The Price of Information in Combinatorial Optimization[*]

Sahil Singla [†]

July 9, 2017

Abstract

Consider a network design application where we wish to lay down a minimum-cost spanning tree in a given graph; however, we only have stochastic information about the edge costs. To learn the precise cost of any edge, we have to conduct a study that incurs a price. Our goal is to find a spanning tree while minimizing the *disutility*, which is the sum of the tree cost and the total price that we spend on the studies. In a different application, each edge gives a stochastic *reward value*. Our goal is to find a spanning tree while maximizing the *utility*, which is the tree reward *minus* the prices that we pay.

Situations such as the above two often arise in practice where we wish to find a good solution to an optimization problem, but we start with only some partial knowledge about the parameters of the problem. The missing information can be found only after paying a *probing* price, which we call the *price of information*. What strategy should we adopt to optimize our expected utility/disutility?

A classical example of the above setting is Weitzman's "Pandora's box" problem where we are given probability distributions on values of n independent random variables. The goal is to choose a *single* variable with a large value, but we can find the actual outcomes only after paying a price. Our work is a generalization of this model to other combinatorial optimization problems such as matching, set cover, facility location, and prize-collecting Steiner tree. We give a technique that reduces such problems to their non-price counterparts, and use it to design exact/approximation algorithms to optimize our utility/disutility. Our techniques extend to situations where there are additional constraints on what parameters can be probed or when we can simultaneously probe a subset of the parameters.

[*]Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing. We are grateful to Anupam Gupta and Viswanath Nagarajan for several discussions on this project.

[†]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Email: `ssingla@cmu.edu`. Research supported in part by a CMU Presidential Fellowship and NSF awards CCF-1319811, CCF-1536002, CCF-1540541, and CCF-1617790.

1 Introduction

Suppose we want to purchase a house. We have some idea about the value of every available house in the market, say based on its location, size, and photographs. However, to find the exact value of a house we have to hire a house inspector and pay her a price. Our *utility* is the difference in the value of the best house that we find and the total inspection prices that we pay. We want to design a strategy to maximize our utility.

The above problem can be modeled as Weitzman's "Pandora's box" problem [Wei79]. Given probability distributions of n independent random variables X_i and given their probing prices π_i, the problem is to adaptively probe a subset $\mathsf{Probed} \subseteq [n]$ to maximize the expected utility:

$$\mathbb{E}\left[\max_{i \in \mathsf{Probed}} \{X_i\} - \sum_{i \in \mathsf{Probed}} \pi_i\right].$$

Weitzman gave an optimal adaptive strategy that maximizes the expected utility (naïve greedy algorithms can behave arbitrarily bad: see Section A.1). However, suppose instead of probing values of elements, we probe weights of edges in a graph. Our utility is the maximum-weight matching that we find *minus* the total probing prices that we pay. What strategy should we adopt to maximize our expected utility?

In a different scenario, consider a network design *minimization* problem. Suppose we wish to lay down a minimum-cost spanning tree in a given graph; however, we only have stochastic information about the edge costs. To find the precise cost X_i of any edge, we have to conduct a study that incurs a price π_i. Our *disutility* is the sum of the tree cost and the total price that we spend on the studies. We want to design a strategy to minimize our expected disutility. An important difference between these two scenarios is that of maximizing utility vs minimizing disutility.

Situations like the above often arise where we wish to find a "good" solution to an optimization problem; however, we start with only some partial knowledge about the parameters of the problem. The missing information can be found only after paying a *probing* price, which we call the *price of information*. What strategy should we adopt to optimize our expected utility/disutility? In this work we design optimal/approximation algorithms for several combinatorial optimization problems in an uncertain environment where we jointly optimize the value of the solution and the price of information.

1.1 Utility/Disutility Optimization

To begin, the above maximum-weight matching problem can be formally modeled as follows.

Max-Weight Matching Given a graph G with edges E, suppose each edge $i \in E$ takes some random weight X_i independently from a known probability distribution. We can find the exact outcome X_i only after paying a *probing price* π_i. The goal is to adaptively probe a set of edges $\mathsf{Probed} \subseteq E$ and select a matching $\mathbb{I} \subseteq \mathsf{Probed}$ to maximize the expected *utility*,

$$\mathbb{E}\left[\sum_{i \in \mathbb{I}} X_i - \sum_{i \in \mathsf{Probed}} \pi_i\right],$$

where the expectation is over random variables $\mathbf{X} = (X_1, \ldots, X_n)$ and any internal randomness of the algorithm. We observe that we can only select an edge if it has been probed and we might select only a subset of the probed edges. This matching problem can be used to model kidney exchanges where testing compatibility of donor-receiver pairs has an associated price.

To capture value functions of more general combinatorial problems in a single framework, we define the notion of *semiadditive* functions.

Definition 1.1 (Semiadditive function)**.** *We say a function $f(\mathbb{I}, \mathbf{X}) : 2^V \times \mathbb{R}_{\geq 0}^{|V|} \to \mathbb{R}_{\geq 0}$ is* semiadditive *if there exists a function $h : 2^V \to \mathbb{R}_{\geq 0}$ such that*

$$f(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i + h(\mathbb{I}).$$

For example, in the case of max-weight matching our value function $f(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i$ is *additive*, i.e. $h(\mathbb{I}) = 0$. We call these functions semiadditive because the second term $h(\mathbb{I})$ is allowed to effect the function in a "non-additive" way; however, not depending on $\mathbf{X}$. Here are some other examples.

- *Uncapacitated Facility Location*: Given a graph $G = (V, E)$ with metric (V, d), $\textsc{Clients} \subseteq V$, and facility opening costs $\mathbf{X} : V \to \mathbb{R}_{\geq 0}$, we wish to open facilities at some locations $\mathbb{I} \subseteq V$. The function is the sum of facility opening costs and the connection costs to $\textsc{Clients}$. Hence,

$$f(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i + \sum_{j \in \textsc{Clients}} \min_{i \in \mathbb{I}} d(j, i). \tag{1}$$

  Here $h(\mathbb{I}) = \sum_{j \in \textsc{Clients}} \min_{i \in \mathbb{I}} d(j, i)$ only depends on $\mathbb{I}$, and not on facility opening costs $\mathbf{X}$.

- *Prize-Collecting Steiner Tree*: Given a graph $G = (V, E)$ with some edge costs $\mathbf{c} : E \to \mathbb{R}_{\geq 0}$, a root node $r \in V$, and *penalties* $\mathbf{X} : V \to \mathbb{R}_{\geq 0}$. The goal is to find a tree that connects a subset of nodes to r, while trying to minimize the cost of the tree and the sum of the penalties of nodes $\mathbb{I}$ not connected to r. Hence,

$$f(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i + \text{Min-Steiner-Tree}(V \setminus \mathbb{I}),$$

  where Min-Steiner-Tree$(V \setminus \mathbb{I})$ denotes the minimum cost tree connecting all nodes in $V \setminus \mathbb{I}$ to r.

We can now describe an abstract utility-maximization model that captures problems such as Pandora's box, max-weight matching, and max-spanning tree, in a single unifying framework,

Utility-Maximization Suppose we are given a downward-closed (packing)[1] constraint $\mathcal{F} \subseteq 2^V$ and a semiadditive function val. Each element $i \in V$ takes a value X_i independently from a known probability distribution. To find the outcome X_i we have to pay a known probing price π_i. The goal is to adaptively probe a set of elements $\mathsf{Probed} \subseteq V$ and select $\mathbb{I} \subseteq \mathsf{Probed}$ that is feasible (i.e., $\mathbb{I} \in \mathcal{F}$) to maximize the expected *utility*,

$$\mathbb{E}\left[\mathsf{val}(\mathbb{I}, \mathbf{X}) - \sum_{i \in \mathsf{Probed}} \pi_i\right],$$

where the expectation is over random variables $\mathbf{X}$ and any internal randomness of the algorithm.

For example, in the max-weight matching problem val is an additive function and a subset of edges $\mathbb{I}$ is feasible if they form a matching. Similarly, when val is additive and $\mathcal{F}$ is a matroid, this framework captures max-weight matroid rank function, which contains Pandora's box and max-spanning tree as special cases.

The following is our main result for the utility-maximization problem:

[1]An independence family $\mathcal{F} \subseteq 2^V$ is called *downward-closed* if $A \in \mathcal{F}$ implies $B \in \mathcal{F}$ for any $B \subseteq A$. A set-system is called *upward-closed* if its complement is downward-closed.

Theorem 1.2. *For the utility-maximization problem for additive value functions and various packing constraints $\mathcal{F}$, we obtain the following efficient algorithms.*

- k-system[2]*: For stochastic element values, we get a k-approximation.*
- Knapsack*: For stochastic item values and known item sizes, we get a 2-approximation.*

Some important corollaries of Theorem 1.2 are an optimal algorithm for the max-weight matroid rank problem[3] and a 2-approximation algorithm for the max-weight matching problem. Theorem 1.2 is particularly interesting because it gives approximation results for mixed-sign objectives, which are usually difficult to handle.

We also show that if val is allowed to be any monotone submodular function then one *cannot* obtain good approximation results: there is an $\tilde{\Omega}(\sqrt{n})$ hardness even in a deterministic setting (see Section A.3).

Next, we describe a disutility-minimization model that captures problems like the *min*-cost spanning tree.

