Scenario Submodular Cover

Nathaniel Grammel[1], Lisa Hellerstein[1(✉)], Devorah Kletenik[2], and Patrick Lin[3]

[1] Department of Computer Science and Engineering,
NYU Tandon School of Engineering, Brooklyn, NY, USA
{ngrammel,lisa.hellerstein}@nyu.edu
[2] Department of Computer and Information Science, Brooklyn College,
City University of New York, New York, NY, USA
kletenik@sci.brooklyn.cuny.edu
[3] Department of Computer Science, University of Illinois at Urbana-Champaign,
Champaign, IL, USA
plin15@illinois.edu

Abstract. We introduce the Scenario Submodular Cover problem. In this problem, the goal is to produce a cover with minimum expected cost, with respect to an empirical joint probability distribution, given as input by a weighted sample of realizations. The problem is a counterpart to the Stochastic Submodular Cover problem studied by Golovin and Krause [6], which assumes independent variables. We give two approximation algorithms for Scenario Submodular Cover. Assuming an integer-valued utility function and integer weights, the first achieves an approximation factor of $O(\log Qm)$, where m is the sample size and Q is the goal utility. The second, simpler algorithm achieves an approximation factor of $O(\log QW)$, where W is the sum of the weights. We achieve our bounds by building on previous related work (in [4,6,15]) and by exploiting a technique we call the Scenario-OR modification. We apply these algorithms to a new problem, Scenario Boolean Function Evaluation. Our results have applciations to other problems involving distributions that are explicitly specified by their support.

1 Introduction

The Submodular Cover problem is a fundamental problem in submodular optimization that generalizes the classical NP-complete Set Cover problem. Adaptive versions of this problem have applications to a number of other problems, notably machine learning problems where the goal is to build a decision tree or strategy of minimum expected cost. Examples of such problems include entity identification (exact learning with membership queries), classification (equivalence class determination), and decision region identification (cf. [1,6,7,11]). Other applications include reducing expected prediction costs for learned Boolean classifiers, given attribute costs [5].

Previous work on *Stochastic* Submodular Cover assumes independence of the variables of the probability distribution. Optimization is performed with respect to this distribution. We consider a new version of the problem that we call

K. Jansen and M. Mastrolilli (Eds.): WAOA 2016, LNCS 10138, pp. 116–128, 2017.
DOI: 10.1007/978-3-319-51741-4_10

Scenario Submodular Cover, that removes the independence assumption. In this problem, optimization is with respect to an input distribution given explicitly by its support (with associated probability weights). We give approximation algorithms solving Scenario Submodular Cover over discrete distributions.

In generic terms, an adaptive submodular cover problem is a sequential decision problem where we must choose items one by one from an item set $N = \{1, \ldots, n\}$. Each item has an initially unknown state, which is a member of a finite state set Γ. The state of an item is revealed only after we have chosen the item. We represent a subset $S \subseteq N$ of items and their states by a vector $x \in (\Gamma \cup \{*\})^n$ where $x_i = *$ if $i \notin S$, and x_i is the state of item i otherwise. We are given a monotone, submodular[1] utility function $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$. It assigns a non-negative integer value to a subset of the items where the value can depend on the states of the items. There is a non-negative goal utility value Q, such that $g(a) = Q$ for all $a \in \Gamma^n$. There is a cost associated with choosing each item, which we are given. In distributional settings, we are also given the joint distribution of the item states. We continue choosing items until their utility value is equal to the goal utility, Q. The problem is to determine the adaptive order in which to choose items so as to minimize expected cost (in distributional settings) or worst-case cost (in adversarial settings).

Stochastic Submodular Cover is an adaptive submodular cover problem in a distributional setting. In this problem, the state of each item is an independent random variable. The distributions of the variables are given as input. Golovin and Krause introduced a simple algorithm for this problem, called Adaptive Greedy, achieving an approximation factor of $O(\log Q)$. Another algorithm for the problem, called Adaptive Dual Greedy, was presented by Deshpande et al. [5]. These algorithms have been useful in solving other stochastic optimization problems, which can be reduced to Stochastic Submodular Cover through the construction of appropriate utility functions (e.g., [2,5,7,11]).

