Buying Information for Stochastic Optimization

Mingchen Ma [1] **Christos Tzamos** [1]

Abstract

Stochastic optimization is one of the central problems in Machine Learning and Theoretical Computer Science. In the standard model, the algorithm is given a fixed distribution known in advance. In practice though, one may acquire at a cost extra information to make better decisions.

In this paper, we study how to buy information for stochastic optimization and formulate this question as an online learning problem. Assuming the learner has an oracle for the original optimization problem, we design a 2-competitive deterministic algorithm and a $e/(e-1)$-competitive randomized algorithm for buying information. We show that this ratio is tight as the problem is equivalent to a robust generalization of the ski-rental problem, which we call super-martingale stopping.

We also consider an adaptive setting where the learner can choose to buy information after taking some actions for the underlying optimization problem. We focus on the classic optimization problem, Min-Sum Set Cover, where the goal is to quickly find an action that covers a given request drawn from a known distribution. We provide an 8-competitive algorithm running in polynomial time that chooses actions and decides when to buy information about the underlying request.

1. Introduction

1.1. Offline and Adaptive Stochastic Optimization

Stochastic optimization is one of the core problems in machine learning and theoretical computer sciences. In stochastic optimization, the input parameters of the problems are random variables drawn from a known distribution. Given the distribution of the parameters, a learner constructs a feasible solution in advance (offline stochastic optimization) or adaptively (adaptive stochastic optimization) to optimize the objective function in expectation. Formally, the two types of stochastic optimization problems can be defined in the following way.

Definition 1.1 (Offline Stochastic Optimization)**.** Let $\mathcal{S}$ be a set of scenarios and $\mathcal{X}(\mathcal{S})$ be a set of actions. Let $\ell(A, s) : 2^{\mathcal{X}} \times \mathcal{S} \to R_+$ be a loss function. An offline stochastic optimization problem $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ is to find a set of actions A that minimize $\mathbf{E}_{s\sim\mathcal{D}}\ell(A, s)$, where $\mathcal{D}$ is a distribution over $\mathcal{S}$.

Definition 1.2 (Adaptive Stochastic Optimization)**.** Let $\mathcal{S}$ be a set of scenarios and $\mathcal{X}(\mathcal{S})$ be a set of actions. Initially, a random scenario s is drawn according to a distribution $\mathcal{D}$. Then, the learner sequentially chooses actions $a_1, a_2, \ldots$ and after the t-th action observes a (possibly randomized) outcome $r((a_1, a_2, \ldots, a_t), s) \in \mathbb{R}$. The goal of the learner is to take a sequence of actions A that minimizes $\mathbf{E}_{s\sim\mathcal{D}}\ell(A, s)$ for a given loss function $\ell(A, s)$, possibly exploiting the information gained about s along the way.

A huge body of work among different communities such as machine learning, theoretical computer science, statistics, and operations research has studied stochastic optimization problems given their numerous applications. For example, methods of offline stochastic optimization have been widely applied to problems such as training machine learning models (Shalev-Shwartz et al., 2009; Bottou, 2010; Kingma & Ba, 2014) and mechanism design (Nisan & Ronen, 1999; Hartline, 2013; Roughgarden, 2016). On the other hand, many adaptive stochastic optimization problems such as Pandora's Box problem (Weitzman, 1979; Chawla et al., 2020), active learning (Dasgupta, 2004; Settles, 2012) and optimal decision tree (Adler & Heeringa, 2012; Li et al., 2020) have also been applied to areas like artificial intelligence, microeconomics, and operations research.

A common assumption in these works is that the distribution $\mathcal{D}$ of the scenario s is considered as a given. However, such an assumption is not realistic in practice. A learner in practice has many ways to gain extra knowledge on the optimization problem he is going to solve. With the extra knowledge, it is reasonable that the learner updates the prior distribution $\mathcal{D}$ to some posterior distribution $\mathcal{D}'$ and uses a better strategy to solve the problem. As a concrete example, consider n bidders that compete over an item in an

[*]Equal contribution [1]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA. Correspondence to: Mingchen Ma <mma54@wisc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

auction. Classic auction theory assumes that the auctioneer only knows a prior distribution $\mathcal{D}$ over the buyer values and wants to design an auction to optimize a target objective such as welfare or revenue. In practice though, there is a number of information sources available to the auctioneer that provide information about the bidders such as their demographics, their preferences or their purchase history. Such information can be very useful.

However, this information does not come for free. It may cost significant amounts of money or time and it is not clear in advance, how helpful this information will be. In the example, the auctioneer may pay an information provider only to receive irrelevant pieces of information or information already known.

1.2. Our Contribution and Techniques

In this paper, we study the problem of buying information for stochastic optimization. We consider a learner that wants to minimize the total cost spent on solving the optimization problem and the cost of acquiring information.

We model the information acquisition process using a signaling scheme (Emek et al., 2014). A signaling scheme is a (randomized) function f from the set of scenarios $\mathcal{S}$ to a signal space $\mathbb{R}$. If a learner asks for feedback from f, he will receive a signal y and the prior distribution can be updated as $\mathcal{D}|_{f(s)=y}$. In our model, we assume there is a sequence of signaling schemes $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ arriving online. At any timestep t, based on the signals received so far, the learner has the choice to continue purchasing the next signal given by f_t or stop. Our goal is to construct a learner who is competitive to the cost of a prophet who knows the structure of $\mathcal{F}$ in advance and can take optimal actions.

For offline optimization, all signals must be purchased before taking any actions in the underlying stochastic optimization problem. We assume the learner is able to compute an (approximate) optimal solution for the underlying problem given the available information at any point in time. The goal of the learner is then to adaptively decide when to stop buying feedback. Our main results in this setting are summarized below:

Theorem 1.3 (Informal Version of Theorem 3.6 and Theorem 3.8)**.** *There exist a 2-competitive deterministic learner and an $\frac{e}{e-1}$-competitive randomized learner to buy information for offline stochastic optimizations.*

We show that both learners can be implemented efficiently and have competitive ratios that are information theoretically optimal. Thus, we give a comprehensive understanding of buying feedback for offline stochastic optimization. To solve the problem, we formulate it as a super-martingale stopping problem: There is an unknown sequence of random variables $(X_0, X_1, \dots)$ satisfying $\mathbf{E}(X_{i+1} \mid X_i) \leq X_i$. The realizations of the random variables arrive online and an algorithm outputs a stopping index i^* adaptively to minimize $\mathbf{E}(i^* + X_{i^*})$. The super-martingale stopping problem can be seen as a generalization of the classic ski-rental problem introduced in (Karlin et al., 1994) where all $X_i \in \{0, B\}$ and its variant introduced in (Chawla et al., 2020), where $(X_0, X_1, \dots)$ are monotone decreasing constants. In the more general setting of super-martingale stopping though, the values of X_i may not be monotone, and they are only monotone in expectation. This makes the problem significantly more challenging and as we show in Appendix C, natural algorithms for ski-rental problems are not competitive for our problem.

For adaptive stochastic optimization, it is also natural to intertwine purchasing information with taking actions. For example, several actions may be taken first in the problem and then information may be purchased conditional on their outcome. As this setting is more problem dependent, we focus on a paradigmatic case of adaptive stochastic optimization, where there is a random set of good actions, and the learner takes actions in each round until a good action is chosen. Such a problem is called Min Sum Set Cover (MSSC), a well-studied adaptive stochastic optimization problem (Bar-Noy et al., 1998; 1999; Feige et al., 2004). In our model, the learner has an extra action at each round to buy information getting a better estimate of the probability that an action is good.

We provide an algorithm for this problem competitive to a prophet that knows the sequence of signaling schemes in advance:

Theorem 1.4 (Informal Version of Theorem 4.6)**.** *There is a poly-time learner that is 8-competitive for buying information for Min Sum Set Cover.*

We achieve this in two steps. In the first step, we show we can shrink the action space so that we don't need to consider when to buy feedback. We introduce a simpler model called adaptive stochastic optimization with time dependent feedback, where a learner takes an action in each round, and feedback arrives for free after an action is taken. We show in Theorem 4.3 that if there is a learner that is α-competitive for adaptive stochastic optimization with time dependent feedback, then we can use it to construct a 2α-competitive learner to buy information for adaptive stochastic optimization. Our second step is to prove the following technical theorem, which is of independent interest.

Theorem 1.5 (Informal Version of Theorem 4.4)**.** *The greedy algorithm is 4-competitive for MSSC with time dependent feedback.*

There is a lot of work done for analysis of the greedy algorithm of min sum coverage objective under different settings (Feige et al., 2004; Streeter & Golovin, 2008; Golovin &

Krause, 2011). The analysis is usually based on an elegant histogram approach proposed in (Feige et al., 2004). However, in our model, the decision made by the learner is fully adaptive and it is hard to adapt such an analysis directly. Instead, we bypass such difficulty and use an interesting linear programming dual approach to analyze the greedy algorithm. Besides algorithmic results, we also present hard instances to build information theoretic lower bound for MSSC under our models.

1.3. Applications of our Model

Buying information is very common in practice. In fact, our model fits well in both theory and practical applications. In this section, we give several applications of our model. We first give a typical example of buying information for offline stochastic optimization.

Selling One Item with Feedback There is a seller who wants to sell an item to a buyer. The seller sets a price p for the item. The buyer has a value v for the item and would like to pay the price p for the item if $p \leq v$. However, if $p > v$, the buyer will not buy the item. Given a pair of (v, p), denote by $P(v, p) = p\mathbf{1}_{p \leq v}$ the payment of the buyer. The value of the buyer may depend on his nationality, education, or other factors. The information can be collected from the historic trade and thus the seller has a prior distribution $\mathcal{D}$ of the value v. The goal of the learner is to set up the price p to minimize $\mathbf{E}_{v \sim \mathcal{D}} (v - P(v, p))$. However, instead of setting the price immediately, the seller may pay some money to collect more information about the buyer. This can help the seller update the prior distribution of the value v. In practice, it is hard to predict the quality of the information. The question for the seller is how much information is sufficient for him to set up a good price.

Our second example is on buying information for adaptive stochastic optimization.

Optimal Decision Tree with Feedback A doctor wants to diagnose the disease of a patient. There are $\mathcal{X} = [n]$ different tests that can be performed by the doctor and $\mathcal{S} = [m]$ different possible diseases. If the patient has a disease $s \in \mathcal{S}$ and a test $a \in \mathcal{X}$ is performed, then the doctor will receive an outcome $r(s, a)$. The doctor has a prior distribution $\mathcal{D}$ of the disease s based on the symptom of the patient. In the standard optimal decision tree problem, based on the knowledge of $\mathcal{D}$, the goal of the doctor is to perform a sequence of tests adaptively to identify the disease while minimizing the expected cost of the tests. In practice, the doctor may choose not to run tests but instead send the patient home to see whether the symptoms worsen. However, this is also costly and it may be challenging to predict what symptoms will appear and how much time it will take for them to appear. Combined with an algorithm for computing approximately optimal decision trees, our work shows how to incorporate the symptom monitoring component to efficiently identify the disease.

Beyond these applications, our model fits well with many existing theoretical frameworks in learning theory. Here we take adaptive submodular optimization, a recently popular research direction in the field of machine learning as our example.

Adaptive Submodularity with Feedback Motivated by applications on artificial intelligence, (Golovin & Krause, 2011) introduces the notion of adaptive submodularity, which was a popular research topic in the last decade. A function $f(A, s)$ of a set of actions A and a random scenario s is adaptive submodular if $\mathbf{E}_s f(A, s)$ is a submodular function. After an action a is taken, the learner will see an outcome $s(a)$. Given the distribution of s, the learner will construct the action set A adaptively to optimize classic objectives for submodular functions (Fujishige, 2005) such as submodular maximization, min submodular coverage, and min sum submodular coverage. Many natural questions arise when feedback is involved in this framework. For example, if feedback is costly, how can we buy feedback to help us make adaptive decisions? If the feedback is free and time dependent, are existing policies still competitive?

1.4. Organization of paper

In Section 2, we formally introduce the model studied by the paper. In Section 3, we introduce the super-martingale stopping problem to study buying information for offline stochastic optimization. We give a tight deterministic algorithm and a tight randomized algorithm for the super-martingale stopping problem. Furthermore, we will discuss the robustness of these algorithms. In Section 4, we focus on buying information for adaptive stochastic optimization. We introduce the model of time dependent feedback and build a connection between adaptive stochastic optimization with time dependent feedback and buying information for adaptive stochastic optimization in Section 4.1. In Section 4.2, we show a simple greedy learner is 4-competitive for Min Sum Set Cover with time dependent feedback. And in Section 4.3, we design an 8-competitive algorithm for buying information for Min Sum Set Cover. Furthermore, we discuss the information theoretic lower bound for Min Sum Set Cover under both settings.

2. Stochastic Optimization with Feedback

2.1. Feedback Signals for Stochastic Optimization

Let $\mathcal{S}$ be a set of scenarios with a distribution $\mathcal{D}$ over $\mathcal{S}$ and let $\mathcal{Y}$ be a set of random variables over $\mathbb{R}$. A randomized signaling scheme $f : \mathcal{S} \to \mathcal{Y}$ is a map from $\mathcal{S}$ to $\mathcal{Y}$. Let s

be a scenario drawn from $\mathcal{D}$. A signal received from f is a realization $y \in \mathbb{R}$ of the random variable $f(s)$. Similarly, a deterministic signaling scheme $f : \mathcal{S} \to \mathbb{R}$ is a function from $\mathcal{S}$ to $\mathbb{R}$. When a scenario s is drawn, a signal received from f is defined by $y = f(s) \in \mathbb{R}$. In particular, any deterministic signaling scheme gives a partition of $\mathcal{S}$. Given the definition of a signaling scheme, we are able to define feedback for stochastic optimization problems.

Definition 2.1 (Feedback)**.** Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be a stochastic optimization problem. A sequence of feedback $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ is a sequence of **unknown** randomized (deterministic) signaling scheme. The tth feedback received by a learner is the pair $(y_t, \mathcal{D} \mid_{f_t(s)=y_t, f_{t-1}(s)=y_{y-1}, \ldots, f_0(s)=y_0})$, where y_t is the signal from f_t.

For convenience, we assume f_0 is a constant for every scenario, throughout the paper. Such an assumption is used to reflect the fact that the learner has no extra knowledge at time 0. In fact, for our model, randomized signaling schemes are equivalent to deterministic ones. We leave a discussion for this in Appendix A. In this paper, we consider deterministic signaling schemes. A deterministic signaling scheme can simplify our analysis and provide more intuition. In particular, if each signaling scheme $f \in \mathcal{F}$ is deterministic, then $\mathcal{F}$ can be represented as a tree. For such feedback $\mathcal{F}$, we define a feedback tree $T(\mathcal{F})$ as follows.

