A new approach to high-dimensional online decision-making

Yilun Chen
The School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850, yc2436@cornell.edu

David Goldberg
The School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850, dag369@cornell.edu

Exogenous factors, in particular, commodity prices, can significantly impact the optimal decisions in dynamic pricing and revenue management, option trading, and many other real-world online decision-making scenarios. Realistic modelling paradigms track the complex dynamics of such factors, and the associated decision making problems often boil down to solving stochastic dynamic programs (DP) with high-dimensional and/or path-dependent underlying state, which are well known to be computationally intractable in general, suffering from the so-called “curse of dimensionality”. To sidestep this fundamental computational issue, much of the literature either imposes strong probabilistic/structural assumptions such as i.i.d. state evolution or linear reward structures, or resorts to approximate methods such as approximate dynamic programming (ADP) that are mostly heuristic in nature.

Building on the recent algorithmic progress of Goldberg and Chen (2018) in optimal stopping, we propose a new approach that overcomes the computational barrier for a fairly general class of high-dimensional stochastic DPs. Our results come with strong theoretical guarantees (both runtime and accuracy), guaranteeing an $(1 - \epsilon)$-approximate optimal policy in a runtime scaling polynomially in the time horizon and effectively independent of the dimension, subject to only a “limited-move” constraint, which in the dynamic pricing setting translates to “limited price change”. Furthermore, our results are data-driven and completely model-free, allowing for arbitrarily complex and non-stationary non-Markovian state evolution, assuming only access to a black-box simulator of the model, and requiring no prior knowledge of the underlying distributions/reward structures.

Key words: Online decision-making, commodity prices, dynamic pricing, swing option pricing, stochastic DP, curse of dimensionality, optimal stopping, efficient approximation algorithms, PTAS

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 cer-
tain 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 online 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. Pindyck (2004), Nguyen and Walther (2020), or a vivid real-world case: WTI Crude Oil prices). Meanwhile, they can significantly impact companies’ profits especially 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 online decision-making paradigm is tailor-made for this task, as we illustrate through concrete examples.

**Example 1 (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, i.e. 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 online decision-making paradigm.

**Example 2 (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 Jaillet et al. (2004), Carmona and Ludkovski (2010) or Vayanos et al. (2011) 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 online decision-making paradigm.
Unfortunately, baking commodity prices and/or other relevant exogenous factors into online 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 Helms et al. (1984), Peterson et al. (1992)), meaning that their future movements are based on their historical trajectories (hence path-dependent) rather than simply their current positions. Online 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 precision) the optimal solution as the dimension/the time horizon scale up. We note that, 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.

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 underlying 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 online 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 \( \epsilon \). More specifically, for any fixed \( \epsilon > 0 \), the algorithm can output an \( \epsilon \)-optimal
policy and a \((1 - \epsilon)\) approximation of the optimal total reward in a runtime scaling only \textit{polynomially} in the time horizon, and effectively \textit{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 as input a “system simulator”, rather than any knowledge of the system’s actual underlying probability distribution/state evolution formula or reward functions. Formally, such system simulator is able to generate independent copies of state trajectories and the corresponding rewards. Typical real world examples include commodity price and/or demand forecasting tools. Our algorithm only uses the outputting simulated data from the system simulator, essentially treating the latter as a black box.

- The key building block of the algorithm is a recently proposed approach (Goldberg and Chen (2018)) that, for the first time, addresses the curse of dimensionality for \textit{optimal stopping} in a general sense. In particular, their approach builds on a novel representation of the optimal value, works in a non-DP fashion, and outputs provably near-optimal stopping policies in poly-time. With the observation that any online decision-making problem can be framed as recursively solving one-move problems (Proposition 1), we are able to leverage Goldberg and Chen (2018) to devise our efficient approximation algorithm.

1.1. Main contributions

We make the following contributions.

- \textbf{Approximability results and PTAS.}

We prove that a large class of online decision-making problems, in particular those subject to the limited-move constraint, are in theory \textit{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. The term \textit{polynomial-time-approximation-schemes (PTAS)} refers to such algorithms that for any fixed \(\epsilon\), 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 online decision-making problem, 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.
Algorithm.

We devise an algorithm for coping with high-dimensional path-dependent exogenous factors such as commodity prices in online decision-making. Our algorithm is particularly effective for deriving constant approximations in the long-horizon limited-move regime. Such regime makes practical sense. Indeed, in dynamic pricing applications, (1) decision-making processes often last for a long selling season and (2) manufacturers/retailers often set business rules that limit the number of total price changes. For instance, chemicals market leaders typically implement one or two price changes in a year. As another example, supermarket retailers often impose certain constraints such as “varying the price of a particular product at most $L = 3$ times during the quarter (the 90-day selling season)” (Cohen et al. (2016a)). In these cases, our algorithm can efficiently lock in a pricing strategy that guarantees a constant approximation, say, 75% of the theoretically optimal profit, even in the presence of multiple raw materials with fluctuating prices. In practice, such “robust” pricing strategy is often preferred over those with uncertain performance.

1.2. Literature review

There are a large amount of papers addressing online 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 decision-making problems.

The main application of finding pricing strategy to manage fluctuating prices of multiple raw materials relates our paper to the dynamic pricing literature. Recently, there are a stream of works considering dynamic pricing with high-dimensional underlying features (see Amin et al. (2014), Javanmard and Nazerzadeh (2016), Qiang and Bayati (2016), Cohen et al. (2016b), Keskin and Zeevi (2014), Javanmard (2017), Ban and Keskin (2018), Lobel et al. (2018)). Their high-dimensional feature vectors are often used to describe different customers/products. Common assumptions of these works include (1) the demand/willingness-to-pay has some special structures (e.g. linear in the underlying features), and (2) the features are i.i.d. over time. 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. Besbes and Zeevi (2009), Harrison et al. (2012), Cheung et al. (2017), Farias and Van Roy (2010), Araman and Caldentey (2009), Broder and Rusmevichientong (2012), Keskin and Zeevi (2014), den Boer and Zwart (2013), Lobel and Perakis (2010), Perakis and Roels
(2010), Ferreira et al. (2018), Bertsimas and Perakis (2006). 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 utilize data drawn from a black-box simulator to 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, Greenleaf (1995), Kopalle et al. (1996), Popescu and Wu (2007), Nasiry and Popescu (2011), Guler et al. (2014), Hu et al. (2016), Chen et al. (2016b,a), Cohen et al. (2016a, 2017), Hsieh and Dye (2017), Wang (2018), Cohen-Hillel et al. (2019a,b) 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 Bensaid and Lesne (1996), Ajourlou et al. (2016), Shen et al. (2013), Candogan et al. (2010), Cabral et al. (1999), Radner et al. (2014), Shin (2017)), the scarcity effect ( demands depending negatively on remaining inventory availability; see Van Herpen et al. (2009), Balachander et al. (2009), Cui et al. (2018), Li and Graves (2012)), and the billboard effect(demands depending positively on remaining inventory availability, see e.g. Wolfe (1968), Achabal et al. (1990), Larson and DeMarais (1990), Caro and Gallien (2012)). 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 Ahn et al. (2019), Cohen et al. (2016a), Cohen-Hillel et al. (2019a). Ahn et al. (2019) 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. Cohen et al. (2016a) 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 Cohen-Hillel et al. (2019a), the authors propose an algorithm that outputs 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 Cohen-Hillel et al. (2019a) high-low pricing marks price down only once.

The works studying dynamic pricing with limited price changes are also relevant to this paper. 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. Gallego and Van Ryzin (1994), Gallego et al. (2008), Chen et al. (2018), Ma et al. (2018), Besbes et al. (2019)). Bitran and Mondschein (1997), Celik et al. (2009), Netessine (2006) study pricing contexts where frequent changes of price is either costly or inapplicable. Chen et al. (2015) propose a heuristic algorithm with performance guarantee that adjusts the prices infrequently. Cheung et al. (2017) proves a bound that concisely links the performance guarantee with the number of price changes. Ahn et al. (2019), Cohen et al. (2016a, 2017), as we discussed above, also assume limited price changes.

Our other application of pricing swing options in commodity markets draws a connection between this paper 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 settings in low dimension and with special underlying distributions. (Thompson (1995), Lari-Lavassani et al. (2001), Jaillet et al. (2004) 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, Barrera-Esteve et al. (2006), Dorr (2003), Figueroa (2006), Ibanez (2004) extend the least square Monte Carlo approach of Longstaff and Schwartz (2001); Haarbrucker and Kuhn (2009) proposes a method that reduce the dimension of both the state space and the action space; Bender and Schoenmakers (2006) extends the policy iteration type of method of Kolodko and Schoenmakers (2006). Leow (2013) uses piecewise linear functions as value function approximations; Marshall and Reesor (2011) extends the stochastic mesh methods proposed in Broadie et al. (2004); Bardou et al. (2009) 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 Benth et al. (2012), Dahlgren (2005), Wilhelm and Winter, Dahlgren and Korn (2005), Eriksson et al. (2014), Kjaer (2008), looks at the continuous problem and borrows techniques from PDE. The discussion of these approaches is beyond the scope of this paper. However they in general can’t escape the curse of dimensionality while enjoying strong performance guarantee.

Since the seminal work Davis and Karatzas (1994) and the follow-up works Rogers (2002), Haugh and Kogan (2004), 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, Meinshausen and Hambly (2004) and Schoenmakers (2012) provide two different dual representations and propose the corresponding simulation algorithms. Based on these dual representations, Chandramouli and Haugh (2012), Bender et al. (2015), Bender (2011a,b), Aleksandrov and Hambly (2010), Balder et al. (2013) 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.