The problem we study, *Scenario Submodular Cover* (Scenario SC), is also a distributional, adaptive submodular cover problem. The distribution is given by a weighted sample. Each element of the sample is a vector in Γ^n, representing an assignment of states to the items in N. Associated with each assignment is a positive integer weight. The sample and its weights define a joint distribution on Γ^n, where the probability of a vector γ in the sample is proportional to its weight. (The probability of a vector in Γ^n that is not in the sample is 0.) As in Stochastic Submodular Cover, the problem is to choose the items and achieve utility Q, while minimizing expected cost. However, because proofs of results for the Stochastic Submodular Cover problem typically rely on the independence assumption, they do not apply to the Scenario SC problem.

Results. We present *Mixed Greedy*, an approximation algorithm for the Scenario SC problem that uses two different greedy criteria. It is a generalization of the

[1] The definitions "monotone" and "submodular," for state-dependent utility functions, has not been standardized. We define these terms in Sect. 2. In the terminology used by Golovin and Krause [6], g is *pointwise* monotone and pointwise submodular.

algorithm of Cicalese et al. [4] for Equivalence Class Determination (also called Group Identification and Discrete Function Evaluation). Our analysis uses the the same basic approach as that used by Cicalese et al., but the proof of their main technical lemma does not apply to our problem. We replace it with a histogram proof similar to that used in [15] for Min-Sum Submodular Cover.

The approximation factor achieved by Mixed Greedy for Scenario SC is $O(\frac{1}{\rho}\log Q)$, where ρ is a technical quantity associated with utility function g. The utility function constructed for the Equivalence Class Determination problem has constant ρ, but this is not the case in general.

To achieve a better bound for other problems, we present a modified version of Mixed Greedy, which uses an existing construction in a novel way. The existing construction produces the OR of two monotone submodular functions with goal values (cf. [5,9]). We apply this construction to g and to another utility function based on the sample, to get a new monotone, submodular function g_S, for which ρ is constant. We call the transformation of g and the sample into g_S the *Scenario-OR* modification.

Once g_S is constructed, Mixed Greedy is run on g_S with goal value Qm, where m is the size of the sample. We show that the resulting algorithm, *Scenario Mixed Greedy*, achieves an $O(\log Qm)$ approximation factor for any Scenario SC problem.

In addition to Mixed Greedy, we also present a simpler, more efficient algorithm for the Scenario SC problem, *Scenario Adaptive Greedy*, with a worse approximation bound. It is based on the Adaptive Greedy algorithm of Golovin and Krause. However, the approximation bound proved by Golovin and Krause [6] for Adaptive Greedy depends on the assumption that g and the distribution defined by the sample weights jointly satisfy *adaptive submodularity*. This is not the case for general Scenario SC instances. Scenario Adaptive Greedy is obtained by modifying Adaptive Greedy using a weighted version of the Scenario-OR modification. Scenario Adaptive Greedy combines g and the weighted sample to obtain a modified utility function g_W, having goal utility QW. Scenario Adaptive Greedy then applies Adaptive Greedy to g_W. We prove that g_W and the distribution defined by the weights jointly satisfy adaptive submodularity. Using the existing approximation bound for Adaptive Greedy then implies a bound of $O(\log QW)$ for Scenario Adaptive Greedy, where W is the sum of the weights.

The constructions of g_S and g_W are similar to constructions in work on Equivalence Class Determination and related problems (cf. [1–3,7]). Our proof of adaptive submodularity uses the approach of showing that a certain function is non-decreasing along a path between two points. This approach was used before (cf. [2,3,7]) but our problem is more general and our proof differs.

Previously, applying ordinary Adaptive Greedy to solve sample-based problems required constructing a utility function g, and then proving adaptive submodularity of g and the distribution on the weighted sample. The proof could be non-trivial (see, e.g., [1,3,7,11]). With our approach, one can get an approximation bound with Adaptive Greedy by proving only submodularity of g, rather

than adaptive submodularity of g and the distribution. Proofs of submodularity are generally easier. Also, the OR construction used in Sect. 2 preserves submodularity, but not Adaptive Submodularity [2].

Given monotone, submodular g with goal value Q, we can use our algorithms to obtain three approximation results for the associated Scenario SC problem: $O(\frac{1}{\rho}\log Q)$ with Mixed Greedy, $O(\log Qm)$ with Scenario Mixed Greedy, and $O(\log QW)$ with Scenario Adaptive Greedy.

Assuming the costs c_i are integers, and letting $C = \sum_i c_i$, we note that applying the "Kosaraju trick" (first used by Kosaraju et al. in [13]) to Scenario Adaptive Greedy yields a bound of $O(\log QmC)$ instead of $O(\log QW)$. See [6] for a similar use of the trick.

