Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Adaptive Submodular Ranking and Routing

Fatemeh Navidi, Prabhanjan Kambadur, Viswanath Nagarajan

To cite this article:
Fatemeh Navidi, Prabhanjan Kambadur, Viswanath Nagarajan (2020) Adaptive Submodular Ranking and Routing. Operations Research 68(3):856-877. https://doi.org/10.1287/opre.2019.1889

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 68, No. 3, May–June 2020, pp. 856–877
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Adaptive Submodular Ranking and Routing

Fatemeh Navidi,[a] Prabhanjan Kambadur,[b] Viswanath Nagarajan[a]

[a] Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109; [b] Bloomberg LP, New York, New York 10022

Contact: navidi@umich.edu (FN); pkambadur@bloomberg.net (PK); viswa@umich.edu, https://orcid.org/0000-0002-9514-5581 (VN)

Received: May 14, 2018
Revised: December 28, 2018
Accepted: May 14, 2019
Published Online in Articles in Advance: April 7, 2020

Subject Classifications: analysis of algorithms: suboptimal algorithms; networks/graphs: stochastic
Area of Review: Optimization

https://doi.org/10.1287/opre.2019.1889

Abstract. We study a general stochastic ranking problem in which an algorithm needs to adaptively select a sequence of elements so as to "cover" a random scenario (drawn from a known distribution) at minimum expected cost. The coverage of each scenario is captured by an individual submodular function, in which the scenario is said to be covered when its function value goes above a given threshold. We obtain a logarithmic factor approximation algorithm for this adaptive ranking problem, which is the best possible (unless $P = NP$). This problem unifies and generalizes many previously studied problems with applications in search ranking and active learning. The approximation ratio of our algorithm either matches or improves the best result known in each of these special cases. Furthermore, we extend our results to an adaptive vehicle-routing problem, in which costs are determined by an underlying metric. This routing problem is a significant generalization of the previously studied adaptive traveling salesman and traveling repairman problems. Our approximation ratio nearly matches the best bound known for these special cases. Finally, we present experimental results for some applications of adaptive ranking.

Funding: V. Nagarajan and F. Navidi were supported in part by the National Science Foundation Division of Computing and Communication Foundations [CAREER Grant CCF-1750127].

Keywords: submodularity • stochastic optimization • approximation algorithms

1. Introduction

Many stochastic optimization problems can be viewed as sequential decision processes of the following form. There is a known distribution $\mathcal{D}$ over a set of scenarios, and the goal is to cover an unknown realized scenario i^* drawn from $\mathcal{D}$. In each step, an algorithm chooses an element that partially covers i^* and receives some feedback from that element. This feedback is then used to update the distribution over scenarios (using conditional probabilities). So any solution in this setting is an adaptive sequence of elements. The objective is to minimize the expected cost incurred to cover the realized scenario i^*.

Furthermore, many different criteria to cover a scenario can be modeled as covering a suitable submodular function. Submodular functions are widely used in many domains, for example, game theory, social networks, search ranking, and document summarization; see Shapley (1971), Lin and Bilmes (2011), Prasad et al. (2014), and Kempe et al. (2015).

As an example of the class of problems that we address, consider a medical diagnosis application. There is a patient with an unknown disease, and there are several possible tests that can be performed. Each test has a certain cost, and its outcome (feedback) can be used to restrict the set of possible diseases. There are also a priori probabilities associated with each disease. The task here is to obtain an adaptive sequence of tests so as to identify the disease at minimum expected cost.

As another example, consider a search engine application. On any query, different user types are often interested in viewing different search results. Each user type is associated with the set of results the user type is interested in and a threshold number of results the user type would like to see. There is also a probability distribution over user types. After displaying each result (or a block of a small number of results), the search engine receives feedback on which of those results were of interest to the realized user type. The goal is to provide an adaptive sequence of results so as to minimize the expected number of results until the user type is satisfied.

Yet another example arises in route planning for disaster management. After a major disaster such as an earthquake, normal communication networks are usually unavailable. So rescue operators would not know the precise locations of victims before actually visiting them. However, probabilistic information is often available based on geographical data, etc. Then the task is to plan an adaptive route for a rescue vehicle that visits all the victims within a minimum expected time.

In this paper, we study an abstract stochastic optimization problem in the setting described that unifies and generalizes many previously studied problems, such as *optimal decision trees* (ODTs) studied in Hyafil and Rivest (1976), Kosaraju et al. (1999), Dasgupta (2004), Chakaravarthy et al. (2011), Cicalese et al. (2014), and Gupta et al. (2017); *equivalence class determination* (see Golovin et al. 2010 and Bellala et al. 2012); *decision region determination* studied in Javdani et al. (2014); and *submodular ranking* studied in Azar and Gamzu (2011) and Im et al. (2016). We obtain an algorithm with the best possible approximation guarantee in all these special cases. We also obtain the first approximation algorithms for some other natural problems that are captured by our framework, such as stochastic versions of knapsack cover and matroid basis with correlated distributions. Moreover, our algorithm is very simple to state and implement. We also present experimental results on the optimal decision tree problem, and our algorithm performs very well.

We extend our framework to a vehicle-routing setting as well, in which the elements are located in a metric and the cost corresponds to travel distance/time between these locations. As special cases, we recover the adaptive traveling salesman (TSP) and repairman (TRP) problems that were studied in Gupta et al. (2017). Our approximation ratio almost matches the best result known for these special cases. Our approach has the advantage of being able to solve a more general problem while allowing for a simpler analysis. We note that submodular objectives are also commonly utilized in vehicle-routing problems; see Chekuri and Pal (2005) and references therein for theoretical work and Singh et al. (2009) for applications in information acquisition and robotics.

For some stochastic optimization problems, one can come up with approximately optimal solutions using static (nonadaptive) solutions that are insensitive to the feedback obtained; see, for example, stochastic (maximization) knapsack in Dean et al. (2008) and stochastic matching in Bansal et al. (2012). However, this is not the case for the adaptive submodular ranking problem. For all the special cases mentioned, there are instances in which the optimal adaptive value is much less than the optimal nonadaptive value. Thus, it is important to come up with an adaptive algorithm.

1.1. Adaptive Submodular Ranking

We start with some basics. A set function $f : 2^U \to \mathbb{R}_+$ on ground set U is said to be submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all $A, B \subseteq U$. The function f is said to be monotone if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq U$. We assume that set functions are given in the standard *value oracle* model; that is, we can evaluate $f(S)$ for any $S \subseteq U$ in polynomial time. To reduce notation, for any subset $S \subseteq U$ and element $e \in U$, we use $f(S \cup e) = f(S \cup \{e\})$.

In the adaptive submodular ranking problem (ASR), we have a ground set U of n *elements* with positive costs $\{c_e\}_{e \in U}$. We also have m *scenarios* with a probability distribution $\mathcal{D}$ given by probabilities $\{p_i\}_{i=1}^m$ totaling to one. Each scenario $i \in [m] := \{1, \cdots, m\}$ is specified by

i. A *monotone submodular* function $f_i : 2^U \to [0, 1]$, where $f_i(\emptyset) = 0$ and $f_i(U) = 1$ (any monotone submodular function can be expressed in this form by scaling).

ii. A *feedback* function $r_i : U \to G$, where G is a set of possible feedback values.

We note that f_i and r_i need not be related in any way: this flexibility allows us to capture many different applications. Scenario $i \in [m]$ is said to be *covered* by any subset $S \subseteq U$ of elements such that $f_i(S) = 1$. The goal in ASR is to adaptively find a sequence of elements in U that minimizes the expected cost to cover a *random scenario* i^* drawn from $\mathcal{D}$. The identity of i^* is initially unknown to the algorithm. When the algorithm selects an element $e \in U$, it receives some feedback value $g = r_{i^*}(e) \in G$, which can be used to update the probability distribution of i^* using conditional probabilities. In particular, the probability of any scenario $i \in [m]$ with $r_i(e) \neq g$ would become zero. The sequence of selected elements is *adaptive* because it depends on the feedback received.

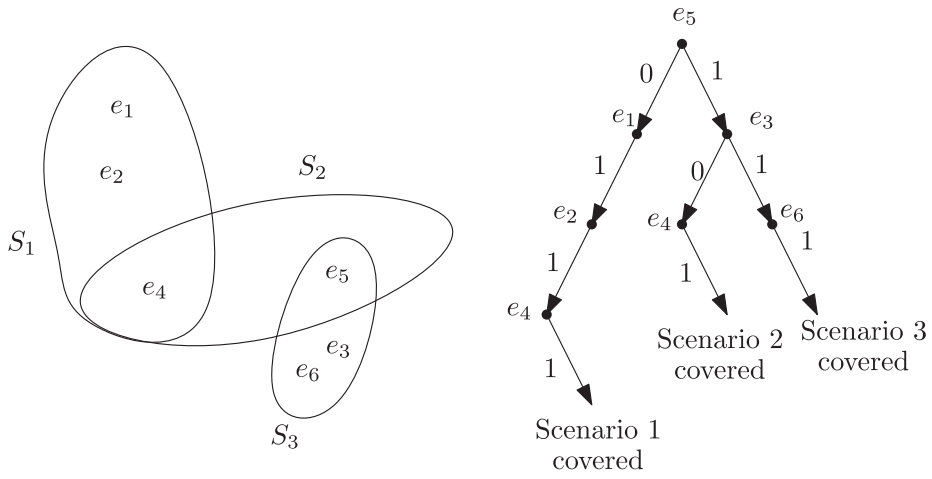
Example 1. Figure 1 demonstrates an example for ASR. In this example, we have elements $U = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ and three scenarios. Each element has cost 1, and there is a uniform probability distribution over scenarios. Each senario $i \in \{1, 2, 3\}$ is associated with a subset S_i with submodular function $f_i(S) = \frac{|S \cap S_i|}{|S_i|}$ and binary feedback function $r_i(e) = \mathbb{1}[e \in S_i]$. So the realized scenario i^* is covered with subset $S \subseteq U$ if and only if $S_{i^*} \subseteq S$. And the feedback from an element e is one if and only if $e \in S_{i^*}$. The decision tree in Figure 1 represents a feasible solution with expected cost $\frac{1}{3} \cdot 4 + \frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 3 = \frac{10}{3}$.

A solution to ASR is represented by a decision tree $\mathcal{T}$, where each node is labeled by an element $e \in U$, and the branches out of such a node are labeled by the possible feedback we can receive after selecting e. Each node in $\mathcal{T}$ also corresponds to a *state* that is specified by the set E of previously selected elements and the feedback $\theta_e \in G$ of each $e \in E$. From this information, we can obtain a more abbreviated version of the state as (E, H), where H denotes the set of uncovered and compatible scenarios based on the observed feedback. Formally,

$$H = \{i \in [m] \ : \ f_i(E) < 1, \ r_i(e) = \theta_e \ \text{ for all } e \in E\}.$$

Every scenario $i \in [m]$ traces a root–leaf path in the decision tree $\mathcal{T}$, which, at any node labeled by element $e \in U$, takes the branch labeled by feedback $r_i(e)$.

Figure 1. An Example for ASR and a Feasible Solution



Let T_i denote the sequence of elements on this path. In a feasible decision tree $\mathcal{T}$, each scenario $i \in [m]$ must be covered, that is, $f_i(T_i) = 1$. The cost $C_{\mathcal{T}}(i)$ of $\mathcal{T}$ under scenario i is the total cost of the shortest prefix $\overline{T}_i$ of T_i such that $f_i(\overline{T}_i) = 1$. The objective in ASR is to minimize the expected cost $\sum_{i=1}^{m} p_i \cdot (\sum_{e \in \overline{T}_i} c_e)$. We emphasize that multiple scenarios may trace the same path in $\mathcal{T}$: in particular, it is *not* necessary to identify the realized scenario i^* in order to cover it.

We also note that cost is only incurred until the realized scenario i^* gets covered even though the algorithm may not know this. In applications in which scenarios correspond to users and the goal is to minimize cost incurred by the users, this is the natural definition. An example is the *multiple intent reranking* problem, which models the search engine application (see Section 4.2). However, in some other applications (such as optimal decision tree), we are interested in algorithms that know exactly when to stop. For the applications that we consider, it turns out that this is still possible using the preceding definition; see Section 4 for details.

An important parameter in the analysis of our algorithm is the following:

$$\epsilon \quad := \quad \min_{\substack{e \in U : f_i(S \cup e) > f_i(S) \\ i \in [m],\, S \subseteq U}} \quad f_i(S \cup e) - f_i(S). \tag{1}$$

It measures the minimum positive incremental value of any element. Such a parameter appears in all results on the submodular cover problem, for example, Wolsey (1982) and Azar and Gamzu (2011).

1.2. Adaptive Submodular Routing

In the adaptive submodular routing problem (ASP), we have a ground set U of n elements that are located at vertices of a metric $(U \cup \{s\}, d)$, where s is a specified root vertex. Here, $d : U \times U \to \mathbb{R}_+$ is a cost function that is symmetric (i.e., $d(x, y) = d(y, x)$ for all $x, y \in U$) and satisfies triangle inequality (i.e., $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in U$). We use the terms "element" and "vertex" interchangeably. As before, we have m scenarios with a probability distribution $\mathcal{D}$ given by probabilities $\{p_i\}_{i=1}^{m}$ totaling to one, and each scenario $i \in [m]$ is associated with functions f_i and r_i. A feasible solution to ASP can again be represented by a decision tree $\mathcal{T}$, at the end of which each scenario is covered. Note that, in the actual solution, we need to return to the root s after visiting the last vertex in $\mathcal{T}$. For any scenario i, let τ_i denote the root–leaf path traced in decision tree $\mathcal{T}$ and let π_i denote the shortest prefix of τ_i such that $f_i(\pi_i) = 1$. The cost $C_{\mathcal{T}}(i)$ of $\mathcal{T}$ under scenario i is the total cost of path π_i. Specifically, if $\pi_i = s, e_1, e_2, \cdots e_k$, the cost under scenario i would be $C_{\mathcal{T}}(i) = d(s, e_1) + \sum_{i=1}^{k-1} d(e_i, e_{i+1})$. The objective is to minimize the expected cost $\sum_{i=1}^{m} p_i \cdot C_{\mathcal{T}}(i)$. As with ASR, cost in ASP is only incurred until the realized scenario i^* is covered.

This problem differs from ASR only in the definition of the cost: here, we want to minimize the expected metric cost of the walk that covers i^*. Note also that ASP generalizes ASR (at the loss of factor 2 in the approximation ratio). To see this, for any ASR instance, consider the ASP instance on the metric $(U \cup \{s\}, d)$ induced by a star with center s and leaves U, where $d(s, e) = c_e$ for all $e \in U$.

