficacy of static prices for revenue management in the face of strategic customers. *Management Science*, 2018.

[88] Wang Chi Cheung, David Simchi-Levi, and He Wang. Dynamic pricing and demand learning with limited price experimentation. *Operations Research*, 65(6):1722–1731, 2017.

[89] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. *arXiv preprint arXiv:1901.03254*, 2019.

[90] YS Chow, H Robbins, and D Siegmund. Great expectations: The theory of optimal stopping. 1971.

[91] Sören Christensen. A method for pricing american options using semi-infinite linear programming. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 24(1):156–172, 2014.

[92] Dragos Florin Ciocan and Velibor V Mišić. Interpretable optimal stopping. *Management Science*, 2020.

[93] Emmanuelle Clément, Damien Lamberton, and Philip Protter. An analysis of a least squares regression method for american option pricing. *Finance and Stochastics*, 6(4):449–471, 2002.

[94] Edith Cohen. Approximate max-flow on small depth networks. *SIAM Journal on Computing*, 24(3):579–597, 1995.

[95] Maxime Cohen, Swati Gupta, Jeremy Kalas, and Georgia Perakis. An efficient algorithm for dynamic pricing using a graphical representation. *Available at SSRN 2772231*, 2016.

[96] Maxime Cohen, Ilan Lobel, and Renato Paes Leme. Feature-based dynamic pricing. *Available at SSRN 2737045*, 2016.

[97] Maxime C Cohen, Ngai-Hang Zachary Leung, Kiran Panchamgam, Georgia Perakis, and Anthony Smith. The impact of linear optimization on promotion planning. *Operations Research*, 65(2):446–468, 2017.

[98] Tamar Cohen-Hillel, Kiran Panchamgam, and Georgia Perakis. Bounded memory peak end models can be surprisingly good. *Available at SSRN 3360643*, 2019.

[99] Tamar Cohen-Hillel, Kiran Panchamgam, and Georgia Perakis. High-low promotion policies for peak-end demand models. *Available at SSRN 3360680*, 2019.

[100] William L Cooper and Bharath Rangarajan. Performance guarantees for empirical markov decision processes with applications to multiperiod inventory models. *Operations Research*, 60(5):1267–1281, 2012.

[101] José Correa, Paul Dütting, Felix Fischer, and Kevin Schewior. Prophet inequalities for iid random variables from an unknown distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 3–17, 2019.

[102] José Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms for a random stream of customers. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 169–186, 2017.

[103] Ruomeng Cui, Dennis J Zhang, and Achal Bassamboo. Learning from inventory availability information: Evidence from field experiments on amazon. *Management Science*, 65(3):1216–1235, 2018.

[104] M Dahlgren. A continuous time model to price commodity-based swing options. *Review of derivatives research*, 8(1):27–47, 2005.

[105] Martin Dahlgren and Ralf Korn. The swing option on the stock market. *International Journal of Theoretical and Applied Finance*, 8(01):123–139, 2005.

[106] Mark HA Davis and Ioannis Karatzas. A deterministic approach to optimal stopping. *Probability, Statistics and Optimisation (ed. FP Kelly). NewYork Chichester: John Wiley & Sons Ltd*, pages 455–466, 1994.

[107] Arnoud V den Boer and Bert Zwart. Simultaneously learning and optimizing using controlled variance pricing. *Management science*, 60(3):770–783, 2013.

[108] Vijay V Desai, Vivek F Farias, and Ciamac C Moallemi. Pathwise optimization for optimal stopping problems. *Management Science*, 58(12):2292–2308, 2012.

[109] Uwe Dorr. Valuation of swing options and examination of exercise strategies by monte carlo techniques. *Mathematical Finance*, 10:27, 2003.

[110] Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning?, 2020.

[111] Daniel Egloff et al. Monte carlo algorithms for optimal stopping and statistical learning. *The Annals of Applied Probability*, 15(2):1396–1432, 2005.