We want to mention the information relaxation techniques (see Brown et al. (2010), Brown and Smith (2011) among others) that generalize the duality-based ideas to deal with general online decision-making problems. However, we note that 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.

We also want to mention several papers using ideas from robust optimization to sidestep the challenging DP (see e.g. Lim and Shanthikumar (2007), Adida and Perakis (2006), Chen and Farias (2018)). These approaches achieve various levels of tractability via fundamentally different modelling paradigms, and are generally in comparable to the results presented in this paper.

1.3. Organization

The rest of the paper is organized as follows. In section 2, we introduce the model and assumptions. In section 3, we build the theoretical foundation of our algorithm. The main algorithms and the main results on their performance and complexity will be presented in section 4. Finally, we sketch the key steps of the analysis in section 5.
2. Model

We consider a decision maker (DM) in a discrete-time system driven by some exogenous stochastic processes. The DM faces an online 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. There is a limit on the number of active periods. 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.

2.1. Formal model description

Let the discrete time, $d$-dimensional stochastic process $(x_t)_{t \geq 0}$ be the system’s underlying state. $x_t$ starts in a fixed initial state $x_0 \in \mathbb{R}^d$ at time 0, and evolves according to some probability law in a fixed time span $[1, T]$. Let $\mathcal{F}_t \equiv \sigma(x_1, ..., x_t)$ denote the associated $\sigma$-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 $a_t \in \{p^*\} \cup A_t$, where $p^*$ is the “passive action” and $A_t$ 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. $r_t(a_0, a_1, ..., a_t, x_0, ..., x_t)$ \footnote{For notational convenience, we restrict $a_0 \in A_0$ and assume $r_0(a_0, x_0) = 0$ for all $a_0$.} where $r_t$ is a deterministic function \footnote{This is without loss of generality since any extra randomness can be taken care of by replacing the original reward by its (conditional) expectation.}. 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. We want to remind the reader that $r_t(a_0, a_1, ..., a_t)$ is a $\mathcal{F}_t$-measurable random variable.

Denote a dynamic policy by $\pi = \{\pi_0, ..., \pi_T\}$, where each $\pi_t$ is a mapping from the $\sigma$-algebra $\mathcal{F}_t$ to the extended action set $\{p^*\} \cup A_t$. The DM’s task is to find a policy to maximize the expected total reward (with the expectation taken implicitly over $(x_t)_{1 \leq t \leq T}$):

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

2.2. Examples

Our model incorporates various real-world online 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 (possibly) demand-relevant economic indicators. The active action set $A_t$ 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 $A_t$, or chooses $p^*$ and sticks with the price at time $t-1$. 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 $D_t$ is then generated and fulfilled, and the manufacturer collects the one-period profit $(p_t - c_t)D_t$. We note that in paint manufacturing, due to the existing complex inventory management strategies and possibly long lead times on the supply side, current unit costs often rely on past prices of raw materials, i.e. $c_t \Delta c_t(x_0, \ldots, x_t)$. 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, launched/announced in the past few months, i.e. $D_t \Delta D_t(p_t, x_0, \ldots, x_t)$. 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 $r_t(a_0, \ldots, a_t, x_0, \ldots, x_t)$ 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 requires to determine the best exercising times of the swing rights and the corresponding optimal purchasing amounts from a profit-maximizing agent’s perspective. In this case, the state $x_t$ characterizes the market price of electricity. We note that $x_t$ 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 $A_t$ consists of all possible purchasing amount allowed by the swing right, i.e. $[m^-, m^+]$. The passive action $p^*$ simply leads to not exercising a right at the current time. The reward $r_t(a_t)$ captures the extra payoff relative to the forward contract: if $a_t = p^*, r_t(a_t) = 0$, otherwise $r_t(a_t) = (p - p_t(x_t))(a_t - m)$.

2.3. **Assumptions**
Motivated by these 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 company brand 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” (Cohen et al. (2016a)). 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 online decision-making under the one-move constraint.

Assumption 2 (The 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 $a_0, \ldots, a_t$ and any trajectory $(x_0, \ldots, x_t)$, and outputs in $C$ units of computational time:

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

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 those model-based approaches to online decision-making.

We also note that, 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 online decision-making tasks (simply by plugging in the corresponding system simulators).
We specify the computational cost \( C \) for calling the simulator once, either for sampling the underlying state or for evaluating the reward. Here we (implicitly) require that \( C \) depends on dimension \( d \) and time horizon \( T \) only polynomially (if not linearly). We note that 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 paper.

Assumption 3. Technical conditions required in our algorithmic analysis:

i. (Finite action set). \( \mathcal{A}_t \) is a static, discrete set, with size \( |\mathcal{A}_t| = M \).

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

iii. (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 \( (a_t)_{0 \leq t \leq T} \), \( OPT \geq \alpha \sum_{t=1}^{T} r_t(a_0, ..., a_t) \) a.s.

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 paper 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.

Finally, 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 \( \alpha \). 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 $\alpha$ 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.

2.4. Additional notation
Let $[t]$ denote the sequence $(0, 1, ..., t)$. Accordingly, $z_{[t]} \triangleq (z_0, ..., z_t)$. 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 $N(\epsilon, \delta) \triangleq [\frac{\epsilon}{2} e^{-2 \log(\frac{2}{\delta})}]$ be a constant that is frequently used in our analysis. Let $U(0, z) \triangleq z$ for any $y, z > 0$. Define recursively $kU(y, z) \triangleq y^{k-1}U(y, z)$ for $k \geq 1$.

3. Theory and preliminaries
We first formulate the online decision-making problem as a set of recursive equations.

3.1. Optimality equations
Let $\mathcal{J}_t^k(a_{[t-1]}, x_{[t]} )$ denote the value-to-go function at time $t$, with 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 dependence on the underlying state $x_t$, writing $\mathcal{J}_t^k(a_{[t-1]})$ 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 $a_0, ..., a_{t-1}$,

\[
\mathcal{J}_t^k(a_{[t-1]}) = \max_{a \in A_t} \sup_{t+1 \leq \tau \leq T} \left[ E \left[ \sum_{s=t}^{\tau-1} r_s(a_{[s]}) + \mathcal{J}_{s}^{k-1}(a_{[\tau-1]}) \bigg| \mathcal{F}_t \right] \right],
\]

\[\text{s.t.} \quad a_t = a \quad \text{and} \quad a_s = p^* \text{ for } s \in [t+1, \tau - 1]. \tag{1}\]

\[
\mathcal{J}_t^1(a_{[t-1]}) = \max_{a \in A_t} E \left[ \sum_{s=t}^{T} r_s(a_{[s]}) \bigg| \mathcal{F}_t \right],
\]

\[\text{s.t.} \quad a_t = a \quad \text{and} \quad a_s = p^* \text{ for } s \in [t+1, T]. \tag{2}\]

Here in equation (1), $\tau$ 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 $\tau$, and making the best decisions subject to $k - 1$ active-action-taking opportunities starting from time $\tau$. Formally we have

$^3$ For notational convenience, we assume $\mathcal{J}_{t}^k = 0$ for $t \geq T + 1$.
Proposition 1. The class of functions $(J_k^t)_{t \geq 0, k \geq 1}$ satisfying the equations (1) and (2) are the value-to-go functions of the online decision-making problem. Specifically, $J_0^{K+1} = \text{OPT}$.

A rigorous proof of Proposition 1 is given in technical appendix 7.1. We briefly discuss its implications. The set of equations (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 (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 suffices whereas DP requires depth-$T$). On the other hand, each recursion of equation (1) contains an optimal stopping problem, which may itself be computationally challenging in high dimension. Efficiently solving the online decision-making problem via equations (1) and (2) would be in theory possible if there exists a computationally efficient approach to each corresponding optimal stopping problem.

Our main algorithm leverages such an approach to optimal stopping, proposed by Goldberg and Chen (2018), that enjoys provably near-optimal performance guarantee with runtime scaling only polynomially in time horizon $T$, effectively independent of dimension $d$. We next briefly introduce the main theoretical and algorithmic results of Goldberg and Chen (2018).

3.2. A new approach to optimal stopping

3.2.1. Formulation. In view of equation (1), we adopt 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 $g_t : \mathbb{R}^d \rightarrow \mathbb{R}^+$ is an arbitrary deterministic and non-negative cost function. Let $\mathcal{T}$ be the set of stopping times with respect to $(\mathcal{F}_t)_{1 \leq t \leq T}$. An optimal stopping problem is formally defined as $Z^* \triangleq \inf_{\tau \in \mathcal{T}} E[Z_\tau]$.

Remark 1. Immediately, a maximization formulation seems to fit more naturally into the context of the current paper (as appearing in equation (1)). Nevertheless, we adopt the minimization formulation $\inf_{\tau \in \mathcal{T}} E[Z_\tau]$ in order to more clearly present the results. In fact, under the boundedness condition of Assumption 3, a maximization problem is literally equivalent to a minimization problem via some straightforward algebra.\(^4\)

3.2.2. An expansion representation of $Z^*$. Goldberg and Chen (2018) develops a novel, fundamentally non-DP theory for optimal stopping. They discover a new representation of the optimal value $Z^*$ (in the form of an infinite sum of certain expectations), and prove rate-of-convergence-type results when truncating this infinite sum.