1.3. Results

Our main result is an $O(\log \frac{1}{\epsilon} + \log m)$-approximation algorithm for ASR in which $\epsilon > 0$ is as defined in (1) and m is the number of scenarios. Assuming $P \neq NP$, this result is asymptotically the best possible even when $m = 1$. This is because the set cover problem on k elements is a special case of ASR with $m = 1$ and parameter $\epsilon = 1/k$, and Dinur and Steurer (2014) showed that approximating set cover to within a $(1-\delta)\ln k$ factor (for any $\delta > 0$) is NP-hard. Our algorithm is a simple adaptive, greedy-style algorithm. At each step,

we assign a score to each remaining element and select the element with maximum score. Such a simple algorithm was previously unknown even in the special case of an optimal decision tree despite a large number of papers on this topic, including Hyafil and Rivest (1976), Kosaraju et al. (1999), Dasgupta (2004), Guillory and Bilmes (2009), Chakaravarthy et al. (2011), Golovin and Krause (2011), Adler and Heeringa (2012), Cicalese et al. (2014), and Gupta et al. (2017).

For ASP, we provide an $O(\log^{2+\delta} n \cdot (\log \frac{1}{\epsilon} + \log m))$-approximation algorithm in which $\delta > 0$ is any fixed constant and ϵ is as defined in (1). This algorithm utilizes some ideas from the algorithm for ASR and involves combining a number of smaller tours into the final solution. We also make use of an algorithm for the (deterministic) submodular orienteering problem in a black box fashion. Our result is nearly the best possible because the group Steiner problem studied in Garg et al. (2000) is a special case of ASP with $m = 1$ and parameter $\epsilon = 1/k$, where k denotes the number of groups. There is an $\Omega(\log^{2-\delta} n)$-factor hardness of approximation for group Steiner by Halperin and Krauthgamer (2003), and the best approximation ratio known is $O(\log^2 n \cdot \log k)$ from Garg et al. (2000).

We show that our framework is widely applicable by demonstrating a number of previously studied stochastic optimization problems as special cases. In many cases, we match or improve upon prior approximation guarantees. We also obtain the first approximation algorithms for some other stochastic problems. More details on these applications can be found in Section 4.

Finally, in Section 5, we provide computational results for the optimal decision tree problem (and its generalized version). We use a data set arising in the identification of toxic chemicals based on binary symptoms. Our algorithm performs very well compared with some other natural algorithms.

1.3.1. Outline of Key Techniques.

Our algorithm for ASR involves repeatedly selecting an element that maximizes a combination of (i) the expected increase in function value relative to the target of one and (ii) a measure of gain in identifying the realized scenario. See Equation (2) for the formal selection criterion. Our analysis provides new ways of reasoning about adaptive decision trees. At a high level, our approach is similar to that for the minimum latency TSP in Blum et al. (1994) and Chaudhuri et al. (2003). We upper bound the probability that the algorithm incurs a certain power-of-two cost 2^k in terms of the probability that the optimal solution incurs cost $2^k/\alpha$, which is then used to establish an $O(\alpha)$ approximation ratio. Our main technical contribution is in relating these completion probabilities in the algorithm and the optimal solution (see Lemma 2). In particular, a key step in our proof is a coupling of "bad" states in the algorithm (in which the gain in terms of our selection criterion is small) with bad states in the optimum (in which the cost incurred is high). This is reflected in the classification of the algorithm's states as good/okay/bad (Definition 1), and the proof that the *expected* gain of the algorithm is large (Lemma 3). Our algorithm and analysis for the ASP are along similar lines.

1.4. Related Works

The basic submodular cover problem (select a min-cost subset of elements that covers a given submodular function) was first considered by Wolsey (1982), who proved that the natural greedy algorithm is a $(1 + \ln \frac{1}{\epsilon})$-approximation algorithm. This result is tight because set cover is a special case. The submodular cover problem corresponds to the special case of ASR with $m = 1$.

The deterministic submodular ranking problem was introduced by Azar and Gamzu (2011), who obtained an $O(\log \frac{1}{\epsilon})$-approximation algorithm when all costs are unit. This is a special case of ASR when there is no feedback (i.e., $G = \emptyset$) and costs are uniform; note that the optimal ASR solution in this case is just a fixed sequence of elements. The result in Azar and Gamzu (2011) was based on an interesting "reweighted" greedy algorithm: the second term in our selection criterion (2) is similar to this. A different proof of the submodular ranking result, using a min-latency type analysis, was obtained in Im et al. (2016), which also implied an $O(\log \frac{1}{\epsilon})$-approximation algorithm with nonuniform costs. We also use a min-latency type analysis for ASR.

The first $O(\log m)$-approximation algorithm for an optimal decision tree was obtained in Gupta et al. (2017), which is known to be best possible from Chakaravarthy et al. (2011). This result was extended to the equivalence class determination problem in Cicalese et al. (2014). Previous results based on a simple greedy "splitting" algorithm had a logarithmic dependence on either costs or probabilities, which can be exponential in m; see Kosaraju et al. (1999), Dasgupta (2004), Guillory and Bilmes (2009), Chakaravarthy et al. (2011), and Adler and Heeringa (2012). The algorithms in Gupta et al. (2017), and Cicalese et al. (2014) were significantly more complex than what we obtain here as a special case of ASR. In particular, these algorithms proceeded in $O(\log m)$ phases, each of which required solving an auxiliary subproblem that reduced the number of possible scenarios by a constant factor. Using such a "phase-based" algorithm and analysis for the general ASR problem only leads to an $O(\log m \cdot \log \frac{1}{\epsilon})$-approximation ratio because the subproblem to be solved in each phase is submodular ranking, which only has an $O(\log \frac{1}{\epsilon})$-approximation ratio. Our work is based on a much

simpler greedy-style algorithm and a new analysis, which leads to the $\mathbb{O}(\log m + \log 1/\epsilon)$ approximation ratio.

A different stochastic version of submodular ranking was considered in Im et al. (2016), in which (i) the feedback was independent across elements and (ii) all the submodular functions needed to be covered. In contrast, ASR involves a correlated scenario-based distribution, and only the submodular function of the "realized" scenario i^* needs to be covered. Because of these differences, both the algorithm and analysis for ASR are different from Im et al. (2016): our selection criterion (2) involves an additional "information gain" term, and our analysis requires a lot more work in order to handle correlations. We note that, unlike ASR, the stochastic submodular ranking problem in Im et al. (2016) does not capture the optimal decision tree problem and its variants (equivalence class, decision region determination, etc.).

For some previous special cases of ASR studied in Golovin et al. (2010), Bellala et al. (2012), and Javdani et al. (2014), one could obtain approximation algorithms via the framework of "adaptive submodularity" introduced by Golovin and Krause (2011). However, this approach does not apply to the general ASR problem, and the approximation ratio obtained is at least $\Omega(\log^2 1/p_{min})$, where $p_{min} = \min_{i=1}^{m} p_i$ can be exponentially small in m. We note that the original paper by Golovin and Krause (2011) claimed an $O(\log 1/p_{min})$ bound, which was found to be erroneous by Nan and Saligrama (2017); an updated version in Golovin and Krause (2017) addresses this error but only obtains an $O(\log^2 1/p_{min})$ bound. So, even in the case of uniform probabilities, our result provides an improved $O(\log m)$ approximation ratio compared with the $O(\log^2 m)$ ratio from Golovin and Krause (2017). We also note that our analysis is based on a completely different approach, which might be of independent interest.

Recently, Grammel et al. (2016) considered the scenario submodular cover problem, which can also be seen as a special case of ASR. This involves a *single* integer-valued submodular function for all scenarios, which is defined on an expanded ground set $U \times G$ (i.e., pairs of "element, feedback" values). For this problem, our algorithm matches (in fact, improves slightly) the approximation ratio in Grammel et al. (2016) with a much simpler algorithm and analysis. We note that ASR is strictly more general than scenario submodular cover. For example, deterministic submodular ranking studied in Azar and Gamzu (2011) is a special case of ASR but not of scenario submodular cover.

A special case of the ASP, the *adaptive TSP* was studied in Gupta et al. (2017), in which the goal is to visit vertices in a random demand set. Gupta et al. (2017) obtained an $O(\log^2 n \cdot \log m)$-approximation algorithm for adaptive TSP and showed that any improvement on this would translate to a similar improvement for the group Steiner problem, which is a long-standing open question. Although our approximation ratio for ASP is slightly worse, it is much more general and involves a simpler analysis. For example, using ASP, we can directly obtain an approximation algorithm for the variant of adaptive TSP in which only a target number of demand vertices need to be visited.

A problem formulation similar to ASP was also studied in Lim et al. (2015), where approximation algorithms were obtained under certain technical assumptions on the underlying submodular functions and probability distribution. To the best of our knowledge, the approach in Lim et al. (2015) is not applicable to the general ASP problem considered here.

2. Algorithm for Adaptive Submodular Ranking

Recall that the state of our algorithm (i.e., any node in its decision tree) can be represented by (E, H), where (i) $E \subseteq U$ is the set of previously selected elements and (ii) $H \subseteq [m]$ is the set of scenarios that are compatible with feedback (on E) received so far and are still uncovered.

At each state (E, H), our algorithm selects an element that maximizes the value computed in Equation (2). This can be viewed as the cost-effectiveness of any element e: the terms inside the parantheses measure the gain from element e, and this gain is normalized by the element's cost c_e. The gain of any element e comes from two sources:

1. *Information gain:* this corresponds to the first term in (2). Note that the feedback from element e can be used to define a partition of the scenarios in H, where all scenarios in a part have the same feedback from e. Then, subset $L_e(H)$ is defined to be the complement of the largest cardinality part; note that each part within $L_e(H)$ has size at most $|H|/2$. If the realized scenario happens to be in $L_e(H)$, then we make good progress in identifying the scenario: this is because the number of compatible scenarios decreases by (at least) a factor of two. The term $\sum_{j \in L_e(H)} p_j$ in (2) is just the probability that the realized scenario is in $L_e(H)$.

2. *Function coverage:* this corresponds to the second term in (2) and is based on the algorithm for *deterministic* submodular ranking from Azar and Gamzu (2011). An important point here is that we consider the relative gain of each function f_i (for $i \in H$), which is the ratio of the increase in function value (i.e., $f_i(e \cup E) - f_i(E)$) to the remaining target (i.e., $1 - f_i(E)$), rather than just the absolute increase.

Algorithm 1 gives a formal description. Note that we may not incur the cost for all selected elements

under scenario i^* as the cost is only considered up to the point at which i^* is covered.

Algorithm 1 (ASR Algorithm)

1. $E \leftarrow \emptyset$, $H \leftarrow [m]$.
2. **while** $H \neq \emptyset$ **do**
3. For any element $e \in U$, let $B_e(H)$ denote the set with maximum cardinality among
$$\{i \in H : r_i(e) = t\}, \quad \text{for } t \in G.$$
Define $L_e(H) = H \setminus B_e(H)$.
4. Select element $e \in U \setminus E$ that maximizes
$$\frac{1}{c_e} \cdot \left(\sum_{j \in L_e(H)} p_j + \sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} \right). \tag{2}$$
5. $E \leftarrow E \cup \{e\}$.
6. Remove incompatible and covered scenarios from H based on the feedback from e.
7. end.
8. Output E.

Note that H only contains uncovered scenarios. So, for all $i \in H$, we have $f_i(E) < 1$, and the denominator in Equation (2) is always positive. We have the following theorem:

Theorem 1. *Algorithm 1 is an $\mathbb{O}(\log 1/\epsilon + \log m)$-approximation algorithm for ASR.*

Now, we analyze the performance of this algorithm. For any subset $T \subseteq [m]$ of scenarios, we use $\Pr(T) = \sum_{i \in T} p_i$. For any subset $W \subseteq U$ of elements, we use $c(W) = \sum_{e \in W} c_e$. Let OPT denote an optimal solution to the ASR instance and ALG be the solution found by the algorithm. Set $L := 15(1 + \ln 1/\epsilon + \log_2 m)$, and its choice will be clear later. We refer to the total cost incurred until any point in a solution as the *time*. We assume (without loss of generality by scaling) that all costs are at least one. For any $k = 0, 1, \cdots$, we define the following quantities:

- A_k is the set of uncovered scenarios of ALG at time $L \cdot 2^k$, and $a_k = \Pr(A_k)$. More formally, we have $A_k = \{i \in [m] : C_{ALG}(i) \geq L \cdot 2^k\}$, where $C_{ALG}(i)$ is the cost of scenario i in ALG.
- X_k is the set of uncovered scenarios of OPT at time 2^{k-1}, and $x_k = \Pr(X_k)$. That is, we have $X_k = \{i \in [m] : C_{OPT}(i) \geq 2^{k-1}\}$, where $C_{OPT}(i)$ is the cost of scenario i in OPT. Note that $x_0 = 1$.

To keep things simple, we assume that all costs are integers. However, the analysis extends directly to the case of noninteger costs by replacing summations (over time t) with integrals.

Lemma 1. *The expected cost of ALG and OPT can be bounded as follows:*

$$C_{ALG} \leq L \sum_{k \geq 0} 2^k a_k + L \quad \text{and} \quad C_{OPT} \geq \frac{1}{2} \sum_{k \geq 0} 2^{k-1} x_k. \tag{3}$$

Proof of Lemma 1. In ALG, for all $k \geq 1$, the probability of scenarios for which the cover time is in $[2^{k-1}L, 2^k L)$ is equal to $a_{k-1} - a_k$. So we have

$$\begin{aligned} C_{ALG} &= \sum_{i \in [m]} p_i \cdot C_{ALG}(i) = \sum_{k \geq 1} \sum_{i \in A_{k-1} \setminus A_k} p_i \cdot C_{ALG}(i) \\ &\leq \sum_{k \geq 1} \sum_{i \in A_{k-1} \setminus A_k} p_i \cdot 2^k L \\ &\leq \sum_{k \geq 1} 2^k L(a_{k-1} - a_k) + L(1 - a_0) \\ &= \sum_{k \geq 1} 2^k L a_{k-1} - \sum_{k \geq 1} 2^k L a_k + L(1 - a_0) \\ &= 2 \sum_{k \geq 0} 2^k L a_k - \left(\sum_{k \geq 0} 2^k L a_k - L a_0 \right) + L(1 - a_0) \\ &= \sum_{k \geq 0} 2^k L a_k + L. \end{aligned}$$

On the other hand, in OPT, for all $k \geq 1$, the probability of scenarios for which the cover time is in $[2^{k-2}, 2^{k-1})$ is equal to $x_{k-1} - x_k$. So we have

$$\begin{aligned} C_{OPT} &= \sum_{i \in [m]} p_i \cdot C_{OPT}(i) = \sum_{k \geq 1} \sum_{i \in X_{k-1} \setminus X_k} p_i \cdot C_{OPT}(i) \\ &\geq \sum_{k \geq 1} \sum_{i \in X_{k-1} \setminus X_k} p_i \cdot 2^{k-2} \geq \sum_{k \geq 1} 2^{k-2}(x_{k-1} - x_k) \\ &= \sum_{k \geq 1} 2^{k-2} x_{k-1} - \sum_{k \geq 1} 2^{k-2} x_k \\ &= \sum_{k \geq 0} 2^{k-1} x_k - \frac{1}{2} \left(\sum_{k \geq 0} 2^{k-1} x_k - \frac{1}{2} \right) \geq \frac{1}{2} \sum_{k \geq 0} 2^{k-1} x_k. \end{aligned}$$

We used the fact that $x_0 = 1$. □

Thus, if we could upper bound each a_k by some multiple of x_k, it would be easy to obtain the approximation factor. However, this is not the case, and instead we prove Lemma 2.