[112] Daniel Egloff, Michael Kohler, Nebojsa Todorovic, et al. A dynamic look-ahead monte carlo algorithm for pricing bermudan options. *The Annals of Applied Probability*, 17(4):1138–1171, 2007.

[113] Marcus Eriksson, Jukka Lempa, and Trygve Kastberg Nilssen. Swing options in commodity markets: a multidimensional levy diffusion model. *Mathematical Methods of Operations Research*, 79(1):31–67, 2014.

[114] Guy Even, Moti Medina, and Dana Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. In *European Symposium on Algorithms*, pages 394–405. Springer, 2014.

[115] Vivek F Farias and Benjamin Van Roy. Dynamic pricing with a prior on market response. *Operations Research*, 58(1):16–29, 2010.

[116] Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Conference on Learning Theory*, pages 637–657, 2015.

[117] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.

[118] Kris Johnson Ferreira, David Simchi-Levi, and He Wang. Online network revenue management using thompson sampling. *Operations research*, 66(6):1586–1602, 2018.

[119] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[120] Marcelo Figueroa. Pricing multiple interruptible-swing contracts. 2006.

[121] Masaaki Fujii, Akihiko Takahashi, and Masayuki Takahashi. Asymptotic

expansion as prior knowledge in deep learning method for high dimensional bsdes. *Asia-Pacific Financial Markets*, 26(3):391–408, 2019.

[122] Guillermo Gallego, Robert Phillips, and Ozge Sahin. Strategic management of distressed inventory. *Production and Operations Management*, 17(4):402–415, 2008.

[123] Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020, 1994.

[124] David Gamarnik and David Goldberg. Randomized greedy algorithms for independent sets and matchings in regular graphs: Exact results and finite girth corrections. *arXiv preprint arXiv:0807.1277*, 2008.

[125] David Gamarnik, David A Goldberg, and Theophane Weber. Correlation decay in random decision networks. *Mathematics of Operations Research*, 39(2):229–261, 2014.

[126] David Gamarnik, Tomasz Nowicki, and Grzegorz Swirszcz. Maximum weight independent sets and matchings in sparse random graphs. exact results using the local weak convergence method. *Random Structures & Algorithms*, 28(1):76–106, 2006.

[127] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems.

[128] John P Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. In *Selected Papers of Frederick Mosteller*, pages 355–398. Springer, 2006.

[129] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[130] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

[131] Paul Glasserman, Bin Yu, et al. Number of paths versus number of basis functions in american option pricing. *The Annals of Applied Probability*, 14(4):2090–2119, 2004.

[132] Oliver Göbel, Martin Hoefer, Thomas Kesselheim, Thomas Schleiden, and Berthold Vöcking. Online independent set beyond the worst-case: Secretaries, prophets, and periods. In *International Colloquium on Automata, Languages, and Programming*, pages 508–519. Springer, 2014.

[133] Karan Goel, Christoph Dann, and Emma Brunskill. Sample efficient policy search for optimal stopping domains. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1711–1717, 2017.

[134] David A Goldberg and Yilun Chen. Beating the curse of dimensionality in options pricing and optimal stopping. *arXiv preprint arXiv:1807.02227*, 2018.

[135] A Goldenshluger and A Zeevi. Optimal stopping of a random sequence with unknown distribution. Technical report, Tech. rep., Workig paper, Graduate School of Business, Columbia University. 183, 2017.

[136] Eric A Greenleaf. The impact of reference price effects on the profitability of price promotions. *Marketing science*, 14(1):82–104, 1995.

[137] M Guray Guler, Taner Bilgic, and Refik Gullu. Joint inventory and pricing decisions with reference effects. *IIE Transactions*, 46(4):330–343, 2014.

[138] Gido Haarbrucker and Daniel Kuhn. Valuation of electricity swing options by multistage stochastic programming. *Automatica*, 45(4):889–899, 2009.