\(^4\) Suppose $Z_t \in [0, \mathcal{L}] \ a.s.$ Then $Y_t = \mathcal{L} - Z_t$ satisfies: 1. $Y_t \in [0, \mathcal{L}] \ a.s.$ and 2. $\inf_{\tau \in \mathcal{T}} E[Z_\tau] = \mathcal{L} - \sup_{\tau \in \mathcal{T}} E[Y_\tau]$
For $t \in [1, T]$, let $Z_t^1 \triangleq Z_t$. For $k \geq 1$ and $t \in [1, T]$, recursively define $Z_t^{k+1} \triangleq Z_t^k - E\left[\min_{i \in [1, T]} Z_i^k | F_t^k\right]$. Let $L_k \triangleq E\left[\min_{t \in [1, T]} Z_t^k\right]$.

**Lemma 1.** \(Z^* = \sum_{k=1}^{\infty} L_k\).

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

**Lemma 2.** \(0 \leq Z^* - \sum_{k=1}^{n} L_k \leq \frac{U}{n+1}, \text{ for all } n \geq 1\).

We note that in many ways the statements of Lemma 1 and Lemma 2 are quite surprising. Indeed, Lemma 1 essentially asserts that the value of a general optimal stopping problem has an expansion representation which looks very much like a closed-form solution, whereas Lemma 2 guarantees a linear rate of convergence when truncating the expansion representation, regardless of the distribution of the underlying state \(x_t\) or the structure of the cost functions \(g_t\), requiring only that \((g_t)_{t \in [1, T]}\) are bounded. We note that the two lemmas suggest a fundamentally non-DP approach to optimal stopping, which leads to efficient algorithms especially in the high-dimensional and/or path-dependent cases.

The two lemmas are Theorem 1 and Theorem 2 of Goldberg and Chen (2018), respectively. We omit the proof, instead pointing the interested reader directly to that paper.

Hereinafter we consider only the case where \(Z_t\) is bounded by \(U\).

**3.2.3. Algorithmic implications.** In light of Lemma 1 and Lemma 2, Goldberg and Chen (2018) proposes algorithms for approximating the optimal value \(Z^*\) and finding the (near-)optimal stopping policy. There 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 one in Assumption 2.

**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 \(\epsilon\) and \(\delta\).

We note that our simulator model allows for inexact evaluation of the costs, effectively extending Goldberg and Chen (2018) to incorporate a broader class of scenarios including that of equation (1). We next present two algorithms \(W_a\) and \(W_b\), as variants of Goldberg and Chen (2018) under
Assumption 4, that can (with high probability) 1. compute an \(\epsilon\)-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 3 (Wa: value approximation).** Algorithm \(W_a\) takes as input \(\epsilon, \delta \in (0,1)\), and achieves the following. It returns a random number \(X\) satisfying \(P(|X - Z^*| > \epsilon U) < \delta\) in total computational time at most

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

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 \(H_i(\epsilon, \delta, T) \triangleq h_i(4^{-6\epsilon^{-1}}, \delta \exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1 + \log(\frac{1}{\delta}))^{-3\epsilon^{-1}})\) \(i \in \{1, 2\}\).

**Lemma 4 (Wb: good policy).** For all \(\epsilon \in (0,1)\), algorithm \(W_b\) outputs a randomized stopping time \(\tau\), s.t. \(E[Z_{\tau}] - Z^* \leq \epsilon U\), 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

\[
\exp(200\epsilon^{-2})T^{6\epsilon^{-1}} \times \left( C \times h_2(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}) + h_1(4^{-6\epsilon^{-1}}, \exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}) \right)
\]

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 \(W_a\) and \(W_b\) will be given in the technical appendix 7.2. Here we sketch the main steps: \(W_a\) 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. \(W_b\) repeatedly calls \(W_a\), compares the obtained value-to-go approximation with the cost-at-stop, and determines whether to stop. The proof of Lemma 3 and 4 combines the result of Lemma 2 with some general tools in probability (e.g. concentration inequalities such as Azuma-Hoeffding). We also leave it to the technical appendix 7.2.

### 4. Algorithm

We now present our main online decision-making algorithms. Following the ideas proposed in section 3, our algorithms use the optimal stopping solvers \(W_a\) and \(W_b\) to recursively solve equations (1) and (2). We begin by describing a set of subroutines for approximating value-to-go functions \(J^k_t\).
4.1. Subroutines for computing $J^k_t$

The subroutines, denoted by $(Q^k)_{k \geq 1}$, 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 $J^k_t(a_{t-1}, x_t)$. For the ease of analysis, we assume that these subroutines know parameters $\alpha$ and $M$ (defined in Assumption 3), which is not required when used in practice.

**Subroutine** $Q^1$ (for computing $J^1_t$)

<table>
<thead>
<tr>
<th>Inputs: control parameters $\epsilon, \delta$, history $a_{t-1}, x_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $a \in A_t$ do</td>
</tr>
<tr>
<td>fix a policy $\pi^a \triangleq (a_{t-1}, a, p^<em>, ..., p^</em>)$,</td>
</tr>
<tr>
<td>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 $S$</td>
</tr>
<tr>
<td>and store their average as $Y^a$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>$a^* \leftarrow \arg \max_{a \in A_t} Y^a$</td>
</tr>
</tbody>
</table>

**Outputs:** $a^*$ (approximate optimal action $a^*_t$) and $Y^a$ (value-to-go $J^1_t$ approximation)

**Subroutine** $Q^k$ (for computing $J^k_t$)

<table>
<thead>
<tr>
<th>Inputs: control parameters $\epsilon, \delta$, history $a_{t-1}, x_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $a \in A_t$ do</td>
</tr>
<tr>
<td>fix a policy $\pi^a \triangleq (a_{t-1}, a, p^<em>, ..., p^</em>)$,</td>
</tr>
<tr>
<td>Call $W_a$ with parameters $\alpha\epsilon$ and $\delta/M$ to solve</td>
</tr>
<tr>
<td>$\sup_{\tau \in [t,T]} E \left[ \sum_{s=t}^{\tau-1} r_s(\pi^a_s, x_s) + J^{k-1}<em>s(\pi^a</em>{\tau-1}, x_{\tau}) \right]</td>
</tr>
<tr>
<td>with $(J^{k-1}<em>j)</em>{j \geq 1}$ approximated by $Q^{k-1}$</td>
</tr>
<tr>
<td>store the output as $Y^a$</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>$a^* \leftarrow \arg \max_{a \in A_t} Y^a$</td>
</tr>
</tbody>
</table>

**Outputs:** $a^*$ (approximate optimal action $a^*_t$) and $Y^a$ (value-to-go $J^k_t$ approximation)

Remark 2. $Q^k$ makes calls to $W_a$ to solve the optimal stopping problem in equation (1). $W_a$’s access to the underlying model is through simulation. More specifically, $W_a$ 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 $S$ by Assumption 2.
Evaluating the “reward-at-stop”, however, requires the knowledge of \((r_j)_{j \geq 1}\) and \(J_j^{k-1}\). The former can again be obtained by calling simulator \(S\) (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.

### 4.2. 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** \(Q_a\) (for computing \(OPT\))

Call \(Q^{K+1}\) with control parameters \((\epsilon, \delta)\)

**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 during the rest of the times. It repeatedly solves an optimal stopping problem to determine when to end the current epoch and start a new one.

**Sketch of Algorithm** \(Q_b\) (for computing a decision-making policy)

```
for \(k = 0 : K\) do
    at the beginning of the \(k\)-th epoch, \(t_k \leftarrow \text{the current time}\), \(x_{[t_k-1]} \leftarrow \text{the state trajectory}\), \(a_{[t_k-1]} \leftarrow \text{the historical action sequence}\)
    call \(Q^{K+1-k}\) 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 \(\pi^k \Delta = (a_{[t_k-1]}, a_k, p^*, ..., p^*)\)
    call \(W_b\) with control parameter \(\frac{\alpha\epsilon}{2(K+1)}\) to solve the optimal stopping problem
    \[
    \sup_{\tau \in [t_k+1,T]} E[\sum_{s=t_k}^{\tau-1} r_s(\pi^k_s, x_s) + J_\tau^{K+1-k}(\pi^k_{[\tau-1]}, x_{[\tau]}) | x_{[t_k]} = x_{[t_k]}]
    \]
    with \((J_j^{K+1-k})_{j \geq 0}\) approximated by \(Q^{K+1-k}\)
    end the epoch when \(W_b\) outputs \textit{STOP}
    if the current period == \(T\) \ OR \ \(k == K\)
        Output \(\pi^k\) break
    end if
end for
```
4.3. Main results: performance and complexity

We present our main results on the performance and complexity of the algorithms \( Q_a \) and \( Q_b \). Our first theorem establishes that algorithm \( Q_a \) efficiently computes a provably near-optimal approximation of the optimal value.