After the appearance of a preliminary version of this paper [8], Navidi et al. [14] presented a new algorithm solving a generalization of the Scenario SC problem. It achieves the $O(\log Qm)$ bound of Scenario Mixed Greedy using a single greedy rule, different from the one used in Scenario Adaptive Greedy. Their algorithm can be applied to problems where there is a distinct monotone submodular function for each scenario.

Applications. The Scenario SC problem has many applications. As an example, consider the query learning problem of identifying an unknown hypothesis h from a hypothesis class $\{h_1, \ldots, h_m\}$ by asking queries from the set $\{q_1, \ldots, q_n\}$. The answer to each query is 0 or 1, and we are given an $m \times n$ table D where $D[i, j]$ is the answer to q_i for h_j. Each pair of hypotheses differs on at least one query. Suppose there is a given cost c_i for asking query q_i, and each h_j has a given prior probability p_j. The problem is to build a decision tree (querying procedure) for identifying h, minimizing expected query cost, assuming h is drawn with respect to the p_j. View the q_i as items $i \in N$, the h_j as scenarios, and the answer to q_i as the state of item i. Represent answers to queries asked so far as a partial assignment $b \in \{0, 1, *\}^n$ where $b_i = *$ means q_i has not been asked. Define utility function $g\colon \{0, 1, *\}^n \to \mathbb{Z}_{\geq 0}$ whose value on $b \in \{0, 1, *\}^n$ is $\min\{m - 1, r(b)\}$ where $r(b) = |\{h_j \mid \exists i \text{ such that } b_i \neq * \text{ and } D[i, j] \neq b_i\}|$. Function g is monotone and submodular. Further, $g(b) = m-1$ iff the answers in b uniquely identify h. Building a decision tree with minimum expected decision cost is equivalent to solving Scenario SC for g with goal value $Q = m - 1$, for costs c_i and weights proportional to the p_i. An algorithm with an approximation bound of $O(\log m)$ for this problem was first presented by [10].

Equivalence Class Determination is a generalization of the query learning problem where in addition to D, we are given a partition of the h_j into equivalence classes. The decision tree must just identify the class to which h belongs. This problem can also be seen as a Scenario SC problem, using the "Pairs" utility function of Cicalese et al., which has goal value $Q = O(m^2)$ [4]. Applying our Scenario Mixed Greedy bound to this utility function yields an approximation bound of $O(\log m)$, matching the bound of Cicalese et al.

Our bound on Scenario Mixed Greedy yields a new approximation bound for the Decision Region Identification problem studied by Javdani et al. [11], which is

an extension of Equivalence Class Determination. They define a utility function whose value is a weighted sum of hyperedges cut in a certain hypergraph. We define a utility function whose value is the *number* of hyperedges cut. Using Mixed Greedy with this function yields an approximation bound of $O(k \log m)$, where k is a parameter associated with the problem, and m is the sample size. In contrast, the bound in [11] is $O(k \log(\frac{W}{w_{min}}))$, where w_{min} is the minimum weight on a realization in the sample. (The recent paper of Navidi et al. [14] gives a further bound.)

We can apply our algorithms to Scenario BFE (Boolean Function Evaluation) problems, which we introduce here. These problems are a counterpart to the Stochastic BFE problems[2] studied in AI, operations research, and in learning with attribute costs (see e.g., [5,12,16]). In a Scenario BFE problem, we are given a representation of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. For each $i \in \{1,\ldots,n\}$, we are given $c_i > 0$, the cost of obtaining the value of the ith bit of an initially unknown $a \in \{0,1\}^n$. We are given a weighted sample $S \subseteq \{0,1\}^n$. The problem is to compute a (possibly implicit) decision tree computing f, minimizing the expected cost of evaluating f on $a \in \{0,1\}^n$ using the tree. The expectation is with respect to the distribution defined by the sample weights.

Deshpande et al. [5] gave approximation algorithms for some Stochastic BFE problems that work by constructing a monotone, submodular utility function g and running Adaptive Greedy. By substituting the sample-based algorithms in this paper in place of Adaptive Greedy, we obtain results for analogous Scenario BFE problems. For example, using Mixed Greedy, we obtain an $O(k \log n)$ approximation for the Scenario BFE problem for k-of-n functions, a bound that is independent of sample size. Details are in the full version of the paper.

We note that the Scenario BFE problem differs from the function evaluation problem considered by Cicalese et al. [4]. In that problem, the decision tree must only compute f correctly on assignments $a \in \{0,1\}^n$ in the sample, while in Scenario BFE the tree must compute f correctly on all $a \in \{0,1\}^n$. Also, in Scenario BFE we assume function f is given with the sample, and we consider particular types of functions f.