Definition 2.2 (Feedback Tree)**.** Let feedback $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ be a set of deterministic signaling schemes. The feedback tree $T(\mathcal{F})$ for $\mathcal{F}$ is a tree that is defined as follows. Each node $v \in T(\mathcal{F})$ contains a set of scenarios and the children of v form a partition of the set of scenarios contained in v. The root of $T(\mathcal{F})$ contains all scenarios. For every $s \in \mathcal{S}$, let $P(s) = (v_0, v_1, \ldots, v_n)$ be the longest path in $T(\mathcal{F})$ such that every node in $P(s)$ contains s. Then the set of scenarios contained in v_i is defined by $\{s' \in v_{i-1} \mid f_i(s') = f_i(s)\}$.

2.2. Problem Formulation

Although feedback is helpful for a learner to make better decisions for stochastic optimization problems, obtaining feedback always requires some cost. The cost can be either time or money. Thus, it is natural for a learner to consider how to balance the cost of asking for feedback and the cost of solving the optimization problem. We consider formulating this problem in an online fashion for offline and adaptive stochastic optimization problems.

Definition 2.3 (Buying Information for Offline Stochastic Optimization)**.** Let $(\mathcal{X}, \mathcal{S}_0, \ell, \mathcal{D}_0)$ be an offline stochastic optimization problem and $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ be a sequence of **unknown** feedback. Let $\mathcal{C} = \{c_t\}_{t=0}^{\infty}$ be a sequence of cost for receiving a signal from $f_{t+1} \in \mathcal{F}$. Here, $c_t : \mathbb{R} \to \mathbb{Z}_+$ is a nonnegative function that depends on the last received signal. In each time round $t \geq 0$, a learner receives an offline stochastic optimization problem $(\mathcal{X}, \mathcal{S}_t, \ell, \mathcal{D}_t)$ and a cost $c_t(y_t)$ to obtain a signal from f_{t+1}, where y_t is the signal received from f_t. Here, $\mathcal{S}_t = \{s \in \mathcal{S}_{t-1} \mid y_t \in \textbf{dom}(f_t(s))\}$ and $\mathcal{D}_t = \mathcal{D}_{t-1} \mid_{f_t(s)=y_t}$ for $t \geq 1$. The learner can either stop and pay $\sum_{j=0}^{t-1} c_j(y_j) + \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_t} \ell(A, s)$ or enter the next time round. An offline stochastic optimization with feedback $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$ is to decide a stopping time T adaptively to minimize $\mathbf{E}_T \left(\sum_{j=0}^{T-1} c_j(y_j) + \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_T} \ell(A, s) \right)$.

Let $I = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$ be an instance of offline stochastic optimization with feedback, denote by $\textbf{cost}(\mathcal{A}, I)$ the cost of the stopping time output by a learner $\mathcal{A}$ for the given instance. A learner is α-competitive if for every instance $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$, $\textbf{cost}(\mathcal{A}, I) \leq \alpha \text{OPT}(I) = \alpha \min_{\mathcal{A}} \textbf{cost}(\mathcal{A}, I)$.

We can describe the problem in a more intuitive way in terms of the feedback tree. Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be a stochastic optimization problem and $T(\mathcal{F})$ be a feedback tree. Each node v of $T(\mathcal{F})$ represents a new stochastic optimization problem $(\mathcal{X}, \mathcal{S}_v, \ell, \mathcal{D}_v)$, where $\mathcal{S}_v$ is the set of scenarios contained in v and $\mathcal{D}_v = \mathcal{D} \mid_{s \in \mathcal{S}_v}$. Solving this optimization problem needs a cost $\min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_v} \ell(A, s)$. Each node also has a cost c_v to move down for one step. The stochastic optimization problem and the cost will be revealed to the learner when the learner reaches v. $T(\mathcal{F})$ is unknown to the learner and a path of $T(\mathcal{F})$ is selected according to $\mathcal{D}$ initially. The learner will keep moving along the path by paying the cost c_v and will decide when to stop and solve the optimization problem. The benchmark we want to compare is a learner who knows the whole feedback tree in advance and thus can compute the optimal stopping time.

Definition 2.4 (Buying Information for Adaptive Stochastic Optimization)**.** Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be an adaptive stochastic optimization problem. $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ be a sequence of **unknown** feedback. Let $\mathcal{C} = \{c_t\}_{t=0}^{\infty}$ be a sequence of cost for receiving a signal from $f_{t+1} \in \mathcal{F}$. Here, $c_t : \mathbb{R} \to \mathbb{Z}_+$ is a nonnegative function that depends on the last received signal. Initially, a scenario s is drawn according to $\mathcal{D}$. In each time round t, a learner first adaptively receives an arbitrary number of signals $y(s)$ from the sequence $\mathcal{F}$ by paying the corresponding cost, then selects an action $a_t \in \mathcal{X}$. Let $T(s)$ be the number of signals received by the learner if s is drawn. An adaptive stochastic optimization problem with feedback is to make decisions to ask for feedback and take actions adaptively in each time round to minimize $\mathbf{E}_{s \sim \mathcal{D}} \left(\ell(A, s) + \sum_{j=0}^{T(s)-1} c_j(y_j(s)) \right)$.

Let $I = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$ be an instance of adaptive stochastic optimization with feedback, denote by $\textbf{cost}(\mathcal{A}, I)$ the expected cost of the decisions made by a learner $\mathcal{A}$ for the given instance. A learner is α-competitive if for every instance $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$, $\textbf{cost}(\mathcal{A}, I) \leq \alpha \text{OPT}(I) = \alpha \min_{\mathcal{A}} \textbf{cost}(\mathcal{A}, I)$.

3. Buying Information for Offline Stochastic Optimization and Super-Martingale Stopping Problem

Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be an offline stochastic optimization problem and f be a signaling scheme. Denote by $\mathcal{D}_y$ the posterior distribution of $\mathcal{D}$ after receiving signal y from f. Although it is possible that $\min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}} \ell(A, s) < \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_y} \ell(A, s)$, it is always true that

$$\mathbf{E}_y \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_y} \ell(A, s) \le \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}} \ell(A, s).$$

That is to say, feedback is always helpful in expectation. This implies the sequence of minimum value of the stochastic optimization problems is a super-martingale. Formally, given a sequence of feedback $\mathcal{F}$, denote by D_i the posterior distribution after receiving signals from $f_0, f_1, \ldots, f_i$. Let random variable $X_i = \min_{A \subseteq \mathcal{X}} \mathbf{E}_{s \sim \mathcal{D}_i} \ell(A, s)$. Then for every $i \ge 0$, we have $\mathbf{E}\left(X_{i+1} \mid X_i\right) \le X_i$. This motivates us to formulate the problem of buying information as the following super-martingale stopping problem. As we discuss in Appendix B, super-martingale stopping is equivalent to buying information for stochastic optimization.

3.1. Super-Martingale Stopping Problem

Definition 3.1 (Super-Martingale Stopping Problem)**.** Let $X_0, X_1, \ldots, X_n$ be a sequence of nonnegative random variables unknown to the learner. Assume for every i, $\mathbf{E}(X_{i+1} \mid X_i) \le X_i$. The problem has $n+1$ rounds. In the ith round, given an observed realization of $X_0, \ldots, X_i$, a learner decides either to stop and pay $i + X_i$ or to obtain the realization of X_{i+1} and go to the next round. The goal of the learner is to compute a decision rule to obtain a stopping time i^* **only based on the observed realization of the sequence** to minimize $\mathbf{E}(i^* + X_{i^*})$.

For convenience, we assume X_0 is a constant throughout the paper. Suppose each random variable X_i has finite support, then the sequence can be represented by a tree T, where a node v with depth i stores a realization of X_i. To simplify the notation, we use v to denote both the node and the value stored at the node. When we make a single movement from node v, we will reach a child v' of v with probability $\Pr(X_{i+1} = v' \mid X_i = v)$. An optimal learner knows tree T in advance and can decide in advance which node to stop to optimize the expected cost. Formally, a set of stopping nodes S is feasible for T if every path of T with length n contains one and only one stopping node. The cost of S is $\sum_{v \in S} \Pr(v)(\text{depth}(v) + v)$. We denote by $\text{OPT}(T)$ the minimum cost among all feasible sets of stopping nodes of T. An algorithm is α-competitive if for every instance of the super-martingale stopping problem with a representation T, the expected cost of the algorithm $\text{ALG}(T) = \mathbf{E}(i^* + X_{i^*}) \le \alpha \text{OPT}(T)$.

In the ski-rental problem studied in (Karlin et al., 1994), there is a pair of positive numbers (B, T) such that $X_i = B$ if $i < T$ and $X_i = 0$ if $i \ge T$. This implies that ski-rental problem is a special case of the super-martingale stopping problem. Thus, we have the following information theoretic lower bound for the super-martingale stopping problem.

Theorem 3.2. *For every $\epsilon > 0$, no randomized algorithm is $\frac{e}{e-1} - \epsilon$-competitive for the super-martingale stopping problem.*

Theorem 3.3. *For every $\epsilon > 0$, no deterministic algorithm is $2 - \epsilon$-competitive for super-martingale stopping problem.*

Recall that the key idea in the design of algorithms for ski-rental problem is to balance the payment X_i and the index i. However, this idea cannot be simply applied to the super-martingale stopping problem. There are two difficulties faced in the super-martingale stopping problem. First, since any algorithm can only get information from one path of the tree, it is hard to estimate the expected stopping time for the whole tree. Second, unlike most ski-rental type problems, the value X_i is not necessarily decreasing. It is possible that an algorithm moves for one step but sees an X_i with a very large value. We will show in Appendix C that some natural algorithms that work for ski-rental problems are not competitive for the super-martingale stopping problem. On the other hand, in Appendix D, we establish a simple randomized 2-competitive algorithm for the super-martingale stopping problem using a completely novel idea. Although the algorithm we present in Appendix D shows competitive algorithms do exist for super-martingale stopping problem, the competitive ratio doesn't match the information theoretic lower bound in Theorem 3.2 and Theorem 3.3. In the following sections, we will give a tight deterministic algorithm and randomized algorithm for the super-martingale stopping problem. Furthermore, we will also discuss the robustness of these algorithms, when the input is not a super-martingale.

The key idea for designing our algorithms is to maintain the following estimator $Q_p(t)$ throughout the execution of the algorithms. Let $(X_1, \ldots, X_n)$ be an instance of super-martingale stopping and let T be its tree representation. Initially, a path $p = (v_0, v_1, \ldots, v_n)$ of T will be drawn randomly according to the joint distribution of $(X_0, \ldots, X_n)$. We define a function $v_p(t) = v_i$, if $t \in [i, i+1)$. Furthermore, we define $Q_p(t) = \int_0^t \frac{1}{v_p(t)} dt$. In particular, $Q_p(t)$ only depends on our observed realization and doesn't depend on the realization of the random variables we have not seen. We notice that $Q_p(t)$ is strictly increasing with respect to t and thus for every $s \ge 0$, we can define its inverse function $Q_p^{-1}(s) = t$, where $Q_p(t) = s$. The power of Q is that it can be used to upper bound and lower bound the optimal stopping time, which can be summarized by the following two lemmas that we will frequently used in our proof. The proof of Lemma 3.5 can be found in Appendix E.1 due to a lack of space.

Lemma 3.4. *Let T be a tree representation of an instance of the super-martingale stopping problem and let p be a path of T. Then for every $s > r > 0$, $Q_p^{-1}(s) - Q_p^{-1}(r) = \int_r^s v_p(Q_p^{-1}(w))dw$.*

Proof. The proof follows a change of variable. We write $w = Q_p(t)$. Then we have

$$\int_r^s v_p(Q_p^{-1}(w))dw = \int_{Q_p^{-1}(r)}^{Q_p^{-1}(s)} v_p(t)dQ_p(t)$$
$$= \int_{Q_p^{-1}(r)}^{Q_p^{-1}(s)} \frac{v_p(t)}{v_p(t)}dt = Q_p^{-1}(s) - Q_p^{-1}(r).$$

□

Lemma 3.5. *Let $v \in T$ be a node with depth i and let $\{p_j\}_{j=1}^k$ be the set of paths that passes v. For every $\rho^* \in [Q_{p_j}(i), Q_{p_j}(i+1)]$ and for every $\rho \geq \rho^*$, $\sum_{j=1}^k \Pr(p_j)(Q_{p_j}^{-1}(\rho) - Q_{p_j}^{-1}(\rho^*)) \leq \Pr(v)(\rho - \rho^*)v$.*

3.2. A Tight Deterministic Algorithm for Martingale Stopping

In this section, we propose a simple deterministic 2-competitive algorithm for the super-martingale stopping problem. The competitive ratio is tight according to Theorem 3.3. We leave the proof for Appendix E.2 due to the space limit.

Theorem 3.6. *There is a deterministic poly-time algorithm that is 2-competitive for the super-martingale stopping problem.*

Algorithm 1 DETERMINISTICSTOPPING (2-competitive deterministic algorithm for super-martingale stopping)

```
for i=0,1,2,... do
  Observe $X_i = v_i$ and compute $Q_p(t)$ for $t \in [i, i+1]$ based on the realization of $X_0, \dots, X_i$.
  if $\exists t \in (i, i+1]$ such that $Q_p(t) = 1$ then
    Stop at time $i$ and pay $i + v_i$.
  end if
end for
```

In particular, if the sequence of random variables is monotone decreasing, then our algorithm can even compete against a prophet who knows the realization of the sequence in advance.

Corollary 3.7. *Let I be an instance of the super-martingale stopping problem and $(X_0, X_1, \dots)$ be the input sequence. Denote by ALG(I) the cost of Algorithm 1 over instance I. If $(X_0, X_1, \dots)$ is monotone decreasing, then ALG$(I) \leq 2\mathbf{E}\min_i (i + X_i)$.*

Proof. Let $x = (x_0, x_1, \dots)$ be a realization of $(X_0, X_1, \dots)$ and denote by ALG(x) the cost of Algorithm 1 if the realization is x. Since x is monotone decreasing, we have ALG$(x) \leq 2\min_i (i + x_i)$. Thus,

$$\text{ALG}(I) \leq \mathbf{E}_x 2 \min_i (i + x_i) = 2\mathbf{E}\min_i (i + X_i).$$

□

3.3. A Tight Randomized Algorithm for Martingale Stopping

In this section, we extend the idea of Theorem 3.6 to obtain a $\frac{e}{e-1}$-competitive randomized algorithm for the super-martingale stopping problem. Notice that according to Theorem 3.2, the competitive ratio is tight. Recall that in the Algorithm 1, we maintain an estimator $Q_P(t)$ throughout the execution of the algorithm and stop when $Q_P(t) = 1$. To obtain a better randomized algorithm, we select a random threshold ρ initially, and stop when $Q_P(t)$ exceeds this threshold. The proof of Theorem 3.8 can be found in Appendix E.3.