Lemma 2. *For all $k \geq 1$, we have $a_k \leq 0.2a_{k-1} + 3x_k$.*

Using this lemma, we can prove Theorem 1.

Proof of Theorem 1. Let $Q = \sum_{k \geq 0} L \cdot 2^k a_k + L$, which is the bound on C_{ALG} from (3). Using Lemma 2,

$$\begin{aligned} Q &\leq L \cdot \sum_{k \geq 1} 2^k (0.2a_{k-1} + 3x_k) + L(a_0 + 1) \\ &\leq 0.4L \cdot \sum_{k \geq 0} 2^k a_k + 6L \cdot \sum_{k \geq 1} 2^{k-1} x_k + L(a_0 + 1) \\ &\leq 0.4(Q - L) + 6L \left(\sum_{k \geq 0} 2^{k-1} x_k - \frac{x_0}{2} \right) + 2L \\ &\leq 0.4 \cdot Q + 12L \cdot C_{OPT}. \end{aligned} \tag{4}$$

The first inequality in (4) is by definition of Q and $a_0 \leq 1$, and the second inequality uses the bound on C_{OPT} from (3). Finally, we have $Q \leq 20L \cdot C_{OPT}$.

Because $L = 15(1 + \ln 1/\epsilon + \log m)$ and $C_{ALG} \le Q$, we obtain the theorem. □

2.2. Proof of Lemma 2

We now prove Lemma 2 for a fixed $k \ge 1$. Consider any time t between $L \cdot 2^{k-1}$ and $L \cdot 2^k$. Note that ALG's decision tree induces a partition of all the uncovered scenarios at time t, where each part H consists of all scenarios that are at a particular state (E, H) at time t. Let $R(t)$ denote the set of parts in this partition. We also use $R(t)$ to denote the collection of states corresponding to these parts. Note that all scenarios in A_k appear in $R(t)$ as these scenarios are uncovered even at time $L \cdot 2^k \ge t$. Similarly, all scenarios in $R(t)$ are in A_{k-1}. See Figure 3.

Proposition 1. *Consider any state (E, H) and element $e \in E$. Then* (i) *the feedback values $\{r_i(e) : i \in H\}$ are all identical, and* (ii) $L_e(H) = \emptyset$.

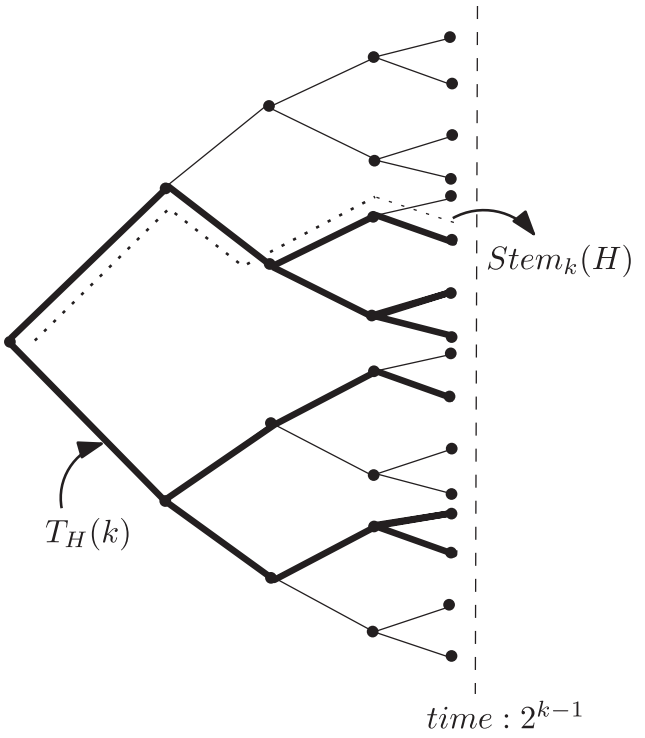
Proof of Proposition 1. Note that, by definition, at state (E, H), all scenarios in H are compatible with the feedback received from elements in E. Thus, all of them should have the same feedback for any element in E. Furthermore, for any $e \in E$, $L_e(H)$ is the complement of the largest part in the partition of H based on element e's feedback. According to the fact that all scenarios in H have the same feedback for element e, they are all in the same part, which is the largest part. So the complement of the largest part of the partition, which is $L_e(H)$, is empty. □

For any $(E, H) \in R(t)$, note that E consists of all elements that have been completely selected before time t. The element that is being selected at state (E, H) is *not* included in E. Let $T_H(k)$ denote the subtree of OPT that corresponds to paths traced by scenarios in H up to time 2^{k-1}; this only includes elements that are completely selected by time 2^{k-1}. Note that each node (labeled by any element $e \in U$) in $T_H(k)$ has at most $|G|$ outgoing branches, and one of them is labeled by the feedback corresponding to $B_e(H) = H \setminus L_e(H)$. We define $\mathsf{Stem}_k(H)$ to be the path in $T_H(k)$ that, at each node (labeled e), follows the branch corresponding to $H \setminus L_e(H)$. See Figure 2 for an example. We also use $\mathsf{Stem}_k(H)$ to denote the set of elements that are completely selected on this path.

Definition 1. Each state (E, H) in ALG is exactly one of the following types:

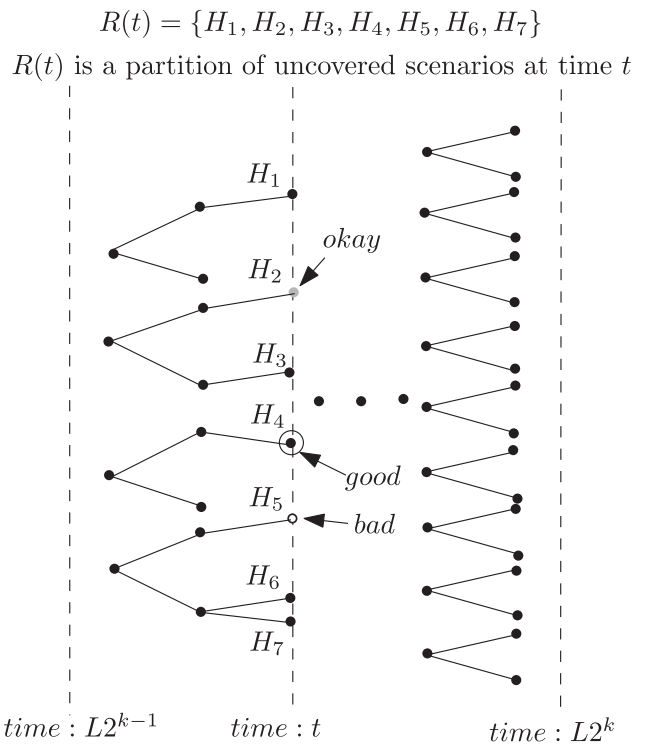
- Bad if the probability of uncovered scenarios in H at the end of $\mathsf{Stem}_k(H)$ is at least $\frac{\Pr(H)}{3}$.
- Okay if it is not bad and $\Pr(\cup_{e \in \mathsf{Stem}_k(H)} L_e(H))$ is at least $\frac{\Pr(H)}{3}$.
- Good if it is neither bad nor okay and the probability of scenarios in H that get covered by $\mathsf{Stem}_k(H)$ is at least $\frac{\Pr(H)}{3}$.

Figure 2. $\mathsf{Stem}_k(H)$ in OPT for $|G| = 2$



See Figure 3. This is well defined because, by definition of $\mathsf{Stem}_k(H)$, each scenario in H is (i) uncovered at the end of $\mathsf{Stem}_k(H)$ or (ii) in $L_e(H)$ for some $e \in \mathsf{Stem}_k(H)$ or (iii) covered by some prefix of $\mathsf{Stem}_k(H)$;

Figure 3. Bad, Good, and Okay States in ALG



that is, the function value reaches one on $\mathsf{Stem}_k(H)$. So the total probability of the scenarios in one of these three categories must be at least $\frac{\Pr(H)}{3}$. Therefore, each state (E,H) is exactly one of these three types.

The following quantity turns out to be very useful in our proof of Lemma 2.

$$Z := \sum_{t>L2^{k-1}}^{L2^k} \sum_{(E,H)\in R(t)} \max_{e\in U\setminus E} \frac{1}{c_e} \cdot \left(\Pr(L_e(H)) + \sum_{i\in H} p_i \cdot \frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)}\right), \quad (5)$$

$$= \sum_{t>L2^{k-1}}^{L2^k} \sum_{(E,H)\in R(t)} \max_{e\in U\setminus E} \frac{1}{c_e} \cdot \left(\sum_{i\in H} p_i \cdot \left(\mathbb{1}[i\in L_e] + \frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)}\right)\right). \quad (6)$$

Basically Z corresponds to the total "gain" according to our algorithm's selection criterion (2) accrued from time $L2^{k-1}$ to $L2^k$ over all the decision paths. We note that, if costs are not integer, we can consider an integral over time $t\in(L2^{k-1},L2^k]$ instead of the summation, and the rest of the analysis is essentially unchanged. Now, we obtain a lower and upper bound for Z and combine them to prove Lemma 2. The lower bound views Z as a sum of terms over t and uses the fact that the gain is "high" for good/okay states as well as the bound on probability of bad states (Proposition 2). The upper bound views Z as a sum of terms over scenarios and uses the fact that if the total gain for a scenario is high then it must be already covered.

Proposition 2. *For any $t\in\left[(L2^{k-1},L2^k\right]$, we have $\sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H) \le 3x_k$.*

Proof of Proposition 2. Note that $Stem_k(H)\subseteq T_H(k)$. Recall that $T_H(k)$ is the subtree of OPT up to time 2^{k-1} that only contains the scenarios in H. So the probability of uncovered scenarios at the end of $Stem_k(H)$ is at most the probability of scenarios in H that are not covered in OPT by time 2^{k-1}. This probability is at least $\Pr(H)/3$ for the bad state (E,H) based on its definition. Now, because states in $R(t)$ induce a subpartition of scenarios, we have

$$x_k = \sum_{i\in X_k} p_i \ge \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \sum_{i\in H\cap X_k} p_i \ge \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H)/3.$$

Rearranging, we obtain the desired inequality. □

Lemma 3. *We have $Z \ge L\cdot(a_k-3x_k)/3$.*

Proof of Lemma 3. Considering only the good/okay states in each $R(t)$ in expression (5),

$$Z \ge \sum_{t>L2^{k-1}}^{L2^k} \left(\sum_{\substack{(E,H)\in R(t)\\(E,H):okay}} \max_{e\in U\setminus E} \frac{\Pr(L_e(H))}{c_e} + \sum_{\substack{(E,H)\in R(t)\\(E,H):good}} \max_{e\in U\setminus E} \sum_{i\in H} \frac{p_i}{c_e}\cdot\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} \right).$$

Fix any time t. For any state $(E,H)\in R(t)$, define $W(H)=\mathsf{Stem}_k(H)\setminus E$. The total cost of elements in $\mathsf{Stem}_k(H)$ is at most 2^{k-1}, so $c(W(H))\le 2^{k-1}$.

Case 1. (E,H) is an okay state. Because $W(H)\subseteq U\setminus E$, we can write

$$\max_{e\in U\setminus E}\frac{\Pr(L_e(H))}{c_e} \ge \max_{e\in W(H)}\frac{\Pr(L_e(H))}{c_e} \ge \frac{\sum_{e\in W(H)}\Pr(L_e(H))}{c(W(H))} \ge \frac{\Pr(\cup_{e\in W(H)}L_e(H))}{2^{k-1}} = \frac{1}{2^{k-1}}\cdot\Pr(\cup_{e\in \mathsf{Stem}_k(H)}L_e(H)) \ge \frac{\Pr(H)}{3\cdot 2^{k-1}}. \quad (7)$$

The equality in (7) uses $\cup_{e\in E}L_e(H)=\emptyset$ (by Proposition 1), and the last inequality is by definition of an okay state.

Case 2. (E,H) is a good state. We use $F\subseteq H$ to denote the set of scenarios that get covered in $\mathsf{Stem}_k(H)$; by definition of a good state, we have $\Pr(F)\ge\Pr(H)/3$. Again using $W(H)\subseteq U\setminus E$, we have

$$\max_{e\in U\setminus E}\frac{1}{c_e}\sum_{i\in H}p_i\cdot\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} \ge \max_{e\in W(H)}\frac{1}{c_e}\sum_{i\in H}p_i\cdot\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} \ge \frac{1}{c(W(H))}\sum_{e\in W(H)}\sum_{i\in H}p_i\cdot\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} = \frac{1}{c(W(H))}\sum_{i\in H}p_i\sum_{e\in W(H)}\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} \ge \frac{1}{2^{k-1}}\sum_{i\in H}p_i\cdot\frac{f_i(W(H)\cup E)-f_i(E)}{1-f_i(E)}, \quad (8)$$

$$= \frac{1}{2^{k-1}}\sum_{i\in H}p_i\cdot\frac{f_i(\mathsf{Stem}_k(H))-f_i(E)}{1-f_i(E)} \ge \sum_{i\in F}\frac{p_i}{2^{k-1}} = \frac{\Pr(F)}{2^{k-1}} \ge \frac{\Pr(H)}{3\cdot 2^{k-1}}. \quad (9)$$

The last inequality in (8) is by submodularity of the f_i's, and the next equality is by definition of $W(H)$. The first inequality in (9) is based on this fact that $f_i(Stem_k(H))=1$ for any $i\in F$, and the last

inequality is by definition of a good state. Now, we combine (7) and (9):

$$\begin{aligned} Z &\geq \sum_{t>L2^{k-1}}^{L2^k} \sum_{\substack{(E,H)\in R(t)\\(E,H):okay}} \frac{\Pr(H)}{3\cdot 2^{k-1}} + \sum_{t>L2^{k-1}}^{L2^k} \sum_{\substack{(E,H)\in R(t)\\(E,H):good}} \frac{\Pr(H)}{3\cdot 2^{k-1}} \\ &= \sum_{t>L2^{k-1}}^{L2^k} \frac{\Pr(R(t)) - \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H)}{3\cdot 2^{k-1}} \\ &\geq \sum_{t>L2^{k-1}}^{L2^k} \frac{a_k - 3x_k}{3\cdot 2^{k-1}} = \frac{L\cdot(a_k-3x_k)}{3}. \end{aligned} \tag{10}$$

The first equality uses the fact that the states $(E,H)\in R(t)$ are exactly one of the types bad/okay/good. The last inequality uses Proposition 2 and that $\{H:(E,H)\in R(t)\}$ contains all scenarios in A_k. □

Lemma 4. *We have* $Z \leq a_{k-1}\cdot(1+\ln 1/\epsilon+\log m)$.