Disutility-Minimization Suppose we are given an upward-closed (covering) constraints $\mathcal{F}' \subseteq 2^V$ and a semiadditive function cost. Each element $i \in V$ takes a value X_i independently from a known probability distribution. To find the outcome X_i we have to pay a known probing price π_i. The goal is to adaptively probe a set of elements $\mathsf{Probed} \subseteq V$ and select $\mathbb{I} \subseteq \mathsf{Probed}$ that is feasible (i.e., $\mathbb{I} \in \mathcal{F}'$) to minimize the expected *disutility*,

$$\mathbb{E}\left[\mathsf{cost}(\mathbb{I}, \mathbf{X}) + \sum_{i \in \mathsf{Probed}} \pi_i\right],$$

where the expectation is over random variables $\mathbf{X}$ and any internal randomness of the algorithm.

For example, in the min-cost spanning tree problem, cost is an additive function and a subset of edges $\mathbb{I}$ are in $\mathcal{F}'$ if they contain a spanning tree. Similarly, when val is the semiadditive facility location function as defined in Eq. (1) and every non-empty subset of V is feasible in $\mathcal{F}'$, this captures the min-cost facility location problem.

Remark: The disutility-minimization problem can be also modeled as a utility-maximization problem by allowing item values to be negative and working with the infeasibility constraints (if $A \in \mathcal{F}'$ then $V \setminus A \in \mathcal{F}$), but such a transformation is not approximation factor preserving.

We now mention our results in this model. (See Section 4 for formal descriptions of these problems).

Theorem 1.3. *For the disutility-minimization problem for various covering constraints $\mathcal{F}'$, we obtain the following efficient algorithms.*

- Matroid Basis*: For stochastic element costs, we get the optimal adaptive algorithm.*
- Set Cover*: For stochastic costs of the sets, we get a $\min\{O(\log|V|), f\}$-approximation, where V is the universe and f is the maximum number of sets in which an element can occur.*
- Uncapacitated Facility Location*: For stochastic facility opening costs in a given metric, we get a 1.861-approximation.*

[2]An independence family $\mathcal{F} \subseteq 2^V$ is a *k-system* if for any $Y \subseteq V$ we have $\frac{\max_{A \in \mathcal{B}(Y)}(|A|)}{\min_{A \in \mathcal{B}(Y)}(|A|)} \leq k$, where $\mathcal{B}(Y)$ denotes the set of maximal independent sets of $\mathcal{F}$ included in Y [CCPV11]. These are more general than intersection of k matroids: e.g., a 2-system captures matching in general graphs and a k-system captures matching in a hypergraph with edges of size at most k.

[3]For weighted matroid rank functions, Kleinberg et al. [KWW16] independently obtain a similar result.

- Prize-Collecting Steiner Tree*: For stochastic penalties in a given graph with given edge costs, we get a* 3*-approximation.*
- Feedback Vertex Set*: For stochastic vertex costs in a given graph we get an* $O(\log n)$*-approximation.*

1.2 Constrained Utility-Maximization

Our techniques can extend to settings where we impose restrictions on the set of elements that we can probe. In particular, we are given a downward-closed set system $\mathcal{J}$ and the constraints allow us to only probe a subset of elements $\mathsf{Probed} \in \mathcal{J}$. This is different from the model discussed in Section 1.1 as earlier we could probe any set of elements but could get value for only a subset elements that belong to $\mathcal{F}$. As an example, consider a generalization of the Pandora's box problem where besides paying probing prices, we can only probe at most k elements. We now formally define our problem.

Constrained Utility-Maximization Suppose we are given downward-closed probing constraints $\mathcal{J} \subseteq 2^V$ and probability distributions of independent non-negative variables X_i for $i \in V$. To find X_i we have to pay a probing price π_i. The goal is to probe a set of elements $\mathsf{Probed} \in \mathcal{J}$ to maximize the expected *utility*,

$$\mathbb{E}\left[\max_{i \in \mathsf{Probed}}\{X_i\} - \sum_{i \in \mathsf{Probed}} \pi_i\right].$$

Remark: One can define an even more general version of this problem where we simultaneously have both downward-closed set systems $\mathcal{F}$ and $\mathcal{J}$, and the goal is to maximize a semiadditive function val corresponding to $\mathcal{F}$, while probing a set feasible in $\mathcal{J}$. For ease of exposition, we do not discuss it here and consider our value function to be the $\max$ function, as in the original Pandora's box problem.

Depending on the family of constraints $\mathcal{J}$, we design efficient approximation algorithms for some settings of the above problem. The following is our main result for this problem (proof in Section 5.1).

Theorem 1.4. *If the constraints* $\mathcal{J}$ *form an* ℓ*-system then the constrained utility-maximization problem has a* $3(\ell+1)$*-approximation algorithm.*

Since the cardinality (or any matroid) constraint forms a 1-system, an application of Theorem 1.4 gives a 6-approximation algorithm for the Pandora's box problem under a cardinality probing constraint.

The above constrained utility-maximization problem is powerful and can be used as a framework to study variants of Pandora's box. For example, consider the Pandora's box problem where we also allow to select an *unprobed* box i and get value $\mathbb{E}[X_i]$, without even paying its probing price π_i. This can be modeled using a partition matroid constraint where each box has two copies and the constraints allow us to probe at most one of them. The first copy has a deterministic value $\mathbb{E}[X_i]$ with zero probing price and the second copy has a random value X_i with price π_i. Using Theorem 1.4, we get a 6-approximation for this variant.

As a non-trivial application of this constrained utility-maximization framework, in Section 5.3 we discuss a *set-probing utility-maximization* problem where the costs are on subsets of random variables, instead of individual variables. Thus for a subset $S \subseteq V$, we pay price π_S to simultaneously probe all the random variables X_i for $i \in S$. This complicates the problem because to find X_i, we can probe a "small" or a "large" set containing i, but at different prices. Formally, we define the problem as follows.

Set-Probing Utility-Maximization Given probability distributions of independent non-negative variables X_i for $i \in V$ and given set family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where $S_j \subseteq V$ for $j \in [m]$ has a probing price

$\pi_j \geq 0$. The problem is to probe some of the sets in $\mathcal{S}$ with indices in $\mathsf{Probed} \subseteq [m]$ to maximize

$$\mathbb{E}\left[\max_{\exists j \in \mathsf{Probed} \ s.t. \ S_j \ni i} \{X_i\} - \sum_{j \in \mathsf{Probed}} \pi_j\right].$$

Note that when we probe multiple sets containing an element i, we find the same value X_i and not a fresh sample from the distribution.

Remark: If the sets S_j are pairwise disjoint then one can solve the above problem *optimally*: replacing each set S_j with a new random variable $X'_j = \max_{i \in S_j}\{X_i\}$ having probing price π_j reduces it to Pandora's box.

We use the constrained utility-maximization problem framework to show the following result.

Theorem 1.5. *The set-probing utility-maximization problem has a $3(\ell + 1)$-approximation efficient algorithm, where ℓ is the size of the largest set in $\mathcal{S}$. Moreover, no efficient algorithm can be $o(\ell/\log \ell)$-approximation, unless $P = NP$.*

1.3 Our Techniques

How do we bound the utility/disutility of the optimal adaptive strategy? The usual techniques in approximation algorithms for stochastic problems (see related work in Section 1.4) either use a linear program (LP) to bound the optimal strategy, or directly argue about the adaptivity gap of the optimal decision tree. Neither of these techniques is helpful because the natural LPs fail to capture a mixed-sign objective—they wildly overestimate the value of the optimal strategy. On the other hand, the adaptivity gap of our problems is large even for the special case of the Pandora's box problem—see an example in Section A.2.

We need two crucial ideas for both our utility-maximization and disutility-minimization results. Our first idea is to show that for semiadditive functions, one can bound the utility/disutility of the optimal strategy in the price-of-information world (hereafter, the *PoI world*) using a related instance in a world where there is no price to finding the parameters, i.e., $\pi_i = 0$ (hereafter, the *Free-Info world*). This proof crucially relies on the semiadditive nature of our value/cost function.

Our second idea is to show that any algorithm with "nice" properties in the Free-Info world can be used to get an algorithm with a similar expected utility/disutility in the PoI world. We call such a nice algorithm FRUGAL (and define it formally in Section 3.1). For intuition, imagine a FRUGAL algorithm to be a greedy algorithm, or an algorithm that is not "wasteful"—it picks elements irrevocably. This also includes simple primal-dual algorithms that do not have the *reverse-deletion* step.

Theorem 1.6. *If there exists a* FRUGAL *α-approximation Algorithm $\mathcal{A}$ to maximize (minimize) a semiadditive function over some packing constraints $\mathcal{F}$ (covering constraints $\mathcal{F}'$) in the Free-Info world then there exists an α-approximation algorithm for the corresponding utility-maximization (disutility-minimization) problem in the PoI world.*

Finally, to prove our results from Section 1.1, in Section 4 we show why many classical algorithms, or their suitable modifications, are FRUGAL.

Our techniques for the constrained utility-maximization problem in Section 5 again use the idea of bounding this problem in the PoI world with a similar problem in the Free-Info world. This latter problem turns out to be the same as the *stochastic probing* problem studied in [GN13, ASW14, GNS16, GNS17]. By proving an extension of the adaptivity gap result of Gupta et al. [GNS17], we show that one can further simplify these Free-Info problems to non-adaptive utility-maximization problems (by losing a constant factor). This we can now (approximately) solve using our techniques for the utility-maximization problem.