Theorem 3.8. *There is a randomized poly-time algorithm for the super-martingale stopping problem that is $\frac{e}{e-1}$-competitive.*

Algorithm 2 RANDOMIZEDSTOPPING ($\frac{e}{e-1}$-competitive algorithm for super-martingale stopping)

```
Randomly draw a threshold $\rho \in [0, 1]$ with a probability density function $p(\rho) = \frac{e^\rho}{e-1}$.
for i=0,1,2,... do
  Observe $X_i = v_i$ and compute $Q_p(t)$ for $t \in [i, i+1]$ based on the realization of $X_0, \dots, X_i$.
  if $\exists t \in [i, i+1]$ such that $Q_p(t) = \rho$ then
    Stop at time $t$ and pay $i + v_i$. {Every time we stop, $i \leq t$.}
  end if
end for
```

Similarly, we have the following corollary, when the input sequence is monotone decreasing.

Corollary 3.9. *Let I be an instance of the super-martingale stopping problem and $(X_0, X_1, \dots)$ be the input sequence. Denote by ALG(I) the cost of Algorithm 2 over instance I. If $(X_0, X_1, \dots)$ is monotone decreasing, then ALG$(I) \leq \frac{e}{e-1}\mathbf{E}\min_i (i + X_i)$.*

Proof. Let $x = (x_0, x_1, \dots)$ be a realization of $(X_0, X_1, \dots)$ and denote by ALG(x) the cost of Algorithm 2 if the realization is x. Since x is monotone decreasing, we have ALG$(x) \leq \frac{e}{e-1}\min_i (i + x_i)$. Thus,

$$\text{ALG}(I) \leq \mathbf{E}_x \frac{e}{e-1} \min_i (i + x_i) = \frac{e}{e-1}\mathbf{E}\min_i (i + X_i).$$

□

3.4. A Discussion on Benchmark

In this section, we discuss the benchmark of the super-martingale stopping problem. According to Corollary 3.7 and Corollary 3.9, if the input sequence is monotone decreasing, then our algorithms can compete with a prophet who knows the realization of the sequence in advance. However, in general, it is not possible to compete against such a strong benchmark, since the gap between the two benchmarks can be arbitrarily large. Thus, it is only reasonable to compete with an algorithm that knows the structure of the feedback in advance. We formalize the discussion as the following theorem, whose proof is in Appendix E.4.

Theorem 3.10. *No algorithm is competitive against* $\mathbf{E}\min_i (i + X_i)$ *for the super-martingale stopping problem.*

3.5. On the Robustness of Algorithm 1 and Algorithm 2

In this part, we consider the robustness of Algorithm 1 and Algorithm 2. Back to our motivation, buying information for offline stochastic optimization. In the model of buying information for offline stochastic optimization, we assume that given a stochastic optimization problem, the learner can solve the problem exactly. However, since most stochastic optimization problems are NP-hard, usually, the learner might only have an α-approximate algorithm to solve it. If X_i is the optimal value of the stochastic optimization problem after receiving the ith feedback, then the cost to solve the problem for the learner is instead $\tilde{X}_i$, where $\tilde{X}_i \in [X_i, \alpha X_i]$. That is to say, if the learner stops at X_i, he will pay $i + \tilde{X}_i$. We remark that in this case, $\tilde{X}_0, \ldots, \tilde{X}_n$ may not satisfies the super-martingale property anymore, thus we cannot apply the analysis of Algorithm 1 and Algorithm 2 directly. However, we will show that the two algorithms are robust under such perturbation. In other words, Algorithm 1 is 2α-competitive and Algorithm 2 is $\frac{e}{e-1}\alpha$-competitive. Formally, we have the following theorem, whose proof is deferred to Appendix E.5.

Theorem 3.11. *Let T be a tree representation of an instance of the super-martingale stopping problem. Let $\tilde{T}$ be any tree constructed by changing the value of every leaf $v \in T$ by some value $\tilde{v} \in [v, \alpha v]$. If we run Algorithm 1 over $\tilde{T}$, then ALG$(\tilde{T}) \leq 2\alpha$OPT(T) and if we run Algorithm 2 over $\tilde{T}$, then ALG$(\tilde{T}) \leq \frac{e}{e-1}\alpha$OPT$(T)$.*

4. Buying Information for Adaptive Stochastic Optimization and Prophet Inequality

Unlike offline stochastic optimization with feedback, buying information for adaptive stochastic optimization is much more problem-dependent. For this reason, we consider designing competitive learners to buy information for specific problems. We choose Min Sum Set Cover, an extreme case of the adaptive stochastic optimization problem as the first problem studied under the feedback setting.

Definition 4.1 (Min Sum Set Cover)**.** Let $\mathcal{B} = [n]$ be a set of boxes, each box i contains an unknown number $b_i \in \{0, 1\}$. A learner can know b_i by querying box i, i.e. the action space $\mathcal{X} = \mathcal{B}$. A scenario $s \in \{0, 1\}^n$ is a binary vector that represents the number contained in each box. If scenario s is realized, then for every box $i \in \mathcal{B}$, $s_i = b_i$. A scenario s is covered if a box i such that $s_i = 1$ is queried. Let $\mathcal{S}$ be a set of scenarios and $\mathcal{D}$ be a probability distribution over $\mathcal{S}$. Let f be a sequence of feedback. A scenario s^* is drawn from $\mathcal{D}$ initially. In each round t, a learner takes an action $a_t \in \mathcal{X}$ to query the box a_t and observes the number contained in that box. Given an instance $(\mathcal{B}, \mathcal{S}, \mathcal{D})$ of Min Sum Set Cover, the goal of a learner is to construct the sequence of boxes A to query to minimize $\mathbf{E}_{s\sim\mathcal{D}}\ell(A, s)$, where $\ell(A, s)$ is the number of boxes in A to query until the drawn scenario s is covered.

The main contribution of this section can be broken down into two parts. In the first part, we give a general strategy to shrink the action space of buying information for a broad class of stochastic optimization problems. For such a class of problems, we show that if an α-prophet inequality exists for an adaptive stochastic optimization problem with time dependent feedback, which we will define later, then there is a 2α-competitive learner for buying information for adaptive stochastic optimization. In the second part, using such an idea, we construct an 8-competitive learner to buy information for Min Sum Set Cover(MSSC) by showing a 4-prophet inequality for MSSC with time dependent feedback. Furthermore, we will establish information theoretic lower bounds for MSSC under both settings.

4.1. Time Dependent Feedback and Prophet Inequality

A prophet inequality for an adaptive stochastic optimization is established when a signal arrives from f_t for free in each round. Formally, we have the following model.

Definition 4.2 (Adaptive Stochastic Optimization with Time Dependent Feedback)**.** Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be an adaptive stochastic optimization problem. $\mathcal{F} = \{f_t\}_{t=0}^{\infty}$ be a sequence of feedback. Initially, a scenario s is drawn according to $\mathcal{D}$. In each time round t, a learner receives a signal $y_t(s)$ from $f_t(s)$, then takes an action $a_t \in \mathcal{X}$. An adaptive stochastic optimization problem with time dependent feedback $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F})$ is to make decisions to construct a sequence of actions A adaptively to minimize $\mathbf{E}_{s\sim\mathcal{D}}\ell(A, s)$.

If we denote by $\mathbf{cost}(\mathcal{A}, I)$ be the expected cost of a learner $\mathcal{A}$ at a given instance I, then a learner is α-competitive if for every instance I, $\mathbf{cost}(\mathcal{A}, I) \leq \min_{\mathcal{A}} \mathbf{cost}(\mathcal{A}, I)$. In particular, here we are competing with a learner who knows $\mathcal{F}$ in advance. We say a stochastic optimization $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ satisfies an α-prophet inequality if there is an α-competitive learner for the corresponding stochastic optimization problem with time dependent feedback. We have the following theorem to establish the relation between the two problems.

Theorem 4.3. *If there is an α-competitive learner for Min Sum Set Cover with Time Dependent Feedback, then there is a 2α-competitive learner for Buying Information for Min Sum Set Cover.*

Although the statement of Theorem 4.3 is on MSSC here, the same results actually hold for a broader class of problems, where the loss function can be written as a covering function. Due to space limitations, we leave the general statement and the proof of Theorem 4.3 for Appendix F.1.

4.2. Min Sum Set Cover with Time Dependent Feedback

In this part, we establish a 4-prophet inequality for MSSC with time dependent feedback via the following theorem.

Theorem 4.4. *Algorithm 3, a simple greedy learner is 4-competitive for Min Sum Set Cover with Time Dependent Feedback.*

Algorithm 3 GREEDY (4-competitive Learner for MSSC with Time Dependent Feedback)

```
for t=0,1,2,... do
    Receive scenario set $S_t$ that are consistent with the feedback and outcomes received so far.
    Compute $\Pr(i) := \sum_{s\in S_t: s_i=1} \Pr(s \mid S_t)$
    Query any box $i^* \in \arg\max\{\Pr(i) \mid i \in \mathcal{B}\}$.
    if $s^*_{i^*} = 1$ then
        return
    end if
end for
```

Here we give an overview of our proof, the whole proof is deferred to Appendix F.2. Our proof is based on a linear programming approach. Assume the feedback is known in advance, then the problem becomes to assign a box for each node of the feedback tree $T(\mathcal{F})$ to minimize the average number of boxes used to cover the drawn scenario. This problem can be naturally lower bounded by a linear program, and thus every feasible solution to the dual of the linear program gives a lower bound for OPT. We will show that a simple greedy algorithm with no knowledge of $T(\mathcal{F})$ can be used to construct a feasible solution to the dual program such that the cost of the greedy algorithm is at most a quarter times the dual objective of the solution it constructs.

By Theorem 13 in (Feige et al., 2004), we know that for every $\epsilon > 0$, it is NP-hard to approximate MSSC within a ratio of $4 - \epsilon$. MSSC is a very special case of MSSC with Time Dependent Feedback, thus the result given by Theorem 4.4 is tight if we only consider learners that can be implemented in poly-time. However, in the classic MSSC, if we allow a learner to be implemented in super-polynomial time, then we can simply compute the optimal order of box to query using a brute force method. This gives a natural question. Is the knowledge of $\mathcal{F}$ useful? We show that such knowledge is indeed useful by giving the following information theoretical lower bound for MSSC with Time Dependent Feedback. That is to say, we consider all learners regardless of their running time. We establish the following information theoretic lower bound for MSSC with time dependent feedback. The proof is deferred to Appendix F.3.

Theorem 4.5. *For every $\epsilon > 0$, there is no deterministic learner that is $2-\epsilon$-competitive for Min Sum Set Cover with Time Dependent Feedback.*

4.3. Buying Information for Min Sum Set Cover

In the last section, we establish a prophet inequality for MSSC. In this section, we go back to the original motivation of buying feedback for adaptive stochastic optimization to discuss the upper bound and information theoretic lower bound for MSSC when asking for feedback requires some cost. The model of the problem is given as follows.

According to Theorem 4.4 and Theorem 4.3, we can immediately obtain an efficient competitive learner to buy feedback for Min Sum Set Cover, which is described in Algorithm 4.

Theorem 4.6. *There is a poly-time learner that is 8-competitive for buying information for Min Sum Set Cover.*

Algorithm 4 GREEDYBUYING (8-competitive learner for MSSC with Feedback)

```
for $t = 0, 1, 2, \ldots$ do
    Receive scenario set $S_t$ that are consistent with the feedback and outcomes received so far.
    Receive the cost $c_t$ to receive a signal $y_{t+1}$ from $f_{t+1}$
    for $j = 1 \ldots, c_t$ do
        Compute $\Pr(i) := \sum_{s\in S_t: s_i=1} \Pr(s \mid S_t)$.
        Query any box $i^* \in \arg\max\{\Pr(i) \min i \in \mathcal{B}\}$.
        if $s^*_{i^*} = 1$ then
            return
        else
            Update $S_t \leftarrow S_t \cap \{s \in \mathcal{S} \mid s_{i^*} = 0\}$.
        end if
    end for
    Pay $c_t$ to obtain signal $y_{t+1}$ from $f_{t+1}$.
end for
```

The main goal of this section is to obtain an information theoretical lower bound for buying information for MSSC. We establish the information theoretic lower bound via the following theorem, whose proof is in Appendix F.4.

Theorem 4.7. *For every $\epsilon > 0$, there is no deterministic algorithm that is $2 - \epsilon$-competitive for buying information for Min Sum Set Cover.*

References

Adler, M. and Heeringa, B. Approximating optimal binary decision trees. *Algorithmica*, 62(3):1112–1121, 2012.

Bar-Noy, A., Bellare, M., Halldórsson, M. M., Shachnai, H., and Tamir, T. On chromatic sums and distributed resource allocation. *Information and Computation*, 140 (2):183–202, 1998.

Bar-Noy, A., Halldórsson, M. M., and Kortsarz, G. A matched approximation bound for the sum of a greedy coloring. *Information Processing Letters*, 71(3-4):135–140, 1999.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Chawla, S., Gergatsouli, E., Teng, Y., Tzamos, C., and Zhang, R. Pandora's box with correlations: Learning and approximation. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1214–1225. IEEE, 2020.

Dasgupta, S. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17, 2004.

Emek, Y., Feldman, M., Gamzu, I., PaesLeme, R., and Tennenholtz, M. Signaling schemes for revenue maximization. *ACM Transactions on Economics and Computation (TEAC)*, 2(2):1–19, 2014.

Feige, U., Lovász, L., and Tetali, P. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

Fujishige, S. *Submodular functions and optimization*. Elsevier, 2005.

Golovin, D. and Krause, A. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Hartline, J. D. Mechanism design and approximation. *Book draft. October*, 122(1), 2013.

Karlin, A. R., Manasse, M. S., McGeoch, L. A., and Owicki, S. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, R., Liang, P., and Mussmann, S. A tight analysis of greedy yields subexponential time approximation for uniform decision tree. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 102–121. SIAM, 2020.

Nisan, N. and Ronen, A. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 129–140, 1999.

Roughgarden, T. *Twenty lectures on algorithmic game theory*. Cambridge University Press, 2016.

Settles, B. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114, 2012.

Shalev-Shwartz, S., Shamir, O., Srebro, N., and Sridharan, K. Stochastic convex optimization. In *COLT*, pp. 5, 2009.