Proof of Lemma 4. For any scenario $i\in A_{k-1}$ (i.e., uncovered in ALG by time $L2^{k-1}$), let $\widehat{\pi}_i$ be the path traced by i in ALG's decision tree. For each element e that appears in $\widehat{\pi}_i$, we say that e is selected during the interval $K_{e,i}=(D_e, D_e+c_e]$, where D_e is the total cost of elements in $\widehat{\pi}_i$ before e. Let π_i be the subpath of $\widehat{\pi}_i$ consisting of elements selected between time $2^{k-1}L$ and 2^kL or when i gets covered (whatever happens earlier). Let $t_{e,i}\leq c_e$ denote the width of the interval $K_{e,i}\cap(L2^{k-1}, L2^k]$. Note that there can be at most two elements e in π_i with $t_{i,e}<c_e$: one that is being selected at time $L2^{k-1}$ and another at $L2^k$.

Recall that, for any $L2^{k-1}<t\leq L2^k$, every scenario in $R(t)$ appears in A_{k-1}. So only scenarios in A_{k-1} can contribute to Z, and we rewrite (6) by interchanging summations as follows:

$$\begin{aligned} Z &= \sum_{i\in A_{k-1}} p_i \cdot \sum_{e\in\pi_i} t_{e,i}\cdot\frac{1}{c_e}\left(\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} + \mathbb{1}[i\in L_e(H)]\right) \\ &\leq \sum_{i\in A_{k-1}} p_i \cdot \left(\sum_{e\in\pi_i}\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} + \sum_{e\in\pi_i}\mathbb{1}[i\in L_e(H)]\right). \end{aligned} \tag{11}$$

For any $e\in\pi_i$, we use (E,H) to denote the state at which e is selected.

Fix any scenario $i\in A_{k-1}$. For the first term, we use Claim 1 (see below) and the definition of ϵ in (1). This implies $\sum_{e\in\pi_i}\frac{f_i(e\cup E)-f_i(E)}{1-f_i(E)} \leq 1+\ln\frac{1}{\epsilon}$. To bound the second term, note that, if scenario $i\in L_e(H)$ when ALG selects element e, then the number of compatible scenarios decreases by at least a factor of *two* in path π_i. So such an event can happen at most $\log_2 m$ times along the path π_i. Thus, we can write $\sum_{e\in\pi_i}\mathbb{1}[i\in L_e(H)]\leq\log_2 m$. The lemma now follows from (11). □

We now complete the proof of Lemma 2.

Proof of Lemma 2. By Lemmas 3 and 4, we have

$$\begin{aligned} L\cdot(a_k-3x_k)/3 &\leq Z \\ &\leq a_{k-1}\cdot(1+\ln 1/\epsilon+\log m) = a_{k-1}\cdot\frac{L}{15}. \end{aligned}$$

Rearranging, we obtain $a_k\leq 0.2\cdot a_{k-1}+3x_k$ as needed. □

Claim 1 (Claim 2.1 in Azar and Gamzu (2011))**.** Let $f: 2^U\to[0,1]$ be any monotone function with $f(\emptyset)=0$ and $\epsilon=\min\{f(S\cup\{e\})-f(S): e\in U, S\subseteq U, f(S\cup\{e\})-f(S)>0\}$. Then, for any sequence $\emptyset=S_0\subseteq S_1\subseteq\cdots S_k\subseteq U$ of subsets, we have

$$\sum_{t=1}^{k}\frac{f(S_t)-f(S_{t-1})}{1-f(S_{t-1})} \leq 1+\ln\frac{1}{\epsilon}.$$

3. Algorithm for Adaptive Submodular Routing

Recall that the ASP is a generalization of ASR to a vehicle-routing setting in which costs correspond to a metric $(U\cup\{s\},d)$. Here, U denotes the set of elements, and s is a special root vertex. The rest of the input is exactly as in ASR: we are given m scenarios in which each scenario $i\in[m]$ has some probability p_i, a submodular function f_i, and a feedback function r_i. The goal is to compute an adaptive tour (that begins and ends at s) and covers a random scenario i^* at minimum expected cost, and the cost corresponds to the cost of the path of elements we need to take until we cover the realized scenario. For any walk P, when it is clear from context, we also use P to refer to the vertices/elements on this walk.

An important subproblem in our algorithm for ASP is the *submodular orienteering problem* (SOP), defined as follows. There is a metric $(U\cup\{s\},d)$ with root s, a monotone submodular function $f:2^V\to\mathbb{R}_+$, and a bound B. The goal is to compute a tour P originating from s of cost at most B that maximizes $f(P)$. A (ρ,σ)-bicriteria approximation algorithm for SOP returns a tour P such that the cost of P is at most $\sigma\cdot B$ and $f(P)\geq \mathsf{OPT}/\rho$, where OPT is the maximum value of a tour of cost at most B. Our ASP algorithm makes use of a (ρ,σ)-bicriteria approximation algorithm denoted ALG-SOP.

Our algorithm involves concatenating a sequence of smaller tours (each originating from s), where the tour costs increase geometrically. Each such tour is obtained as a solution to a suitably defined instance of SOP. The SOP instance encountered at state (E,H) involves the function $g_{(E,H)}$ defined in (12). Similar to the definition (2) of gain of an individual element in the ASR algorithm, function $g_{(E,H)}(T)$ measures the collective gain from any subset T of elements. This again comprises two parts:

1. *Information gain:* this is the first term in (12). The definition of subsets $L_e(H)$ is the same as for ASR. If the realized scenario happens to be in $L_e(H)$ for any $e \in T$, then it is clear that we make good progress in identifying the scenario: the number of compatible scenarios decreases by (at least) a factor of two. The term $\Pr(\cup_{e\in T} L_e(H))$ in (12) is just the probability that the realized scenario is in $L_e(H)$ for some $e \in T$.

2. *Function coverage:* this is the second term in (12) and is based on the algorithm for deterministic submodular routing from Im et al. (2016).

Crucially, both of these terms in $g_{(E,H)}$ are monotone submodular functions, so SOP can be used.

Algorithm 2 (ASP Algorithm)

1. $E \leftarrow \emptyset, \pi \leftarrow \emptyset, H \leftarrow [m]$ and $D = 15\rho(1 + \ln\frac{1}{\epsilon} + \log m)$.
2. **for** *phase* $k = 0, 1, 2, \ldots,$ **do**
3. If $H = \emptyset$, then output π and end the algorithm.
4. for *iteration* $u = 1, 2, \ldots, D,$ **do**
5. For any element $e \in U \setminus E$, let $B_e(H)$ denote the set with maximum cardinality among $\{i \in H : r_i(e) = t\}$ for $t \in G$ and define $L_e(H) := H \setminus B_e(H)$.
6. Define the submodular function
$$g_{(E,H)}(T) := \Pr(\cup_{e\in T} L_e(H)) + \sum_{i\in H} p_i \cdot \frac{f_i(E \cup T) - f_i(E)}{1 - f_i(E)}, \quad \forall T \subseteq U. \tag{12}$$
7. Use ALG-SOP to approximately solve the SOP instance on metric $(U \cup \{s\}, d)$ with root s, submodular function $g_{(E,H)}$, and cost bound 2^k to obtain tour P_u.
8. $E \leftarrow E \cup P_u$ and concatenate P_u to π to form a new tour.
9. Remove incompatible and covered scenarios from H based on the feedback from P_u.
10. **end**
11. **end**

As with ASR, the algorithm for ASP may not incur the cost of the entire walk traced under scenario i^*: recall that the cost is only incurred until i^* gets covered.

We can always assume that $P_u \subseteq U \setminus E$ in line 7: this is because $g_{(E,H)}(e) = 0$ for all $e \in E$ as in Proposition 1. In the rest of this section, we prove the following result.

Theorem 2. *If ALG-SOP is any (ρ, σ)-bicriteria approximation algorithm for SOP, our algorithm for* ASP *is an $\mathbb{O}(\sigma\rho(\log 1/\epsilon + \log m))$-approximation algorithm.*

We can use the following known result on SOP.

Theorem 3 (Calinescu and Zelikovsky 2005)**.** *For any constant $\delta > 0$, there is a polynomial time $(\mathbb{O}(1), \mathbb{O}(\log^{2+\delta} n))$-bicriteria approximation algorithm for submodular orienteering.*

By combining Theorems 2 and 3, we obtain the following:

Corollary 1. *For any constant $\delta > 0$, there is an $\mathbb{O}((\log 1/\epsilon + \log m) \cdot \log^{2+\delta} n)$-approximation algorithm for the adaptive submodular routing problem.*

Instead of Theorem 3, we can also use the *quasi-polynomial* time $\mathbb{O}(\log n)$-approximation algorithm for SOP from Chekuri and Pal (2005), which implies the following:

Corollary 2. *There is a quasi-polynomial time $\mathbb{O}((\log 1/\epsilon + \log m) \cdot \log n)$-approximation algorithm for the adaptive submodular routing problem.*

3.2. Analysis

We start by showing that the use of SOP is well defined.

Proposition 3. *For any state (E, H) in Algorithm 2, the function $g_{(E,H)}$ is monotone and submodular.*

Proof of Proposition 3. First note that, for any monotone submodular function f_i and $E \subseteq U$, we have that $f_i(E \cup T) - f_i(E)$ is a monotone submodular function of T. Also $f(T) = \Pr(\cup_{e\in T} L_e(H))$ is a weighted coverage function, so it is monotone submodular. Now, because a nonnegative weighted sum of submodular functions is submodular, the following function is submodular:

$$\sum_{i\in H} p_i \cdot \frac{f_i(E \cup T) - f_i(E)}{1 - f_i(E)} + \Pr(\bigcup_{e\in T} L_e(H)),$$

which is equal to $g_{(E,H)}(T)$. □

In the following, we use cost and time interchangeably. We refer to the outer loop in Algorithm 2 by *phase* and the inner loop by *iteration*. Define $\bar{L} := 2D \cdot \sigma$. Then we have the following proposition:

Proposition 4. *All vertices that are added to E in the jth phase are visited in π by time $\bar{L} \cdot 2^j$.*

Proof of Proposition 4. In each phase k, we add D tours of cost at most $2^k\sigma$ each. So a vertex that is added in phase j is visited by time $\sum_{k=0}^{j} 2^k D \cdot \sigma \le 2^{j+1} D \cdot \sigma = \bar{L} \cdot 2^j$. □

Let ALG be the solution produced by Algorithm 2 and OPT be the optimal solution. For any $k = 0, 1, \cdots$, we define the following quantities:

- A_k is the set of uncovered scenarios of ALG at the end of phase k, and $a_k = \Pr(A_k)$.
- X_k is the set of uncovered scenarios of OPT at time 2^{k-1}, and $x_k = \Pr(X_k)$. Note that $x_0 = 1$.

Lemma 5. *The expected cost of ALG and OPT can be bounded as follows:*

$$C_{ALG} \le \bar{L} \sum_{k\ge 0} 2^k a_k + \bar{L} \quad \text{and} \quad C_{OPT} \ge \frac{1}{2} \sum_{k\ge 0} 2^{k-1} x_k . \quad (13)$$

Proof of Lemma 5. By Proposition 4, for all $k \ge 1$, every scenario in $A_{k-1} \setminus A_k$ in ALG is covered by time $\bar{L}2^k$. So we can write exactly the same inequalities as in the proof of Lemma 1. □

As for ASR, in order to prove Theorem 2, it suffices to prove the following:

Lemma 6. *For any $k \ge 0$, we have $a_k \le 0.2a_{k-1} + 3x_k$.*

3.3. Proof of Lemma 6

Throughout this subsection, we fix phase k to its value in Lemma 6. Consider any iteration u in phase k of the algorithm. ALG's decision tree induces a partition of all the uncovered scenarios at iteration u, where each part H consists of all scenarios that are at a particular state (E, H) at the start of iteration u. Let $R_k(u)$ denote the set of parts in this partition. We also use $R_k(u)$ to denote the collection of states corresponding to these parts. Note that all scenarios in A_k appear in $R_k(u)$ as these scenarios are uncovered even at the end of phase k. Similarly, all scenarios in $R_k(u)$ are in A_{k-1}.

The analysis is similar to that for Lemma 2. Analogous to the quantity Z in the proof of Lemma 2, we use

$$\bar{Z} := \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \max_{P\in\mathcal{A}(E,H,k)} g_{(E,H)}(P) . \quad (14)$$

Here, $\mathcal{A}(E, H, k)$ denotes the set of feasible tours to the SOP instance solved in iteration u of phase k, and (E, H) denotes the state at the beginning of this iteration. We prove Lemma 6 by upper/lower bounding $\bar{Z}$.

For any $(E, H) \in R_k(u)$, note that E consists of all elements that have been selected before iteration u. The set of elements that are selected at iteration u are *not* included in E. We also define $T_H(k)$ and $\mathsf{Stem}_k(H)$ as in Section 2. Recall, $T_H(k)$ is the subtree of OPT that corresponds to paths traced by scenarios in H up to time 2^{k-1}; this only includes elements that are completely selected by time 2^{k-1}. And $\mathsf{Stem}_k(H)$ is the path in $T_H(k)$ that, at each node (labeled e), follows the branch corresponding to $H \setminus L_e(H)$. Again, we use $\mathsf{Stem}_k(H)$ to also denote the set of elements that are on this path. We also use the definition of bad, okay, and good states from Definition 1. Then, exactly as in Proposition 2, we have the following:

Proposition 5. *For any iteration u in phase k, we have $\sum_{\substack{(E,H)\in R_k(u)\\(E,H):bad}} \Pr(H) \le 3x_k$.*

Lemma 7. *We have $\bar{Z} \ge D \cdot (a_k - 3x_k)/3$.*

Proof of Lemma 7. Considering only the good/okay states in each $R_k(u)$ in expression (14),

$$\begin{aligned}
\bar{Z} &= \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \max_{P\in\mathcal{A}(E,H,k)} \\
&\quad \cdot \left(\sum_{i\in H} p_i \cdot \frac{f_i(E\cup P) - f_i(E)}{1 - f_i(E)} + \Pr\left(\bigcup_{e\in P} L_e(H) \right) \right) \\
&\ge \sum_{u=1}^{D} \left(\sum_{\substack{(E,H)\in R_k(u)\\(E,H):okay}} \max_{P\in\mathcal{A}(E,H,k)} \Pr\left(\bigcup_{e\in P} L_e(H) \right) \right. \\
&\quad \left. + \sum_{\substack{(E,H)\in R_k(u)\\(E,H):good}} \max_{P\in\mathcal{A}(E,H,k)} \sum_{i\in H} p_i \cdot \frac{f_i(E\cup P) - f_i(E)}{1 - f_i(E)} \right).
\end{aligned}$$

Fix any iteration u. For any state $(E, H) \in R_k(u)$, define $W(H) = \mathsf{Stem}_k(H) \setminus E$. Note that the cost of $\mathsf{Stem}_k(H)$ is at most 2^{k-1}, so the tour obtained by doubling this path is in $\mathcal{A}(E, H, k)$: that is, the tour originates from s and has cost at most 2^k. We call this tour $\overline{W}(H)$.