1.4 Related Work

An influential work of Dean et al. [DGV04] considered the stochastic knapsack problem where we have stochastic knowledge about the sizes of the items. Chen et al. [CIK$^+$09] studied stochastic matchings where we find about an edge's existence only after probing, and Asadpour et al. [ANS08] studied stochastic submodular maximization where the items may or may not be present. Several followup papers have appeared, e.g. for knapsack [BGK11, Ma14], packing integer programs [DGV05, CIK$^+$09, BGL$^+$12], budgeted multi-armed bandits [GM12, GM07, GKMR11, LY13, Ma14], orienteering [GM09, GKNR12, BN14], matching [Ada11, BGL$^+$12, BCN$^+$15, AGM15], and submodular objectives [GN13, ASW14]. Most of these results proceed by showing that the stochastic problem has a small adaptivity gap and then focus on the non-adaptive problem. In fact, Gupta et al. [GNS17, GNS16] show that adaptivity gap for submodular functions over any packing constraints is $O(1)$.

Most of the above works do not capture mixed-sign objective of maximizing the value *minus* the prices. Some of them instead model this as a knapsack constraint on the prices. Moreover, most of them are for maximization problems as for the minimization setting even the non-adaptive problem of probing k elements to minimize the expected minimum value has no polynomial approximation [GGM10]. This is also the reason we do not consider constrained (covering) disutility-minimization in Section 1.2. There is also a large body of work in related models where information has a price. We refer the readers to the following papers and the references therein [GK01, CFG$^+$02, KK03, GMS07, CJK$^+$15, AH15, CHKK15].

The Pandora's box solution can be written as a special case of the Gittins index theorem [Git74]. Dumitriu et al. [DTW03] consider a minimization variant of the Gittins index theorem when there is no discounting. Another very relevant paper is that of Kleinberg et al. [KWW16], while their results are to design auctions. Their proof of the Pandora's box problem inspired this work.

Organization In Section 2 we show how to bound the optimal strategy in the PoI world using a corresponding problem in the Free-Info world. In Section 3 we introduce the idea of using a FRUGAL algorithm to design a strategy with a good expected utility/disutility in the PoI world. In Section 4 we show why many classical algorithms, or their suitable modifications, are FRUGAL. Finally, in Section 5 we discuss the settings where we have probing constraints, and its application to the set-probing problem.

2 Bounding the Optimal Strategy for Utility/Disutility Optimization

In this section we bound the expected utility/disutility of the optimal adaptive strategy for a combinatorial optimization in the PoI world in terms of a surrogate problem in the Free-Info world. We first define the *grade* τ and *surrogate* Y of non-negative random variables.

Definition 2.1 (*Grade* τ). *For any non-negative random variable X_i, let $\tau_i^{\max}$ be the solution to equation $\mathbb{E}[(X_i - \tau_i^{\max})^+] = \pi_i$ and let $\tau_i^{\min}$ be the solution to equation $\mathbb{E}[(\tau_i^{\min} - X_i)^+] = \pi_i$.*

Definition 2.2 (*Surrogate Y*)**.** *For any non-negative random variable X_i, let $Y_i^{\max} = \min\{X_i, \tau_i^{\max}\}$ and let $Y_i^{\min} = \max\{X_i, \tau_i^{\min}\}$.*

Note that $\tau_i^{\max}$ could be negative in the above definition. The following lemmas bound the optimal strategy in the PoI world in terms of the optimal strategy of a surrogate problem in the Free-Info world.

Lemma 2.3. *The expected utility of the optimal strategy to maximize a semiadditive function* val *over packing constraints $\mathcal{F}$ in the PoI world is at most*

$$\mathbb{E}_{\mathbf{X}}[\max_{\mathbb{I}\in\mathcal{F}}\{\mathsf{val}(\mathbb{I}, \mathbf{Y}^{\max})\}].$$

Lemma 2.4. *The expected disutility of the optimal strategy to minimize a semiadditive function* cost *over covering constraints $\mathcal{F}'$ in the PoI world is at least*

$$\mathbb{E}_{\mathbf{X}}[\min_{\mathbb{I}\in\mathcal{F}'}\{\mathsf{cost}(\mathbb{I}, \mathbf{Y}^{\min})\}].$$

We only prove Lemma 2.3 as the proof of Lemma 2.4 is similar. The ideas in this proof are similar to that of Kleinberg et al. [KWW16, Lemma 1] to bound the optimal adaptive strategy for Pandora's box.

Proof of Lemma 2.3. Consider a fixed optimal adaptive strategy. Let A_i denote the indicator variable that element i is selected into $\mathbb{I}$ and let $\mathsf{1}_i$ denote the indicator variable that element i is probed by the optimal strategy. Note that these indicators are correlated and the set of elements with non-zero A_i is feasible in $\mathcal{F}$. Now, the optimal strategy has expected utility

$$\begin{aligned}
&= \mathbb{E}\left[\mathsf{val}(\mathbb{I}, \mathbf{X}) - \sum_{i\in\mathsf{Probed}} \pi_i\right] \\
&= \mathbb{E}\left[\sum_i (A_i X_i - \mathsf{1}_i \pi_i)\right] + \mathbb{E}[h(\mathbb{I})] \\
&= \mathbb{E}\left[\sum_i \left(A_i X_i - \mathsf{1}_i \mathbb{E}_{X_i}[(X_i - \tau_i^{\max})^+]\right)\right] + \mathbb{E}[h(\mathbb{I})],
\end{aligned}$$

using the definition of π_i. Since value of X_i is independent of whether it's probed or not, we simplify to

$$= \mathbb{E}\left[\sum_i \left(A_i X_i - \mathsf{1}_i (X_i - \tau_i^{\max})^+\right)\right] + \mathbb{E}[h(\mathbb{I})].$$

Moreover, since we can select an element into $\mathbb{I}$ only after probing, we have $\mathsf{1}_i \geq A_i$. This implies that the expected utility of the optimal strategy is

$$\begin{aligned}
&\leq \mathbb{E}\left[\sum_i \left(A_i X_i - A_i (X_i - \tau_i^{\max})^+\right)\right] + \mathbb{E}[h(\mathbb{I})] \\
&= \mathbb{E}\left[\sum_i A_i Y_i^{\max}\right] + \mathbb{E}[h(\mathbb{I})] \\
&= \mathbb{E}[\mathsf{val}(\mathbb{I}, \mathbf{Y}^{\max})].
\end{aligned}$$

Finally, since elements in $\mathbb{I}$ form a feasible set, this is at most $\mathbb{E}[\max_{\mathbb{I}\in\mathcal{F}}\{\mathsf{val}(\mathbb{I}, \mathbf{Y}^{\max})\}]$. ☐

3 Designing an Adaptive Strategy for Utility/Disutility Optimization

In this section we introduce the notion of a FRUGAL algorithm and prove Theorem 1.6. We need the following notation.

Definition 3.1 ($\mathbf{Y}_M$). *For any vector $\mathbf{Y}$ with indices in V and any $M \subseteq V$, let $\mathbf{Y}_M$ denote a vector of length $|V|$ with entries Y_j for $j \in M$ and a symbol $*$, otherwise.*

3.1 A FRUGAL Algorithm

The notion of a FRUGAL algorithm is similar to that of a greedy algorithm, or any other algorithm that is not "wasteful"—it selects elements one-by-one and irrevocably. Its definition captures "non-greedy" algorithms such as the primal-dual algorithm for set cover that does not have the *reverse-deletion* step.

We define a FRUGAL algorithm in the packing setting. Consider a packing problem in the Free-Info world (i.e., $\forall i, \pi_i = 0$) where we want to find a feasible set $\mathbb{I} \in \mathcal{F}$ and $\mathcal{F} \subseteq 2^V$ are some downward-closed constraints, while trying to maximize a semiadditive function $\mathsf{val}(\mathbb{I}, \mathbf{Y}) = \sum_{i\in\mathbb{I}} Y_i + h(\mathbb{I})$.

Definition 3.2 (FRUGAL Packing Algorithm)**.** *For a packing problem with constraints $\mathcal{F}$ and value function* val, *we say Algorithm $\mathcal{A}$ is* FRUGAL *if there exists a* marginal-value *function $g(\mathbf{Y}, i, y) : \mathbb{R}^V \times V \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is increasing in y, and for which the pseudocode is given by Algorithm 1. We note that this algorithm always returns a feasible solution if we assume $\emptyset \in \mathcal{F}$.*

Algorithm 1 FRUGAL Packing Algorithm $\mathcal{A}$

```
1: Start with $M = \emptyset$ and $v_i = 0$ for each element $i \in V$.
2: For each element $i \notin M$, compute $v_i = g(\mathbf{Y}_M, i, Y_i)$. Let $j = \mathrm{argmax}_{i\notin M \text{ \& } M\cup i\in\mathcal{F}}\{v_i\}$.
3: If $v_j > 0$ then add $j$ into $M$ and go to Step 2. Otherwise, return $M$.
```

A simple example of a FRUGAL packing algorithm is the greedy algorithm to find the maximum weight spanning tree (or to maximize any weighted matroid rank function), where $g(\mathbf{Y}_M, i, Y_i) = Y_i$.

We similarly define a FRUGAL algorithm in the covering setting. Consider a covering problem in the Free-Info world where we want to find a feasible set $\mathbb{I} \in \mathcal{F}'$, where $\mathcal{F}' \subseteq 2^V$ is some upward-closed constraint, while trying to minimize a semiadditive function $\mathsf{cost}(\mathbb{I}, \mathbf{Y}) = \sum_{i\in\mathbb{I}} Y_i + h(\mathbb{I})$.