Streeter, M. and Golovin, D. An online algorithm for maximizing submodular functions. *Advances in Neural Information Processing Systems*, 21, 2008.

Weitzman, M. L. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pp. 641–654, 1979.

A. Equivalence of Randomized and Deterministic Signaling Schemes

In our model, it is sufficient to study the case when each signaling scheme is deterministic. In this part, we give a brief discussion on the equivalence of randomized and deterministic signaling schemes.

Given a set of scenarios $\mathcal{S}$, a distribution $\mathcal{D}$ over $\mathcal{S}$, and a randomized signaling scheme f. We show we can construct a modified triple $(\mathcal{S}', \mathcal{D}', f')$ such that f' is a deterministic signaling scheme and $(\mathcal{S}', \mathcal{D}', f')$ is equivalent to $(\mathcal{S}, \mathcal{D}, f)$. The triple is constructed in the following way. $\mathcal{S}'$ contains multiple copies for each $s \in \mathcal{S}$. $\mathcal{D}'$ is a uniform distribution over $\mathcal{S}'$. For every $s \in \mathcal{S}$, assume the range of $f(s)$ is $\{y_1(s), \dots, y_k(s)\}$ and the set of copies is $\{Y_1(s), \dots, Y_k(s)\}$ accordingly. The sizes of the copies are made such that if we draw a scenario according to $\mathcal{D}'$, the probability that it is a copy of s is equal to the probability of obtaining s from $\mathcal{D}$. Furthermore, if we uniformly draw a copy from $\{Y_1(s), \dots, Y_k(s)\}$ the probability that we obtain a copy from $Y_i(s)$ is equal to the probability that we receive $y_i(s)$ from $f(s)$. In this way, we define $f'(s') = y_i(s)$ if $s' \in Y_i(s)$. Thus, we obtain an equivalent triple $(\mathcal{S}', \mathcal{D}', f')$ with a deterministic signaling scheme.

B. Equivalence of Super-Martingale Stopping and Buying Information for Offline Stochastic Optimization

In this part, we give a brief discussion on the equivalence of the super-martingale stopping problem and buying information for offline stochastic optimization problems.

We have seen that the super-martingale stopping problem is a special case of buying information for offline stochastic optimization. To see the other direction, it remains to see that given an instance $I = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$ of buying information for offline stochastic optimization, we can assume each $c_t = 1 \in \mathcal{C}$. We give the intuition here via the definition of feedback tree. Let $T(\mathcal{F})$ be a feedback tree. Assume a learner arrives at a node v of $T(\mathcal{F})$, the posterior distribution of the stochastic optimization problem at v is D_v and the cost to move to the next node v' is c_v. Then we can add $c_v - 1$ virtual nodes between v and v' such that the posterior distribution at each node is D_v and the cost to move to the next node is 1. After the modification, we can run any algorithm for the super-martingale stopping problem over the modified instance. We pay c_v to move to v' if and only if we reach v' in the modified instance. In this way, any α-competitive algorithm for the super-martingale stopping problem can be used to construct an α-competitive learner to buy information for offline stochastic optimization problems.

C. Natural Algorithms Fail for Martingale Stopping Problem

In this section, we show some natural algorithms that work for ski-rental problems but fail for the super-martingale stopping problem. According to (Karlin et al., 1994), it is well-known that the following algorithm is 2-competitive for the ski-rental problem.

Algorithm 5 ClassicSkiRental (2-competitive deterministic algorithm for ski-rental)

```
for $i = 0, 1, 2, \dots$ do
    Observe $X_i = v_i$
    if $v_i \le i$ then
        Stop and pay $i + v_i$.
    end if
end for
```

Theorem C.1. *Algorithm 5 is not competitive for the super-martingale stopping problem.*

Proof. We construct a sequence of instance I_n of the super-martingale stopping problem. Denote by ALG(I_n) the cost of Algorithm 5 over instance I_n and denote by OPT(I_n) the optimal cost of I_n. We will show that ALG$(I_n) \ge H_n$OPT(I_n), where H_n is the nth harmonic number.

Let $(X_0, \dots, X_n)$ be the sequence of random variables for instance I_n. Define $X_0 = 1$ to be a constant. For every $i \ge 1$, X_i can take two possible values. Given X_{i-1}, $X_i = 0$ with probability $\frac{1}{i+1}$ and with probability $\frac{i}{i+1}$, $X_i = \frac{i+1}{i} X_{i-1}$. That is to say, X_i is either 0 or $i + 1$ and $\mathbf{E}X_i = 1$.

Assume we run Algorithm 5 over instance I_n. Suppose we just observe X_i. If $X_i = 0$, then we stop and pay i right away.

If $X_i = i+1$, then Algorithm 5 will keep querying X_{i+1}. Denote by i^* the random variable of the stopping time of Algorithm 5. Then, we have

$$\mathbf{E}i^* = \sum_{i=1}^{n} \frac{i}{i+1} \prod_{j=1}^{i-1} \frac{j}{j+1} = \sum_{i=1}^{n} \frac{1}{i+1} = H_n - 1.$$

On the other hand, we know from the construction of the instance that $\mathbf{E}X_{i^*} = 1$, since $\mathbf{E}X_i = 1$ for every i. Thus the total cost of the algorithm is ALG$(I_n) = \mathbf{E}i^* + X_{i^*} = H_n$. On the other hand, we have OPT$(I_n) \le 1$, since it can simply stop at the beginning. This gives ALG$(I_n) \ge H_n$OPT(I_n), which implies that Algorithm 5 is not competitive.

□

The reason why Algorithm 5 fails is that $(X_1, \dots, X_n)$ might be an increasing sequence, which forces the algorithm to keep querying the next box forever. To avoid keeping querying boxes forever, a natural idea is to change the stopping rule by looking at the smallest value we have seen so far. However, it turns out that such a stopping rule still fails. We consider the following algorithm.

Algorithm 6 REVISEDSKIRENTAL (2-competitive deterministic algorithm for ski-rental)

```
for $i = 0, 1, 2, \dots$ do
    Observe $X_i = v_i$ and set $v^* = \min\{v_0, \dots, v_i\}$.
    if $v^* \le i$ then
        Stop and pay $i + v_i$.
    end if
end for
```

Theorem C.2. *Algorithm 6 is not competitive for the super-martingale stopping problem.*

Proof. We construct a sequence of instance I_n of the super-martingale stopping problem. Denote by ALG(I_n) the cost of Algorithm 6 over instance I_n and denote by OPT(I_n) the optimal cost of I_n. We will show that ALG$(I_n) \ge \Omega(n)$OPT(I_n).

Let $(X_0, \dots, X_n, X_{n+1})$ be the sequence of random variables of instance I_n of the super-martingale stopping problem. Define $X_0 = n$ and $X_{n+1} = 0$. For $i \in [n]$, X_i can take two possible values. Given X_{i-1}, $X_i = e^n X_{i-1}$ with probability e^{-n} and $X_i = 0$ with probability $1 - e^{-n}$. That is to say for $i \in [n]$, $\mathbf{E}X_n = n$. Notice that according to the stopping rule of Algorithm 6, X_{n+1} will never be queried by the algorithm. Thus, we have ALG$(I_n) \ge \mathbf{E}X_i = n$.

On the other hand, we consider an algorithm that keeps querying X_{i+1} if $X_i \ne 0$. Denote by i^* the stopping time of this algorithm. We know that $\mathbf{E}X_{i^*} = 0$. Furthermore, we have

$$\mathbf{E}i^* = \sum_{i=1}^{n+1} i(1 - e^{-n}) \prod_{j=1}^{i-1} e^{-n} \le e^{-n} \sum_{i=1}^{n+1} i \in O(1).$$

This implies that OPT$(I_n) \le \mathbf{E}i^* + X_{i^*} \in O(1)$, while ALG$(I_n) \in \Omega(n)$. Thus, Algorithm 6 is not competitive.

□

D. A Simple Randomized Algorithm for Martingale Stopping Problem

In this section, we give a simple randomized 2-competitive algorithm for the super-martingale stopping problem.

Algorithm 7 THROWCOIN (Simple 2-competitive randomized algorithm for super-martingale stopping)

```
for $i = 0, 1, 2, \dots$ do
    Observe $X_i = v_i$.
    Stop and pay $i + v_i$ with probability $\min\{1, 1/v_i\}$.
end for
```

Theorem D.1. *Algorithm 7 is 2-competitive for the super-martingale stopping problem.*

Proof. Let $(X_0, \dots, X_n)$ be a sequence of random variables, and let T be the tree representation of the sequence. Denote by OPT(T) the optimal cost of the instance and denote by ALG(T) the cost of Algorithm 7 over the instance. We prove the theorem using inductions on the number of random variables, which is also the depth of T.

If depth$(T) = 0$, which means there is only one random variable X_0 in the sequence, the cost of any algorithm is X_0 and the theorem holds trivially. Assuming the theorem holds for any tree with depth $n-k$, we show the theorem holds for any tree with depth $n-k-1$. Let T be a tree of an instance of super-martingale stopping problem such that depth$(T) = n-k-1$. Let v be the root of T and let $v^1, \dots, v^k$ be the children of v. Denote by T^i the subtree rooted at v^i. By a dynamic programming approach, we know that

$$\text{OPT}(T) = \min\{v, 1 + \sum_{i=1}^{k} \Pr(v^i)\text{OPT}(T^i)\}.$$

We consider two cases. In the first case, OPT$(T) = v$. Without loss of generality, we assume $v > 1$, otherwise, the algorithm will simply stop at v. The cost of Algorithm 7 is

$$\begin{aligned}
\text{ALG}(T) &= \frac{1}{v}v + (1 - \frac{1}{v})(1 + \sum_{i=1}^{k} \Pr(v^i)\text{ALG}(T^i)) \\
&\le \frac{1}{v}v + (1 - \frac{1}{v})(1 + 2\sum_{i=1}^{k} \Pr(v^i)\text{OPT}(T^i)) \\
&\le \frac{1}{v}v + (1 - \frac{1}{v})(1 + 2\sum_{i=1}^{k} \Pr(v^i)v^i) \\
&\le 1 + 1 - \frac{1}{v} + 2v - 2 \le 2v.
\end{aligned}$$

Here, in the first inequality, we use the induction hypothesis, in the third inequality, we use the super-martingale property.

In the second case, OPT$(T) = 1 + \sum_{i=1}^{k} \Pr(v^i)\text{OPT}(T^i)$. Similarly, we have

$$\begin{aligned}
\text{ALG}(T) &= \frac{1}{v}v + (1 - \frac{1}{v})(1 + \sum_{i=1}^{k} \Pr(v^i)\text{ALG}(T^i)) \\
&\le \frac{1}{v}v + (1 - \frac{1}{v})(1 + 2\sum_{i=1}^{k} \Pr(v^i)\text{OPT}(T^i)) \\
&\le 2\left(1 + \sum_{i=1}^{k} \Pr(v^i)\text{OPT}(T^i)\right).
\end{aligned}$$

This shows that for every instance with a tree representation T, ALG$(T) \le 2$OPT(T). This implies Algorithm 7 is 2-competitive.

□

E. Miss Proof in Section 3

E.1. Proof of Lemma 3.5

Proof. We prove this lemma using induction on the depth of v. If v has a depth of n (v is a leaf), then Lemma 3.5 follows directly by Lemma 3.4, since $Q_{p_j}^{-1}(\rho) - Q_{p_j}^{-1}(\rho^*) = \int_{\rho^*}^{\rho} v dw = (\rho - \rho^*)v$. Assume Lemma 3.5 holds for every node v' with depth $n-k$, we show this for a node v with depth $n-k-1$. We notice that if $\rho \le Q_{p_j}(n-k)$, then this is correct by Lemma 3.4. So in the rest of the proof, we assume $\rho > Q_{p_j}(n-k)$. Let $\{u_j\}_{j=1}^{\ell}$ be the set of children of v and let

$D(u_j) \subseteq \{p_j\}_{j=1}^k$ be the set of paths that passes u_j. then we have

$$\begin{aligned}\sum_{j=1}^{k} \Pr(p_j)(Q_{p_j}^{-1}(\rho) - Q_{p_j}^{-1}(\rho^*)) &= \sum_{j=1}^{k} \Pr(p_j)(n-k-Q_{p_j}^{-1}(\rho^*)) + \sum_{j=1}^{\ell}\sum_{p\in D(u_j)} \Pr(p)(Q_{p_j}^{-1}(\rho) - (n-k)) \\ &\le \sum_{j=1}^{k} \Pr(p_j)(Q_{p_j}(n-k) - \rho^*)v + \sum_{j=1}^{\ell} \Pr(u_j)(\rho - (Q_p(n-k)))u_j \\ &= \Pr(v)(Q_p(n-k) - \rho^*)v + \sum_{j=1}^{\ell} \Pr(u_j)(\rho - (Q_p(n-k)))u_j \\ &\le \Pr(v)(Q_p(n-k) - \rho^*)v + \Pr(v)(\rho - (Q_p(n-k)))v \\ &= \Pr(v)(\rho-\rho^*)v.\end{aligned}$$

Here, in the first inequality, we use the assumption of induction and in the second inequality, we use the fact that $\sum_{j=1}^{\ell} \Pr(u_j \mid v)u_j \le u_j$. □

E.2. Proof of Theorem 3.6

Proof. We show Algorithm 1 is 2-competitive. Let T be a tree representation of an instance of the super-martingale stopping problem. We maintain two sets of nodes $\tilde{O}$ and $\tilde{A}$ in the following way. For each path $p \subseteq T$. We travel down p from the root of T and stop traveling at a node v of p if either v is a stopping node of OPT(T) or it is a stopping node of Algorithm 1. In the first case, we add v to $\tilde{O}$, otherwise, we add it to $\tilde{A}$. We denote by $\tilde{T}$ the subtree of T with the set of leaves $\tilde{A} \cup \tilde{O}$. Furthermore, let $P(v)$ be the path of $\tilde{T}$ that ends at $v \in \tilde{O} \cup \tilde{A}$. Then we have the following lower bound for OPT(T).

$$\text{OPT}(T) \ge \sum_{v\in\tilde{O}} \Pr(v)\,(v + \text{depth}(v)) + \sum_{v\in\tilde{A}} \Pr(v)\text{depth}(v)$$

To upper bound ALG(T), we will need to establish the following inequality and claim. Let P be a path such that there is some $v \in P \cap \tilde{O}$, then there must be some stopping node $f_P(v) \in P$ of ALG(T) that has v as its ancestor. Let $D(v)$ be the set of paths that passes v. We know from the stopping rule of Algorithm 1 that for every $P \in D(v)$, $Q_P(\text{depth}(f_P(v))) \le 1$. By Lemma 3.5, we have

$$\sum_{P\in D(v)} \Pr(P)(\text{depth}(f_P(v)) - \text{depth}(v)) \le \sum_{P\in D(v)} \Pr(P)(Q_P^{-1}(1) - \text{depth}(v)) \le \Pr(v)\,(1 - Q_P(v))\,v \le \Pr(v)v. \quad (1)$$

Furthermore, we next prove the following claim.