[139] Nir Halman, Diego Klabjan, Chung-Lun Li, James Orlin, and David Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *SIAM Journal on Discrete Mathematics*, 28(4):1725–1796, 2014.

[140] Nir Halman, Giacomo Nannicini, and James Orlin. A computationally efficient fptas for convex stochastic dynamic programs. *SIAM Journal on Optimization*, 25(1):317–350, 2015.

[141] J Michael Harrison, N Bora Keskin, and Assaf Zeevi. Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. *Management Science*, 58(3):570–586, 2012.

[142] Martin B Haugh and Leonid Kogan. Pricing american options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

[143] Elad Hazan. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*, 2019.

[144] Billy P Helms, Fred R Kaen, and Robert E Rosenman. Memory in commodity futures contracts. *The Journal of Futures Markets (pre-1986)*, 4(4):559, 1984.

[145] Theodore P Hill, Robert P Kertz, et al. Comparisons of stop rule and supremum expectations of iid random variables. *The Annals of Probability*, 10(2):336–345, 1982.

[146] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[147] Tsu-Pang Hsieh and Chung-Yuan Dye. Optimal dynamic pricing for deteriorating items with reference price effects when inventories stimulate demand. *European Journal of Operational Research*, 262(1):136–150, 2017.

[148] Zhenyu Hu, Xin Chen, and Peng Hu. Dynamic pricing with gain-seeking reference price effects. *Operations Research*, 64(1):150–157, 2016.

[149] Alfredo Ibanez. Valuation by simulation of contingent claims with multiple early exercise opportunities. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 14(2):223–248, 2004.

[150] Alfredo Ibáñez and Carlos Velasco. Recursive lower and dual upper bounds for bermudan-style options. *European Journal of Operational Research*, 280(2):730–740, 2020.

[151] Patrick Jaillet, Damien Lamberton, and Bernard Lapeyre. Variational inequalities and the pricing of american options. *Acta Applicandae Mathematica*, 21(3):263–289, 1990.

[152] Patrick Jaillet, Ehud I Ronn, and Stathis Tompaidis. Valuation of commodity-based swing options. *Management science*, 50(7):909–921, 2004.

[153] Farshid Jamshidian. The duality of optimal exercise and domineering claims: A doob–meyer decomposition approach to the snell envelope. *Stochastics An International Journal of Probability and Stochastic Processes*, 79(1-2):27–60, 2007.

[154] Adel Javanmard. Perishability of data: dynamic pricing under varying-coefficient models. *The Journal of Machine Learning Research*, 18(1):1714–1744, 2017.

[155] Adel Javanmard and Hamid Nazerzadeh. Dynamic pricing in high-dimensions. *arXiv preprint arXiv:1609.07574*, 2016.

[156] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low bellman rank are pac-learnable, 2016.

[157] Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms, 2021.

[158] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[159] Mark S Joshi. A new class of dual upper bounds for early exercisable derivatives encompassing both the additive and multiplicative bounds. *Operations Research Letters*, 43(6):581–585, 2015.

[160] Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

[161] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

[162] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[163] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2-3):193–208, 2002.

[164] Michael J Kearns, Yishay Mansour, and Andrew Y Ng. Approximate planning in large pomdps via reusable trajectories. In *NIPS*, pages 1001–1007. Citeseer, 1999.

[165] Michael J Kearns, Yishay Mansour, and Andrew Y Ng. Approximate plan-

ning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems*, pages 1001–1007, 2000.

[166] N Bora Keskin and Assaf Zeevi. Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. *Operations Research*, 62(5):1142–1167, 2014.

[167] Thomas Kesselheim, Robert Kleinberg, and Rad Niazadeh. Secretary problems with non-uniform arrival order. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 879–888, 2015.

[168] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European symposium on algorithms*, pages 589–600. Springer, 2013.