Definition 3.3 (FRUGAL Covering Algorithm)**.** *For a covering problem with constraints $\mathcal{F}'$ and cost function* cost, *we say Algorithm $\mathcal{A}$ is* FRUGAL *if there exists a* marginal-value *function $g(\mathbf{Y}, i, y) : \mathbb{R}^V_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that is increasing in y, and for which the pseudocode is given by Algorithm 2.*

Algorithm 2 FRUGAL Coverage Algorithm $\mathcal{A}$

```
1: Start with $M = \emptyset$ and $v_i = 0$ for each element $i \in V$.
2: For each element $i \notin M$, compute $v_i = g(\mathbf{Y}_M, i, Y_i)$. Let $j = \mathrm{argmax}_{i\notin M}\{v_i\}$.
3: If $v_j > 0$ then add $j$ into $M$ and go to Step 2. Otherwise, return $M$.
```

We note that for a covering problem it is unclear whether Algorithm 2 returns a feasible solution as we do not appear to be looking at our covering constraints $\mathcal{F}'$. To overcome this, we say the marginal-value *function g* encodes $\mathcal{F}'$ *if whenever M is infeasible then there exists an element $i \notin M$ with $v_i > 0$. This means that the algorithm will return a feasible solution as long as $V \in \mathcal{F}'$.*

A simple example of a FRUGAL covering algorithm is the greedy min-cost set cover algorithm, where $g(\mathbf{Y}_M, i, Y_i) = \left(|\bigcup_{j \in M \cup i} S_j| - |\bigcup_{j \in M} S_j|\right) / Y_i$. Note that here g *encodes* our coverage constraints.

Remark: Observe that a crucial difference between FRUGAL packing and covering algorithms is that a FRUGAL packing algorithm has to handle $\mathbf{Y} \in \mathbb{R}^V$ (i.e. some entries in $\mathbf{Y}$ could be negative) but a FRUGAL covering algorithm has to only handle $\mathbf{Y} \in \mathbb{R}^V_{\geq 0}$. The intuition behind this difference is that unlike the disutility minimization problem, the utility maximization problem has a mixed-sign objective.

3.2 Using a FRUGAL Algorithm to Design an Adaptive Strategy

After defining the notion of a FRUGAL algorithm, we can now prove Theorem 1.6 (restated below).

Theorem 1.6. *If there exists a* FRUGAL *α-approximation Algorithm $\mathcal{A}$ to maximize (minimize) a semiadditive function over some packing constraints $\mathcal{F}$ (covering constraints $\mathcal{F}'$) in the Free-Info world then there exists an α-approximation algorithm for the corresponding utility-maximization (disutility-minimization) problem in the PoI world.*

We prove Theorem 1.6 only for the utility-maximization setting as the other proof is similar. Lemma 2.3 already gives us an upper bound on the expected utility of the optimal strategy for the utility-maximization problem in terms of the expected value of a problem in the Free-Info world. This Free-Info problem can be solved using Algorithm $\mathcal{A}$. The main idea in the proof of this theorem is to show that if Algorithm $\mathcal{A}$ is FRUGAL then we can also run a modified version of $\mathcal{A}$ in the PoI world and get the same expected utility.

Proof of Theorem 1.6. Let $Alg(\mathbf{Y}^{\max}, \mathcal{A})$ denote the set $\mathbb{I} \in \mathcal{F}$ returned by Algorithm $\mathcal{A}$ when it runs with element weights $\mathbf{Y}^{\max}$. Since $\mathcal{A}$ is an α-approximation algorithm (where $\alpha \geq 1$), we know

$$\mathsf{val}(Alg(\mathbf{Y}^{\max}, \mathcal{A}), \mathbf{Y}^{\max}) \geq \frac{1}{\alpha} \cdot \max_{\mathbb{I} \in \mathcal{F}} \{\mathsf{val}(\mathbb{I}, \mathbf{Y}^{\max})\}. \tag{2}$$

The following crucial lemma shows that one can design an adaptive strategy in the PoI world with the same expected utility.

Lemma 3.4. *If Algorithm $\mathcal{A}$ is* FRUGAL *then there exists an algorithm in the PoI world with expected utility*

$$\mathbb{E}_{\mathbf{X}}[\mathsf{val}(Alg(\mathbf{Y}^{\max}, \mathcal{A}), \mathbf{Y}^{\max})].$$

Before proving Lemma 3.4, we finish the proof of Theorem 1.6. Recollect that Lemma 2.3 shows that $\mathbb{E}[\max_{\mathbb{I} \in \mathcal{F}}\{\mathsf{val}(\mathbb{I}, \mathbf{Y}^{\max})]$ is an upper bound on the expected optimal utility in the PoI world. Combining this with Lemma 3.4 and Eq. (2) gives an α-approximation algorithm in the PoI world. □

We first give some intuition for the missing Lemma 3.4. The lemma is surprising because it says that there exists an algorithm in the PoI world that has the same expected utility as Algorithm $\mathcal{A}$ in the Free-Info world, where there are no prices. The fact that in the Free-Info world Algorithm $\mathcal{A}$ can only get the smaller surrogate values $Y_i^{\max} = \min\{X_i, \tau_i^{\min}\}$, instead of the actual value X_i, comes to our rescue. We show that $Y^{\max}$ is defined in a manner to balance this difference in the values with the probing prices.

Proof of Lemma 3.4. Since $\mathcal{A}$ is FRUGAL, we would like to run Algorithm 1 in the PoI world. The difficulty is that we do not know $\mathbf{Y}^{\max}$ values of the unprobed elements. To overcome this hurdle, consider Algorithm 3 that uses the grade $\tau^{\max}$ as a proxy for $\mathbf{Y}^{\max}$ values of the unprobed elements.

Claim 3.5. *The set of elements returned by Algorithm 3 is the same as that by Algorithm 1 running with* $\mathbf{Y} = \mathbf{Y}^{\max}$.

Proof of Claim 3.5. We prove the claim by induction on the number of elements selected by Algorithm 3. Suppose the set of elements selected by both the algorithms into M are the same till now and Algorithm 3 decides to select element j in Step 3(a). This means that j is already probed before this step. The only concern is that Algorithm 3 selects j without probing some other element i based on its grade $\tau_i^{\max}$. We observe that this step is consistent with Algorithm 1 because $Y_i^{\max} \leq \tau_i^{\max}$ and $g(\mathbf{Y}_M^{\max}, i, Y_i^{\max})$ is an increasing function in $Y_i^{\max}$, which implies

$$g(\mathbf{Y}_M^{\max}, i, Y_i^{\max}) \quad \leq \quad g(\mathbf{Y}_M^{\max}, i, \tau_i^{\max}) \quad \leq \quad g(\mathbf{Y}_M^{\max}, i, Y_j^{\max}). \qquad \square$$

An immediate corollary is that value of Algorithm 3 in the Free-Info world is

$$\mathbb{E}_{\mathbf{X}}[\mathsf{val}(Alg(\mathbf{Y}^{\max}, \mathcal{A}), \mathbf{Y}^{\max})]. \tag{3}$$

In Claim 3.6 we argue that this expression also gives expected utility of Algorithm 3 in the PoI world, which completes the proof of Lemma 3.4. $\square$

Claim 3.6. *The expected utility of Algorithm 3 in PoI world is*

$$\mathbb{E}_{\mathbf{X}}[\mathsf{val}(Alg(\mathbf{Y}^{\max}, \mathcal{A}), \mathbf{Y}^{\max})].$$

Proof of Claim 3.6. We first expand the claimed expression,

$$\mathbb{E}_{\mathbf{X}}[\mathsf{val}(Alg(\mathbf{Y}^{\max}, \mathcal{A}), \mathbf{Y}^{\max})] = \mathbb{E}\left[\sum_{i \in Alg(\mathbf{Y}^{\max}, \mathcal{A})} Y_i^{\max}\right] + \mathbb{E}[h(Alg(\mathbf{Y}^{\max}, \mathcal{A}))]. \tag{4}$$

Observe that to prove the claim we can ignore the second term, $\mathbb{E}[h(Alg(\mathbf{Y}^{\max}, \mathcal{A}))]$, because it contributes the same in both the worlds (it is only a function of the returned feasible set). We now argue that in every step of Algorithm 3 the expected change in $\sum_{i \in M} Y_i^{\max}$ in the Free-Info world is the same as the expected increase in $\sum_{i \in M} X_i$ minus the probing prices in the PoI world.

Algorithm 3 Utility-Maximization

```
1: Start with $M = \emptyset$ and $v_i = 0$ for all elements $i$.
2: For each element $i \notin M$:
   (a) if $i$ is probed let $v_i = g(\mathbf{Y}_M^{\max}, i, Y_i^{\max})$.
   (b) if $i$ is unprobed let $v_i = g(\mathbf{Y}_M^{\max}, i, \tau_i^{\max})$.
3: Consider the element $j = \text{argmax}_{i \notin M \text{ \& } M \cup i \in \mathcal{F}}\{v_i\}$ and $v_j > 0$.
   (a) If $j$ is already probed then select it into $M$ and set $v_j = 0$.
   (b) If $j$ is not probed then probe it. If $X_j \geq \tau_j^{\max}$ then select $j$ into $M$ and set $v_j = 0$.
4: If every element $i \notin M$ has $v_i = 0$ then return set $M$. Else, go to Step 2.
```

We first consider the case when the next highest element j in Step 3 of Algorithm 3 is already probed and has $v_j > 0$. In this case, the algorithm selects element j. Since this element has been already probed before (but not selected then), it means $X_j < \tau_j^{\max}$ and $X_j = Y_j^{\max}$. Hence the increase in the value of the algorithm in both the worlds is X_j.