Claim 1. Let T' be a subtree of T with the same root of T. Let S be the set of leaves of T'. For each $v \in S$, denote by $P(v)$ the path from the root to v. If every path of T has a node in S, then $\sum_{v\in S} \Pr(v)vQ_{P(v)}(\text{depth}(v)) \le \sum_{v\in S} \Pr(v)\text{depth}(v)$.

Proof of Claim. Let $v \in S$ be a leave of T'. Assume that depth$(v) = i$ and $P(v) = (v_0, \dots, v_i)$. Then

$$vQ_{P(v)}(i) = v_i \sum_{j=0}^{i-1} \frac{1}{v_j}. \quad (2)$$

Now we prove this claim by induction on the depth of T'. If T' has a depth of 0, then the claim holds trivially. Now we assume the claim for any tree with depth k, we show this holds for a tree T' with depth $k+1$. We remove the nodes with depth $k+1$ in T' and denote by the remaining tree $\bar{T}$. Denote by $\bar{S}$ the leaves of $\bar{T}$ and denote by K the set of leaves of $\bar{T}$

with depth k. For every node v, let $N(v)$ be the set of children of v. Then we have

$$\begin{aligned}
\sum_{v\in S} p(v)vQ_{P(v)}(\text{depth}(v)) &= \sum_{v\in \bar{S}} \Pr(v)vQ_{P(v)}(\text{depth}(v)) + \sum_{v\in K}\Pr(v)\sum_{u\in N(v)}\big(\Pr(u\mid v)(uQ_{P(u)}(k+1) - vQ_{P(v)}(k))\big)\\
&\le \sum_{v\in \bar{S}} \Pr(v)\text{depth}(v) + \sum_{v\in K}\Pr(v)\sum_{u\in N(v)}\big(\Pr(u\mid v)(uQ_{P(u)}(k+1) - vQ_{P(v)}(k))\big)\\
&\le \sum_{v\in \bar{S}} \Pr(v)\text{depth}(v) + \sum_{v\in K}\Pr(v)\sum_{u\in N(v)}\big(\Pr(u\mid v)u(Q_{P(u)}(k+1) - Q_{P(v)}(k))\big)\\
&= \sum_{v\in \bar{S}} \Pr(v)\text{depth}(v) + \sum_{v\in K}\Pr(v) = \sum_{v\in S}\Pr(v)\text{depth}(v).
\end{aligned}$$

Here, the first inequality follows by our induction, the second inequality follows by the super-martingale property and the second equality follows by (2).

◇

This gives the following upper bound for $\text{ALG}(T)$.

$$\begin{aligned}
\text{ALG}(T) &\le \sum_{v\in\tilde{O}}\left(\Pr(v)\,(v+\text{depth}(v)) + \sum_{P\in D(v)}\Pr(P)(\text{depth}(f_P(v)) - \text{depth}(v))\right) + \sum_{v\in\tilde{A}}\Pr(v)\,(v+\text{depth}(v))\\
&\le \sum_{v\in\tilde{O}}\Pr(v)\,(\text{depth}(v)+2v) + \sum_{v\in\tilde{A}}\Pr(v)\,(v+\text{depth}(v))\\
&\le \sum_{v\in\tilde{O}}\Pr(v)\,(\text{depth}(v)+2v) + \sum_{v\in\tilde{A}}\Pr(v)\,\big(v(Q_{P(v)}(\text{depth}(v))+1)+\text{depth}(v)\big)\\
&\le \sum_{v\in\tilde{O}}\Pr(v)\,(\text{depth}(v)+2v) + \sum_{v\in\tilde{A}}\Pr(v)\,\big(v(Q_{P(v)}(\text{depth}(v))+1)+\text{depth}(v)\big) + \sum_{v\in\tilde{O}}\Pr(v)\,\big(vQ_{P(v)}(\text{depth}(v))\big)\\
&\le 2\sum_{v\in\tilde{O}}\Pr(v)v + 2\sum_{v\in\tilde{A}}\Pr(v)v + 2\sum_{v\in\tilde{A}}\Pr(v)\text{depth}(v) + 2\sum_{v\in\tilde{A}}\Pr(v)\text{depth}(v)\\
&\le 2\text{OPT}(T).
\end{aligned}$$

Here, in the first inequality, we used the super-martingale property of T. In the second inequality, we use (1). In the third inequality, we use the stopping rule of Algorithm 1. The second last inequality follows by Claim 1.

□

E.3. Proof of Theorem 3.8

Proof. We show Algorithm 2 is $e/(e-1)$-competitive. Let T be a representation of an instance of the super-martingale stopping problem. Let $S = \{v_j^*\}_{j=1}^k$ be the set of stopping nodes of $\text{OPT}(T)$. Let $u \in S$ and let $P(u) \subseteq T$ be the path from the root to u. We notice that we can assume the depth of u is at most $Q^{-1}_{P(u)}(1)$. Since the cost of Algorithm 2 only depends on the value of nodes with depth strictly less than $Q^{-1}_{P(u)}(1)$, we can assume every node with a depth larger than $Q^{-1}_{P(u)}(1)$ has a value of 0. This assumption doesn't affect the cost of Algorithm 2 but will force every $u \in S$ has depth at most $\lceil Q^{-1}_{P(u)}(1)\rceil$. Under this assumption, if a node u has depth exactly $\lceil Q^{-1}_{P(u)}(1)\rceil$, we can furthermore assume the contribution of u to the cost of $\text{OPT}(T)$ is $Q^{-1}_{P(u)}(1)$. This will only decrease the cost of $\text{OPT}(T)$. So in the rest of the proof, every u in S has a depth at most $Q^{-1}_{P(u)}(1)$. In particular, this implies for every $u \in S$, there exists some $\rho_u \in [0,1]$ such that $Q_{P(u)}(\text{depth}(u)) = \rho_u$. Thus, we can write

$$\text{OPT}(T) = \sum_{u\in S}\Pr(u)\left(u + Q^{-1}_{P(u)}(\rho_u)\right).$$

On the other hand, we can decompose the cost of the algorithm according to S. For every $u \in S$, we define $D(u)$ to be the set of paths in T from the root to a leaf that passes u. Then we can write the cost of the algorithm

$$\text{ALG}(T) \leq \sum_{u \in S} \Pr(u) \sum_{p' \in D(u)} \Pr(p' \mid u) \int_0^1 \left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) p(\rho) d\rho,$$

where we use the fact that when the algorithm stops at time t, the depth of the stopping node is at most t. This implies

$$\begin{aligned}
\text{ALG}(T) - \text{OPT}(T) &\leq \sum_{u \in S} \Pr(u) \int_0^1 \sum_{p' \in D(u)} \Pr(p' \mid u) \left[\left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) - \left(u + Q_{P(u)}^{-1}(\rho_u) \right) \right] p(\rho) d\rho \\
&= \sum_{u \in S} \Pr(u) \int_0^{\rho_u} \sum_{p' \in D(u)} \Pr(p' \mid u) \left[\left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) - \left(u + Q_{P(u)}^{-1}(\rho_u) \right) \right] p(\rho) d\rho \\
&- \sum_{u \in S} \Pr(u) \int_{\rho_u}^1 \sum_{p' \in D(u)} \Pr(p' \mid u) \left[\left(u + Q_{P(u)}^{-1}(\rho_u) \right) - \left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) \right] p(\rho) d\rho \\
&\leq \sum_{u \in S} \Pr(u) \int_0^{\rho_u} \sum_{p' \in D(u)} \Pr(p' \mid u) \left[\left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) - \left(u + Q_{P(u)}^{-1}(\rho_u) \right) \right] p(\rho) d\rho \\
&+ \sum_{u \in S} \Pr(u) \int_{\rho_u}^1 \sum_{p' \in D(u)} \Pr(p' \mid u) \left(Q_{p'}^{-1}(\rho) - Q_{P(u)}^{-1}(\rho_u) \right) p(\rho) d\rho \\
&\leq \sum_{u \in S} \Pr(u) \int_0^{\rho_u} \sum_{p' \in D(u)} \Pr(p' \mid u) \left[\left(v_{p'}(Q_{p'}^{-1}(\rho)) + Q_{p'}^{-1}(\rho) \right) - \left(u + Q_{P(u)}^{-1}(\rho_u) \right) \right] p(\rho) d\rho \\
&+ \sum_{u \in S} \Pr(u) \int_{\rho_u}^1 (\rho - \rho_u) u p(\rho) d\rho \\
&= \sum_{u \in S} \Pr(u) \int_0^{\rho_u} \left(v_{P(u)}(Q_{P(u)}^{-1}(\rho)) + Q_{P(u)}^{-1}(\rho) - Q_{P(u)}^{-1}(\rho_u) \right) p(\rho) d\rho \\
&+ \sum_{u \in S} \Pr(u) \left(\int_{\rho_u}^1 (\rho - \rho_u) u p(\rho) d\rho - \int_0^{\rho_u} u p(\rho) d\rho \right).
\end{aligned}$$

Here, the second inequality follows the super-martingale property of the sequence of random variables starting from node u. The second inequality follows by Lemma 3.5.

Recall our goal is to show that $\text{ALG}(T) - \frac{e}{e-1}\text{OPT}(T) = \text{ALG}(T) - \text{OPT}(T) - \frac{1}{e-1}\text{OPT}(T) \leq 0$. For every $u \in S$, we define two functions $F_u(s)$ and $G_u(\tau)$ as follows. Let

$$F_u(s) := \int_0^s \left(v_{P(u)}(Q_{P(u)}^{-1}(\rho)) + Q_{P(u)}^{-1}(\rho) - Q_{P(u)}^{-1}(s) \right) p(\rho) d\rho - \frac{1}{e-1} Q_{P(u)}^{-1}(s)$$

and

$$G_u(\tau) := \int_{\rho_u}^1 (\rho - \rho_u) \tau p(\rho) d\rho - \int_0^{\rho_u} \tau p(\rho) d\rho - \frac{1}{e-1}\tau.$$

From our above discussion, we know that

$$\text{ALG}(T) - \frac{e}{e-1}\text{OPT}(T) = \text{ALG}(T) - \text{OPT}(T) - \frac{1}{e-1}\text{OPT}(T) \leq \sum_{u \in S} \Pr(u) \left(F_u(\rho_u) + G_u(u) \right).$$

It is sufficient to show for every u, $F_u(s) \leq 0, \forall s \in [0, \rho_u]$ and $G_u(\tau) \leq 0, \forall \tau \geq 0$. We first look at $F_u(s)$. Recall the definition of the density function is $p(\rho) = \frac{e^\rho}{e-1}$. We know from Lemma 3.4 that

$$\begin{aligned}
F_u(s) &= \int_0^s \left(v_{P(u)}(Q^{-1}_{P(u)}(\rho)) - \int_\rho^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw \right) p(\rho)d\rho - \frac{1}{e-1}\int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw \\
&= \int_0^r v_{P(u)}(Q^{-1}_{P(u)}(w)dw \frac{e^r}{e-1}\Big|_{r=0}^{r=s} - \int_0^s \int_0^\rho v_{P(u)}(Q^{-1}_{P(u)}(w)dwp(\rho)d\rho \\
&- \int_0^s \int_\rho^s v_{P(u)}(Q^{-1}_{P(u)}(w)dwp(\rho)d\rho - \frac{1}{e-1}\int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw \\
&= \frac{e^s}{e-1}\int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw - \int_0^s \int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dwp(\rho)d\rho - \frac{1}{e-1}\int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw \\
&= \left(\frac{e^s}{e-1} - \frac{e^s-1}{e-1} - \frac{1}{e-1}\right)\int_0^s v_{P(u)}(Q^{-1}_{P(u)}(w)dw = 0.
\end{aligned}$$

Then we look at $G_u(\tau)$. We have

$$\begin{aligned}
\frac{dG_u(\tau)}{d\tau} &= \int_{\rho_u}^1 (\rho - \rho_u)p(\rho)d\rho - \int_0^{\rho_u} p(\rho)d\rho - \frac{1}{e-1} \\
&= \frac{\rho e^\rho}{e-1}\Big|_{\rho=\rho_u}^{\rho=1} - \int_{\rho_u}^1 p(\rho)d\rho - \frac{\rho_u(e-e^{\rho_u})}{e-1} - \int_0^{\rho_u} p(\rho)d\rho - \frac{1}{e-1} \\
&= -\frac{\rho_u e}{e-1} \le 0.
\end{aligned}$$

This implies that $G_u(\tau) \le G_u(0) = 0$. Put the above arguments together, we obtain ALG$(T) \le \frac{e}{e-1}$OPT(T). □

E.4. Proof of Theorem 3.10

Proof. Let I be an instance of super-martingale stopping problem. Let OPT$(I) = \min \mathbf{E}\left(i^* + X_{i^*}\right)$ and OPT$' = \mathbf{E}\min_i\left(i + X_i\right)$. We will construct a sequence of instance I_N such that OPT$(I_N) \ge \Omega(N)$OPT$'(I_N)$, showing that the gap between the two benchmarks can be arbitrarily large.

Let $(X_0, X_1, \dots)$ be the sequence of random variables of instance I_N. Define $X_0 = N$. For every $i \ge 1$, X_i can take two possible values. Given X_i, $X_{i+1} = e^N X_i$ with probability e^{-N} and $X_{i+1} = 0$ with probability $1 - e^{-N}$. That is to say, the sequence of random variables is a super-martingale with a mean equal to N. Thus, the optimal stopping rule is to simply stop at X_0 and OPT$(I_N) = N$. On the other hand, consider any realization $(x_0, x_1, \dots)$ of the sequence. We notice from the construction that if $x_i = 0$ then for every $j > i$, $x_j = 0$. Denote by i' the smallest index such that $x_{i'} = 0$. Then we have $\min i + x_i = i'$ if $i' \le N$ and $\min i + x_i = N$ if $i' > N$. Since $\Pr(i' = i) = (1 - e^{-N})e^{-(i-1)N}$, we have

$$\text{OPT}'(I_N) = \sum_{i=1}^N i(1-e^{-N})e^{-(i-1)N} + \sum_{i=N+1}^\infty N(1-e^{-N})e^{-(i-1)N} \in O(1).$$

This implies OPT$(I_N) \ge \Omega(N)$OPT$'(I_N)$.