[169] Samir Khuller, Uzi Vishkin, and Neal Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *Journal of Algorithms*, 17(2):280–289, 1994.

[170] Mats Kjaer. Pricing of swing options in a mean reverting model with jumps. *Applied mathematical finance*, 15(5-6):479–502, 2008.

[171] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 123–136, 2012.

[172] Michael Kohler. A regression-based smoothing spline monte carlo algorithm for pricing american options in discrete time. *AStA Advances in Statistical Analysis*, 92(2):153–178, 2008.

[173] Michael Kohler, Adam Krzyżak, and Nebojsa Todorovic. Pricing of high-dimensional american options by neural networks. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 20(3):383–410, 2010.

[174] Anastasia Kolodko and John Schoenmakers. An efficient dual monte carlo upper bound for bermudan style derivative. 2004.

[175] Anastasia Kolodko and John Schoenmakers. Iterative construction of the optimal bermudan stopping time. *Finance and Stochastics*, 10(1):27–49, 2006.

[176] Praveen K Kopalle, Ambar G Rao, and Joao L Assuncao. Asymmetric reference price effects and dynamic pricing policies. *Marketing Science*, 15(1):60–85, 1996.

[177] Christos Koufogiannakis and Neal E Young. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 171–179, 2009.

[178] Christos Koufogiannakis and Neal E Young. Distributed algorithms for covering, packing and maximum weighted matching. *Distributed Computing*, 24(1):45–63, 2011.

[179] Tze Leung Lai and SP-S Wong. Valuation of american options via basis functions. *IEEE transactions on automatic control*, 49(3):374–385, 2004.

[180] Bernard Lapeyre and Jérôme Lelong. Neural network regression for bermudan option pricing. *arXiv preprint arXiv:1907.06474*, 2019.

[181] Ali Lari-Lavassani, Mohamadreza Simchi, and Antony Ware. A discrete valuation of swing options. 2001.

[182] Paul D Larson and Robert A DeMarais. Psychic stock: An independent variable category of inventory. *International Journal of Physical Distribution & Logistics Management*, 20(7):28–34, 1990.

[183] Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model, 2020.

[184] Jérôme Lelong. Dual pricing of american options by wiener chaos expansion. *SIAM Journal on Financial Mathematics*, 9(2):493–519, 2018.

[185] Kai-Siong Leow. *Pricing of swing options: A Monte Carlo simulation approach*. PhD thesis, Kent State University, 2013.

[186] Hans Rudolf Lerche and Mikhail Urusov. On minimax duality in optimal stopping. *Sequential Analysis*, 29(3):328–342, 2010.

[187] Retsef Levi, Robin O Roundy, and David B Shmoys. Provably near-optimal sampling-based policies for stochastic inventory control models. *Mathematics of Operations Research*, 32(4):821–839, 2007.

[188] Reut Levi, Moti Medina, et al. A (centralized) local guide. *Bulletin of EATCS*, 2(122), 2017.

[189] Hongmin Li and Stephen C Graves. Pricing decisions during inter-generational product transition. *Production and Operations Management*, 21(1):14–28, 2012.

[190] Andrew EB Lim and J George Shanthikumar. Relative entropy, exponential utility, and robust dynamic pricing. *Operations Research*, 55(2):198–214, 2007.

[191] Ilan Lobel, Renato Paes Leme, and Adrian Vladu. Multidimensional binary search for contextual decision-making. *Operations Research*, 66(5):1346–1361, 2018.

[192] Ruben Lobel and Georgia Perakis. Dynamic pricing through sampling based optimization. *Available at SSRN 1748426*, 2010.

[193] 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.

[194] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *Journal of the ACM (JACM)*, 62(5):1–17, 2015.

[195] Brendan Lucier. An economic view of prophet inequalities. *ACM SIGecom Exchanges*, 16(1):24–47, 2017.

[196] Will Ma, David Simchi-Levi, and Jinglong Zhao. Dynamic pricing under a static calendar. *Available at SSRN 3251015*, 2018.