Next, consider the case that the next highest element j in Step 3 has not been probed before. Let μ_j denote the probability density function of random variable X_j. Now the expected increase in the value in the Free-Info world is

$$\tau_j^{\max} \cdot \Pr[X_j \geq \tau_j^{\max}] \quad = \quad \tau_j^{\max} \cdot \int_{t=\tau_j^{\max}}^{\infty} \mu_j(t) dt.$$

This is because the algorithm selects this element in this step only if its value is at least $\tau_j^{\max}$, in which case $Y_j^{\max} = \tau_j^{\max}$. On the other hand, the expected increase in the value in the PoI world is given by

$$-\pi_j + \int_{t=\tau_j^{\max}}^{\infty} t \cdot \mu_j(t) dt.$$

This is because we pay the probing cost π_j and get a positive value X_j only when $X_j \geq \tau_j^{\max}$. Now using the definition of $\tau_j^{\max}$, we can simplify the above equation to

$$-\int_{t=\tau_j^{\max}}^{\infty} (t - \tau_j^{\max}) \mu_j(t) dt + \int_{t=\tau_j^{\max}}^{\infty} t \cdot \mu_j(t) dt \quad = \quad \tau_j^{\max} \cdot \int_{t=\tau_j^{\max}}^{\infty} \mu_j(t) dt.$$

This shows that in every step of the algorithm the expected increase in the value in both the worlds is the same, thereby proving Claim 3.6. □

4 Applications to Utility/Disutility Optimization

In this section we show that for several combinatorial problems there exist FRUGAL algorithms. Hence we can use Theorem 1.6 to obtain optimal/approximation algorithms for the corresponding utility-maximization or disutility-minimization problem in PoI world.

4.1 Utility-Maximization

To recollect, in the utility-maximization setting we are given a semiadditive value function $\mathsf{val} \geq 0$ and a packing constraint $\mathcal{F}$. Our goal is to probe a set of elements Probed and select a feasible set $\mathbb{I} \subseteq \mathsf{Probed}$ in $\mathcal{F}$ to maximize expected utility,

$$\mathbb{E}\left[\mathsf{val}(\mathbb{I}, \mathbf{X}) - \sum_{i \in \mathsf{Probed}} \pi_i \right].$$

We assume that $\emptyset \in \mathcal{F}$ and hence there always exist a solution of utility zero.

4.1.1 k-System

Let $\mathsf{val}(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i$ be an additive function and let $\mathcal{F}$ denote a k-system constraint in this setting. To prove Theorem 1.2, we observe that the greedy algorithm that starts with an empty set and at every step selects the next feasible element maximum marginal-value is an α-approximation algorithm is a FRUGAL algorithm as defined in Definition 3.2. We know that this greedy algorithm is a k-approximation for additive functions over a k-system in Free-Info world [Jen76, KH78]. Hence, Theorem 1.6 combined with the greedy algorithm gives the k-system part of Theorem 1.2 as a corollary.

4.1.2 Knapsack

Given a knapsack of size B, suppose each item i has a known size s_i ($\leq B$) but a stochastic value X_i. To find X_i, we have to pay probing price π_i. The goal is to probe a subset of items Probed and select a subset $\mathbb{I} \subseteq$ Probed, where $\sum_{i\in\mathbb{I}} s_i \leq B$, to maximize the expected utility

$$\mathbb{E}\left[\sum_{i\in\mathbb{I}} X_i - \sum_{i\in\mathsf{Probed}} \pi_i\right].$$

We can model this problem in our utility-maximization framework by taking $\mathsf{val}(\mathbb{I}, \mathbf{X}) = \sum_{i\in\mathbb{I}} X_i$ and $\mathcal{F}$ to contain every subset S of items that fit into the knapsack.

In the Free-Info world, consider a greedy algorithm that sorts items in decreasing order based on the ratio of their value and size, and then selects items greedily in this order until the knapsack is full. This greedy algorithm does not always give a constant approximation to the knapsack problem. Similarly, an algorithm that selects only the most valuable item is not always a constant approximation algorithm (recollect that we can pick every item i because $s_i \leq B$). However, it's known that for any knapsack instance if we randomly run one of the previous two algorithms, each w.p. half, then this is a 2-approximation algorithm.

From Theorem 1.6, we can simulate the greedy algorithm in the PoI world. Also, using the solution to the Pandora's box problem, we can simulate selecting the most valuable item in the PoI world. Hence, consider an algorithm that for any given knapsack problem in the PoI world, runs either the simulated greedy algorithm or the Pandora's box solution, each with probability half. Such an algorithm is a 2-approximation to the knapsack problem in the PoI world.

4.2 Disutility-Minimization

To recollect, in the disutility-minimization setting we are given a semiadditive cost function cost ≥ 0 and a covering constraint $\mathcal{F}'$. Our goal is to probe a set of elements Probed and select a feasible set $\mathbb{I} \subseteq$ Probed in $\mathcal{F}'$ to minimize expected disutility,

$$\mathbb{E}\left[\mathsf{cost}(\mathbb{I}, \mathbf{X}) + \sum_{i\in\mathsf{Probed}} \pi_i\right].$$

We will assume that $V \in \mathcal{F}$, and hence there always exists a feasible solution.

4.2.1 Matroid Basis

Given a matroid $\mathcal{M}$ of rank r on n elements, we consider the additive function $\mathsf{cost}(\mathbb{I}, \mathbf{X}) = \sum_{i\in\mathbb{I}} X_i$ and let $\mathcal{F}'$ be subsets of elements that contain a basis of $\mathcal{M}$. To ensure that a feasible set of finite value exists, we make the following assumption.

Assumption 4.1. *We can always extend a set $\mathbb{I} \in \mathcal{M}$ to a basis by probing and selecting items with zero penalty but with large probing cost π_0. Thus it incurs an additional penalty of $(r - |rank(\mathbb{I})|) \cdot \pi_0$.*

To use Theorem 1.6, we notice that the greedy algorithm that always selects the minimum cost independent element is FRUGAL. This is true because we choose marginal-value function g to be the reciprocal of the weight of an element in Definition 3.3. Now since the greedy algorithm is optimal for min-cost matroid basis, this proves the first part of Theorem 1.3.

4.2.2 Set Cover

Consider a problem where we are given sets $S_1, \dots, S_m \subseteq V$ that have some unknown stochastic costs X_i. The goal is to select a set cover with minimum disutility, which is the sum of the set cover solution costs and the probing prices. We can model this problem in our framework by considering a the additive function $\mathsf{cost}(\mathbb{I}, \mathbf{X}) = \sum_{i \in \mathbb{I}} X_i$ and $\mathcal{F}'$ be set covers of V.

To ensure that the solution is always bounded, we make the following assumption.

Assumption 4.2. *There exists $S_0 = [n]$ that covers all elements and has $X_0 = 0$, but a finite large π_0.*

To prove the set cover part of Theorem 1.3, we first notice that the classical $O(\log |V|)$ greedy algorithm for the min-cost set cover problem is FRUGAL. This is because the marginal-value function $g(\mathbf{Y}_M, i, Y_i)$ in Definition 3.3 is equal to $\Big(|\bigcup_{j \in M \cup i} S_j| - |\bigcup_{j \in M} S_j| \Big) / Y_i$.

Next we give an f-approximation algorithm, where f is the maximum number of sets in which an element can appear. We observe that the primal-dual f-approximation algorithm (see pseudocode in Algorithm 4) for the min-cost set cover [BYE81, WS11] is also FRUGAL. This is because we can encode the information about the order σ and the dual variables y_j for $j \in M$ in the marginal-value function $g(\mathbf{Y}_M, i, Y_i)$ in Definition 3.3.

Algorithm 4 Primal-dual algorithm for min-cost set cover

```
1: Fix an order $\sigma$ on the ground elements. Start with $M = \emptyset$ and $y_j = 0$ for every ground element $j$.
2: Select the next element $j \notin \bigcup_{i \in M} S_i$ according to $\sigma$ and raise its dual variable $y_j$ until some set $i$
   becomes tight, i.e., $\sum_{j \in S_i} y_j = Y_i$.
3: Select every tight set into $M$.
4: If every ground element is covered in $\bigcup_{i \in M} S_i$ then return $M$, else go to Step 2.
```

4.2.3 Uncapacitated Facility Location

Consider an uncapacitated facility location problem where we are given a graph $G = (V, E)$ with metric (V, d) and CLIENTS $\subseteq V$, however, facility opening costs X_i for $i \in V$ are stochastic and can be found by paying a probing price π_i. The goal is to probe a set of facility locations $\mathsf{Probed} \subseteq V$ and open a non-empty subset $I \subseteq \mathsf{Probed}$ to minimize expected disutility

$$\mathbb{E}\left[\sum_{u \in \text{CLIENTS}} d(u, \mathbb{I}) + \sum_{i \in \mathbb{I}} X_i + \sum_{i \in \mathsf{Probed}} \pi_i \right],$$

where $d(u, \mathbb{I}) = \min_{i \in \mathbb{I}} d(u, i)$.

We model the above problem in our framework by defining exponential number of elements that are indexed by (i, S), for $i \in V$ and $S \subseteq$ CLIENTS, which denotes that facility i will serve clients S. Any subset of elements, say $\mathbb{I} = \{(i_1, S_1), (i_2, S_2), \ldots\}$, is feasible if the union of their clients covers CLIENTS. The semiadditive $\mathsf{cost}(\mathbb{I}, \mathbf{X})$ is given by $\sum_{(i,S) \in \mathbb{I}} \Big(X_i + \sum_{j \in S} d(i, j) \Big)$.

We notice that the 1.861-approximation greedy algorithm of Jain et al. [JMM+03] for the uncapacitated facility location problem is FRUGAL. In each step, their algorithm selects the next best element with minimum cost per client, where already opened facilities now have zero opening costs. The reciprocal of this value gives the marginal-value function g. Hence, we can use Theorem 1.6 to obtain a 1.861-approximation strategy.