**Theorem 1.** Under Assumptions 1, 2 and 3, and for any \( \epsilon, \delta \in (0,1) \), algorithm \( Q_a \) can output a random number \( X \) satisfying

\[
P(\|X - OPT\| > \epsilon \times OPT) \leq \delta.
\]

Furthermore, the above can be achieved in total computational time at most

\[
C \times (10TM^2)^{K_U(10^6, \frac{1}{M})} \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^{K_U(10^6, \frac{1}{M})}
\]

and with number of samples required from the simulator \( S \) at most

\[
(10TM^2)^{K_U(10^6, \frac{1}{M})} \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^{K_U(10^6, \frac{1}{M})}
\]

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 2.** Under Assumptions 1, 2 and 3, and for any \( \epsilon \in (0,1) \), algorithm \( Q_b \) can compute a randomized decision-making policy \( \pi \) subject to at most \( K \) active action-taking opportunities, such that

\[
|E[\sum_{t=1}^{T} r_t(\pi_t)] - OPT| \leq \epsilon \times OPT.
\]

Furthermore, at each time, \( \pi \) outputs its decision in total computational time at most

\[
2C(100TM)^{k+1_U\left(10^6, \frac{2(K+1)}{\epsilon}\right)}
\]

and with number of samples required from simulator \( S \) at most

\[
2(100TM)^{k+1_U\left(10^6, \frac{2(K+1)}{\epsilon}\right)}
\]

**Remark 3.** At this point, our algorithmic analyses serve more as a proof-of-concept that there does exist a PTAS (analogy) for such general online decision-making problems. The bounds we derive 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) action-taking that is either costly or bearing significant consequences, limited to a small number. 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.
5. Analysis

In this section we outline the proof of Theorem 1 and Theorem 2. In subsection 5.1, we present the analysis of the subroutines \( (Q^k)_{k \geq 1} \), from which Theorem 1 follows directly. We then apply the results to the analysis of \( Q_b \), proving the performance guarantee of its output policy in subsection 5.2 and the computational/sampling complexity bounds in subsection 5.3. We complete the proof of our main results in subsection 5.4.

5.1. Analysis of subroutines \( (Q^k)_{k \geq 1} \)

Algorithm \( Q^k \) approximates the value functions \( (J^k_t)_{t \in [0,T]} \) 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 5.** 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

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

and the number of samples required from simulator \( S \) at most

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

return a random number \( X \) satisfying

\[
P(|X - J^{k+1}_t(a_{[t-1]}, x_{[t]}))| > \epsilon \times OPT) < \delta.
\]

\( Q^{k+1} \) is recursively defined, calling \( Q^k \) multiple times, with the optimal stopping subroutine \( W_a \) 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 3, the algorithm analysis of \( W_a \). The full proof of Lemma 5 is given in Appendix 7.3.

5.2. Performance guarantee of \( Q_b \)

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

**Lemma 6.** 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 \( \pi^b \) with \( \pi^b_{[t-1]} = a_{[t-1]} \), and taking at most \( k \) active actions in \([t+1, T]\), such that

\[
\left| E \left[ \sum_{s=t}^{T} r_s(\pi^b_{[s]} | F_t) - J^{k+1}_t(a_{[t-1]}, x_{[t]}) \right] \right| \leq \frac{k+1}{K+1} \epsilon \times OPT.
\]

We use induction, combined with the algorithm analysis of \( W_b \) and \( Q^k \) (Lemma 4 and 5) 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 7.3.
5.3. 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 7.** 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 $Q_b$ outputs an action in computational time at most $2C(100TM)^{k+1}U(10^6, \frac{2(K+1)}{\alpha \epsilon})$, and with number of independent samples required from simulator $S$ at most $2(100TM)^{k+1}U(10^6, \frac{2(K+1)}{\alpha \epsilon})$.

Recall that at each time, $Q_b$ makes one call to $W_b$ to determine whether to take active action, and (possibly) one call to $Q^k$ to pick a new action. Therefore the computational and sampling costs of $Q_b$ can be bounded by that of $W_b$ and $Q^k$, which are the contents of Lemma 4 and 5. We leave the full proof to appendix 7.3.

5.4. Proof of the main results

With Lemma 5,6 and 7, we complete the proof of our main results, Theorem 1 and 2.

**Proof of Theorem 1** Recall that $Q_a$ simply makes one call to $Q^{K+1}$ with control parameters $(\epsilon, \delta)$. Therefore, the result follows immediately from Lemma 5. Q.E.D.

**Proof of Theorem 2** The fact that the output policy is a good policy, i.e.

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

is an immediate corollary of Lemma 6. The computational and sampling costs of algorithm $Q_b$ follow directly from Lemma 7. Combining the above then completes our proof. Q.E.D.

6. 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 $\epsilon$, our algorithm can obtain an $\epsilon$-approximation of the optimal value, as well as an $\epsilon$-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.
7. Technical appendix

We provide here technical details of the proofs and analysis. Let’s begin by stating a familiar result in probability theory, which is often used to prove concentration of estimators. We omit the proof.

**Lemma 8 (Hoeffding’s inequality).** Suppose that for some \( n \geq 1 \) and \( N > 0 \), \( \{X_i, i \in [1,n]\} \) are i.i.d., and \( P(X_i \in [0,N]) = 1 \). Then \( \mathbb{P}\left| n^{-1} \sum_{i=1}^{n} X_i - E[X_1] \right| \geq \eta \leq 2 \exp \left( -\frac{2\eta^2 n}{N^2} \right) \).

7.1. The optimality equations and Proposition 1

This subsection 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 \( (J_t^k)_{1 \leq j \leq k, t \geq 0} \) solve the equations (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 (1) associated with \( J_t^{k+1} \) as follows

\[
J_t^{k+1}(a_{[t-1]}) = \max_{a \in A_t} \sup_{t+1 \leq \tau \leq T} E\left[ \sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \sup_{\pi^{a,\tau,k}} E\left[ \sum_{s=\tau}^{T} r_s(\pi^{a,\tau,k}_{[s]}) \mid F_\tau \right] \mid F_t \right]
\]

\[
= \max_{a \in A_t} \sup_{t+1 \leq \tau \leq T} E\left[ \sum_{s=t}^{\tau-1} r_s(\pi^a_{[s]}) + \sum_{s=\tau}^{T} r_s(\pi^{a,\tau,k}_{[s]}) \mid F_\tau \right] \mid F_t
\]

\[
= \sup_{\pi} E\left[ \sum_{s=t}^{T} r_s(\pi_{[s]}) \mid F_t \right] \quad \text{s.t.} \quad \pi_{[t-1]} = a_{[t-1]}, \quad \sum_{j=t+1}^{T} 1_{\{\pi_j \neq p^*\}} \leq k \quad \text{a.s.}
\]

where the second equality follows from the tower rule. Notice that the last equality indicates that \( J_t^{k+1} \), as the solution to equation (1) is the \((k+1)\)-st value function. Combining with the induction hypothesis then completes the proof. Q.E.D.

7.2. Optimal stopping subroutines and proof of Lemma 3 and 4

This subsection is devoted to the proof of Lemma 3 and Lemma 4 in section 3. Exploiting Lemma 1 and 2, algorithm \( W_a \) 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 \( (B^k)_{k \geq 1} \), 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:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( B^1 ) (for computing ( Z_t^1(x_{[t]}) ))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>( t, x_{[t]}, \epsilon, \delta )</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td>cost evaluation with parameters ( \epsilon, \delta, t, x_{[t]} )</td>
</tr>
</tbody>
</table>
For $k \geq 1$, we define $B^{k+1}$ inductively as follows:

**Algorithm $B^{k+1}$ (for computing $Z_i^{k+1}(x_{[t]})$)**

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

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

- generate an ind. sample of $x_{[t+1,T]}$ conditional on $x_{[t]} = x_{[t]}$
  - store in $A^1 ← x_{[T]}$

for $j = 1 : T$

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

end for

$A^3[i] ← \min_{j \in [T]} A^2[j]$

end for

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

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

We now formally analyze $B^k$. 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

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

$$q_k(\epsilon, \delta, T) \triangleq \delta \times (f_k(\epsilon, \delta, T))^{-1}.$$

We will need the following auxiliary lemma, which demonstrates that functions $f_k$ and $q_k$ satisfies certain recursive bounds. We refer the reader to Goldberg and Chen (2018) for the proof, which we omit here.

**Lemma 9.** For all $\epsilon, \delta \in (0, 1)$ and $k \geq 1$,

$$f_{k+1}(\epsilon, \delta, T) \geq (N(\frac{\epsilon}{4}, \frac{\delta}{4}) + 1) \times (T + 2) \times f_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T),$$

$$q_{k+1}(\epsilon, \delta, T) \leq q_k(\frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}, T).$$

With Lemma 9 in hand, we now prove the following lemma, which certifies that $(B^k)_{k \geq 1}$ is indeed a set of “good” approximation algorithms.

**Lemma 10.** For all $k \geq 1$, $t \in [1, T]$, $x_{[t]} \in \mathbb{R}^{d \times t}$, $\epsilon, \delta \in (0, 1)$, algorithm $B^k$ 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 \( P(|X - Z^k_i(x_{[i]})| > \epsilon U) \leq \delta \).