[197] Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 260–273. Springer, 2013.

[198] Michael D Marcozzi. On the approximation of optimal stopping problems with application to financial mathematics. *SIAM journal on scientific computing*, 22(5):1865–1884, 2001.

[199] T James Marshall and R Mark Reesor. Forest of stochastic meshes: A new

method for valuing high-dimensional swing options. *Operations Research Letters*, 39(1):17–21, 2011.

[200] Aranyak Mehta. Online matching and ad allocation. 2013.

[201] Nicolai Meinshausen and Ben M Hambly. Monte carlo methods for the valuation of multiple-exercise options. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 14(4):557–583, 2004.

[202] Adam Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

[203] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 406–415. IEEE, 2001.

[204] Aditya Modi, Nan Jiang, Ambuj Tewari, and Satinder Singh. Sample complexity of reinforcement learning using linearly combined model ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 2010–2020. PMLR, 2020.

[205] Javad Nasiry and Ioana Popescu. Dynamic pricing with loss-averse consumers and peak-end anchoring. *Operations research*, 59(6):1361–1368, 2011.

[206] Serguei Netessine. Dynamic pricing of inventory/capacity with infrequent price changes. *European Journal of Operational Research*, 174(1):553–580, 2006.

[207] Duc Khuong Nguyen and Thomas Walther. Modeling and forecasting commodity market volatility with long-term economic and financial variables. *Journal of Forecasting*, 39(2):126–142, 2020.

[208] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 327–336. IEEE, 2008.

[209] Marcel Nutz, Jianfeng Zhang, et al. Optimal stopping under adverse nonlinear expectation and related games. *The Annals of Applied Probability*, 25(5):2503–2534, 2015.

[210] Sechan Oh and Özalp Özer. Characterizing the structure of optimal stopping policies. *Production and Operations Management*, 25(11):1820–1838, 2016.

[211] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[212] Christos H Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 121–129, 1993.

[213] Andrea Pascucci. *PDE and martingale methods in option pricing*. Springer Science & Business Media, 2011.

[214] Georgia Perakis and Guillaume Roels. Robust controls for network revenue management. *Manufacturing & Service Operations Management*, 12(1):56–76, 2010.

[215] Goran Peskir and Albert Shiryaev. *Optimal stopping and free-boundary problems*. Springer, 2006.

[216] Richard L Peterson, Christopher K Ma, and Robert J Ritchey. Dependence in commodity prices. *The Journal of Futures Markets (1986-1998)*, 12(4):429, 1992.

[217] Robert S Pindyck. Volatility and commodity price dynamics. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 24(11):1029–1047, 2004.

[218] Ioana Popescu and Yaozhong Wu. Dynamic pricing strategies with reference effects. *Operations research*, 55(3):413–429, 2007.

[219] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[220] Sheng Qiang and Mohsen Bayati. Dynamic pricing with demand covariates. *Available at SSRN 2765257*, 2016.

[221] Roy Radner, Ami Radunskaya, and Arun Sundararajan. Dynamic pricing of network goods with boundedly rational consumers. *Proceedings of the National Academy of Sciences*, 111(1):99–104, 2014.

[222] LCG Rogers. Pathwise stochastic optimal control. *SIAM Journal on Control and Optimization*, 46(3):1116–1132, 2007.

[223] LCG Rogers. Dual valuation and hedging of bermudan options. *SIAM Journal on Financial Mathematics*, 1(1):604–608, 2010.

[224] LCG Rogers. Bermudan options by simulation. *arXiv preprint arXiv:1508.06117*, 2015.

[225] Leonard CG Rogers. Monte carlo valuation of american options. *Mathematical Finance*, 12(3):271–286, 2002.

[226] Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011.

[227] Sabato Santaniello, Samuel P Burns, Alexandra J Golby, Jedediah M Singer, William S Anderson, and Sridevi V Sarma. Quickest detection of drug-resistant seizures: An optimal control approach. *Epilepsy & Behavior*, 22:S49–S60, 2011.