4.2.4 Prize-Collecting Steiner Tree

Consider a Prize-Collecting Steiner tree problem (PCST) where we are given a graph $G = (V, E)$ with some edge costs $\mathbf{c} : E \to \mathbb{R}_{\geq 0}$, a root node $r \in V$, and probability distributions on the independent penalties X_i for $i \in V$. The stochastic penalties X_i can be found by paying a probing price π_i. The goal is to probe a set of nodes $\mathsf{Probed} \subseteq V \setminus \{r\}$ and select a subset $I \subseteq \mathsf{Probed}$ to minimize expected disutility,

$$\mathbb{E}\left[\sum_{i \in \mathbb{I}} X_i + \text{Min-Steiner-Tree}(V \setminus \mathbb{I}) + \sum_{i \in \mathsf{Probed}} \pi_i \right],$$

where Min-Steiner-Tree$(V \setminus \mathbb{I})$ denotes the minimum cost Steiner tree connecting all nodes in $V \setminus \mathbb{I}$ to r.

As discussed in Section 1.1, we can model the PCST in our disutility-minimization framework by noticing that the function $\mathsf{cost}(\mathbf{X}, \mathbb{I}) = \sum_{i \in \mathbb{I}} X_i + \text{Min-Steiner-Tree}(V \setminus \mathbb{I})$ is semiadditive. We show that although the 2-approximation Goemans-Williamson [GW95] algorithm (hereafter, GW-algorithm) for PCST in the Free-Info world is not FRUGAL, it can be modified to obtain a 3-approximation FRUGAL algorithm for PCST. Combining this with Theorem 1.6 gives a 3-approximation algorithm for PCST in the PoI world.

We quickly recollect the 2-approximation primal-dual GW-algorithm. (We do not repeat their proof and refer to [WS11, Chapter 14] for details.) Their algorithm starts by making each node $i \in V \setminus \{r\}$ active with initial charge $p(\{i\}) = X_i$. At any time, the algorithm grows a *moat* around each active component C and discharges C at the same rate. If a component C runs out of charge, we make it inactive and mark every unlabeled node in the component with label C. If an edge e becomes *tight*, we pick e, merge the two components C, C' connected by e, make both C, C' inactive, and make $C \cup C'$ active with an initial charge of $p(C) + p(C')$. Any component that hits the component containing r is made inactive. In the *cleanup phase* we remove all edges that do not disconnect an unmarked node from r, while ensuring that if a component with label C is connected to r then every node with label $C' \supseteq C$ is also connected to r.

We first observe that the GW-algorithm is not FRUGAL. This is because whenever a node i is labeled with a component $C \ni i$, the algorithm looks at the penalty X_i; however, the decision of whether to select i into $\mathbb{I}$ (i.e., not connecting i to r) is not made until the cleanup phase. The reason is that some other active component C' might later come and merge with C, and eventually connect i to r. To fix this, we modify this algorithm to make it FRUGAL. The idea is to immediately include the labeled vertices into $\mathbb{I}$.

Consider an algorithm that creates the same tree as the GW-algorithm; however, any node that ever gets labeled during the run of the algorithm is imagined to be included into $\mathbb{I}$. This means that although our final tree might connect a labeled node i to r, our algorithm still pays its penalty X_i. We argue that these additional penalties are at most the optimal PCST solution in the Free-Info world, which gives us a 3-approximation FRUGAL algorithm.

Finally, to argue that the additional penalties are not large, consider the state of the GW-algorithm before the cleanup phase. Let $\mathcal{C}$ denote the set of maximal inactive components. Clearly, each node i that was every labeled belongs to some maximal tight component $C \in \mathcal{C}$. Hence, the sum of the additional penalties is upper bounded by $\sum_{C\in\mathcal{C}} \sum_{i\in C} X_i$. Since each component $C \in \mathcal{C}$ is tight, we know $\sum_{C\in\mathcal{C}} \sum_{i\in C} X_i = \sum_{C\in\mathcal{C}} \sum_{S\subseteq C} y_S$, where y_S are the dual variables corresponding to the moats. Since the dual solutions form a feasible dual-solution, they are a lower bound on the optimal solution for the problem. This proves that the additional penalty paid by our FRUGAL algorithm in comparison to GW-algorithm is at most the optimal solution.

4.2.5 Feedback Vertex Set

Given an undirected graph $G = (V, E)$, suppose each node $i \in V$ has a stochastic weight X_i, which we can find by probing and paying price π_i. The problem is to probe a set $\mathsf{Probed} \subseteq V$ and select a subset $\mathbb{I} \subseteq \mathsf{Probed}$ s.t. the induced graph $G[V \setminus \mathbb{I}]$ contains no cycle, while minimizing the expected disutility

$$\mathbb{E}\left[\sum_{i\in\mathbb{I}} X_i + \sum_{i\in\mathsf{Probed}} \pi_i\right].$$

The above problem can be modeled in our framework by considering the additive function $\mathsf{cost}(\mathbb{I}, \mathbf{X}) = \sum_{i\in\mathbb{I}} X_i$ and $\mathcal{F}'$ contains a set of nodes S if $G[V \setminus S]$ has no cycle. Becker and Geiger [BG96] showed that the greedy Algorithm 5 is an $O(\log n)$-approximation algorithm for the feedback vertex set problem in the Free-Info world. Since this algorithm is FRUGAL, using Theorem 1.6 we get an $O(\log n)$-approximation algorithm for minimizing disutility for the feedback vertex set problem in the PoI world.

Algorithm 5 Greedy Algorithm for Feedback Vertex Set

```
1: Start with $R = M = \emptyset$ and $v_i = 0$ for each element $i \in V$.
2: While $\exists i \in V \setminus (R \cup M)$ s.t. degree of $i$ in $G[V \setminus (R \cup M)]$ is 0 or 1, add $i$ to $R$.
3: For each element $i \notin R \cup M$, compute $v_i = degree(i, G[V \setminus (R \cup M)])/w(i)$, where $degree(i, G)$ is
   the degree of vertex $i$ in $G$ and $w(i)$ is the weight of vertex $i$.
4: Let $j = \mathrm{argmax}_{i \notin R\cup M}\{v_i\}$. Add $j$ to $M$.
5: If $R \cup M \neq V$, go to Step 2. Otherwise, return $M$.
```

Remark: The $O(\log n)$-approximation primal-dual algorithm in [BYGNR98] (or in Chapter 7.2 of [WS11]) can be also shown to be FRUGAL. This gives another $O(\log n)$-approximation algorithm for minimizing disutility for feedback vertex set problem.

5 Constrained Utility-Maximization

In this section we consider a generalization of the Pandora's box problem where we have an additional constraint that allows us to only probe a subset of elements $\mathsf{Probed} \subseteq V$ that belongs to a downward-closed constraint $\mathcal{J}$ (e.g., a cardinality constraint allowing us to probe at most k elements). We restate our main result for the constrained utility-maximization problem (see problem definition in Section 1.2).

Theorem 1.4. *If the constraints $\mathcal{J}$ form an ℓ-system then the constrained utility-maximization problem has a $3(\ell + 1)$-approximation algorithm.*

Similar to Section 2, our strategy to prove Theorem 1.4 is to bound the constrained utility-maximization problem in the PoI world (a mixed-sign objective function) with a surrogate constrained utility-maximization problem in the Free-Info world (i.e., where $\pi_i = 0$ for all $i \in V$). This latter problem turns out to be the same as the *stochastic probing* problem, which we define below in the form that is relevant to this paper.

Stochastic Probing Given downward-closed probing constraints $\mathcal{J} \subseteq 2^V$ and probability distributions of independent non-negative variables Y_i for $i \in V$, the stochastic probing problem is to *adaptively* probe a subset $\mathsf{Probed} \in \mathcal{J}$ to maximize the expected value $\mathbb{E}[\max_{i \in \mathsf{Probed}}\{Y_i\}]$. Here, *adaptively* means that the decision to probe which element next can depend on the outcomes of the already probed elements.

5.1 Reducing Constrained Utility-Maximization to Non-adaptive Stochastic Probing

The following lemma bounds the expected utility of the constrained utility-maximization problem in the PoI world by the expected value of a stochastic probing problem in the Free-Info world.

Lemma 5.1. *The expected utility of the optimal strategy for the constrained utility-maximization problem is at most the expected value of the optimal adaptive strategy for a stochastic probing problem with the same constraints $\mathcal{J}$ and where the random variables Y_i for $i \in V$ have probability distributions $Y_i^{\max}$ (recollect, Definition 2.2).*

Proof of Lemma 5.1. We start by noticing that the optimal strategy for our problem is given by a decision tree T with leaves l. For any root leaf path P_l, the value of the optimal strategy is $\max_{i \in P_l}\{X_i\} - \pi(P_l)$. Thus the expected value of the optimal strategy is

$$\mathbb{E}_l \left[\max_{i \in P_l}\{X_i\} - \pi(P_l) \right]. \tag{5}$$

Now we design an adaptive strategy for the stochastic probing problem on random variables $Y_i^{\max}$ with expected value at least as given by Eq. (5). Consider the adaptive strategy that follows the same decision tree T (note, it pays no probing price). The expected value of such an adaptive strategy is given by

$$\mathbb{E}_l \left[\max_{i \in P_l}\{Y_i^{\max}\} \right]. \tag{6}$$

The following claim finishes the proof of this lemma.

Claim 5.2. *Eq.* (5) $\leq$ *Eq.* (6).