□

E.5. Proof of Theorem 3.11

Proof. It is sufficient to show that if we run Algorithm 1 or Algorithm 2 then ALG$(\tilde{T}) \le \alpha$ALG(T). Recall that the only difference between Algorithm 1 and Algorithm 2 is that they use different threshold ρ. Algorithm 1 uses $\rho = 1$ and Algorithm 2 uses a random threshold. Let $\rho \in [0, 1]$ be a realization of the random threshold used in Algorithm 1 and Algorithm 2. We denote by ALG$_\rho(\tilde{T})$ and ALG$_\rho(T)$ the cost of the Algorithm on the corresponding instances with a threshold ρ. In the rest of the proof, we will show ALG$_\rho(\tilde{T}) \le \alpha$ALG$_\rho(T)$ for every $\rho \in [0, 1]$. This will directly imply that ALG$(\tilde{T}) \le \alpha$ALG(T).

Since the only difference between T and $\tilde{T}$ is the value of each node, let $p = (v_0, v_1, \dots, v_n)$ be a path in T, we define

$\tilde{v}_p(t) = \tilde{v}_i$ if $t \in [i, i+1)$. We can also define $\tilde{Q}_p(t)$ and $\tilde{Q}_p^{-1}(t)$ in the similar way. Using these notations, we have

$$\mathrm{ALG}_\rho(T) = \mathbf{E}_p\left(Q_p^{-1}(\rho) + v_p(Q_p^{-1}(\rho))\right) = \int_0^\rho \mathbf{E}_p v_p(Q_p^{-1}(w))dw + \mathbf{E}_p v_p(Q_p^{-1}(\rho) \geq \rho\mathbf{E}_p v_p(Q_p^{-1}(\rho)) + \mathbf{E}_p v_p(Q_p^{-1}(\rho)).$$

Here, the second equality follows by Lemma 3.4 and the inequality follows by the super-martingale property of T.

On the other hand, for every path p, since for every $v \in T$, $\tilde{v} \geq v$, we know that $\tilde{Q}_p^{-1}(\rho) \geq Q_p^{-1}(\rho)$. If we denote by $t' = Q_p^{-1}(\rho)$, then this implies there exists some $\rho' \leq \rho$ such that $\tilde{Q}_p(t') = \rho'$. In particular, since $\tilde{v}_p(t) \leq \alpha v_p(t)$ for every t, it follows that $\rho' \geq \frac{1}{\alpha}\rho$. Thus, we can write

$$\begin{aligned}
\mathrm{ALG}_\rho(\tilde{T}) &= \mathbf{E}_p\left(Q_p^{-1}(\rho) + \tilde{Q}_p^{-1}(\rho) - \tilde{Q}_p^{-1}(\rho') + \tilde{v}_p(\tilde{Q}_p^{-1}(\rho))\right) \\
&= \mathbf{E}_p Q_p^{-1}(\rho) + \mathbf{E}_p \int_{\rho'}^{\rho} \tilde{v}_p(\tilde{Q}_p^{-1}(w))dw + \mathbf{E}_p\tilde{v}_p(\tilde{Q}_p^{-1}(\rho)) \\
&\leq \mathbf{E}_p Q_p^{-1}(\rho) + \alpha(\rho - \rho')\mathbf{E}_p v_p(Q_p^{-1}(\rho)) + \alpha\mathbf{E}_p v_p(Q_p^{-1}(\rho)) \\
&\leq \mathbf{E}_p Q_p^{-1}(\rho) + (\alpha - 1)\rho\mathbf{E}_p v_p(Q_p^{-1}(\rho)) + \alpha\mathbf{E}_p v_p(Q_p^{-1}(\rho)).
\end{aligned}$$

Here, the equality follows Lemma 3.4, the first inequality follows by the super-martingale property of the T and the second inequality follows by the fact that $\rho' \geq \frac{1}{\alpha}\rho$. Thus, we obtain

$$\frac{\mathrm{ALG}_\rho(\tilde{T})}{\mathrm{ALG}_\rho(T)} \leq \frac{\mathbf{E}_p Q_p^{-1}(\rho) + (\alpha-1)\rho\mathbf{E}_p v_p(Q_p^{-1}(\rho)) + \alpha\mathbf{E}_p v_p(Q_p^{-1}(\rho))}{\mathbf{E}_p Q_p^{-1}(\rho) + \ \mathbf{E}_p v_p(Q_p^{-1}(\rho))} \leq \max\{\alpha, 1 + (\alpha-1)\frac{\rho\mathbf{E}_p v_p(Q_p^{-1}(\rho))}{\mathbf{E}_p Q_p^{-1}(\rho)}\}.$$

Here we use the fact that if $a, b, c, d \geq 0$, then $\frac{a+b}{c+d} \leq \max\{\frac{a}{c}, \frac{b}{d}\}$. By Lemma 3.4 and the super-martingale property, we know that

$$\mathbf{E}_p Q_p^{-1}(\rho) = \int_0^\rho \mathbf{E}_p v_p(Q_p^{-1}(w))dw \geq \rho\mathbf{E}_p v_p(Q_p^{-1}(\rho)).$$

This implies

$$1 + (\alpha - 1)\frac{\rho\mathbf{E}_p v_p(Q_p^{-1}(\rho))}{\mathbf{E}_p Q_p^{-1}(\rho)} \leq 1 + \alpha - 1 = \alpha.$$

Thus, $\mathrm{ALG}_\rho(\tilde{T}) \leq \alpha\mathrm{ALG}_\rho(T)$ for every $\rho \in [0,1]$.

□

F. Missing Proof in Section 4

F.1. Proof of Theorem 4.3

As we mentioned in the main body of the paper, Theorem 4.3 not only holds for MSSC but also holds for a broader class of stochastic optimization problems. In this part, we give the general statement and the proof for Theorem 4.3. To begin with, we define a broad class of adaptive stochastic optimization problems for which Theorem 4.3 holds.

Definition F.1 (Adaptive Stochastic Optimization with Covering Loss)**.** Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be an adaptive stochastic optimization problem. For every $s \in \mathcal{S}$, we define a family of sets of actions $C(s) \subseteq 2^{\mathcal{X}}$. We say a scenario s is covered if a set of actions $A \in C(s)$ are taken. We say the loss function ℓ is a covering loss if for every scenario $s \in \mathcal{S}$ and every sequence of actions $\vec{a} = (a_1, a_2, \dots)$, $\ell(\vec{a}, s) = \min\{t \mid \exists A \in C(s), \text{s.t.} A \subseteq \{a_1, \dots, a_t\}\}$, which is the time for $\vec{a}$ to cover s.

Many adaptive stochastic optimization problems such as MSSC and optimal decision tree problems have covering objective functions. Next, we give a general statement of Theorem 4.3, which builds a connection between adaptive stochastic optimization with time dependent feedback and buying information for adaptive stochastic optimization.

Theorem F.2 (General Version of Theorem 4.3)**.** *Let $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ be an adaptive stochastic optimization problem with a covering loss function. If $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ satisfies an α-prophet inequality, then there is a 2α-competitive learner for the adaptive stochastic optimization problem with feedback $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$.*

Proof. Denote by $I = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$ the instance of buying information for stochastic optimization and OPT(I) be the optimal value of the instance. Since $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D})$ satisfies an α-prophet inequality, let $\mathcal{A}$ be an α-competitive learner for the stochastic optimization problem with feedback. At time round t, denote by R_t the set of outcomes received after taking a set of actions and denote by Y_t a set of signals received from the signaling schemes. Notice that R_t, Y_t are random sets that depend on the random scenarios s. Then $a = \mathcal{A}(R_t, Y_t)$ is the next action taken by the learner $\mathcal{A}$. Based on the notations, we design the following algorithm, which will be shown as 2α-competitive for $I = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$.

Algorithm 8 BUYINGINFORMATION (2α-competitive algorithm for $(\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}, \mathcal{C})$)

```
for $t = 0, 1, 2, \dots$ do
    Receive $R_{t-1}$ the set of outcomes received so far and $Y_{t-1}$ the signals received so far.
    Receive a cost $c_t$ to receive a signal from $f_{t+1}$.
    for $i = 1, \dots, c_t$ do
        Take action $a_i = \mathcal{A}(R_{t-1}, Y_{t-1})$ and receive outcome $r_i$.
        Update $R_{t-1} \leftarrow R_{t-1} \cup \{r_i\}$.
        if $s$ is covered then
            return
        end if
    end for
    Pay $c_t$ to get signal $y_{t+1}$ from $f_{t+1}$.
    Update $R_t \leftarrow R_{t-1}, Y_t \leftarrow Y_{t-1} \cup \{y_{t+1}\}$
end for
```

We notice that during the execution of Algorithm 8, we count the time round in a different way for convenience. This doesn't affect the final cost of the algorithm. We now decompose the cost of Algorithm 8 into two parts. Denote by $\mathcal{A}'$ Algorithm 8. For each scenario $s \in \mathcal{S}$, let **cost**(s) be the total cost of $\mathcal{A}'$ when s is drawn. We write

$$\mathbf{cost}(s) = \mathbf{cost}_f(s) + \mathbf{cost}_c(s),$$

where $\mathbf{cost}_f(s)$ is the feedback cost, the total cost $\mathcal{A}'$ spends on buying signals when s is drawn and $\mathbf{cost}_c(s)$ is the coverage cost, the number of actions taken by $\mathcal{A}'$ to cover s. That is to say

$$\begin{aligned}\mathbf{cost}(\mathcal{A}', I) = \mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}(s) &= \mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_f(s) + \mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_c(s) \\ &\leq 2\mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_c(s),\end{aligned}$$

since the feedback cost is always less than the coverage cost.

In the rest of the proof, we will construct an instance $\tilde{I} = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}')$ of stochastic optimization with time dependent feedback based on I such that $\mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_c(s) \leq \alpha\text{OPT}(\tilde{I})$ and $\text{OPT}(\tilde{I}) \leq \text{OPT}(I)$. We construct the feedback $\mathcal{F}'$ by constructing every possible sequence of signals received from $\mathcal{F}'$. Let $(y_0, y_1, \dots)$ be a sequence of signals received from the signaling schemes $(f_0, f_1, \dots)$. Let $c_t(y_t)$ be the cost to obtain signal y_{t+1} for the sequence. Then for every $t \geq 0$, we make $c_t(y_t)$ copies for y_t. Thus, the corresponding signals sequence in $\mathcal{F}'$ is $(y_0, \dots, y_0, y_1, \dots, y_1, \dots)$, where y_t appears $c_t(y_t)$ times.

Now we consider Algorithm 8. If we ignore the step where we pay c_t to get y_{t+1}, then the remaining algorithm is exactly running $\mathcal{A}$ over $\tilde{I} = (\mathcal{X}, \mathcal{S}, \ell, \mathcal{D}, \mathcal{F}')$. Since $\mathcal{A}$ is an α-competitive learner for $\tilde{I}$, we know that

$$\mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_c(s) = \mathbf{cost}(\mathcal{A}, \tilde{I}) \leq \alpha\text{OPT}(\tilde{I}).$$

It remains to show that $\text{OPT}(\tilde{I}) \leq \text{OPT}(I)$. Consider instance I. Assume s is the drawn scenario and the corresponding sequence of signals is $(y_0, y_1, \dots)$. Assume the sequence of actions taken in OPT(I) for this sequence of signals is $(a_0, a_1, \dots)$. We construct a sequence of actions taken for instance $\tilde{I}$ with sequence of signals $(y_0, \dots, y_0, y_1, \dots, y_1, \dots)$ by modifying $(a_0, a_1, \dots)$. Assume that in OPT(I), the learner pays a cost c to obtain the next signal after taking actions a_i. Then in the modified sequence, we take c arbitrary actions after a_i. For every drawn scenario s, the modified sequences can take less cost to cover s. This implies that $\text{OPT}(\tilde{I}) \leq \text{OPT}(I)$. Putting things together, we have

$$\mathbf{cost}(\mathcal{A}', I) \leq 2\mathbf{E}_{s\sim\mathcal{D}}\mathbf{cost}_c(s) \leq 2\alpha\text{OPT}(\tilde{I}) \leq 2\alpha\text{OPT}(I),$$

which means Algorithm 8 is 2α-competitive for adaptive stochastic optimization with feedback.

□

F.2. Proof of Theorem 4.4

Before presenting the proof, we define the model of MSSC with time dependent feedback as a remainder.

Definition F.3 (Min Sum Set Cover with Time Dependent Feedback)**.** Let $\mathcal{B} = [n]$ be a set of boxes, each box i contains an unknown number $b_i \in \{0, 1\}$ A learner can know b_i by querying box i, i.e. the action space $\mathcal{X} = \mathcal{B}$. A scenario $s \in \{0, 1\}^n$ is a binary vector that represents the number contained in each box. If scenario s is realized, then for every box $i \in \mathcal{B}$, $s_i = b_i$. A scenario s is covered if a box i such that $s_i = 1$ is queried. Let $\mathcal{S}$ be a set of scenarios and $\mathcal{D}$ be a probability distribution over $\mathcal{S}$. Let f be a sequence of feedback. A scenario s^* is drawn from $\mathcal{D}$ initially. In each round t, a learner receives signal $y_t(s^*)$ from signaling scheme f_t and takes an action $a_t \in \mathcal{X}$ to query the box a_t and observed the number contained in that box. Given an instance $(\mathcal{B}, \mathcal{S}, \mathcal{D}, \mathcal{F})$ of Min Sum Set Cover with Time Dependent Feedback, the goal of a learner is to construct the sequence of boxes A to query to minimize $\mathbf{E}_{s\sim\mathcal{D}}\ell(A, s)$, where $\ell(A, s)$ is the number of boxes to query to cover the drawn scenario s.

As a remainder, we restate Algorithm 3, the simple greedy algorithm that we want to analyze here.

Algorithm 9 GREEDY (4-competitive Learner for MSSC with Time Dependent Feedback)

```
for t=0,1,2,... do
    Receive scenario set $S_t$ that are consistent with the feedback and outcomes received so far.
    Compute $\Pr(i) := \sum_{s\in S_t:s_i=1} \Pr(s \mid S_t)$
    Query any box $i^* \in \arg\max\{\Pr(i) \mid i \in \mathcal{B}\}$.
    if $s^*_{i^*} = 1$ then
        return
    end if
end for
```

Proof. Without loss of generality, we can assume $\mathcal{D}$ is a uniform distribution over $\mathcal{S}$ and $\mathcal{F}$ contains only deterministic signaling schemes. This is because given a distribution $\mathcal{D}$, we can modify $\mathcal{S}$ by making multiple copies of each scenario and uniformly draw a scenario from the modified set of scenarios according to our discussion in Appendix A. Under this assumption, we will write a linear program to lower bound OPT(I). We say a scenario s is covered by a box i if $s_i = 1$. For every scenario, s, denote by L_s the set of boxes that cover s.