**Proof of Lemma 10** 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

\[
P\left( \left| \mathbf{A}^2[\cdot, j] - Z^k_i(x_{[i]}, x_{[i+1,j]}) \right| > \frac{\epsilon}{4} U \right) < \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T}.
\]

Applying a union bound and we have

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

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 \( \mathcal{B}^k(t, x_{[i]}, \frac{\epsilon}{4}, \frac{\delta}{4}) \) (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 \( \mathcal{B}^{k+1}(t, x_{[i]}, \epsilon, \delta) \) is at most

\[
N\left( \frac{\epsilon}{4}, \frac{\delta}{4} \right) + N\left( \frac{\epsilon}{4}, \frac{\delta}{4} \right) \times T \times f_k\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) \times h_2\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) \\
+ f_k\left( \frac{\epsilon}{2}, \frac{\delta}{2}, T \right) \times h_2\left( \frac{\epsilon}{2}, \frac{\delta}{2}, T \right) \\
\leq N\left( \frac{\epsilon}{4}, \frac{\delta}{4} \right) \times (T + 2) \times f_k\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) \times h_2\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) \\
\leq f_{k+1}(\epsilon, \delta, T) \times h_2\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right).
\]

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

\[
\left( C \times h_2\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) + h_1\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right) + 1 \right) \times f_k\left( \frac{\epsilon}{4}, \frac{\delta}{4N(\frac{\epsilon}{4}, \frac{\delta}{4})T} \right).
\]
then the minimum of a length-\( T \) array is computed (at computational cost \( T \)). One additional call is then made to \( \mathcal{B}^k(t, x_0, \hat{z}, \bar{z}) \), at computational cost at most

\[
\left(C \times h_2\left(\frac{\epsilon}{2 \times 4^k}, q_k\left(\frac{\epsilon}{2 \times 4^k}, \frac{\delta}{2}, T\right)\right) + h_1\left(\frac{\epsilon}{2 \times 4^k}, q_k\left(\frac{\epsilon}{2 \times 4^k}, \frac{\delta}{2}, T\right)\right) + 1\right) \times f_k\left(\frac{\epsilon}{2}, \frac{\delta}{2}\right);
\]

and finally the average of \( N\left(\frac{\hat{z}}{2}, \frac{\bar{z}}{2}\right) \) is computed and subtracted from \( A^3 \), at computational cost \( N\left(\frac{\hat{z}}{2}, \frac{\bar{z}}{2}\right) + 1 \). 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 computational cost of \( \mathcal{B}^{k+1}(t, x_0, \epsilon, \delta) \) is at most

\[
T \times N\left(\frac{\epsilon}{4}, \frac{\delta}{4}\right) \times f_k\left(\frac{\epsilon}{4}, \frac{\delta}{4}, N\left(\frac{\hat{z}}{2}, \frac{\bar{z}}{2}\right)T\right) + N\left(\frac{\epsilon}{4}, \frac{\delta}{4}\right)C + T + N\left(\frac{\epsilon}{4}, \frac{\delta}{4}\right) + 1
\]

\[
+ \left(C \times h_2\left(\frac{\epsilon}{2 \times 4^k}, q_k\left(\frac{\epsilon}{2 \times 4^k}, \frac{\delta}{2}, T\right)\right) + h_1\left(\frac{\epsilon}{2 \times 4^k}, q_k\left(\frac{\epsilon}{2 \times 4^k}, \frac{\delta}{2}, T\right)\right) + 1\right) \times f_k\left(\frac{\epsilon}{2}, \frac{\delta}{2}, T\right)
\]

\[
\leq \left(C \times h_2\left(\frac{\epsilon}{4^k+1}, q_k\left(\frac{\epsilon}{4^k+1}, \frac{\delta}{4^k+1}, T\right)\right) + h_1\left(\frac{\epsilon}{4^k+1}, q_k\left(\frac{\epsilon}{4^k+1}, \frac{\delta}{4^k+1}, T\right)\right) + 1\right) \times f_k\left(\frac{\epsilon}{4^k+1}, \frac{\delta}{4^k+1}, T\right)
\]

\[
\times \left(N\left(\frac{\epsilon}{4^k+1}\right) + 1\right) \times (T + 2) \times f_k\left(\frac{\epsilon}{4}, \frac{\delta}{4}, N\left(\frac{\hat{z}}{2}, \frac{\bar{z}}{2}\right)T\right).
\]

Combining the above calculation with Lemma 9, we prove that the induction is true in case \( k + 1 \), thus completing the proof. Q.E.D.

With Lemma 10 in hand, we now complete the proof of Lemma 3. We first formally define algorithm \( \mathcal{W}_a \), which combines \( \mathcal{B}^k \) and Lemma 1 and 2 that gives a good approximation of \( \text{OPT} \):

**Algorithm** \( \mathcal{W}_a \)

**Input:** \( \epsilon, \delta \)

set \( G \leftarrow \left\lceil \frac{\hat{z}}{2} \right\rceil \)

for \( i = 1 : G \) Do

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

end for

**Output:** \( \sum_{i=1}^{G} A^0[i] \)

with \( \hat{\mathcal{B}}^k \) a modified version of \( \mathcal{B}^k \) that approximates \( E[\min_{t \in [T]} Z^k_t] \):

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

**Input:** \( \epsilon, \delta; \)

for \( i = 1 : N(\frac{\hat{z}}{2}, \frac{\bar{z}}{2}) \) do

    call the simulator to generate \((x_1, ..., x_T)\) and store in \( A^1 \)
for $j = 1 : T$ do
  generate an ind. call to $B^k(j, A^1_{[j]}, \frac{\delta}{2}, \frac{4}{2N(\frac{\delta}{2})})$ and store in $A^2[j]$
end for

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

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

Proof of Lemma 3  By arguments almost identical to the proof of Lemma 10, we conclude that $\hat{B}^k(\epsilon, \delta)$ in runtime at most $(C \times h_2(\frac{\epsilon}{T+1}, q_{k+1}(\epsilon, \delta, T)) + h_1(\frac{\epsilon}{T+1}, q_{k+1}(\epsilon, \delta, T)) + 1) \times f_{k+1}(\epsilon, \delta, T)$ and calls to the simulator at most $2 \times h_2(\frac{\epsilon}{T+1}, q_{k+1}(\epsilon, \delta, T)) \times f_{k+1}(\epsilon, \delta, T)$, output a number $X$ satisfies $P(|X - E[\min_{t \in [T]} Z^k_t]| > \epsilon U) < \delta$. We omit the details. Now let’s analyze algorithm $W_a$.

We first prove algorithm $W_a$. We state without proving the following results, which are respectively Lemma 8 and Corollary 4 in Goldberg and Chen (2018), and which show how to find a good stopping policy.

Combining Lemma 10 and 12, we complete the proof of Lemma 4. Let’s first formally define algorithm $W_b$.  

LEMMA 11. For all $k \geq 1$, $\min_{t \in [T]} Z^k_t \leq \frac{U}{k}$. a.s.

LEMMA 12. For $k \geq 1$, let $\tau_k$ be the stopping time that stops the first time that $Z^k_t \leq \frac{U}{k}$, where such a time exists w.p.1 by Lemma 11. Then $|E[Z_{\tau_k}] - OPT| \leq \frac{U}{k}$.
Algorithm $W_b$

Input: $\epsilon$

for $t = 1 : T$ do

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

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

Output: STOP break

end if

end for

Output: STOP

Proof of Lemma 4 The fact that $W_b$, as defined, outputs stopping strategies with desired performance guarantee follows from Lemma 2 and 12, and several straightforward union bounds. The argument is in fact identical to the proof of Corollary 3 in Goldberg and Chen (2018) 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 $B^{[4\epsilon^{-1}]}$ with parameters $\frac{\epsilon}{4}, \frac{\epsilon}{4T}$. By directly applying Lemma 10, we conclude that the computational cost for algorithm $W_b$ to output the decision in each time period is at most

$$f_{[4\epsilon^{-1}]}^{(\epsilon,\frac{\epsilon}{4},\frac{\epsilon}{4T},T)} \times \left( C \times h_2(\frac{\epsilon}{4^{1+|4\epsilon^{-1}|}}, q_{[4\epsilon^{-1}]}^{(\epsilon,\frac{\epsilon}{4},\frac{\epsilon}{4T},T)} + 1 \right)$$

$$\leq \exp(200\epsilon^{-2}) \times T^{6\epsilon^{-1}}$$

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

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

7.3. Analysis of main algorithms and proof of Lemma 5, 6 and 7

This subsection is devoted to the detailed analysis of our main algorithms in section 4. Specifically, we provide the analysis of algorithms $(Q_k^k)_{k \geq 1}$, which is the content of Lemma 5; we provide the proof of algorithm $Q_b$’s performance guarantee, which is the content of Lemma 6; and we provide the computational and sampling complexity analysis, which is the content of Lemma 7.

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

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

$$f^k(\epsilon,\delta) \triangleq C(10TM^2)^{kU(10^6, \frac{1}{2})} \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^{kU(10^6, \frac{1}{2})}.$$
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 13.** Functions $g^k(\epsilon, \delta)$ and $f^k(\epsilon, \delta)$ are decreasing in $\epsilon$ and $\delta$. They satisfy

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

Furthermore, they satisfy the recursion

\[
\begin{align*}
g^{k+1}(\epsilon, \delta) & \geq 2g^k\left(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\deltaM^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1 + \log(M/\delta))^{-3\epsilon^{-1}}\right) \\
& \quad \times \exp(200\epsilon^{-2})MT^{6\epsilon^{-1}}(1 + \log(M/\delta))^{6\epsilon^{-1}} \\
f^{k+1}(\epsilon, \delta) & \geq 2\left(C \times g^k\left(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\deltaM^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1 + \log(M/\delta))^{-3\epsilon^{-1}}\right) \\
& \quad + f^k\left(\frac{\epsilon}{2}4^{-6\epsilon^{-1}}, \frac{1}{2}\deltaM^{-1}\exp(-200\epsilon^{-2})T^{-6\epsilon^{-1}}(1 + \log(M/\delta))^{-3\epsilon^{-1}}\right)\right) \\
& \quad \times \exp(200\epsilon^{-2})MT^{6\epsilon^{-1}}(1 + \log(M/\delta))^{6\epsilon^{-1}}
\end{align*}
\]

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 $\text{OPT} \geq \alpha \mathcal{L}$. We note that 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 13 and the maximum cap $\mathcal{L}$, we now complete the proof of Lemma 5.