Proof. Let $A_l(i)$ and $\mathbb{1}_{i \in P_l}$ denote indicator variables that element i is selected and probed on a root-leaf path P_l of the optimal strategy, respectively. Note that these indicator variables are correlated. The expected utility of the optimal strategy equals

$$
\begin{aligned}
\mathbb{E}_l\left[\max_{i\in P_l}\{X_i\} - \pi(P_l)\right] &= \mathbb{E}_l\left[\sum_i (A_l(i)X_i - \mathsf{1}_{i\in P_l}\pi_i)\right] \\
&= \mathbb{E}_l\left[\sum_i \left(A_l(i)X_i - \mathsf{1}_{i\in P_l}\mathbb{E}_i[(X_i - \tau_i^{\max})^+]\right)\right] \\
&= \mathbb{E}_l\left[\sum_i \left(A_l(i)X_i - \mathsf{1}_{i\in P_l}(X_i - \tau_i^{\max})^+\right)\right]
\end{aligned}
$$

since X_i is independent of $\mathsf{1}_{i\in P_l}$. Now using $A_l(i) \le \mathsf{1}_{i\in P_l}$,

$$
\begin{aligned}
&\le \mathbb{E}_l\left[\sum_i \left(A_l(i)X_i - A_l(i)(X_i - \tau_i^{\max})^+\right)\right] \\
&= \mathbb{E}_l\left[\sum_i A_l(i)Y_i^{\max}\right] \\
&\le \mathbb{E}_l\left[\max_{i\in P_l}\{Y_i^{\max}\}\right],
\end{aligned}
$$

where the last inequality uses $\sum_{i\in P_l} A_l(i) \le 1$.

□

The following Lemma 5.3 shows that we can further simplify the stochastic probing problem in the Free-Info world by focusing only on finding the best *non-adaptive* strategy for this problem, i.e. the problem of finding $\text{argmax}_{\mathsf{Probed}\in\mathcal{J}}\{\max_{i\in\mathsf{Probed}}\{Y_i\}\}$. This is because the *adaptivity gap*—ratio of the expected values of the optimal adaptive and optimal non-adaptive strategies—for the stochastic probing problem is small.

Lemma 5.3. *The adaptivity gap for the stochastic probing problem is at most* 3.

We prove Lemma 5.3 in Section 5.2. It tells us about the existence of a feasible set $S \in \mathcal{J}$ such that $\mathbb{E}[\max_{i\in S}\{Y_i^{\max}\}]$ is at least $1/3$ times the optimal adaptive strategy for the stochastic probing problem. Suppose we have an oracle to (approximately) find this feasible set S for probing constraints $\mathcal{J}$.

Assumption 5.4. *Suppose there exists an oracle that finds $S \in \mathcal{J}$ that β approximately maximizes the non-adaptive stochastic probing solution.*

The above assumption is justifiable as it is a constrained submodular maximization problem that we know how to approximately solve for some many constraint families $\mathcal{J}$, e.g., an ℓ-system.

Lemma 5.5 (Greedy Algorithm [FNW78, CCPV11])**.** *The greedy algorithm has an $(\ell+1)$-approximation for monotone submodular maximization over an ℓ-system.*

Finally, we need to show that given S there exists an efficient adaptive strategy in the PoI world with expected utility $\mathbb{E}[\max_{i\in S}\{Y_i^{\max}\}]$. But this is exactly the Pandora's box problem for which we know that Weitzman's index-based policy is optimal with expected utility $\mathbb{E}[\max_{i\in S}\{Y_i^{\max}\}]$. The above discussion can be summarized in the following theorem.

Theorem 5.6. *Given a β-approximation oracle for monotone submodular maximization over downward-closed constraints $\mathcal{J}$, there exists a 3β-approximation algorithm for constrained utility-maximization.*

Combining Lemma 5.5 and Theorem 5.6, we get Theorem 1.4 as a corollary.

5.2 Bounding the Adaptivity Gap

In this section we prove Lemma 5.3 by generalizing a similar result for Bernoulli variables of Gupta et al. [GNS17] to functions that are given by weighted matroid rank function. We do this by reducing the problem for discrete random variables to a Bernoulli setting.

Lemma 5.7. *The adaptivity gap for the stochastic probing problem for discretely distributed non-negative random variables is bounded by that for Bernoulli distributed non-negative random variables.*

Proof. Let us assume that each random variable Y_i takes value in a discrete set $\{v_1, v_2, \ldots, v_m\}$, where it takes value v_j w.p. p_j and $\sum_j p_j = 1$. (Note that p_j is a function of i but we don't write index i for ease of exposition.) Consider the optimal adaptive strategy decision tree $\mathcal{T}$ for the stochastic probing problem; here each node has at most m children. We modify $\mathcal{T}$ to obtain a binary decision tree $\mathcal{T}'$, which shows one can transform the instance to an instance with Bernoulli random variables and the same adaptivity gap. The idea is to replace every node i in $\mathcal{T}$ with m binary decision variables in $\mathcal{T}'$, where variable j is active w.p. $\frac{p_j}{1-\sum_{k<j} p_k}$. If active, variable j leads to the subtree corresponding to the case when Y_i take value v_j in $\mathcal{T}$. The two trees are equivalent because the probability that variable Y_i takes value v_j in $\mathcal{T}'$ is exactly

$$\prod_{j'<j} \left(1 - \frac{p_{j'}}{1-\sum_{k<j'} p_k}\right) \cdot \frac{p_j}{1-\sum_{k<j} p_k} \quad = \quad \prod_{j'<j} \left(\frac{1-\sum_{k\leq j'} p_k}{1-\sum_{k<j'} p_k}\right) \cdot \frac{p_j}{1-\sum_{k<j} p_k} \quad = \quad p_j.$$

□

To finish the proof of Lemma 5.3, we combine Lemma 5.7 with the following result of Gupta et al.

Lemma 5.8 ([GNS17])**.** *The adaptivity gap for the stochastic probing problem for Bernoulli random variables over any given downward-closed constraints is at most* 3.

5.3 An Application to the Set-Probing Utility-Maximization Problem

In this section we see an application of the constrained utility-maximization framework to the *set-probing utility-maximization* problem defined in Section 1.2. This problem is a generalization of Pandora's box where we pay a price to simultaneously find values of a set of random variables. We restate Theorem 1.5 for convenience.

Theorem 1.5. *The set-probing utility-maximization problem has a $3(\ell + 1)$-approximation efficient algorithm, where ℓ is the size of the largest set in $\mathcal{S}$. Moreover, no efficient algorithm can be $o(\ell/\log \ell)$-approximation, unless $P = NP$.*

The remaining section discusses the approximation algorithm. See the hardness proof in Section A.4.

We first observe that WLOG one can assume that the given sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ are downward-closed, i.e., if $S \in \mathcal{S}$ then any subset $T \subseteq S$ is also in $\mathcal{S}$. This is because a simple way to ensure downward-closedness is by adding every subset of S_j for the same price π_j into $\mathcal{S}$. Intuitively, this is equivalent to paying for the original set but choosing not to see the outcome of some of the random variables in it.

To construct our algorithm, we imagine solving a constrained utility-maximization problem. The random variables of this problem are indexed by sets $S \in \mathcal{S}$: variable X_S has value $\max_{i \in S}\{X_i\}$ and has price π_S. The problem is to adaptively probe some elements such that the sets corresponding to them are pairwise disjoint (a downward-closed constraint $\mathcal{J}$), while the goal is to maximize the utility that is given by max element value minus the total probing prices. Intuitively, the reason we need disjointness is to ensure independence between sets in our analysis as disjoint sets of random variables take values independently. We make the following simple observation.

Observation 5.9. *The optimal adaptive policy for this constrained utility-maximization problem with disjointness constraints is the same as the unconstrained set-probing utility-maximization problem.*

Given the above observation and noting that disjointness constraints are downward-closed, we want to use Theorem 5.6 to reduce our problem into a non-adaptive optimization problem. Although it appears that this is not possible because Theorem 5.6 is only for independent variables, and variables corresponding to non-disjoint sets are not independent. Fortunately, the proof of Theorem 5.6 only uses independence of random variables along any root-leaf path of the decision tree. Since our probing constraints ensure that the probed sets are disjoint, we get variables X_S along any root-leaf path to be independent, thereby allowing us to use Theorem 5.6. The final part in the proof of Theorem 1.5 is an approximation algorithm for this non-adaptive constrained utility-maximization problem.

Lemma 5.10. *There exists an efficient $(\ell + 1)$-approximation algorithm for the non-adaptive problem of finding a family $\mathcal{S}' \subseteq \mathcal{S}$ of disjoint sets to maximize $E[\max_{S \in \mathcal{S}'}\{Y_S^{\max}\}]$.*

Proof. Observe that the function $g(\mathcal{S}') = E[\max_{S \in \mathcal{S}'}\{Y_S^{\max}\}]$ is submodular. Also, the disjointness constraints can be viewed as an ℓ-system constraints since each set S has size at most ℓ. Thus we can view the non-adaptive problem as maximizing a submodular function over an ℓ-system, where we know by Lemma 5.5 that the greedy algorithm has an $(\ell + 1)$-approximation.

Moreover, to implement the greedy algorithm efficiently, we note that although $\mathcal{S}$ may contain an exponential number of elements, the initial set system $\mathcal{S}$ was polynomial sized (before we made $\mathcal{S}$ downward-closed). For sets A, B available at the same price, where $A \subseteq B$, it is obvious that the greedy algorithm will always choose B before A. Hence, at every step our greedy algorithm only needs to consider the original sets, which are only polynomial in number, and select the set with the best marginal value. Since, this can be done in polynomial time, this completes the proof of Theorem 1.5. $\square$

Acknowledgments We thank Anupam Gupta, Guru Prashanth, and Bobby Kleinberg for helpful comments on improving the presentation of this paper.