Case 1. (E, H) is an okay state. Because $\overline{W}(H) \in \mathcal{A}(E, H, k)$,

$$\begin{aligned}
\max_{P\in\mathcal{A}(E,H,k)} \Pr\left(\bigcup_{e\in P} L_e(H) \right) &\ge \Pr\left(\bigcup_{e\in W(H)} L_e(H) \right) \\
&= \Pr\left(\bigcup_{e\in \mathsf{Stem}_k(H)} L_e(H) \right) \ge \frac{\Pr(H)}{3}. \quad (15)
\end{aligned}$$

The equality uses $\cup_{e\in E} L_e(H) = \emptyset$ (by Proposition 1), and the last inequality is by Definition 1 of an okay state.

Case 2. (E, H) is a good state. We use $F \subseteq H$ to denote the set of scenarios that get covered in $\mathsf{Stem}_k(H)$; by definition of a good state, we have $\Pr(F) \ge \Pr(H)/3$. Again using $\overline{W}(H) \in \mathcal{A}(E, H, k)$,

$$\begin{aligned}
&\max_{P\in\mathcal{A}(E,H,k)} \sum_{i\in H} p_i \cdot \frac{f_i(P\cup E) - f_i(E)}{1 - f_i(E)} \\
&\ge \sum_{i\in H} p_i \cdot \frac{f_i(W(H)\cup E) - f_i(E)}{1 - f_i(E)} \\
&= \sum_{i\in H} p_i \cdot \frac{f_i(\mathsf{Stem}_k(H)) - f_i(E)}{1 - f_i(E)} \ge \sum_{i\in F} p_i \\
&= \Pr(F) \ge \frac{\Pr(H)}{3}. \quad (16)
\end{aligned}$$

The first equality of (16) is by definition of $W(H)$. The next inequality is based on the fact that $f_i(\mathsf{Stem}_k(H)) = 1$ for any $i \in F$, and the last inequality is by definition of

a good state. Now, we combine (15) and (16) with the definition of $\bar{Z}$:

$$\begin{aligned}\bar{Z} &\geq \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):okay}} \frac{\Pr(H)}{3} + \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):good}} \frac{\Pr(H)}{3}\\ &= \sum_{u=1}^{D} \frac{\Pr(R_k(u)) - \sum_{\substack{(E,H)\in R_k(u)\\(E,H):bad}} \Pr(H)}{3}\\ &\geq \sum_{u=1}^{D} \frac{a_k - 3x_k}{3} \quad = \quad \frac{D\cdot(a_k-3x_k)}{3}.\end{aligned}$$

The first equality uses the fact that the states corresponding to each $(E,H)\in R_k(u)$ are exactly one of the types bad/okay/good. The last inequality uses Proposition 5 and that $R_k(u)$ contains all scenarios in A_k. □

Lemma 8. *We have* $\bar{Z} \leq a_{k-1}\cdot\rho(1+\ln\frac{1}{\epsilon}+\log m)$.

Proof of Lemma 8. For any scenario $i\in A_{k-1}$ (i.e., uncovered in ALG at the end of phase $k-1$), let π_i be the path traced by i in ALG's decision tree, starting from the end of phase $k-1$ to the end of phase k or when i gets covered (whichever happens first). Formally, we represent π_i as a sequence of tuples (E_{iu},H_{iu},P_{iu}) for each iteration u in phase k, where (E_{iu},H_{iu}) is the state at the start of iteration u and P_{iu} is the new tour chosen by ALG at this state.

Recall that, for any iteration u, every scenario in $R_k(u)$ appears in A_{k-1}. So only scenarios in A_{k-1} can contribute to $\bar{Z}$ because every part H in $R_k(u)$ is a subset of A_{k-1}. Furthermore, because ALG-SOP is a (ρ,σ)-bicriteria approximation algorithm, it selects paths P_u such that $\rho\cdot g_{(E,H)}(P_u)\geq \max_{P\in\mathcal{A}(E,H,k)} g_{(E,H)}(P)$. So we can bound $\bar{Z}$ from above as follows:

$$\begin{aligned}\bar{Z} &= \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \max_{P\in\mathcal{A}(E,H,k)} g_{(E,H)}(P)\\ &\leq \quad \rho\cdot\sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} g_{(E,H)}(P_u)\\ &\leq \rho\cdot\sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \left(\sum_{i\in H}\left(p_i\cdot\frac{f_i(E\cup P_u)-f_i(E)}{1-f_i(E)}\right)\right.\\ &\quad \left.+\Pr\left(\bigcup_{e\in P_u} L_e(H)\right)\right)\\ &= \rho\cdot\sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \sum_{i\in H} p_i\cdot\left(\frac{f_i(E\cup P_u)-f_i(E)}{1-f_i(E)}\right.\\ &\quad \left.+\mathbb{1}\left[i\in\cup_{e\in P_u}L_e(H)\right]\right)\\ &\leq \quad \rho\cdot\sum_{i\in A_{k-1}} p_i\cdot\left(\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i}\left(\frac{f_i(P_{iu}\cup E_{iu})-f_i(E_{iu})}{1-f_i(E_{iu})}\right.\right.\\ &\quad \left.\left.+\mathbb{1}[i\in\cup_{e\in P_{iu}}L_e(H_{iu})]\right)\right) \qquad (17)\\ &= \quad \rho\cdot\sum_{i\in A_{k-1}} p_i\cdot\left(\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i}\frac{f_i(P_{iu}\cup E_{iu})-f_i(E_{iu})}{1-f_i(E_{iu})}\right.\\ &\quad \left.+\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i}\mathbb{1}[i\in\cup_{e\in P_{iu}}L_e(H_{iu})]\right), \qquad (18)\end{aligned}$$

where the inequality (17) is due to an interchange of summation and the fact that each part H of $R_k(u)$ is a subset of A_{k-1}. Now, fix any scenario $i\in A_{k-1}$. For the first term in (18), we use Claim 1 and the definition of ϵ in (1). This implies $\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i}\frac{f_i(P_{iu}\cup E_{iu})-f_i(E_{iu})}{1-f_i(E_{iu})}\leq 1+\ln\frac{1}{\epsilon}$. To bound the second term, note that, if at some iteration u with state (E,H), the algorithm selects subset P_u, and if scenario $i\in\cup_{e\in P_u}L_e(H)$, then the number of possible scenarios decreases by at least a factor of *two* in path π_i. So such an event can happen at most $\log_2 m$ times along the path π_i. Thus, we can write $\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i}\mathbb{1}[i\in\cup_{e\in P_{iu}}L_e(H_{iu})]\leq\log_2 m$. The lemma follows from (18). □

Now we can complete the proof of Lemma 6.

Proof of Lemma 6. By Lemmas 7 and 8, we have

$$\begin{aligned}D\cdot(a_k-3x_k)/3 \quad &\leq \quad \bar{Z}\\ &\leq a_{k-1}\cdot\rho(1+\ln 1/\epsilon+\log m) = \quad a_{k-1}\cdot\frac{D}{15}.\end{aligned}$$

Rearranging, we obtain $a_k\leq 0.2\cdot a_{k-1}+3x_k$ as needed. □

4. Applications

In this section, we discuss various applications of ASR. For some of these applications, we obtain improvements over previously known results. For many others, we match (or nearly match) the previous best results using a simpler algorithm and analysis. Some of the applications discussed are new, for which we provide the first approximation algorithms. Table 1 summarizes some of these applications. As defined, cost in ASR and ASP is only incurred until the realized scenario i^* gets covered, and the algorithm may not know this (see Section 1.1). This definition is suitable for the applications discussed in Sections 4.1–4.3 and 4.10. However, for the other applications (Sections 4.4–4.9), the algorithm needs to know explicitly when to stop. For these applications, we also mention the stopping criteria used and show that it coincides with the (usual) criterion of just covering i^*. So Theorem 1 or 2 can be applied in all cases.

4.1. Deterministic Submodular Ranking

In this problem, we are given a set of n elements and m monotone submodular functions $f_1,f_2,\ldots,f_m$, where each $f_i:2^{[n]}\to[0,1]$. We also have a nonnegative weight w_i associated with each $i\in[m]$. The goal is to

Table 1. Some Applications of Adaptive Submodular Ranking

Problem	Previous best result	Our result
Adaptive multiple intent reranking	—	$\mathbb{O}(\log K + \log m)$
Generalized optimal decision tree	—	$\mathbb{O}(\log m)$
Decision region determination	$\mathbb{O}(r \log m)$ in exp time	$\mathbb{O}(r \log m)$ in poly time
Stochastic knapsack cover	—	$\mathbb{O}(\log m + \log W)$
Stochastic matroid basis	—	$\mathbb{O}(\log m + \log q)$
Adaptive traveling repairman problem	$\mathbb{O}(\log^2 n \log m)$	$\mathbb{O}(\log^{2+\delta} n(\log m + \log n))$
Adaptive traveling salesman problem	$\mathbb{O}(\log^2 n \log m)$	$\mathbb{O}(\log^{2+\delta} n(\log m + \log n))$

find a static linear ordering of the elements that minimizes the weighted summation of functions' cover time, with which the cover time of a function f_i is the first time that its value reaches one. This is a special case of ASR in which there is no feedback. Formally, we consider the ASR instance with the same f_is, $G = \emptyset$, and probabilities $p_i = w_i/(\sum_{j=1}^{n} w_j)$. Theorem 1 directly gives an $\mathbb{O}(\log m + \log \frac{1}{\epsilon})$-approximation algorithm. Moreover, by observing that, in (2), for any state (E, H), we have $L_e(H) = \emptyset$, we can strengthen the upper bound in Lemma 4 to $Z \le a_{k-1} \cdot (1 + \ln 1/\epsilon)$. This implies that our algorithm is an $\mathbb{O}(\log \frac{1}{\epsilon})$-approximation, matching the best result in Azar and Gamzu (2011) and Im et al. (2016).

4.2. Adaptive Multiple Intents Reranking

This is an adaptive version of the multiple intents reranking problem, introduced in Azar et al. (2009) with applications to search ranking. There are n results to a particular search query, and m different users. Each user i is characterized by a subset S_i of the results in which the user is interested and a threshold $K_i \le |S_i|$: user i gets covered after seeing at least K_i results from the subset S_i. There is also a probability distribution $\{p_i\}_{i=1}^{m}$ on the m users from which the realized user i^* is chosen. An algorithm displays results one by one and receives feedback on $e \in S_{i^*}$, that is, whether result e is relevant to user i^*. The goal is to find an adaptive ordering of the results that minimizes the expected number of results to cover user i^*. We note that the algorithm need not know when this occurs, that is, when to stop.

This can be modeled as an ASR with results corresponding to elements U and users corresponding to the m scenarios. The feedback values are $G = \{0, 1\}$, and the feedback functions are given by $r_i(e) = \mathbb{1}(e \in S_i)$ for all $i \in [m]$ and $e \in U$. For each scenario $i \in [m]$, the submodular function $f_i(S) = \min(|S \cap S_i|, K_i)/K_i$. Letting $K = \max_{i \in [m]} K_i$, we can see that the parameter ϵ is equal to $1/K$. So Theorem 1 implies an $\mathbb{O}(\log K + \log m)$-approximation algorithm. We note, however, that in the deterministic setting, there are better $\mathbb{O}(1)$-approximation algorithms in Bansal et al. (2010), Skutella and Williamson (2011), and Im et al. (2014). These results are based on a different linear program–based approach: extending such an approach to the stochastic case is still an interesting open question.

4.3. Minimum Cost Matroid Basis

Consider the following stochastic network design problem. We are given an undirected graph (V, E) with edge costs. However, only a random subset $E^* \subseteq E$ of the edges are active. We assume an explicit scenario-based joint distribution for E^*: there are m scenarios in which each scenario $i \in [m]$ occurs with probability p_i and corresponds to active edges $E^* = E_i$. An algorithm learns whether/not an edge e is active only upon *testing* e, which incurs time c_e. An algorithm needs to adaptively test a subset $S \subseteq E$ of edges so that $S \cap E^*$ achieves the maximum possible connectivity in the active graph (V, E^*); that is, $S \cap E^*$ must contain a maximal spanning forest of graph (V, E^*). The objective is to minimize the expected time before the tested edges achieve maximal connectivity in the active graph. The algorithm need not know when this occurs, that is, when to stop.

We can model this as an ASR instance with edges E as elements and scenarios as described. The feedback values are $G = \{0, 1\}$ and $r_i(e) = \mathbb{1}(e \in E_i)$ for all $i \in [m]$ and $e \in E$. The submodular functions are $f_i(S) = \frac{\mathsf{rank}_i(S \cap E_i)}{\mathsf{rank}_i(E_i)}$, where rank_i is the rank function of the graphic matroid on (V, E_i). The f_i's are monotone and submodular because of the submodularity of matroid rank functions. Moreover, the parameter ϵ is at least $\frac{1}{q}$, where $q = |V|$. So Theorem 1 implies an $\mathbb{O}(\log m + \log q)$-approximation algorithm. We note that the same result also holds for a general matroid in which a random (correlated) subset of elements is active and the goal is to find a basis over the active elements at minimum expected cost.

4.4. Optimal Decision Tree

This problem captures many applications in active learning, medical diagnosis, and databases; see, for example, Chakaravarthy et al. (2011) and Dasgupta (2004). There are m possible hypotheses with a probability distribution $\{p_i\}_{i=1}^{m}$, from which an unknown hypothesis i^* is drawn. There are also a number of

binary tests; each test e costs c_e and returns a positive outcome if i^* lies in some subset Y_e of hypotheses and a negative outcome if $i^* \in [m] \setminus Y_e$. It is assumed that i^* can be uniquely identified by performing all tests. The goal is to perform an adaptive sequence of tests so as to identify hypothesis i^* at the minimum expected cost.