**Proof of Lemma 5** 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 $a_0, \ldots, a_{t-1}$ and the state trajectory $x_0, \ldots, x_t$, for each $a \in \mathcal{A}_t$, fix a policy $\pi^a \stackrel{\Delta}{=} (a_{[t-1]}, a, p^*, \ldots, p^*)$. Let $W_a$ be a r.v. that is distributed as $\sum_{s=t}^T r_s(\pi^a_{[s]})$, conditional on $x_a = x_t$. We note that $Q^1$ calls the simulator $\mathcal{S}$ to generate $N(\alpha, \epsilon/\delta)$ i.i.d. samples of $W_a$ and stores their average as $Y^a$. By 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 8, we get

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

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

\[
P\left(\max_{a \in \mathcal{A}_t} |Y^a - J^1_t(a_{[t-1]}, x_t)| \geq \epsilon \times \text{OPT}\right) \leq \delta,
\]
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 \( \mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) \) denotes the value function corresponding to \( a \) at time \( t \). By our definition of \( \mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) \) 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, \( \mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) = \max_{a \in \mathcal{A}_t} \mathcal{J}_t^{k+1,a}(a_{[t-1]}, x_{[t]}) \). 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 3 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 \( Y^a \) satisfying \( P(|Y^a - \mathcal{J}_t^{k+1,a}| > \alpha \epsilon \mathcal{L}) < \frac{\delta}{M} \). Combining the above with the assumption that \( \text{OPT} \geq \alpha \mathcal{L} \) and a union bound we then have

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

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

Next we prove that \( Q^{k+1} \) satisfies the desired computational and sample complexity bounds. By our definition of \( g^k \) and \( f^k \), it suffices to prove that \( Q^{k+1} \) 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 \( Q^1 \) with input parameters \( \epsilon, \delta \) (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 13. 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

\[
\left( 2CTN(\alpha \epsilon, \delta / M) + T + N(\alpha \epsilon, \delta / M) \right) \times M \leq 3CMT(\alpha \epsilon)^{-2}\log(2M\delta^{-1}).
\]

By Lemma 13 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 \( x_0, \ldots, x_t \) and action sequence \( a_0, \ldots, a_{t-1} \), the specific optimal stopping problem here has the following reward function \( \sum_{s=t}^{t-1} r_s(\pi_{[s]}^a) + \mathcal{J}_t^k(\pi_{[t-1]}^a) \) for each \( j \in [t, T] \) and some \( a \in \mathcal{A}_t \), with \( \pi^a = (a_{[t-1]}, a, p^*, \ldots, p^*) \) as defined above. Let \( Z_j \overset{\Delta}{=} \sum_{s=t}^{t-1} r_s(\pi_{[s]}^a) + \)
$\mathcal{J}_j^k(\pi_{j-1}^a)$. 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, notice that $Z_j$ consists of two parts. The first part $\sum_{s=1}^{j-1} r_s(\pi_{[s]}^a)$ 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 OPT $\leq \mathcal{L}$ imply that one can get a number $\tilde{J}$ such that $P(|\tilde{J} - \mathcal{J}_j^k| \geq \eta_1 \mathcal{L}) \leq \eta_2$ at computational 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 $\delta/M$. Combining the previous discussion with Lemma 3, we have that the total number of samples required by $Q^{k+1}$ is at most

$$
H_2(\alpha \epsilon, \delta/M, T) \times \exp(200\alpha^{-2} \epsilon^{-2}) T^{6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1} \epsilon^{-1}} M \\
= h_2(4^{-6\alpha^{-1} \epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2} \epsilon^{-2}) T^{-6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta})))^{6\alpha^{-1} \epsilon^{-1}} \\
\times \exp(200\alpha^{-2} \epsilon^{-2}) MT^{6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1} \epsilon^{-1}} \\
= g^k(\alpha 4^{-6\alpha^{-1} \epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2} \epsilon^{-2}) T^{-6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta})))^{6\alpha^{-1} \epsilon^{-1}} \\
\times \exp(200\alpha^{-2} \epsilon^{-2}) MT^{6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1} \epsilon^{-1}} \\
\leq g^k(\alpha 4^{-6\alpha^{-1} \epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2} \epsilon^{-2}) T^{-6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta})))^{6\alpha^{-1} \epsilon^{-1}} \\
\times \exp(200\alpha^{-2} \epsilon^{-2}) MT^{6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1} \epsilon^{-1}} \\
\leq g^{k+1}(\alpha \epsilon, \delta) 
$$

(5)

where the last inequality follows from Lemma 13.

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 3, we have the total computational cost is at most

$$
\left( C \times H_2(\alpha \epsilon, \delta/M, T) + H_1(\alpha \epsilon, \delta/M, T) \right) \times \exp(200\alpha^{-2} \epsilon^{-2}) T^{6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta}))^{6\alpha^{-1} \epsilon^{-1}} M \\
= \left( C \times h_2(4^{-6\alpha^{-1} \epsilon^{-1}}, \delta M^{-1} \exp(-200\alpha^{-2} \epsilon^{-2}) T^{-6\alpha^{-1} \epsilon^{-1}} (1 + \log(\frac{M}{\delta})))^{6\alpha^{-1} \epsilon^{-1}} \right)
$$
We bound the gap between $\bar{x}_Q$ definition we have at this time point is a constant measuring the expected performance of algorithm $Q$. Expectation is taken not only over the state evolution but also over $B$ action is allowed in later periods. The expected total reward is for the current period, and takes $p$.

Proof of Lemma 6 We proceed by induction. In the base case $k = 1$, the algorithm calls $Q^1$ with $(\frac{\epsilon}{8(\lambda+1)}, \frac{M_0\epsilon}{8(\lambda+1)} \land 1)$, which outputs an action $a^B \in A_t$, where we spell out $B$ as a random variable defined in $[M]$ (due to the randomized nature of $Q^1$). The algorithm then fixes $a^B$ 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 $E[\sum_{s=1}^{T} r_s(\pi^B_{[a]}|x_{[t]} = x_{[t]}$, which we denote by $\tilde{J}^1_B(a_{[t-1]},x_{[t]}$. (here $\pi^B = (a_{[t-1]}, a^B, p^*, ..., p^*)$, as defined previously). Notice that the expectation is taken not only over the state evolution but also over $B$, so that $\tilde{J}^1_B(a_{[t-1]},x_{[t]}$ is a constant measuring the expected performance of algorithm $Q_b$. Suppose the true optimal action at this time point is $a^\ast$. And the corresponding policy $\pi^\ast$ is defined as $(a_{[t-1]}, a^\ast, p^*, ..., p^*)$. By definition we have

$$E\left[\sum_{s=1}^{T} r_s(\pi^1_{[a]}|x_{[t]} = x_{[t]} \right] = J^1_t(a_{[t-1]},x_{[t]}.$$

We bound the gap between $\tilde{J}^1_B$ and $J^1$. For notational simplicity, we make the conditioning $x_{[t]} = x_{[t]}$ implicit in the following probability statements. Recall that for each $a \in A_t$, algorithm $Q^1$ computes a number $Y^a$ that satisfies (with the given control parameters here)

$$P\left(|Y^a - J^1_{a}| > \frac{\alpha \epsilon}{8(\lambda+1)} \right) \leq \frac{\alpha \epsilon}{8(\lambda+1)}.$$

We thus have

$$P\left(J^1_t - \tilde{J}^1_{t} > \frac{\alpha \epsilon}{4(\lambda+1)} \right) = P\left(J^1_t - Y^\ast + Y^B - Y^B - \tilde{J}^1_{t} > \frac{\alpha \epsilon}{4(\lambda+1)} \right) \leq P\left(J^1_t - Y^\ast + Y^B - \tilde{J}^1_{t} > \frac{\alpha \epsilon}{4(\lambda+1)} \right) \leq P\left(|J^1_t - Y^\ast| > \frac{\alpha \epsilon}{8(\lambda+1)} \right) + P\left(|Y^B - \tilde{J}^1_{t} > \frac{\alpha \epsilon}{8(\lambda+1)} \right) \leq \frac{\alpha \epsilon}{8(\lambda+1)} + \frac{\alpha \epsilon}{8(\lambda+1)} = \frac{\alpha \epsilon}{4(\lambda+1)}.$$
where the first inequality follows from the defining property of the algorithm-selected action $a^B$ that $Y^B \geq Y^a$, $a \in \mathcal{A}$, the second inequality follows from a union bound and the third inequality follows from (7). With the above probability bound, we may further bound the expected gap $E[\mathcal{I}_t - \hat{\mathcal{I}}_t^1, B]$ by

$$E[\mathcal{I}_t - \hat{\mathcal{I}}_t^1, B] \leq P\left(\mathcal{I}_t - \hat{\mathcal{I}}_t^1, B > \frac{\alpha \epsilon}{4(K + 1)\mathcal{L}}\right) \times \mathcal{L} + P\left(\mathcal{I}_t - \hat{\mathcal{I}}_t^1, B \leq \frac{\alpha \epsilon}{4(K + 1)\mathcal{L}}\right) \times \frac{\alpha \epsilon}{4(K + 1)\mathcal{L}}$$

$$\leq \frac{\alpha \epsilon}{4(K + 1)\mathcal{L}} + \frac{\alpha \epsilon}{4(K + 1)\mathcal{L}}$$

$$= \frac{\epsilon}{2(K + 1)\alpha \mathcal{L}} \leq \frac{\epsilon}{(K + 1)\text{OPT}},$$