References

[Ada11] Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Inf. Process. Lett.*, 111(15):731–737, 2011.

[AGM15] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. *CoRR*, abs/1505.01439, 2015.

[AH15] Ali E Abbas and Ronald A Howard. *Foundations of decision analysis*. Pearson Higher Ed, 2015.

[AN16] Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 62(8):2374–2391, 2016.

[ANS08] Arash Asadpour, Hamid Nazerzadeh, and Amin Saberi. Stochastic submodular maximization. In *International Workshop on Internet and Network Economics*, pages 477–489. Springer, 2008. Full version appears as [AN16].

[ASW14] Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. In *STACS*, pages 29–40, 2014.

[BCN$^+$15] Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. Improved bounds in stochastic matching and optimization. In *APPROX*, pages 124–134, 2015.

[BG96] Ann Becker and Dan Geiger. Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996.

[BGK11] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *SODA*, pages 1647–1665, 2011.

[BGL$^+$12] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP Is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings. *Algorithmica*, 63(4):733–762, 2012.

[BN14] Nikhil Bansal and Viswanath Nagarajan. On the adaptivity gap of stochastic orienteering. In *IPCO*, pages 114–125, 2014.

[BYE81] Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

[BYGNR98] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM journal on computing*, 27(4):942–959, 1998.

[CCPV11] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[CFG+02] Moses Charikar, Ronald Fagin, Venkatesan Guruswami, Jon M. Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information. *J. Comput. Syst. Sci.*, 64(4):785–819, 2002.

[CHKK15] Yuxin Chen, S Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In *Conference on Learning Theory*, pages 338–363, 2015.

[CIK+09] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating Matches Made in Heaven. In *ICALP (1)*, pages 266–278, 2009.

[CJK+15] Yuxin Chen, Shervin Javdani, Amin Karbasi, J Andrew Bagnell, Siddhartha S Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *AAAI*, pages 3511–3518, 2015.

[DGV04] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 208–217. IEEE, 2004.

[DGV05] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *SODA*, pages 395–404, 2005.

[DTW03] Ioana Dumitriu, Prasad Tetali, and Peter Winkler. On playing golf with two balls. *SIAM Journal on Discrete Mathematics*, 16(4):604–615, 2003.

[FNW78] Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. An analysis of approximations for maximizing submodular set functionsii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

[GGM10] Ashish Goel, Sudipto Guha, and Kamesh Munagala. How to probe for an extreme value. *ACM Transactions on Algorithms (TALG)*, 7(1):12, 2010.

[Git74] John Gittins. A dynamic allocation index for the sequential design of experiments. *Progress in statistics*, pages 241–266, 1974.

[GK01] Anupam Gupta and Amit Kumar. Sorting and selection with structured costs. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 416–425. IEEE, 2001.

[GKMR11] Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *FOCS*, pages 827–836, 2011.

[GKNR12] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In *SODA*, 2012.

[GM07] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *STOC*, pages 104–113. 2007. Full version as: *Approximation Algorithms for Bayesian Multi-Armed Bandit Problems*, http://arxiv.org/abs/1306.3525.

[GM09] Sudipto Guha and Kamesh Munagala. Multi-armed bandits with metric switching costs. In *ICALP*, pages 496–507, 2009.

[GM12] Sudipto Guha and Kamesh Munagala. Adaptive uncertainty resolution in bayesian combinatorial optimization problems. *ACM Transactions on Algorithms (TALG)*, 8(1):1, 2012.

[GMS07] Sudipto Guha, Kamesh Munagala, and Saswati Sarkar. Information acquisition and exploitation in multichannel wireless systems. In *IEEE Transactions on Information Theory*. Citeseer, 2007.

[GN13] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *IPCO*, pages 205–216, 2013.

[GNS16] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1747. SIAM, 2016.

[GNS17] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity Gaps for Stochastic Probing: Submodular and XOS Functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1688–1702. SIAM, 2017.

[GW95] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[Has96] Johan Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 627–636. IEEE, 1996.

[HKT00] Magnús M. Halldórsson, Jan Kratochvıl, and Jan Arne Telle. Independent sets with domination constraints. *Discrete Applied Mathematics*, 99(1):39–54, 2000.

[HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Computational Complexity*, 15(1):20–39, 2006.

[Jen76] Thomas A. Jenkyns. The efficacy of the greedy algorithm. In *Proc. of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing*, pages 341–350, 1976.

[JMM$^+$03] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

[KH78] Bernhard Korte and Dirk Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.

[KK03] Sampath Kannan and Sanjeev Khanna. Selection with monotone comparison costs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 10–17. Society for Industrial and Applied Mathematics, 2003.

[KWW16] Robert Kleinberg, Bo Waggoner, and Glen Weyl. Descending Price Optimally Coordinates Search. *arXiv preprint arXiv:1603.07682*, 2016.

[LY13] Jian Li and Wen Yuan. Stochastic combinatorial optimization via poisson approximation. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 971–980, 2013.

[Ma14] Will Ma. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms: Extended abstract. In *SODA*, pages 1154–1163, 2014.

[Wei79] Martin L. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pages 641–654, 1979.

[WS11] David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

A Illustrative Examples

A.1 Why the naïve greedy algorithm fails for Pandora's box

Suppose $curr$ denotes the maximum value in the currently opened set of boxes. The naïve greedy algorithm selects in any step the unopened box j corresponding to the maximum marginal value, i.e. $\mathrm{argmax}\{\mathbb{E}[(X_j - curr)^+] - \pi_j\}$, and opens it if its marginal value is non-negative. The algorithm stops probing when every unopened box has a negative marginal value. We give an example where this algorithm can be made arbitrarily worse as compared to the optimal algorithm.

Consider $n-1$ iid boxes, each taking value $1/p^2$ w.p. p and 0 otherwise, where $p < 1$. The probing price of each of these boxes is 1. Also, there is a box which takes value $1/p^2$ w.p. 1 but has a probing price of $1/p^2 - 1/p + 1$. Note that in the beginning, the marginal value of every box is $1/p - 1$. Now the optimal strategy is to probe the boxes with price 1, until we see a box with value $1/p^2$. For large enough n, the expected utility of this strategy is $\approx 1/p^2 - 1/p$ because in expectation the algorithm stops after roughly $1/p$ probes. However, the naïve greedy algorithm will open the box with price $1/p^2 - 1/p - 1$ and then stop probing. Thus its expected utility will be $1/p - 1$. By choosing small enough p, the ratio between $1/p^2 - 1/p$ and $1/p - 1$ can be made arbitrarily large.

A.2 The Pandora's box problem has no constant approximation non-adaptive solution

Consider an example where each element independently takes value 1 w.p. p ($\ll 1$) and value 0, otherwise. Suppose the price of probing any element is $1 - \epsilon$, for some small $\epsilon > 0$. The optimal adaptive strategy is to continue probing till we see an element with value 1. Assuming n to be large, it is easy to see that this strategy has expected value $\approx \epsilon/p$. On the other hand, a non-adaptive strategy has to decide in the beginning which all elements S to probe, and then probe them irrespective of the consequence. Since all items are identical, the only decision it has to make is how many items to probe. One can verify that no such non-adaptive strategy can get value more than $O(\epsilon)$. By choosing p to be small enough, we can make the gap arbitrarily large.

A.3 Hardness for general submodular functions

To prove that one cannot obtain good approximation results for any monotone submodular functions f, we show that even when all variables are deterministic, the computational problem of selecting the best set $\mathbb{I} \subseteq V$ to maximize $f(\mathbb{I}) - \pi(\mathbb{I})$ is $\tilde{\Omega}(\sqrt{n})$ hard assuming $P \neq NP$, where $n = |V|$. The idea is to reduce from set packing. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ denote the sets of a set packing instance. For $S \subseteq \mathcal{S}$, let $f(S)$ denote the covering function. Let price of probing S_i be $\pi_i = |S_i| - 1$. Clearly, it doesn't make sense to probe sets that are not disjoint as the marginal utility will be non-positive. The optimal solution therefore equals the maximum number of disjoint sets. But no polynomial time algorithm can be $O(n^{1/2-\epsilon})$-approximation, for any $\epsilon > 0$, unless $P = NP$ [Has96, HKT00].

A.4 Hardness for the set-probing problem

To prove that no polynomial time algorithm for the set-probing problem can be $o(\ell/\log \ell)$-approximation, unless $P = NP$, we reduce ℓ-set packing problem into an instance of the set-probing problem. Given an ℓ-set packing instance with sets $S_1, S_2, \ldots, S_m$, each of size ℓ, we create the following set-probing problem. For every element $i \in \bigcup_j S_j$, w.p. $\frac{1}{n^3}$ variable X_i takes value 1, and is 0, otherwise. Also, let each set S_j have price $\frac{(\ell-0.5)}{n^3}$.

Since probability of two elements taking value 1 is really small $O(1/n^4)$, we see that it only makes sense to probe sets where none of the elements have been already probed: even if a single element is probed before, expected value from probing the set is at most $\frac{(\ell-1)}{n^3} - \frac{(\ell-0.5)}{n^3} = \frac{-0.5}{n^3} < 0$. Hence, $E[Opt] =$ (Max # disjoint sets) $\cdot \frac{0.5}{n^3}$. But this is exactly the ℓ-set packing problem and we know that unless $P = NP$, no poly time algorithm can be $o(\ell/\log \ell)$-approximation [HSS06].