This can be cast as an ASR instance as follows. We associate elements with tests U and scenarios with hypotheses $[m]$. The feedback values are $G = \{0, 1\}$, and the feedback functions are given by $r_i(e) = \mathbb{1}(i \in Y_e)$, which denotes the outcome of test e on hypothesis i. In order to define the submodular functions, let

$$T_e(i) = \begin{cases} [m] \setminus Y_e & \text{if } i \in Y_e, \\ Y_e & \text{if } i \notin Y_e, \end{cases} \qquad \forall e \in U \text{ and } i \in [m].$$

Then, for each scenario $i \in [m]$, define the submodular function $f_i(S) = |\cup_{e \in S} T_e(i)| \cdot \frac{1}{m-1}$. Note that, at any point in the algorithm at which we have performed a set S of tests, the set $\cup_{e \in S} T_e(i^*)$ consists of all hypotheses that have a different outcome from i^* in at least one of the tests in S. So i^* is uniquely identified after performing tests S if and only if $f_{i^*}(S) = 1$. The algorithm's stopping criterion is the first point at which the number of compatible hypotheses/scenarios reaches one: this coincides with the point at which f_{i^*} gets covered. Note that the parameter ϵ is equal to $\frac{1}{m}$, so by Theorem 1, we obtain an $\mathcal{O}(\log m)$-approximation algorithm which is known to be best possible (unless $P = NP$) as shown by Chakaravarthy et al. (2011). Although this problem has been extensively studied, previously such a result was known only via a complex algorithm in Gupta et al. (2017) and Cicalese et al. (2014). We also note that our result extends in a straightforward manner to provide an $O(\log m)$ approximation in the case of *multiway* tests (corresponding to more than two outcomes) as studied in Chakaravarthy et al. (2011).

4.4.1. Generalized Optimal Decision Tree.

Our algorithm also extends to the setting in which we do not have to uniquely identify the realized hypothesis i^*. Here we are given a threshold t such that it suffices to output a subset H^* of at most t hypotheses with $i^* \in H^*$. This can be handled easily by setting

$$f_i(S) = \min\left\{|\cup_{e \in S} T_e(i)| \cdot \frac{1}{m-t}, 1\right\}, \quad \text{for all } S \subseteq U \text{ and } i \in [m].$$

Note that, this time, we have $f_i(S) = 1$ if and only if at least $m - t$ hypotheses differ from i on at least one test in S, so this corresponds to having at most t possible hypotheses. The algorithm's stopping criterion here is the first point at which the number of compatible hypotheses is at most t: Again, this coincides with the point at which f_{i^*} gets covered. And Theorem 1 implies an $\mathcal{O}(\log m)$-approximation algorithm. To the best of our knowledge, this is the first approximation algorithm in this setting.

4.5. Equivalence Class Determination

This is an extension of ODT that was introduced to model noise in Bayesian active learning by Golovin et al. (2010). As in ODT, there are m hypotheses with a probability distribution $\{p_i\}_{i=1}^m$ and binary tests with which each test e has a positive outcome for hypotheses in Y_e. We are additionally given a partition Q of $[m]$. For each $i \in [m]$, let $Q(i)$ be the subset in the partition that contains i. The goal now is to minimize the expected cost of tests until we recognize the *part* of Q containing the realized hypothesis i^*.

We can model this as an ASR instance with tests as elements and hypotheses as scenarios. The feedback functions are the same as in ODT. The submodular functions are

$$f_i(S) = \frac{|\cup_{e \in S} (T_e(i) \cap Q(i)^c)|}{|Q(i)^c|}, \quad \text{for all } S \subseteq U \text{ and } i \in [m].$$

Here, $T_e(i)$ are as defined for ODT and A^c denotes the complement of any set $A \subseteq [m]$. Note that f_i's are monotone submodular with values between zero and one. Furthermore, $f_i(S) = 1$ means that $Q(i)^c \subseteq \cup_{e \in S} T_e(i)$, which means that the set of compatible hypotheses based on the tests S is a subset of $Q(i)$. The algorithm's stopping criterion here is the first point at which the set of compatible hypotheses is a subset of *any* $Q(i)$, which coincides with the point at which f_{i^*} gets covered. Again, Theorem 1 implies an $\mathcal{O}(\log m)$-approximation algorithm. This matches the best previous result of Cicalese et al. (2014), and again our algorithm is much simpler.

4.6. Decision Region Determination

This is an extension of ODT that was introduced in order to allow for *decision making* in Bayesian active learning. As elaborated in Javdani et al. (2014), this problem has applications in robotics, medical diagnosis, and comparison-based learning. Again, there are m hypotheses with a probability distribution $\{p_i\}_{i=1}^m$ and binary tests for which each test e has a positive outcome for hypotheses in Y_e. In addition, there are a number of overlapping decision regions $D_j \subseteq [m]$ for $j \in [t]$. Each region D_j corresponds to the subset of hypotheses under which a particular decision $j \in [t]$ is applicable. The goal is to minimize the expected cost of tests so as to find some decision region D_j containing the realized hypothesis i^*. Following prior work, two additional parameters are useful for this problem: r is the maximum number of decision regions that contain a hypothesis, and d is the maximum

size of any decision region. Our main result here is the following:

Theorem 4. *There is an* $\mathbb{O}(\log m + \min(d, r \log d))$*-approximation algorithm for decision region determination.*

This improves upon a number of previous papers on decision region determination (DRD). Javdani et al. (2014) obtained an $\mathbb{O}(\min(r, d) \cdot \log^2 \frac{1}{\min_i p_i})$-approximation algorithm running in time exponential in $\min(r, d)$. Then, Chen et al. (2015) obtained an $\mathbb{O}(r \cdot \log^2 \frac{1}{\min_i p_i})$-approximation algorithm for this problem in polynomial time. The approximation ratio was later improved by Grammel et al. (2016) to $\mathbb{O}(\min(r, d) \cdot \log m)$, which, however, required time exponential in $\min(r, d)$. In contrast, our algorithm runs in polynomial time.

Before proving Theorem 4, we provide two different algorithms for DRD.

4.6.1. Approach 1: An $O(r \log m)$-Approximation Algorithm for DRD. Here we model DRD as an ASR with tests as elements and hypotheses as scenarios. The feedback functions are the same as in ODT. For each $i \in [m]$ and $j \in [t]$ such that $i \in D_j$, define $f_{i,j}(S) = \frac{|\bigcup_{e \in S}(T_e(i) \cap D_j^c)|}{|D_j^c|}$. Clearly $f_{i,j}$'s are monotone submodular with values between zero and one. Also, $f_{i,j}(S) = 1$ means that $D_j^c \subseteq \bigcup_{e \in S} T_e(i)$, which means that the set of compatible hypotheses based on the tests S is a subset of decision region D_j. However, we may stop when it is determined that the realized hypothesis is in any one of the decision regions. This criterion (for hypothesis i) corresponds to at least one $f_{i,j}(S) = 1$ among $\{j : i \in D_j\}$. Using an idea from Guillory and Bilmes (2011), we can express this criterion as a submodular-cover requirement. Define

$$f_i(S) = 1 - \prod_{j : i \in D_j} (1 - f_{i,j}(S)), \quad \text{for all } S \subseteq U \text{ and } i \in [m].$$

One can verify that $f_i(S) = 1$ if and only if $\exists j : i \in D_j$ and $f_{i,j}(S) = 1$. The algorithm's stopping criterion is the first point at which the set of compatible hypotheses is a subset of any decision region D_j, which coincides with the point at which f_{i^*} gets covered. We can also see that f_i is monotone and submodular. Note that, here, the parameter ϵ is equal to $\min_i \prod_{j : i \in D_j} \frac{1}{|D_j^c|}$, which is much smaller than in previous applications. Still, we have $\epsilon = \Omega(m^{-r})$. So, in this case, Theorem 1 implies an $\mathbb{O}(r \log m)$-approximation algorithm in which r is the maximum number of decision regions that contain a hypothesis.

4.6.2. Approach 2: An m-Approximation Algorithm for DRD. Here we use a simple greedy splitting algorithm. At any state with compatible scenarios $H \subseteq [m]$, the algorithm selects the minimum cost element that splits H. Formally, it selects

$$\arg\min\{c_e : e \in U \text{ with } H \cap Y_e \neq \emptyset \text{ and } H \cap Y_e^c \neq \emptyset\}.$$

The algorithm terminates when the compatible scenario H is contained in any decision region.

As the number of compatible scenarios reduces by at least one after each chosen element, the depth of the algorithm's decision tree is at most m. Consider any depth $k \in \{1, \cdots m\}$ in this decision tree. Note that the states occurring at depth k induce a partition of all scenarios $I \subseteq [m]$ that are yet uncovered (at depth k). For each scenario $i \in I$, let $R_i \subseteq I$ denote all scenarios that are compatible with i at depth k and let C_i denote the minimum cost of an element that splits R_i. Note that all scenarios i occurring at the same state at depth k have the same R_i and C_i. Moreover, the kth element chosen by the algorithm under any scenario $i \in I$ costs exactly C_i. So the algorithm's expected cost at depth k is exactly $\sum_{i \in I} p_i \cdot C_i$. The next claim shows that $OPT \geq \sum_{i \in I} p_i \cdot C_i$, which implies that the total expected cost of the algorithm is at most $m \cdot OPT$.

Claim 2. The optimal cost of the DRD instance $OPT \geq \sum_{i \in I} p_i \cdot C_i$.

Proof of Claim 2. Consider any $i \in I$. Note that $R_i \subseteq I \subseteq [m]$ does not contain any decision region (otherwise, i would have been covered before depth k, which would contradict $i \in I$). So the optimal solution *must* select some element that splits R_i in its decision path for scenario i. As C_i is the minimum cost element that splits R_i, it follows that the optimal cost under scenario i is at least C_i. The claim now follows by taking expectations. □

Proof of Theorem 4. This algorithm involves two phases. The first phase runs the $O(\log m)$-approximation algorithm for *generalized ODT* (Subsection 4.4) on the given set of scenarios and elements with threshold d (this step ignores the decision regions). Crucially, the optimal value of this generalized ODT instance is at most that of the DRD instance. This follows simply from the fact that every decision region has size at most d, so the number of compatible scenarios at the end of any feasible DRD solution is always at most d. So the expected cost in the first phase is $O(\log m) \cdot OPT$. At the end of this phase, we are left with a set M of at most d candidate scenarios and we still need to identify a valid decision region within that set. Let $\{M_1, \cdots M_s\}$ denote the partition of the m scenarios corresponding to the states at the end of the generalized ODT algorithm. So we have $|M_k| \leq d$ for all $k \in [s]$.

Next, in the second phase, we run one of the aforementioned algorithms on the DRD instance conditioned on scenarios M. For any $k \in [s]$, let $\mathcal{I}_k$ denote the DRD instance restricted to scenarios M_k in which

probabilities are normalized so as to sum to one. Crucially,

$$\sum_{k=1}^{s}\left(\sum_{i\in M_k} p_i\right) OPT(\mathcal{I}_k) \le OPT, \tag{19}$$

where OPT is the optimal value of the original DRD instance; (19) follows directly by using the optimal tree for the original DRD instance as a feasible solution for each instance $\mathcal{I}_1, \cdots \mathcal{I}_s$.

Note that the DRD instance in the second phase always has at most d scenarios as $\max_{k=1}^{s} |M_k| \le d$. So the two algorithms have approximation ratios of $O(r \log d)$ and d, respectively, on this instance. Combined with (19), it follows that the expected cost in the second phase is $O(\min\{r \log d, d\}) \cdot OPT$. Adding the cost over both phases proves the theorem. □

4.7. Stochastic Knapsack Cover

In the knapsack cover problem, there are n elements, each with a cost and reward. We are also given a target W, and our goal is to choose a subset of elements with minimum total cost such that the total reward is at least W. Ibarra and Kim (1975) gave a fully polynomial time approximation scheme for this problem. Here, we consider a stochastic version of this problem in which rewards are random and correlated across elements. Previously, Deshpande et al. (2016) considered the case of independent rewards and obtained a three-approximation algorithm. We assume an explicit scenario-based distribution for the rewards. Formally, there are m scenarios, and each scenario $i \in [m]$ occurs with probability p_i and corresponds to element rewards $\{r_i(e)\}_{e=1}^{n}$. We also assume that all rewards are integers between zero and W. An algorithm knows the precise reward of an element $e \in [n]$ only upon selecting e. The goal is to adaptively select a sequence of elements so as to achieve total reward at least W at minimum expected cost.

To model this problem as an instance of ASR, elements and scenarios are as described. The feedback values are $G = \{0, 1, \ldots, W\}$, and the feedback functions are the rewards $r_i(\cdot)$ under each scenario $i \in [m]$. The submodular functions are $f_i(E) = \min(1, \frac{1}{W} \cdot \sum_{e\in E} r_i(e))$, where $r_i(e)$ is the reward of element e under scenario i. Note that $f_i(E) = 1$ if and only if the total reward of elements in E is at least W, which is also used as the stopping criterion for the algorithm. The parameter ϵ would be equal to $w/W \ge 1/W$, where w is the minimum positive reward. Using Theorem 1, we obtain an $\mathcal{O}(\log m + \log \frac{W}{w})$-approximation algorithm.

We note that, in the more general black-box distribution model (in which we can only access the reward distribution through samples), there are hardness results that rule out any subpolynomial approximation ratio by polynomial-time algorithms.

4.8. Scenario Submodular Cover

This was studied recently by Grammel et al. (2016) as a way to model correlated distributions in stochastic submodular cover. We have a set U of elements with costs $\{c_e\}_{e\in U}$. Each element, when selected, provides random feedback from a set G: the feedback is correlated across elements. We are given a scenario-based distribution of elements' feedback values. There are m scenarios with probabilities $\{p_i\}_{i=1}^{m}$ from which the realized scenario i^* is drawn. Each scenario $i \in [m]$ specifies the feedback $r_i(e) \in G$ for each element $e \in U$. Let $*$ denote an unknown feedback value. There is also a "state-based" utility function $f : (G \cup \{*\})^U \to \mathbb{Z}_{\ge 0}$ and an integer target Q. The function f is said to be covered if its value is at least Q. The goal is to (adaptively) select a sequence of elements so as to cover f at the minimum expected cost.

It is assumed f is monotone and submodular: as f is not a usual set function, one needs to extend the notions of monotonicity and submodularity to this setting. For any $g, g' \in (G \cup \{*\})^U$, we say g' is an *extension* of g and write $g' \succcurlyeq g$ if $g'_e = g_e$ for all $e \in U$ with $g_e \ne *$. For any $g \in (G \cup \{*\})^U$, $e \in U$, and $r \in G$, define $g_{e\leftarrow r}$ to be the vector that is equal to g on all coordinates $U \setminus \{e\}$ and has value r in coordinate e. Now, we say f is

- *Monotone* if receiving feedback does not decrease its value; that is, $f(g') \ge f(g)$ for all $g' \succcurlyeq g$.
- *Submodular* if $f(g'_{e\leftarrow r}) - f(g') \le f(g_{e\leftarrow r}) - f(g)$ for all $g' \succcurlyeq g$, $r \in G$, and $e \in U$ with $g'_e = *$.