[228] Sabato Santaniello, David L Sherman, Nitish V Thakor, Emad N Eskandar, and Sridevi V Sarma. Optimal control-based bayesian detection of clinical and behavioral state transitions. *IEEE transactions on neural systems and rehabilitation engineering*, 20(5):708–719, 2012.

[229] John Schoenmakers. A pure martingale dual for multiple stopping. *Finance and Stochastics*, 16(2):319–334, 2012.

[230] John Schoenmakers, Jianing Zhang, and Junbo Huang. Optimal dual martingales, their analysis, and application to new algorithms for bermudan products. *SIAM Journal on Financial Mathematics*, 4(1):86–116, 2013.

[231] Wenjing Shen, Izak Duenyas, and Roman Kapuscinski. Optimal pricing, production, and inventory for new product diffusion under supply constraints. *Manufacturing & Service Operations Management*, 16(1):28–45, 2013.

[232] Euncheol Shin. Monopoly pricing and diffusion of social network goods. *Games and Economic Behavior*, 102:162–178, 2017.

[233] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision pro-

cesses with a generative model. In *Advances in Neural Information Processing Systems*, pages 5186–5196, 2018.

[234] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[235] Justin Sirignano and Konstantinos Spiliopoulos. Stochastic gradient descent in continuous time: A central limit theorem. *Stochastic Systems*, 2020.

[236] Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[237] Lars Stentoft. Convergence of the least squares monte carlo approach to american option valuation. *Management Science*, 50(9):1193–1203, 2004.

[238] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches, 2019.

[239] Chaitanya Swamy and David B Shmoys. Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM Journal on Computing*, 41(4):975–1004, 2012.

[240] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust mdps using function approximation. In *International Conference on Machine Learning*, pages 181–189. PMLR, 2014.

[241] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 217–228, 2019.

[242] Andrew C Thompson. Valuation of path-dependent contingent claims with multiple exercise decisions over time: The case of take-or-pay. *Journal of Financial and Quantitative Analysis*, 30(2):271–293, 1995.

[243] 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.

[244] John N Tsitsiklis and Benjamin Van Roy. Regression methods for pricing

complex american-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.

[245] Erica Van Herpen, Rik Pieters, and Marcel Zeelenberg. When demand accelerates demand: Trailing the bandwagon. *Journal of Consumer Psychology*, 19(3):302–312, 2009.

[246] Benjamin Van Roy. On regression-based stopping times. *Discrete Event Dynamic Systems*, 20(3):307–324, 2010.

[247] Phebe Vayanos, Wolfram Wiesemann, and Daniel Kuhn. Hedging electricity swing options in incomplete markets. *IFAC Proceedings Volumes*, 44(1):846–853, 2011.

[248] Alberto Vera and Siddhartha Banerjee. The bayesian prophet: A low-regret framework for online decision making. *Management Science*, 2020.

[249] Alberto Vera, Siddhartha Banerjee, and Itai Gurvich. Online allocation and pricing: Constant regret via bellman inequalities. *arXiv preprint arXiv:1906.06361*, 2019.

[250] Ruxian Wang. When prospect theory meets consumer choice models: Assortment and pricing management with reference prices. *Manufacturing & Service Operations Management*, 20(3):583–600, 2018.

[251] Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149, 2006.

[252] Martina Wilhelm and Christoph Winter. Finite element valuation of swing options.

[253] David P Williamson. *Network Flow Algorithms*. Cambridge University Press, 2019.

[254] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[255] Harry B Wolfe. A model for control of style merchandise. *IMR; Industrial Management Review (pre-1986)*, 9(2):69, 1968.

[256] Lin F. Yang and Mengdi Wang. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound, 2019.

[257] Lin F. Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features, 2019.

[258] Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms, 2020.