where in the first inequality we use the definition of the maximum cap $\mathcal{L}$ which yields $0 < \hat{\mathcal{I}}_t^1, B \leq \mathcal{I}_t < \mathcal{L}$ and in the second inequality we use Assumption 3 that $\alpha \mathcal{L} \leq \text{OPT}$. 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 $Q_k$. It first makes a call to $Q^{k + 1}$ 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 $Q^k$ 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 $a^B$ be the output action from $Q^{k + 1}$. Notice that $B$ is a random variable due to the randomized nature of algorithm $Q^{k + 1}$. $\pi^B = (a_{[t-1]}, a^B, p^*, ..., p^*)$ is the corresponding policy. We let $\mathcal{J}_t^{k+1, a}(a_{[t-1]}, x_{[t]}) \Delta \sup_a E[\sum_{s=t}^{\tau-1} r_s(\pi^B_{[s]}) + \mathcal{J}^k(\pi^B_{[\tau-1]})|x_{[t]} = x_{[t]}]$ denote the largest total expected reward given that the choice of action in the current period is $a$. Let $\hat{\mathcal{J}}_t^{k+1, B}(a_{[t-1]}, x_{[t]}) \Delta E[\mathcal{J}_t^{k+1, B}(a_{[t-1]}, x_{[t]}))$ where the expectation is taken over $B$. In words, $\hat{\mathcal{J}}_t^{k+1, B}$ is the largest total expected reward given that the current choice of action follows the output of $Q_k$. We let the output stopping time returned by $\mathcal{W}_b$ be $\tau_B$. We denote by $\hat{\mathcal{J}}_t^{k+1, B}(a_{[t-1]}, x_{[t]}) \Delta E[\sum_{s=t}^{\tau_B-1} r_s(\pi^B_{[s]}) + \mathcal{J}_B^k(\pi^B_{[\tau_B-1]})|x_{[t]} = x_{[t]}]$ the expected total reward following the current choice $B$ from $Q^{k + 1}$ and the stopping time $\tau_B$ computed from $Q_k$, and following the true optimal policy afterwards. We note that by the randomized nature of $\mathcal{W}_b$, $\tau_B$ is random even conditional on $B$. The expectation in the definition of $\hat{\mathcal{J}}$ is taken over $x_t, B$ and $\tau_B$. Let $\hat{\mathcal{J}}_t^{k+1}(a_{[t-1]}, x_{[t]})$ denote the expected total reward following the policy returned by $Q_k$. By the definition of $B$ and $\tau_B$, $(\hat{\mathcal{J}}_t^k)_{k \geq 1}$ satisfy the recursion:

$$\hat{\mathcal{J}}_t^{k+1}(a_{[t-1]}, x_{[t]}) = E[\sum_{s=t}^{\tau_B-1} r_s(\pi^B_{[s]}) + \hat{\mathcal{J}}_{\tau_B}^k(\pi^B_{[\tau_B-1]})|x_{[t]} = x_{[t]}].$$

For notational simplicity, in the proof we use $\mathcal{J}_t^{k+1}, \hat{\mathcal{J}}_t^{k+1, B}, \hat{\mathcal{J}}_t^{k+1, B}$ 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

$$\hat{\mathcal{J}}_t^{k+1} - \hat{\mathcal{J}}_t^{k+1} \leq \frac{k + 1}{K + 1}\epsilon \times \text{OPT}, \quad (8)$$
We rewrite the above as
\[
\mathcal{J}_{t}^{k+1} - \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} + \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} - \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} + \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} - \tilde{\mathcal{J}}_{t}^{k+1} \leq \frac{k + 2}{K + 1} \epsilon \times \text{OPT}.
\]
and notice that to prove inequality 8, it suffices to show
\[
\mathcal{J}_{t}^{k+1} - \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} \leq \frac{1}{2(K + 1)} \epsilon \times \text{OPT} \tag{9}
\]
\[
|\tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} - \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}}| \leq \frac{1}{2(K + 1)} \epsilon \times \text{OPT} \tag{10}
\]
\[
|\tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} - \tilde{\mathcal{J}}_{t}^{k+1}| \leq \frac{k}{K + 1} \epsilon \times \text{OPT} \tag{11}
\]
We prove the above bounds one by one.

- **Proof of bound (9).** Recall that for each \(a \in \mathcal{A}_t\), \(Q^{k+1}\) calls \(\mathcal{W}_a\) with control parameters \((\frac{8\epsilon}{8(K + 1)}, \frac{8\epsilon}{8(K + 1)})\), which computes a number \(Y^a\) satisfying

\[
P \left( |Y^a - J_{t}^{k+1,a}| > \frac{\alpha \epsilon}{8(K + 1)} \right) \leq \frac{\alpha \epsilon}{8(K + 1)}.
\]

Notice that the above inequality is identical to the bound (7). The rest of the proof is also exactly identical to that in the base case (right after bound (7)), and we omit here. As a result, we conclude that \(\mathcal{J}_{t}^{k+1} - \tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} \leq \frac{1}{2(K + 1)} \epsilon \times \alpha \times \mathcal{L} \leq \frac{1}{2(K + 1)} \epsilon \times \text{OPT}\), completing the proof of bound (9).

- **Proof of bound (10).** Recall that conditional on \(a^B = a\), \(\mathcal{W}_a\) is called to solve the optimal stopping problem

\[
\mathcal{J}_{t}^{k+1,a} = \sup_{\tau} E \left[ \sum_{s=t}^{r_{B} - 1} r_s(\pi_{[s]}^a) + J^{k}_{\tau} | x_{|\tau]} = x_{|[t]} \right]
\]

with control parameter \(\frac{8\epsilon}{2(K + 1)}\). By Lemma 4, the output stopping time \(\tau_{B}\) satisfies

\[
J_{t}^{k+1,a} - E \left[ \sum_{s=t}^{r_{B} - 1} r_s(\pi_{[s]}^a) + J^{k}_{\tau_{B}}(\pi_{[\tau_{B} - 1]}^B) | x_{|\tau]} = x_{|[t]} \right] \leq \frac{\epsilon}{2(K + 1)} \mathcal{L} \leq \frac{\epsilon}{2(K + 1)} \text{OPT}.
\]

Taking expectation over \(B\), we thus conclude that \(\tilde{\mathcal{J}}_{t}^{k+1,B} - \tilde{\mathcal{J}}_{t}^{k+1,B} \leq \frac{\epsilon}{2(K + 1)} \text{OPT}\), completing the proof of bound (10).

- **Proof of bound (11).** By definition, we have

\[
|\tilde{\mathcal{J}}_{t}^{k+1,\mathcal{B}} - \tilde{\mathcal{J}}_{t}^{k+1}| = \left| E \left[ \sum_{s=t}^{r_{B} - 1} r_s(\pi_{[s]}^B) + J^{k}_{\tau_{B}}(\pi_{[\tau_{B} - 1]}^B) | x_{|\tau]} = x_{|[t]} \right] - E \left[ \sum_{s=t}^{r_{B} - 1} r_s(\pi_{[s]}^B) + J^{k}_{\tau_{B}}(\pi_{[\tau_{B} - 1]}^B) | x_{|\tau]} = x_{|[t]} \right] \right|
\]

\[
\leq E \left[ |\tilde{\mathcal{J}}_{t}^{k}_{\tau_{B}} - J^{k}_{\tau_{B}}| | x_{|\tau]} = x_{|[t]} \right] \leq \frac{k}{K + 1} \epsilon \times \text{OPT},
\]
completing the proof of bound (11). Here the last inequality follows from our induction hypothesis.

Combining bounds 9, 10 and 11 proves inequality 8, the case \( k + 1 \), which then completes our induction and proves the lemma. Q.E.D.

**Proof of Lemma 7** We begin with the sampling cost. At each period in the \((K - k)\)-th epoch, first one call is made to \( W_k \), with control parameter \( \zeta = \frac{\Delta}{2(K + 1)} \), whose sampling cost is at most

\[
\exp(400\zeta^{-2}) T \zeta^{-1} \times h_2 \left( 4^{-6\zeta^{-1}}, e^{-400\zeta^{-2} T^{-6\zeta^{-1}}} \right)
\]

(by Lemma 4)

\[
\leq \exp(400\zeta^{-2}) T \zeta^{-1} \times 2g_k \left( \frac{\alpha}{2} 4^{-6\zeta^{-1}}, \frac{1}{2} e^{-400\zeta^{-2} T^{-6\zeta^{-1}}} \right)
\]

(by arguments in the proof of Lemma 5)

\[
= \exp(400\zeta^{-2}) T \zeta^{-1} \times 2(10TM^2)^{k_U} \left( 10^6, \frac{\alpha}{2} 4\zeta^{-1} \right)
\]

\[
\times \left( 1 + \log(2) + 400\zeta^{-2} + 6\zeta^{-1} \log(T) \right)^{k_U} \left( 10^6, \frac{\alpha}{2} 4\zeta^{-1} \right)
\]

\[
\leq \exp(400\zeta^{-2}) T \zeta^{-1} \times (10TM^2)^{k_U} \left( 10^6, \frac{\alpha}{2} 4\zeta^{-1} \right)
\]

\[
\times \left( 10^3\zeta^{-2} \times T \right)^{k_U} \left( 10^6, \frac{\alpha}{2} 4\zeta^{-1} \right)
\]

\[
\leq \exp(400\zeta^{-2}) T \zeta^{-1} \times (10TM^2)^{k_U} \left( 10^6, 2\zeta^{-1} \right)
\]

\[
\times \left( 10^3\zeta^{-2} \times T \right)^{k_U} \left( 10^6, 2\zeta^{-1} \right)
\]

\[
\leq 10^{2k+1} \left( 10^6, 2\zeta^{-1} \right) \times T^{k+1} \left( 10^6, \zeta^{-1} \right) \times M^{k+1} \left( 10^6, \zeta^{-1} \right)
\]