For any subset $S \subseteq U$ and scenario $i \in [m]$, define $x(S, i) \in (G \cup \{*\})^U$ as

$$x(S,i)_e = \begin{cases} r_i(e) & \text{if } e \in S \\ * & \text{if } e \in U \setminus S. \end{cases}$$

Note that function f is covered by subset $S \subseteq U$ if and only if $f(x(S, i^*)) \ge Q$.

We can model scenario submodular cover as an ASR instance with elements, scenarios, and feedback as previously. The submodular functions are $f_i(S) = \frac{1}{Q} \cdot \min\{f(x(S,i)), Q\}$ for all $S \subseteq U$ and $i \in [m]$. It can be seen that each f_i is monotone submodular (in the usual set function definition). Moreover, the parameter $\epsilon \ge 1/Q$ because function f is assumed to be integer valued. The algorithm's stopping criterion is as follows. If S denotes the set of selected elements and $\theta_e \in G$ the feedback from each $e \in S$, then we stop when $f(\theta) \ge Q$ with which $\theta_e = *$ for all $e \in U \setminus S$. Clearly, this is the same point at which f_{i^*} reaches one.

So Theorem 1 implies an algorithm with approximation ratio of $\mathcal{O}(\log m + \log \frac{1}{\epsilon})$, which is at least as good as the $\mathcal{O}(\log m + \log Q)$ bound in Grammel et al. (2016). We might have $\frac{1}{\epsilon} \ll Q$ for some functions f, in which

case our approximation ratio is better than the previous one.

4.9. Adaptive Traveling Salesman Problem

This is a stochastic version of the basic TSP that was studied in Gupta et al. (2017). We are given a metric $(U \cup \{s\}, d)$, where s is a root vertex, and there is demand at some random subset $S^* \subseteq U$ of vertices. The demand distribution is scenario based: each scenario $i \in [m]$ occurs with probability p_i and has demand subset $S^* = S_i$. We get to know whether $u \in S^*$ or not upon visiting vertex $u \in U$. The goal is to build an adaptive tour originating from s that visits all the demands S^* at minimum expected distance.

As described in Gupta et al. (2017), it suffices to solve the related "isolation problem" in which one wants to *identify* the realized scenario i^* at minimum expected distance and then use an approximate TSP to visit S_{i^*}. The isolation problem, which can be viewed as the metric version of ODT, can be modeled as ASP by considering vertices as elements and scenarios as previously. The feedback values are $G = \{0, 1\}$, and the feedback function is $r_i(e) = \mathbb{1}(e \in S_i)$ for all $e \in U$ and $i \in [m]$. The submodular functions are exactly the same as for the ODT problem (Section 4.4) in which tests correspond to vertices: for each test $e \in U$, we use $Y_e = \{i \in [m] : e \in S_i\}$. Recall that parameter ϵ is equal to $1/m$. So Corollary 1 implies an $\mathbb{O}(\log m \cdot \log^{2+\delta} n)$-approximation algorithm. This almost matches the best result known, which is an $\mathbb{O}(\log^2 n \cdot \log m)$-approximation algorithm by Gupta et al. (2017).

4.9.1. Adaptive k-Traveling Salesman Problem.

The input here is the same as adaptive TSP with an additional number k, and the goal is to minimize the expected distance taken to cover any k vertices of the demand subset S^*. As for adaptive TSP, we can model this problem as an instance of ASP. The only difference is in the definition of the submodular functions, which are now $f_i(T) = \frac{\min(|T \cap S_i|, k)}{k}$ for $T \subseteq U$ and $i \in [m]$. The algorithm stops at the first point at which it has visited k demand vertices, which is the same as f_{i^*} getting covered. Here, parameter $\epsilon = 1/k$ and Corollary 1 imply an $\mathbb{O}((\log m + \log k) \cdot \log^{2+\delta} n)$-approximation algorithm. To the best of our knowledge, this is the first approximation algorithm for this problem.

4.10. Adaptive Traveling Repairman Problem

This is a stochastic version of the TRP, which was also studied in Gupta et al. (2017). The setting is the same as adaptive TSP, but the objective here is to minimize the expected sum of distances to reach the demand vertices S^*.

We now show that this can also be viewed as a special case of ASP. Let $\mathcal{J}$ be a given instance of adaptive TRP with metric $(U \cup \{s\}, d)$, root s, and demand scenarios $\{S_i \subseteq U\}_{i=1}^m$ with probabilities $\{p_i\}_{i=1}^m$. Let $q = \sum_{i=1}^m p_i |S_i|$. We create an instance $\mathcal{I}$ of ASP with elements U, $\sum_{i=1}^m |S_i|$ scenarios and feedback values $G = \{0, 1\}$. For each $i \in [m]$ and $e \in S_i$, we define scenario $h_{e,i}$ as follows:

- $h_{e,i}$ has probability of occurrence p_i/q.
- The submodular function $f_{e,i}(T) = |\{e\} \cap T|$ for $T \subseteq U$.
- $r_{e,i}(e') = \mathbb{1}(e' \in S_i)$ for $e' \in U$.

Note that the total probability of these $\sum_{i=1}^m |S_i|$ scenarios is one. The idea is that covering scenario $h_{e,i}$ in $\mathcal{I}$ corresponds to visiting vertex e when the realized scenario in $\mathcal{J}$ is i. Note that, for any $i \in [m]$, the feedback functions for all the scenarios $\{h_{e,i} : e \in S_i\}$ are identical.

Claim 3. $OPT(\mathcal{I}) = \frac{1}{q} \cdot OPT(\mathcal{J})$.

Proof of Claim 3. Consider an optimal solution R to the adaptive TRP instance $\mathcal{J}$. For each scenario $i \in [m]$, let τ_i denote the tour (originating from s) traced by R; note that τ_i visits every vertex in S_i, and let $C_{e,i}$ denote the distance to vertex $e \in S_i$ along τ_i. So $OPT(\mathcal{J}) = \sum_{i=1}^m p_i \cdot \sum_{e \in S_i} C_{e,i}$. We can also view R as a potential solution for the ASP instance $\mathcal{I}$. To see that this is a feasible solution, note that the tour traced by R under scenario $h_{e,i}$ (for any $i \in [m]$ and $e \in S_i$) is precisely the prefix of τ_i until vertex e, at which point the tour returns to s. So every scenario in $\mathcal{I}$ is covered. Moreover, the expected cost of R for $\mathcal{I}$ is exactly $\sum_{i=1}^m \sum_{e \in S_i} \frac{p_i}{q} C_{e,i} = \frac{1}{q} \cdot OPT(\mathcal{J})$. This shows that $OPT(\mathcal{I}) \le \frac{1}{q} \cdot OPT(\mathcal{J})$.

Now, consider an optimal solution R' to the ASP instance $\mathcal{I}$. For each scenario $h_{e,i}$ (with $i \in [m]$ and $e \in S_i$), let $\sigma_{e,i}$ denote the tour (originating from s) traced by R' and let $\tau_{e,i}$ denote the shortest prefix of $\sigma_{e,i}$ that covers $f_{e,i}$. Let $C'_{e,i}$ denote the cost of the walk $\tau_{e,i}$, which is the cost under scenario $h_{e,i}$. So $OPT(\mathcal{I}) = \sum_{i=1}^m \cdot \sum_{e \in S_i} \frac{p_i}{q} C'_{e,i}$. Note that, for each $i \in [m]$, the tours $\{\sigma_{e,i} : e \in S_i\}$ are identical (call it σ_i) because the feedback obtained under scenarios $\{h_{e,i} : e \in S_i\}$ are identical. So the walks $\{\tau_{e,i} : e \in S_i\}$ must be nested. We now view R' as a potential solution for the adaptive TRP instance $\mathcal{J}$. To see that this is feasible, note that the tour traced under scenario $i \in [m]$ is precisely σ_i, which visits all vertices in S_i. Moreover, because of the nested structure of the walks $\{\tau_{e,i} : e \in S_i\}$, the distance to any vertex $e \in S_i$ under scenario i is exactly $C'_{e,i}$. So the expected cost of R' for $\mathcal{J}$ is $\sum_{i=1}^m p_i \sum_{e \in S_i} C'_{e,i} = q \cdot OPT(\mathcal{I})$. This shows that $OPT(\mathcal{J}) \le q \cdot OPT(\mathcal{I})$.

Combining these two bounds, we obtain $OPT(\mathcal{I}) = \frac{1}{q} \cdot OPT(\mathcal{J})$ as desired. □

Moreover, $\epsilon = 1$ for this ASP instance. Hence, Corollary 1 implies an $\mathbb{O}(\log m \cdot \log^{2+\delta} n)$-approximation algorithm for adaptive TRP. Again, this almost matches the best result known for this problem,

which is an $\mathbb{O}(\log^2 n \log m)$-approximation algorithm by Gupta et al. (2017). Although our approximation ratios for adaptive TSP and TRP are slightly worse than those in Gupta et al. (2017), we obtain these results as direct applications of a more general framework (ASP) with very little problem-specific work.

5. Experiments

We present experimental results for the ODT and generalized ODT problems. We use an expected number of elements as the objective; that is, all costs are unit. The main difference between ODT and generalized ODT is in the stopping criteria, which makes their coverage functions (f_i's) different. Recall that, in ODT, our goal is to uniquely identify the realized scenario. As discussed in Section 4.4,

$$f_i(S) = |\cup_{e\in S} T_e(i)| \cdot \frac{1}{m-1}, \tag{20}$$

where $T_e(i)$ is the set of all scenarios that have a different outcome from scenario i on test e. On the other hand, for generalized ODT, we satisfy the scenario as soon as the number of compatible scenarios is at most t for some input parameter t. Here we have

$$f_i(S) = \min\left\{|\cup_{e\in S} T_e(i)| \cdot \frac{1}{m-t}, 1\right\}. \tag{21}$$

5.1. Data Sets

5.1.1. Real-World Data Set. For our experiments, we used a real-world data set called WISER (http://wiser.nlm.nih.gov/). It contains information related to 79 binary symptoms (corresponding to elements in ODT) for 415 chemicals (equivalent to scenarios in ODT), which is used in the problem of toxic chemical identification of someone who has been exposed to these chemicals. This data set has been used for testing algorithms for similar problems in other papers, for example, Bellala et al. (2011), Bellala et al. (2012), and Bhavnani et al. (2007). For each symptom–chemical pair, the data specifies whether/not that symptom is seen for that chemical. However the WISER data has "unknown" entries for some pairs. In order to obtain instances for ODT from this, we generated 10 different data sets by assigning random binary values to the unknown entries. Then we removed all identical scenarios; otherwise, ODT would not be feasible. The number of scenarios in the resulting data sets ranged from 393 to 407. As probability distributions, we used permutations of the power-law distribution ($\Pr[X = x] = Kx^{\alpha}$) for $\alpha = 0, -1/2, -1$ and -2. To be able to compare results meaningfully, the same permutation was used for each α across all 10 data sets.

5.1.2. Synthetic Data Set. We also used a synthetic data set—SYN-K—that is parameterized by k; this is based on a hard instance for the greedy algorithm Kosaraju et al. (1999). Given k, this instance has $m = 2k + 1$ scenarios and $n = k + 2$ elements as follows:

$$\begin{cases} \text{Scenario } i \in [1,k] \text{ has positive feedback on element } i \text{ and } k+1 \text{ and negative on the others.} \\ \text{Scenario } i \in [k+1,2k] \text{ has positive feedback on element } i-k \text{ and } k+2 \text{ and negative on the others.} \\ \text{Scenario } 2k+1 \text{ has negative feedback on all elements.} \end{cases}$$

Also, the probabilities for the scenarios are as follows:

$$p_i = p_{i+k} = 2^{-i-2} \text{ for } i \in [1, k-1],$$
$$p_k = p_{2k} = 2^{-k-1} \quad \text{and} \quad p_{2k+1} = 2^{-1}.$$

5.2. Algorithms

In our experiments, we compare and contrast the results of four different algorithms:

- ASR: Our algorithm that uses the objective described in (2) with corresponding f_i's described in Equations (20) and (21), for ODT and generalized ODT, respectively.
- Greedy: This is a classic greedy algorithm described in Kosaraju et al. (1999), Dasgupta (2004), Guillory and Bilmes (2009), Chakaravarthy et al. (2011), and Adler and Heeringa (2012). At each iteration, it chooses the element that keeps the decision tree as balanced as possible. More formally, at each state (E, H), we choose an element $e \in U \setminus E$ that minimizes

$$|\Pr(i \in H : r_i(e) = 1) - \Pr(i \in H : r_i(e) = 0)|.$$

Although the rule is the same for ODT and generalized ODT, the set of uncovered compatible scenarios may be different, which affects the sequence of chosen elements.

- Static: This is the algorithm from Azar and Gamzu (2011). This algorithm is not feedback dependent and uses a measure that is similar to the second term in our measure (2). More specifically, this algorithm at each iteration chooses an element e that maximizes

$$\sum_{i\in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)}$$

with corresponding f_i's for each problem, described in Equations (20) and (21).

- AdStatic: This is a modified version of the aforementioned Static algorithm. It uses the observed feedback to skip redundant elements that have the same outcome on all the uncovered compatible scenarios.

5.3. Results

The performance of these four algorithms is reported in the tables. For each data set, we show normalized costs, which is the actual cost divided by the minimum

Table 2. Normalized Costs for ODT with Uniform Distribution and the Information Lower Bound

	Data set									
Algorithm	1	2	3	4	5	6	7	8	9	10
ASR	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.179	1.189	1.180	1.211	1.190	1.191	1.218	1.166	1.193	1.203
AdStatic	1.035	1.038	1.033	1.036	1.033	1.033	1.043	1.035	1.032	1.036
Bestcost	8.704	8.719	8.717	8.706	8.713	8.742	8.717	8.697	8.723	8.736
Lowerbound	8.662	8.626	8.640	8.640	8.626	8.633	8.669	8.640	8.618	8.647

Table 3. Normalized Costs for ODT with Power-Law Distribution $\alpha = -1/2$

	Data set									
Algorithm	1	2	3	4	5	6	7	8	9	10
ASR	1.001	1.002	1.001	1.003	1.002	1.003	1.001	1.003	1.001	1.003
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.203	1.207	1.193	1.231	1.214	1.191	1.233	1.222	1.262	1.222
AdStatic	1.069	1.063	1.065	1.059	1.066	1.058	1.067	1.063	1.071	1.069
Bestcost	8.415	8.427	8.429	8.400	8.422	8.449	8.419	8.403	8.431	8.449

Table 4. Normalized Costs for ODT with Power-Law Distribution $\alpha = -1$

	Data set									
Algorithm	1	2	3	4	5	6	7	8	9	10
ASR	1.038	1.051	1.010	**1.000**	1.005	**1.000**	1.024	1.027	1.041	1.006
Greedy	**1.000**	**1.000**	**1.000**	1.008	**1.000**	1.005	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.308	1.361	1.320	1.320	1.284	1.336	1.335	1.345	1.339	1.383
AdStatic	1.199	1.250	1.193	1.209	1.149	1.195	1.198	1.237	1.187	1.237
Bestcost	7.097	7.075	7.214	7.082	7.302	7.398	7.048	7.099	7.156	7.122

cost over all algorithms. The best algorithm is marked in bold. For ODT, we also report (as "Best cost") the actual minimum cost over the four algorithms. Moreover, for ODT with uniform distribution (Table 2) we also report the information-theoretic lower bound, which is the entropy (equal to $\log_2 m$); this lower bound varies across the 10 data sets as the number m of scenarios varies between 393 and 407.