Fix a sequence of feedback $\mathcal{F}$, let $T(\mathcal{F})$ be the feedback tree induced by $\mathcal{F}$. Let $P(s)$ be the longest path in $T(\mathcal{F})$ such that s is contained in every node in $P(s)$. Any learner $\mathcal{A}$ will assign a box to each node in $T(\mathcal{F})$ such that for every scenario s, there is some node $v \in P(s)$ such that the box assigned to v by $\mathcal{A}$ covers s. We derive the following integer program to capture the cost of a learner. For every node $v \in T(\mathcal{F})$ and for every box $i \in \mathcal{B}$, let $x_{vi} \in \{0, 1\}$ be the indicator if $\mathcal{A}$ assigns box i to node v. For every node, $v \in T(\mathcal{F})$ and for every scenario $s \in v$, let $y_{vs} \in \{0, 1\}$ be the indicator if s is not covered by any box assigned to an ancestor of v. Here, we use the notation $v' < v$ to denote that v' is an ancestor of v. For every scenario s, let **cost**$(s) := \sum_{v\in P(s)} y_{vs}$, which is the time when s is first covered by an assigned box. Then, any learner $\mathcal{A}$ gives a feasible solution to the following integer program.

$$
\begin{aligned}
\min_{x,y}\ & \sum_{s\in\mathcal{S}} \mathbf{cost}(s) \\
\text{s.t.}\ & \sum_{i\in\mathcal{B}} x_{vi} \le 1\ \forall v \in T(\mathcal{F}) \\
& y_{vs} + \sum_{i\in L_s} \sum_{v':v'<v} x_{v'i} \ge 1\ \forall v \in T(\mathcal{F}), \forall s \in v \\
& x_{vi} \in \{0, 1\}\ \forall v \in T(\mathcal{F}), \forall i \in \mathcal{B} \\
& y_{vs} \in \{0, 1\}\ \forall v \in T(\mathcal{F}), \forall s \in \mathcal{S}.
\end{aligned}
\tag{IP}
$$

Here, the first set of constraints implies that for any node v, any learner can assign at most 1 box. The second set of constraints implies that for every node v and every $s \in v$, either s has not been covered so far or there is an ancestor v' of v that is assigned a box $i \in L_s$ by learner $\mathcal{A}$. In particular, since $\mathcal{D}$ is uniform over $\mathcal{S}$, $|\mathcal{S}|$**cost**$(\mathcal{A}, I) = \sum_{s\in\mathcal{S}}$ **cost**(s). Thus, the following linear programming relaxation gives a natural lower bound for $|S|$OPT(I).

$$\begin{aligned}
\min_{x,y} \ & \sum_{s\in\mathcal{S}} \textbf{cost}(s) \\
\text{s.t.} \ & \sum_{i\in\mathcal{B}} x_{vi} \le 1 \ \forall v \in T(\mathcal{F}) \\
& y_{vs} + \sum_{i\in L_s} \sum_{v':v'<v} x_{v'i} \ge 1 \ \forall v \in T(\mathcal{F}), \forall s \in v \\
& x_{vi} \ge 0 \ \forall v \in T(\mathcal{F}), \forall i \in \mathcal{B} \\
& y_{vs} \ge 0 \ \forall v \in T(\mathcal{F}), \forall s \in v.
\end{aligned} \tag{LP}$$

Let $\{A_v\}_{v\in T(\mathcal{F})}$ be the set of dual variables for the first set of constraints in (LP) and let $\{B_{vs}\}_{v\in T(\mathcal{F}),s\in v}$ be the set of dual variables for the second set of constraints in (LP). Then we derive the following dual linear program for (LP).

$$\begin{aligned}
\max_{A,B} \ & \sum_{v\in T(\mathcal{F})} \sum_{s\in v} B_{vs} - \sum_{v\in T(\mathcal{F})} A_v \\
\text{s.t.} \ & B_{vs} \le 1 \ \forall v \in T(\mathcal{F}) \\
& \sum_{s:s\in v, i\in L_s} \sum_{v':v<v'} B_{v's} \le A_v \ \forall v \in T(\mathcal{F}), \forall i \in \mathcal{B} \\
& B_{vs} \ge 0 \ \forall v \in T(\mathcal{F}), \forall s \in v \\
& A_v \ge 0 \ \forall v \in T(\mathcal{F}).
\end{aligned} \tag{DUAL}$$

Since (LP) is feasible and bounded, we know from linear programming dual theory that (DUAL) is feasible, furthermore, (DUAL) and (LP) have the same optimal value. Denote by D the optimal value of (DUAL), then we know that $|S|\text{OPT}(I) \ge D$.

Next, we will show that there is an optimal solution to (DUAL) that has a special structure. We have the following observations.

Observation 1. Let (A, B) be any feasible solution to (DUAL). For every $v \in T(\mathcal{F})$, let $A'_v = \max_i \sum_{s:s\in v,i\in L_s} \sum_{v':v<v'} B_{v's}$. For every $v \in T(\mathcal{F}), s \in v$, let $B'_{vs} = B_{vs}$. Then (A', B') is feasible to (DUAL), furthermore, (A', B') has a larger objective value than (A, B).

The proof of Observation 1 follows directly by the second set of constraints in (DUAL).

Observation 2. Let (A, B) be any feasible solution to (DUAL). For every $s \in \mathcal{S}$, let $C_s := \sum_{v\in P(s)} B_{vs}$. For every $s \in \mathcal{S}$ and for every $v \in P(s)$, define

$$B'_{vs} = \begin{cases} 1 \ \text{if depth}(v) < \lfloor C_s \rfloor, \\ C_s - \lfloor C_s \rfloor \ \text{if depth}(v) = \lfloor C_s \rfloor, \\ 0 \ \text{otherwise.} \end{cases}$$

For every $v \in T(\mathcal{F})$, define $A'_v = \max_i \sum_{s:s\in v,i\in L_s} \sum_{v':v<v'} B'_{v's}$. Then (A', B') is feasible to (DUAL) and (A', B') has a larger objective value than (A, B).

Proof of Observation. The feasibility of (A', B') follows by Observation 1. Thus, we only need to show (A', B') has a larger objective value. We notice that

$$\sum_{v\in T(\mathcal{F})} \sum_{s\in v} B_{vs} = \sum_{s\in\mathcal{S}} \sum_{v\in P(s)} B_{vs} = \sum_{s\in\mathcal{S}} C_s = \sum_{s\in\mathcal{S}} \sum_{v\in P(s)} B'_{vs} = \sum_{v\in T(\mathcal{F})} \sum_{s\in v} B'_{vs}.$$

It remains to show that $\sum_{v\in T(\mathcal{F})} A_v \ge \sum_{v\in T(\mathcal{F})} A'_v$. By Observation 1, we may assume $A_v = \max_i \sum_{s:s\in v,i\in L_s} \sum_{v':v<v'} B_{v's}$ for every $v \in T(\mathcal{F})$. It is sufficient to show for every v and every $s \in v$,

$\sum_{v'\in P(s):v<v'} B_{v's} \geq \sum_{v'\in P(s):v<v'} B'_{v's}$. We have

$$\sum_{v'\in P(s):v<v'} B_{v's} = C_s - \sum_{v'\in P(s):v'\leq v} B_{v's} \geq C_s - \sum_{v'\in P(s):v'\leq v} B_{v's} = \sum_{v'\in P(s):v<v'} B'_{v's}.$$

◇

Observation 1 and Observation 2 imply that an optimal solution to (DUAL) can be constructed in the following way. For each $s \in \mathcal{S}$, assign $C_s \geq 0$ to s. For every $s \in \mathcal{S}$ and for every $v \in P(s)$, define

$$B_{vs} = \begin{cases} 1 \text{ if depth}(v) < \lfloor C_s \rfloor, \\ C_s - \lfloor C_s \rfloor \text{ if depth}(v) = \lfloor C_s \rfloor, \\ 0 \text{ otherwise.} \end{cases}$$

For every $v \in T(\mathcal{F})$, define $A_v = \max_i \sum_{s:s\in v, i\in L_s} \sum_{v':v<v'} B_{v's}$. Let (A, B) be such a solution constructed in the way we discussed above using a vector $C \in R_+^{\mathcal{S}}$. Denote by $D(C)$ the objective value of (A, B). Then we have

$$\begin{aligned} D(C) &= \sum_{v\in T(\mathcal{F})} \sum_{s\in v} B_{vs} - \sum_{v\in T(\mathcal{F})} \max_{i\in\mathcal{B}} \sum_{s:s\in v, i\in L_s} \sum_{v':v<v'} B_{v's} \\ &= \sum_{s\in\mathcal{S}} C_s - \sum_{v\in T(\mathcal{F})} \sum_{i\in\mathcal{B}} \sum_{s:s\in v, i\in L_s} \sum_{v':v<v'} B_{v's}\mathbf{1}_{vi} \\ &= \sum_{s\in\mathcal{S}} C_s - \sum_{v\in T(\mathcal{F})} \sum_{i\in\mathcal{B}} \sum_{v':v<v'} \sum_{s:s\in v, i\in L_s} B_{v's}\mathbf{1}_{vi} \\ &= \sum_{s\in\mathcal{S}} C_s - \sum_{v\in T(\mathcal{F})} \sum_{v':v<v'} \sum_{i\in\mathcal{B}} \sum_{s:s\in v, i\in L_s} B_{v's}\mathbf{1}_{vi} \\ &= \sum_{s\in\mathcal{S}} C_s - \sum_{v'\in T(\mathcal{F})} \sum_{v:v<v'} \sum_{i\in\mathcal{B}} \sum_{s:s\in v, i\in L_s} B_{v's}\mathbf{1}_{vi}, \end{aligned}$$

where $\mathbf{1}_{vi}$ is the indicator function if $i \in \arg\max_j \sum_{s:s\in v, j\in L_s} \sum_{v':v<v'} B_{v's}$. For convenience, we assume there is only one box that achieves the max.

We interpret $D(C)$ via the following physical process. For each scenario $s \in \mathcal{S}$, we generate a particle $\mathbf{P}_s$. $\mathbf{P}_s$ moves along the path $P(s)$ with a rate of 1 and stops at time $t = C_s$. The length of an edge in $T(\mathcal{F})$ is 1. Let $V_s(t)$ be the speed of $\mathbf{P}_s$ at time t. That is to say $C_s = \int_0^\infty V_s(t)dt$ for every $s \in \mathcal{S}$. From this point of view, we can write the first term in $D(C)$ as

$$\sum_{s\in\mathcal{S}} C_s = \int_0^\infty \sum_{s\in\mathcal{S}} V_s(t)dt. \tag{3}$$

On the other hand, for each node $v \in T(\mathcal{F})$, there is a box $i(v)$ such that $\mathbf{1}_{vi(v)} = 1$. At a given time t, we will charge each **moving** particle $G_s(t) := |\{v \mid v \in P(s), \text{depth}(v) \leq t, i(v) \in L_s\}|$. In other words, for every moving particle, we will charge it the number of visited nodes v such that box $i(v)$ covers the corresponding scenario. Next, we build a connection between $G_s(t)$ and $D(C)$. For every $s \in \mathcal{S}$, write $P(s) = (v_0, v_1, \ldots, v_n)$. Then we have

$$G_s(t) = \sum_{i\in L_s} \sum_{v\in P(s):\text{depth}(v)\leq t} V_s(t)\mathbf{1}_{vi} = \sum_{i\in L_s} \sum_{j=0}^{\lfloor t \rfloor} V_s(t)\mathbf{1}_{v_j i},$$

which implies

$$\int_0^\infty G_s(t)dt = \int_0^\infty \sum_{i\in L_s} \sum_{j=0}^{\lfloor t \rfloor} V_s(t)\mathbf{1}_{v_j i}dt = \sum_{v'\in P(s)} \sum_{v:v\leq v'} \sum_{i\in L_s} B_{v's}\mathbf{1}_{vi},$$

according to the construction of B. Thus, we can write the second term in $D(C)$ as

$$\sum_{v' \in T(\mathcal{F})} \sum_{v: v < v'} \sum_{i \in \mathcal{B}} \sum_{s: s \in v, i \in L_s} B_{v's} \mathbf{1}_{vi} \leq \int_0^\infty \sum_{s \in \mathcal{S}} G_s(t) dt. \tag{4}$$

Combine (3) and (4), we get

$$D(C) \geq \int_0^\infty \sum_{s \in \mathcal{S}} V_s(t) - \sum_{s \in \mathcal{S}} G_s(t) dt. \tag{5}$$

In the rest of the proof, instead of constructing the optimal solution to (DUAL), we will construct a vector C_g based on Algorithm 3 such that $|S|\text{ALG}(I) \leq 4D(C^g) \leq 4D \leq 4|S|\text{OPT}(I)$, which implies that Algorithm 3 is 4-competitive.

Consider the implementation of Algorithm 3, the greedy algorithm. We notice that if we arrive at some node $v \in T(\mathcal{F})$, the set of scenarios S_t we received is exactly R_v, the set of scenarios $s \in v$ that has not been covered so far. Since $\mathcal{D}$ is uniform, the box $\mathcal{A}(v)$ queried by the algorithm at node v is the box that can cover most scenarios in R_v. Denote by $X_v = \{s \in R_v \mid \mathcal{A}(v) \in L_s\}$, which is the scenarios in R_v covered by the box that Algorithm 3 queries in this round. Now we define C_s for each scenario s. Let $P = (v_0, v_1, \ldots, v_n)$ be a path of $T(\mathcal{F})$ from the root to some leaf. For each $v_i \in P$, define $C_{v_i} = \max\{C_{v_{i-1}}, \frac{|R_{v_i}|}{c|X_{v_i}|}\}$, where $c > 0$ is a constant that we will determine later and $C_{v_0} = \frac{|R_{v_0}|}{c|X_{v_0}|}$. Notice that $\{X_v\}_{v \in T(\mathcal{F})}$ forms a partition of $\mathcal{S}$, so each s belongs to a unique X_v. For every node $v \in T(\mathcal{F})$ and for every $s \in v$, we set $C_s = C_v$. Denote by C^g the vector we just constructed. We next show that $|S|\text{ALG}(I) \leq 4D(C^g)$. Notice that

$$|\mathcal{S}|\text{ALG}(I) = \sum_{v \in T(\mathcal{F})} (\text{depth}(v) + 1)|X_v| = \sum_{v \in T(\mathcal{F})} |R_v|.$$

Based on this observation, we first derive the following lower bound for $\int_0^\infty \sum_{s \in \mathcal{S}} V_s(t) dt$. We have

$$\int_0^\infty \sum_{s \in \mathcal{S}} V_s(t) dt = \sum_{v \in T(\mathcal{F})} \sum_{s \in X_v} \int_0^\infty V_s(s) dt = \sum_{v \in T(\mathcal{F})} \sum_{s \in X_v} C_s \geq \frac{1}{c} \sum_{v \in T(\mathcal{F})} |R_v| = \frac{1}{c} |\mathcal{S}|\text{ALG}(I).$$

Next, we will show that $\int_0^\infty \sum_{s \in \mathcal{S}} G_s(t) dt \leq \frac{1}{c} \int_0^\infty \sum_{s \in \mathcal{S}} V_s(t) dt$. To do this, we upper bound $\sum_s G_s(t)$ for every $t \geq 0$. For every $s \in \mathcal{S}$, let $P^t(s) = (v_0(s), v_1(s), \ldots, v_{\lceil t \rceil}(s))$ be the truncation of path $P(s)$ with length of $\lceil t \rceil$. We know that for every t, P^t, the set of such truncated paths, forms a partition of $\mathcal{S}$. So we can write $\sum_{s \in \mathcal{S}} G_s(t) = \sum_{P \in P^t} \sum_{s \in P} G_s(t)$.