(by bounds (12) and (13) in the proof of Lemma 13)

\[
= (100TM)^{k+1} \left( 10^6, \frac{2(K+1)}{\alpha} \right).
\]

If the output from \( W_b \) is STOP, \( Q_b \) then makes a call to \( Q^{k-1} \) with control parameters \( \left( \frac{\epsilon}{8(K+1)}, \frac{M\alpha}{8(K+1)} \right) \) (or equivalently \( \left( \frac{\epsilon}{2n}, \frac{M\alpha}{4} \right) \), whose sampling cost is at most

\[
(10TM^2)^{k-1} \left( 10^6, 4\zeta^{-1} \right) \times \left( 1 + \log(4) + \log(\zeta^{-1}) \right)^{k-1} \left( 10^6, 4\zeta^{-1} \right)
\]

(by Lemma 5)

\[
\leq (10TM^2)^{k-1} \left( 10^6, 4\zeta^{-1} \right) \times (2\zeta^{-1})^{k-1} \left( 10^6, 4\zeta^{-1} \right)
\]

\[
\leq (10TM^2)^{k-1} \left( 10^6, 4\zeta^{-1} \right) \times (2\zeta^{-1})^{k-1} \left( 10^6, 4\zeta^{-1} \right)
\]

\[
\leq (10TM^2)^{k-1} \left( 10^6, \zeta^{-1} \right)
\]

(by bounds (12) and (13) in the proof of Lemma 13)

\[
\leq (10TM^2)^{k+1} \left( 10^6, \frac{2(K+1)}{\alpha} \right).
\]

In any case, \( Q_b \) will take at most \( 2(100TM)^{k+1} \left( 10^6, \frac{2(K+1)}{\alpha} \right) \) number of samples from simulator \( f \) 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. Q.E.D.

7.4. Proofs of auxiliary results

This subsection consists of detailed proofs Lemma 13.

Proof of Lemma 13  Recall that
\[
g^k(\epsilon, \delta) = (10TM^2)^kU(10^\frac{6}{1}) \left(1 + \log \left(\frac{1}{\delta}\right)\right)^{kU(10^\frac{6}{1})},
\]
\[
f^k(\epsilon, \delta) = C(10TM^2)^kU(10^\frac{6}{1}) \left(1 + \log \left(\frac{1}{\delta}\right)\right)^{kU(10^\frac{6}{1})}.
\]

The fact that both \(g^k\) and \(f^k\) are monotone in \(\epsilon\) and \(\delta\) is straightforward and we omit the proof.

Next let’s show the bounds on \(g^0\) and \(f^0\). Indeed, \(g^0(\epsilon, \delta) = 10^{-1}(TM^2)^{-1}(1 + \log(\frac{1}{\delta}))^{-1} \geq 10^{-1}TM^2 \log(\frac{\epsilon}{g}) \geq 10^{-1}TM \log(\frac{cM}{\delta}) \geq 2e^{-2}TM \log(\frac{2M}{\delta}).\) Similarly, \(f^0(\epsilon, \delta) = C10^{-1}(TM^2)^{-1}(1 + \log(\frac{1}{\delta}))^{-1} \geq C10^{-1}TM \log(\frac{cM}{\delta}) \geq 3CM\epsilon^{-2}\log(\frac{2M}{\delta}).\)

Finally, let’s prove that \(g^k\) and \(f^k\) satisfy the recursive inequalities. Let’s first state and prove several properties of \(kU\).
\[
kU(10^6, 10^\frac{6}{1}) \geq 2 \times kU(10^6, 4^\frac{8}{1}) + \frac{6}{\epsilon} \tag{12}
\]
\[
10^{kU(10^6, 10^\frac{6}{1})} \geq 10^{kU(10^6, 4^\frac{8}{1})} \times (400\epsilon^{-2})^{kU(10^6, 4^\frac{8}{1})} \times \exp(200\epsilon^{-2}) \tag{13}
\]

The above can be shown via induction. Indeed, in the base case \(k = 0\), inequality (12) becomes \(10^{6\epsilon^{-1}} \geq 2 \times 4^{8\epsilon^{-1}} + 6\epsilon^{-1}\) which can be easily verified for all \(\epsilon \in (0, 1)\). Inequality (13) becomes
\[
10^{10^\frac{6}{1}} \geq 10^{10^\frac{8}{1}} \times (400\epsilon^{-2})^{4^\frac{8}{1}} \times \exp(200\epsilon^{-2}).
\]

Similarly, it can be verified directly that the above holds for all \(\epsilon \in (0, 1)\). Now suppose inequalities (12) and (13) hold true for some \(k \geq 0\), we prove it also holds in case \(k + 1\). Indeed,
\[
k+1U(10^6, 10^\frac{6}{1}) = 10^{6 \times kU(10^6, 10^\frac{6}{1})}
\]
\[
\geq 10^{12 \times kU(10^6, 4^\frac{8}{1}) + 6 \times 4^\frac{8}{1}}
\]
\[
= \left(10^{12 \times kU(10^6, 4^\frac{8}{1}) + 6 \times 4^\frac{8}{1}}\right)^2 \times 10^{3\epsilon}
\]
\[
\geq 2 \times 10^{12 \times kU(10^6, 4^\frac{8}{1}) + 6 \times 4^\frac{8}{1}} \times 10^{3\epsilon}
\]
\[
10^{k+1U(10^6, 10^\frac{6}{1})} = \exp(\log(10) \times 10^{k+1U(10^6, 10^\frac{6}{1})})
\]
\[
\geq \exp\left(\log(10) \times 10^{6 \times kU(10^6, 4^\frac{8}{1})} \times (400\epsilon^{-2})^{6 \times kU(10^6, 4^\frac{8}{1})} \times \exp(1200\epsilon^{-2})\right)
\]
Inequalities (12) and (13) 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

$$g^{k+1}(\epsilon, \delta) = (10TM^2)^{k+1}U(10^6, 1) \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^{k+1}U(10^6, \frac{1}{t})$$

$$= (10TM^2)^4 U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^k U(10^6, 10^6)$$

$$\geq 10^{10^6} \left( 1 + \log \left( \frac{1}{\delta} \right) \right)^k U(10^6, 10^6) \times \exp(100\epsilon^{-2})$$

$$\geq 10^{10^6} \times \exp(100\epsilon^{-2}) \times (10^6)^{k+1} U(10^6, 10^6) \times \exp(100\epsilon^{-2})$$

$$\geq 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$\geq 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$\geq 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$\geq 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$\geq 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$= 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

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

$$= 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

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

$$= 2(10TM^2)^{k+1} U(10^6, 10^6) \left( 1 + \log \left( \frac{1}{\delta} \right) \right) \times T \times M \times \exp(100\epsilon^{-2})$$

$$\times \exp(200\epsilon^{-2}) \times M \times T^{6\epsilon^{-1}} \times (1 + \log(\frac{M}{\delta}))^{6\epsilon^{-1}}$$
\[ f^{k+1}(\epsilon, \delta) = C(10TM^2)^{k+1} U(10^6, \frac{1}{\delta}) \left(1 + \log \left( \frac{1}{\delta} \right) \right)^{k+1} U(10^6, \frac{1}{\delta}) \]

\[ \geq C \times (10TM^2)^{k} U(10^6, \frac{1}{\delta}) \times T^{k} U(10^6, \frac{1}{\delta}) \times M^{1+k} U(10^6, \frac{1}{\delta}) \times (400e^{-2})^{k} U(10^6, \frac{1}{\delta}) \times \exp(200e^{-2} \times T \times 6e^{-1} \times (1 + \log \left( \frac{M}{\delta} \right))^6e^{-1} \]

\[ \geq 4C \times (10TM^2)^{k} U(10^6, \frac{1}{\delta}) \times T^{k} U(10^6, \frac{1}{\delta}) \times M^{1+k} U(10^6, \frac{1}{\delta}) \times (200e^{-2})^{k} U(10^6, \frac{1}{\delta}) \times \exp(200e^{-2} \times T \times 6e^{-1} \times (1 + \log \left( \frac{M}{\delta} \right))^6e^{-1} \]

\[ \geq 4C \times g^{k} \left( 4^{-8e^{-1}}, \frac{1}{2} \delta M^{-1} T^{-6e^{-1}} (1 + \log \left( \frac{M}{\delta} \right))^{-3e^{-1}} \exp(-200e^{-2}) \right) \times \exp(200e^{-2} \times M \times T \times 6e^{-1} \times (1 + \log \left( \frac{M}{\delta} \right))^6e^{-1} \]

\[ = 2 \left( C \times g^{k} \left( 4^{-8e^{-1}}, \frac{1}{2} \delta M^{-1} T^{-6e^{-1}} (1 + \log \left( \frac{M}{\delta} \right))^{-3e^{-1}} \exp(-200e^{-2}) \right) \times \exp(200e^{-2} \times M \times T \times 6e^{-1} \times (1 + \log \left( \frac{M}{\delta} \right))^6e^{-1} \right) \]

Combining the above then completes the proof. Q.E.D.

**References**


Cohen-Hillel T, Panchamgam K, Perakis G (2019a) Bounded memory peak end models can be surprisingly good. *Available at SSRN 3360643*.

Cohen-Hillel T, Panchamgam K, Perakis G (2019b) High-low promotion policies for peak-end demand models. *Available at SSRN 3360680*.


Wilhelm M, Winter C (????) Finite element valuation of swing options.