5.3.1. ODT. Table 2 shows the expected costs of these algorithms for the ODT problem with uniform distribution. It turns out ASR and Greedy algorithms have the same cost for all data sets, and they both outperform Static and AdStatic. Somewhat surprisingly, the resulting cost is very close to the information-theoretic lower bound. Table 3 shows the results for power-law distribution with $\alpha = -1/2$. Greedy does

Table 5. Normalized Costs for ODT with Power-Law Distribution $\alpha = -2$

	Data set									
Algorithm	1	2	3	4	5	6	7	8	9	10
ASR	1.118	1.153	1.011	1.116	**1.000**	**1.000**	**1.000**	1.112	1.124	**1.000**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	1.050	1.193	1.096	**1.000**	**1.000**	1.011
Static	1.684	1.271	1.435	1.397	1.136	1.336	1.867	1.328	1.548	1.531
AdStatic	1.624	1.235	1.414	1.366	1.112	1.293	1.604	1.269	1.468	1.364
Bestcost	3.721	4.085	4.753	4.149	5.884	4.195	4.267	4.373	4.224	4.952

Table 6. Average Cost for Generalized ODT with Uniform Distribution

	Threshold				
Algorithm	1	2	3	4	5
ASR	**1.000**	**1.000**	**1.000**	**1.000**	**1.001**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.001**
Static	1.192	1.088	1.111	1.061	1.008
AdStatic	1.035	1.040	1.088	1.050	1.003

Table 7. Average Cost for Generalized ODT with Power-Law Distribution $\alpha = -1/2$

	Threshold				
Algorithm	1	2	3	4	5
ASR	1.003	**1.000**	**1.000**	**1.000**	1.004
Greedy	**1.000**	1.005	1.010	1.007	**1.002**
Static	1.218	1.126	1.084	1.084	1.054
AdStatic	1.065	1.075	1.060	1.068	1.050

Table 8. Average Cost for Generalized ODT with Power-Law Distribution $\alpha = -1$

	Threshold				
Algorithm	1	2	3	4	5
ASR	1.020	1.010	**1.004**	1.085	1.064
Greedy	**1.001**	**1.004**	1.010	**1.000**	**1.000**
Static	1.333	1.213	1.177	1.120	1.111
AdStatic	1.205	1.176	1.163	1.113	1.108

Table 9. Average Cost for Generalized ODT with Power-Law Distribution $\alpha = -2$

	Threshold				
Algorithm	1	2	3	4	5
ASR	1.063	1.048	1.074	**1.041**	**1.043**
Greedy	**1.035**	**1.038**	**1.045**	1.058	1.059
Static	1.453	1.356	1.324	1.285	1.258
AdStatic	1.375	1.342	1.315	1.282	1.256

slightly better than ASR on all instances; both Greedy and ASR are much better than Static and AdStatic. Table 4 has the results for power-law distribution with $\alpha = -1$. Both Greedy and ASR still outperform Static and AdStatic on all instances. ASR achieves the best solution on 2 out of 10 instances, whereas Greedy is the best on the others. Table 5 is for power-law distribution with $\alpha = -2$. Here, ASR is the best on 4 out of 10 instances, and again both Greedy and ASR outperform Static and AdStatic.

5.3.2. Generalized ODT. For these tests, we report the average (normalized) costs for each distribution and threshold. Each entry is an average over the 10 data sets. Table 6 is for the uniform distribution, Table 7 is for power-law $\alpha = -1/2$, Table 8 is for power-law $\alpha = -1$, and Table 9 is for power-law $\alpha = -2$. ASR performs the best in about half the settings, and Greedy is the best in the others. Note that the best average number is more than one1 in some cases: this shows that the corresponding algorithm was not the best on all 10 data sets. As for ODT, we see that both ASR and Greedy are better than Static and AdStatic in all cases.

5.3.3. Results on Synthetic Data. Table 10 shows the results on the synthetic instances. ASR and AdStatic have the best result simultaneously, and Greedy's performance is much worse. It is somewhat surprising that even Static performs much better than Greedy.

5.3.4. Summary. Both ASR and Greedy perform well on the real data set, and the difference in their objectives is typically small. The largest gaps were for ODT with power-law distribution $\alpha = -2$ (Table 5), with which Greedy is 19% worse than ASR on data set 6 and ASR is 15% worse than Greedy on data set 2. Combined with the fact that Greedy performs poorly on worst-case instances (Table 10), we think that ASR is a good alternative for Greedy in practice. We also observe that it is important to use adaptive algorithms for ODT on the real data set, as Static consistently performs the worst. For ODT, static is, on average, 30% worse than the best algorithm, and for generalized ODT it is, on average, 18% worse.

Table 10. Normalized Cost for ODT and Generalized ODT on SYN-K

Data set:	SYN-50			SYN-100			SYN-150			SYN-200		
	Threshold											
Algorithm	1	3	5	1	3	5	1	3	5	1	3	5
ASR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Static	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09
AdStatic	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Greedy	9.64	9.46	9.27	18.73	18.55	18.36	27.82	27.64	27.46	36.91	36.73	36.55

Acknowledgments

The authors thank Lisa Hellerstein for a clarification on Grammel et al. (2016) regarding the OR construction of submodular functions. They also thank the reviewers for insightful comments, including one that encouraged them to prove Theorem 4. A preliminary version of this paper appeared in the proceedings of Integer Programming and Combinatorial Optimization (IPCO) 2017 as Kambadur et al. (2017). Part of V. Nagarajan's work was done while visiting the Simons Institute for Theoretical Computer Science (UC Berkeley).

References

Adler M, Heeringa B (2012) Approximating optimal binary decision trees. *Algorithmica* 62(3–4):1112–1121.

Azar Y, Gamzu I (2011) Ranking with submodular valuations. *Proc. 22nd Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 1070–1079.

Azar Y, Gamzu I, Yin X (2009) Multiple intents re-ranking. Mitzenmacher M, ed. *Proc. 41st Annual ACM Sympos. Theory Comput.* (ACM, New York), 669–678.

Bansal N, Gupta A, Krishnaswamy R (2010) A constant factor approximation algorithm for generalized min-sum set cover. *Proc. 21st Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 1539–1545.

Bansal N, Gupta A, Li J, Mestre J, Nagarajan V, Rudra A (2012) When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica* 63(4):733–762.

Bellala G, Bhavnani S, Scott C (2011) Active diagnosis under persistent noise with unknown noise distribution: A rank-based approach. *Proc. 14th Internat. Conf. Artificial Intelligence Statist., Fort Lauderdale, FL*, 155–163.

Bellala G, Bhavnani SK, Scott C (2012) Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trans. Inform. Theory* 58(1):459–478.

Bhavnani SK, Abraham A, Demeniuk C, Gebrekristos M, Gong A, Nainwal S, Vallabha GK, Richardson RJ (2007) Network analysis of toxic chemicals and symptoms: Implications for designing first-responder systems. *AMIA Annual Sympos. Proc.* (American Medical Informatics Association, Bethesda, MD), 51–55.

Blum A, Chalasani P, Don Coppersmith BP, Raghavan P, Sudan M (1994) The minimum latency problem. *Proc. 26th Annual ACM Sympos. Theory Comput.* (ACM, New York), 163–171.

Calinescu G, Zelikovsky A (2005) The polymatroid steiner problems. *J. Combin. Optim.* 9(3):281–294.

Chakaravarthy VT, Pandit V, Roy S, Awasthi P, Mohania MK (2011) Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Trans. Algorithms* 7(2):Article 15.

Chaudhuri K, Godfrey B, Rao S, Talwar K (2003) Paths, trees, and minimum latency tours. *Proc. 44th Annual IEEE Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 36–45.

Chekuri C, Pal M (2005) A recursive greedy algorithm for walks in directed graphs. *Proc. 46th Annual IEEE Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 245–253.

Chen Y Javdani S, Karbasi A, Bagnell JA, Srinivasa SS, Krause A (2015) Submodular surrogates for value of information. *Proc. 29th AAAI Conf. Artificial Intelligence, Austin, TX*, 3511–3518.

Cicalese F, Laber ES, Saettler AM (2014) Diagnosis determination: Decision trees optimizing simultaneously worst and expected testing cost. *Proc. 31th Internat. Conf. Machine Learn., Beijing, China*, 414–422.

Dasgupta S (2004) Analysis of a greedy active learning strategy. Saul LK, Weiss Y, Bottou L, eds. *Advances in Neural Information Processing Systems*, vol. 17 (Curran Associates, Red Hook, NY), 337–344.

Dean BC, Goemans MX, Vondrák J (2008) Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.* 33(4):945–964.

Deshpande A, Hellerstein L, Kletenik D (2016) Approximation algorithms for stochastic submodular set cover with applications to Boolean function evaluation and min-knapsack. *ACM Trans. Algorithms* 12(3):Article 42.

Dinur I, Steurer D (2014) Analytical approach to parallel repetition. *Sympos. Theory Comput.* (ACM, New York), 624–633.

Garg N, Konjevod G, Ravi R (2000) A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms* 37(1): 66–84.

Golovin D, Krause A (2011) Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artificial Intelligence Res.* 42:427–486.

Golovin D, Krause A (2017) Adaptive submodularity: A new approach to active learning and stochastic optimization. Submitted December 6, http://arxiv.org/abs/1003.3967.

Golovin D, Krause A, Ray D (2010) Near-optimal Bayesian active learning with noisy observations. *Advances in Neural Information Processing Systems*, vol. 23 (Curran Associates, Red Hook, NY), 766–774.

Grammel N, Hellerstein L, Kletenik D, Lin P (2016) Scenario submodular cover. *Proc. 14th Internat. Workshop Approximation Online Algorithms, Aarhus, Denmark*, 116–128.

Guillory A, Bilmes J (2009) Average-case active learning with costs. *Proc. 20th Internat. Conf. Algorithmic Learning Theory* (Springer, Berlin, Heidelberg), 141–155.

Guillory A, Bilmes J (2011) Simultaneous learning and covering with adversarial noise. *Proc. 28th Internat. Conf. Machine Learn., Bellevue, Washington*, 369–376.

Gupta A, Nagarajan V, Ravi R (2017) Approximation algorithms for optimal decision trees and adaptive TSP problems. *Math. Oper. Res.* 42(3):876–896.

Halperin E, Krauthgamer R (2003) Polylogarithmic inapproximability. *Proc. 35th Annual Sympos. Theory Comput.* (ACM, New York), 585–594.

Hyafil L, Rivest RL (1976) Constructing optimal binary decision trees is *NP*-complete. *Inform. Processing Lett.* 5(1):15–17.

Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM* 22(4):463–468.

Im S, Sviridenko M, Van Der Zwaan R (2014) Preemptive and non-preemptive generalized min sum set cover. *Math. Programming.* 145(1–2):377–401.

Im S, Nagarajan V, van der Zwaan R (2016) Minimum latency submodular cover. *ACM Trans. Algorithms*, 13(1):1–28.

Javdani SH, Chen Y, Karbasi A, Krause A, Bagnell D, Srinivasa SS (2014) Near optimal Bayesian active learning for decision making. *Proc. 17th Internat. Conf. Artificial Intelligence Statistics, Reykjavik, Iceland*, 430–438.

Kambadur P, Nagarajan V, Navidi F (2017) Adaptive submodular ranking. *Integer Programming Combin. Optim. 19th Internat. Conf. Proc.*, Lecture Notes in Computer Science, vol. 10328 (Springer, Cham, Switzerland), 317–329.

Kempe D, Kleinberg JM, Tardos É (2015) Maximizing the spread of influence through a social network. *Theory Comput.* 11:105–147.

Kosaraju SR, Przytycka TM, Borgstrom RS (1999) On an optimal split tree problem. *Proc. 6th Internat. Workshop Algorithms Data Structures, Vancouver, Canada*, 157–168.

Lim ZW, Hsu D, Lee WS (2015) Adaptive stochastic optimization: From sets to paths. Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R, eds. *Advances in Neural Information Processing Systems*, vol. 28 (Curran Associates, Red Hook, NY), 1585–1593.

Lin H, Bilmes J (2011) A class of submodular functions for document summarization. *Proc. 49th Annual Meeting Assoc. Comput. Linguistics, Portland, OR*, 510–520.

Nan F, Saligrama V (2017) Comments on the proof of adaptive stochastic set cover based on adaptive submodularity and its implications for the group identification problem in "group-based active query selection for rapid diagnosis in time-critical situations." *IEEE Trans. Inform. Theory* 63(11):7612–7614.

Prasad A, Jegelka S, Batra D (2014) Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. *Advances in Neural Information Processing Systems*, vol. 27 (Curran Associates, Red Hook, NY), 2645–2653.

Shapley LS (1971) Cores of convex games. *Internat. J. Game Theory* 1(1): 11–26.

Singh A, Krause A, Guestrin C, Kaiser WJ (2009) Efficient informative sensing using multiple robots. *J. Artificial Intelligence Res.* 34: 707–755.

Skutella M, Williamson DP (2011) A note on the generalized min-sum set cover problem. *Oper. Res. Lett.* 39(6):433–436.

Wolsey LA (1982) An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2(4):385–393.

Fatemeh Navidi is a PhD candidate in industrial and operations engineering at the University of Michigan, Ann Arbor. Her research interests are in the area of online stochastic optimization, algorithmic game theory, and machine learning.

Prabhanjan Kambadur heads the AI engineering group at Bloomberg. His research interests include machine learning, natural language processing and understanding, information extraction, knowledge graphs, question answering, and table understanding.

Viswanath Nagarajan is an assistant professor in industrial and operations engineering at the University of Michigan, where he has been since 2014. His research interests are in combinatorial optimization, approximation algorithms, and stochastic optimization.