Let $P = (v_0, \ldots, v_{\lceil t \rceil}) \in P^t$ be such a truncated path. The set of particles that are moving along P corresponds to scenarios in $v_{\lceil t \rceil}$ with $C_s \geq t$. We observe that along the path P, C_{v_i} is a step function with respect to the index i. Based on the definition of C_s, for every v and every $s \in R_v$, we have $C_s \geq C_v$. This implies that along the path P, there must be some $i^* \leq \lfloor t \rfloor$ such that the set of particles that are moving along P at time t corresponds to scenarios exactly in $R_{v_{i^*}} \cap v_{\lceil t \rceil}$. In particular, if we consider the set $P^t(i^*)$ of all paths in P^t that passes v_{i^*}, then at time t, the set of particles moving along these paths is exactly $R_{v_{i^*}}$.

By the greedy property of Algorithm 3, every box can cover at most $|X_{v_{i^*}}|$ scenarios from $R_{v_{i^*}}$. Since each path $P \in P^t(i^*)$ contains at most t nodes and each node is charged by at most $|X_{v_{i^*}}|$ moving particles at time t, we have

$$\sum_{P \in P^t(i^*)} \sum_{s \in P} G_s(t) \leq t|X_{v_{i^*}}| \leq \frac{|R_{v_{i^*}}|}{c|X_{v_{i^*}}|} |X_{v_{i^*}}| = \frac{1}{c} |R_{v_{i^*}}| = \frac{1}{c} \sum_{P \in P^t(i^*)} \sum_{s \in P} V_s(t).$$

Here, the second inequality follows by $C_{v_{i^*}} = \frac{|R_{v_{i^*}}|}{c|X_{v_{i^*}}|} \geq t$. The last equality holds because $\sum_{P \in P^t(i^*)} \sum_{s \in P} V_s(t)$ is the number of moving particles along paths in $P^t(i^*)$, which is $|R_{v_{i^*}}|$. Thus we have

$$\int_0^\infty \sum_{s \in \mathcal{S}} G_s(t) dt \leq \frac{1}{c} \int_0^\infty \sum_{s \in \mathcal{S}} V_s(t) dt.$$

Put the above discussions together, we have

$$D(C^g) \geq \int_0^\infty \sum_{s\in\mathcal{S}} V_s(t) - \sum_{s\in\mathcal{S}} G_s(t)dt \geq (1-\frac{1}{c})\int_0^\infty \sum_{s\in\mathcal{S}} V_s(t)dt \geq \frac{1}{c}(1-\frac{1}{c})|\mathcal{S}|\text{ALG}(I) = \frac{1}{4}|\mathcal{S}|\text{ALG}(I),$$

by setting $c = 2$ to maximize the ratio. This shows Algorithm 3 is 4-competitive.

□

F.3. Proof of Theorem 4.5

Proof. We consider the following instance of min sum set cover with time dependent feedback. Let $\mathcal{B}$ be the set of n boxes. The set of scenarios $\mathcal{S} = \{s^i\}_{i=1}^n$, where $s^i_j = 1$ if $i = j$ and 0 otherwise. $\mathcal{D}$ is a uniform distribution over $\mathcal{S}$. Let $\mathcal{A}$ be any deterministic learner. We design a set of feedback $\mathcal{F}_\mathcal{A}$ such that **cost**$(\mathcal{A}, I) = \frac{n}{2} - o(1)$, while there is a learner $\mathcal{A}'$ such that **cost**$(\mathcal{A}', I) = \frac{n}{4} + o(1)$. Here, $I = (\mathcal{B}, \mathcal{S}, \mathcal{D}, \mathcal{F}_\mathcal{A})$ and **cost**$(\mathcal{A}, I)$ is the cost of a learner $\mathcal{A}$ over instance I.

We describe $\mathcal{F}_A$ via its feedback tree representation $T(\mathcal{F}_\mathcal{A})$. We first fix the structure of $T(\mathcal{F}_\mathcal{A})$, then define the scenario contained in each node of $T(\mathcal{F}_\mathcal{A})$. Let $T(\mathcal{F}_\mathcal{A})$ be a binary tree. Let v be a node in $T(\mathcal{F}_\mathcal{A})$. We denote by $L(v)$ its left child and $R(v)$ its right child. Let $\{v_i\}_{i=1}^n$ be a path of $T(\mathcal{F}_\mathcal{A})$ such that $v_{i+1} = R(v_i)$ and v_1 be the root of $T(\mathcal{F}_\mathcal{A})$. We define the set of scenarios contained in each node in $T(\mathcal{F})$. We know that $v_1 = \mathcal{S}$. Let $\mathcal{A}_i = \mathcal{A}(v_i)$ be the box queried by $\mathcal{A}$ at node v_i. We define $L(v_i) = \{s_{\mathcal{A}_i}\}$ and $R(v_i) = v_i \setminus L(v_i)$. This gives the definition of $\mathcal{F}_\mathcal{A}$. Intuitively, every time $\mathcal{A}$ queries a box, $\mathcal{F}$ only tells $\mathcal{A}$ if it queries the unique box that contains 1. This is to say $\mathcal{F}$ is useless for $\mathcal{A}$ and the cost of $\mathcal{A}$ is

$$\textbf{cost}(\mathcal{A}, I) = \frac{1}{n}\sum_{j=1}^n j = \frac{n-1}{2}.$$

On the other hand, let $\mathcal{A}'$ be the following learner. Let $\mathcal{A}'(v_i) = \mathcal{A}_{n+1-i}$, for $i \in [n]$ and $\mathcal{A}'(L(v_i)) = \mathcal{A}_i$. That is, along the path $\{v_i\}_{i=1}^n$, the order of the queried box by $\mathcal{A}'$ is the inverse of that of $\mathcal{A}$ and at every node $L(v_i)$, $\mathcal{A}'$ queries the box corresponding to the unique scenario contained in $L(v_i)$. This implies

$$\textbf{cost}(\mathcal{A}', I) = \frac{1}{n} + \sum_{j=2}^{\frac{n-1}{2}} \frac{2j}{n} = \frac{n^2-5}{4n}.$$

Thus, we have $\frac{\textbf{cost}(\mathcal{A},I)}{\textbf{cost}(\mathcal{A}',I)} \to 2$, which implies no deterministic learner is $2-\epsilon$-competitive.

□

F.4. Proof of Theorem 4.7

Before presenting the proof, we remind the definition of buying information for MSSC.

Definition F.4 (Buying Information for Min Sum Set Cover)**.** Let $(\mathcal{B}, \mathcal{S}, \mathcal{D})$ be an instance of Min Sum Set Cover, $\mathcal{F} = \{f_t\}_{t=0}^\infty$ be a sequence of feedback and $\mathcal{C} = \{c_t\}_{t=0}^\infty$ be a sequence of cost for receiving a signal from f_{t+1} from $\mathcal{F}$. Initially, a scenario s is drawn from $\mathcal{D}$. In each time round t, before s is covered, a learner adaptively receives an arbitrary number of signals from the sequence $\mathcal{F}$ by paying the corresponding cost and then selects a box to query. An instance $(\mathcal{B}, \mathcal{S}, \mathcal{D}, \mathcal{F}, \mathcal{C})$ of Buying Information for Min Sum Set Cover is to make decisions adaptively to minimize the expected number of the queried box plus the expected cost paid for the feedback to cover the random scenario.

Proof. We consider the following instance of buying information for min sum set cover. Let $\mathcal{B}$ be the set of n boxes. The set of scenarios $\mathcal{S} = \{s^i\}_{i=1}^n$, where $s^i_j = 1$ if $i = j$ and 0 otherwise. $\mathcal{D}$ is a uniform distribution over $\mathcal{S}$. We assume the cost of obtaining any single feedback is 1. Let $\mathcal{A}$ be any deterministic learner. We design a set of feedback $\mathcal{F}_A$ for $\mathcal{A}$.

We describe $\mathcal{F}_A$ via its feedback tree representation $T(\mathcal{F}_\mathcal{A})$. To do this, we will first fix the structure of $T(\mathcal{F}_\mathcal{A})$, then describe the scenarios contained in each node. The structure of $T(\mathcal{F}_\mathcal{A})$ is defined in the following way. There are $n_i + 1$ nodes in $T(\mathcal{F}_\mathcal{A})$ that have depth of i. Here $n_0 = 0$ and for $i \geq 1$, $n_i \geq 0$ is a number that depends on $\mathcal{A}$. Furthermore, for each level of $T(\mathcal{F}_\mathcal{A})$, only the rightmost node has children. In particular, for $i \geq 1$, let $v_{i+1} = R(v_i)$ be the right most child of v_i, where v_1 is the root of $T(\mathcal{F}_\mathcal{A})$.

We notice that given the structure of $T(\mathcal{F}_{\mathcal{A}})$, any deterministic learner $\mathcal{A}$ can be described in the following way using $T(\mathcal{F}_{\mathcal{A}})$. For every node $v \in T(\mathcal{F}_{\mathcal{A}})$, $\mathcal{A}$ will query a set of boxes $\mathcal{B}^v$ in some order, where $|\mathcal{B}^v| \geq 0$. Denote by $\mathcal{B}^i$, the set of boxes queried by $\mathcal{A}$ at node v_i. Let $n_i = |\mathcal{B}^i| \geq 0$, then set of scenarios that contained in $R(v_i)$ is defined by $\{s^j \in v_i \mid j \notin \mathcal{B}^i\}$. Recall that there are $n_i + 1$ nodes in $T(\mathcal{F}_{\mathcal{A}})$ that have depth i and we have defined the set of scenarios contained in one of these nodes. For the rest of n_i nodes, we assign a unique scenario covered by $\mathcal{B}^i$ to each of them. This gives the definition of $\mathcal{F}_{\mathcal{A}}$. In particular, $\mathcal{F}_{\mathcal{A}}$ is useless for $\mathcal{A}$, since every time $\mathcal{A}$ asks for feedback, the feedback only tells $\mathcal{A}$ which scenarios are not covered so far.

Now we compute the cost of $\mathcal{A}$. Consider the path $(v_1, v_2, \ldots, v_k)$ in $T(\mathcal{F}_{\mathcal{A}})$, such that $\sum_{i=1}^k |\mathcal{B}^i| = n$. That is to say, all scenarios are covered before the kth feedback is asked. Notice that $(\mathcal{B}^1, \ldots, \mathcal{B}^k)$ forms a partition of $\mathcal{S}$. Let $s \in \mathcal{B}^i$ be the jth scenario in $\mathcal{B}^i$ covered by $\mathcal{A}$, then the cost of $\mathcal{A}$ when scenario s is drawn is

$$\textbf{cost}(\mathcal{A}, s) = \sum_{\ell=1}^{i-1} n_\ell + i - 1 + j,$$

which implies

$$\begin{aligned}\textbf{cost}(\mathcal{A}, I) = \mathbf{E}_s \textbf{cost}(\mathcal{A}, s) &= \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(\sum_{\ell=1}^{i-1} n_\ell + i - 1 + j \right) \\ &= \frac{1}{n} \left(\sum_{i=1}^{n} i + \sum_{i=1}^{k} \sum_{j=1}^{n_i} i - n \right).\end{aligned}$$

We consider the two different cases. In the first case, $\sum_{i=1}^n i \leq \sum_{i=1}^k \sum_{j=1}^{n_i} i$. We notice that any deterministic learner $\mathcal{A}^*$ that asks for no feedback has a cost $\mathbf{E}_s \textbf{cost}(\mathcal{A}^*, s) = \frac{1}{n} \sum_{i=1}^n i$. This means

$$\frac{\textbf{cost}(\mathcal{A}, I)}{\textbf{cost}(\mathcal{A}^*, I)} \geq \frac{\frac{2}{n} \sum_{i=1}^n i - 1}{\frac{1}{n} \sum_{i=1}^n i} = 2 - o_n(1).$$

In the second case, we assume $\sum_{i=1}^n i > \sum_{i=1}^k \sum_{j=1}^{n_i} i$. In this case, we define a deterministic learner $\mathcal{A}^*$ in the following way. $\mathcal{A}^*$ keeps asking for feedback until the feedback reveals the drawn scenario, then $\mathcal{A}^*$ covers the drawn scenario via the unique box. It is not hard to see, any scenario in $\mathcal{B}^i$ will cost $\mathcal{A}^*$, $i + 1$. Thus, $\mathbf{E}_s \textbf{cost}(\mathcal{A}^*, s) = 1 + \frac{1}{n} \sum_{i=1}^k \sum_{j=1}^{n_i} i$. In this case, we have

$$\frac{\textbf{cost}(\mathcal{A}, I)}{\textbf{cost}(\mathcal{A}^*, I)} \geq \frac{\frac{1}{n} \left(\sum_{i=1}^n i + \sum_{i=1}^k \sum_{j=1}^{n_i} i \right) - 1}{1 + \frac{1}{n} \sum_{i=1}^k \sum_{j=1}^{n_i} i} = \frac{\frac{1}{n} \left(\sum_{i=1}^n i + \sum_{i=1}^k \sum_{j=1}^{n_i} i \right)}{\frac{1}{n} \sum_{i=1}^k \sum_{j=1}^{n_i} i} - o_n(1) \geq 2 - o_n(1).$$

Thus, for every $\epsilon > 0$, there is no deterministic learner that is $2 - \epsilon$ competitive.

$\square$