2 Definitions

Let $N = \{1,\ldots,n\}$ be the set of *items* and Γ be a finite set of states. A *sample* is a subset of Γ^n. A *realization* is an element $a \in \Gamma^n$, representing an assignment of states to items, where for $i \in N$, a_i represents the state of item i. We also refer to an element of Γ^n as an *assignment*.

We call $b \in (\Gamma \cup \{*\})^n$ a *partial* realization. Partial realization b represents the subset $I = \{i \mid b_i \neq *\}$ where each item $i \in I$ has state b_i. For $\gamma \in \Gamma$, the quantity $b_{i \leftarrow \gamma}$ denotes the partial realization produced from b by setting $b_i = \gamma$. For $b, b' \in (\Gamma \cup \{*\})^n$, b' is an *extension* of b, written $b' \succeq b$, if $b'_i = b_i$ for all $b_i \neq *$. We use $b' \succ b$ to denote that $b' \succeq b$ and $b' \neq b$.

[2] In the Operations Research literature, Stochastic Function Evaluation is often called Sequential Testing or Sequential Diagnosis.

Let $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ be a utility function. Function $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ has *goal value* Q if $g(a) = Q$ for all realizations $a \in \Gamma^n$. We define $\Delta g(b, i, \gamma) := g(b_{i \leftarrow \gamma}) - g(b)$.

A standard utility function is a set function $f : 2^N \to \mathbb{R}_{\geq 0}$. It is monotone if for all $S \subset S' \subseteq N$, $f(S) \leq f(S')$. It is submodular if in addition, for $i \in N - S$, $f(S \cup \{i\}) - f(S) \geq f(S' \cup \{i\}) - f(S')$. We extend definitions of monotonicity and submodularity to (state-dependent) function $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ as follows:

- g is *monotone* if for $b \in (\Gamma \cup \{*\})^n$, $i \in N$ such that $b_i = *$, and $\gamma \in \Gamma$, we have $g(b) \leq g(b_{i \leftarrow \gamma})$
- g is *submodular* if for all $b, b' \in (\Gamma \cup \{*\})^n$ such that $b' \succ b$, $i \in N$ such that $b_i = b'_i = *$, and $\gamma \in \Gamma$, we have $\Delta g(b, i, \gamma) \geq \Delta g(b' i, \gamma)$.

Let $\mathcal{D}$ be a probability distribution on Γ^n. Let X be a random variable drawn from $\mathcal{D}$. For $a \in \Gamma^n$ and $b \in (\Gamma \cup \{*\})^n$, we define $\Pr[a \mid b] := \Pr[X = a \mid a \succeq b]$. For i such that $b_i = *$, we define $\mathbb{E}[\Delta g(b, i, \gamma)] := \sum_{a \in \Gamma^n : a \succeq b} \Delta g(b, i, a_i) \Pr[a \mid b]$.

- g is *adaptive submodular with respect to* $\mathcal{D}$ if for all b', b such that $b' \succ b$, $i \in N$ such that $b_i = b'_i = *$, and $\gamma \in \Gamma$, we have $\mathbb{E}[\Delta g(b, i, \gamma)] \geq \mathbb{E}[\Delta g(b', i, \gamma)]$.

Intuitively, we can view b as partial information about states of items i in a random realization $a \in \Gamma^n$, with $b_i = *$ meaning the state of item i is unknown. Then g measures the utility of that information, and $\mathbb{E}[\Delta_g(b, i, \gamma)]$ is the expected increase in utility that would result from discovering the state of i.

For $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ with goal value Q, and $b \in (\Gamma \cup \{*\})^n$ and $i \in N$, where $b_i = *$, let $\gamma_{b,i}$ be the state $\gamma \in \Gamma$ such that $\Delta g(b, i, \gamma)$ is minimized (if more than one exists, choose one arbitrarily). Thus $\gamma_{b,i}$ is the state of item i that would produce the smallest increase in utility, and thus is "worst-case" in terms of utility gain, if we start from b and then discover the state of i.

For fixed $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ with goal value Q, we define an associated quantity ρ, as follows:

$$\rho := \min \frac{\Delta g(b, i, \gamma)}{Q - g(b)} \tag{1}$$

where the minimization is over b, i, γ, where $b \in (\Gamma \cup \{*\})^n$ such that $g(b) < Q$, $i \in N$ such that $b_i = *$, and $\gamma \in \Gamma - \{\gamma_{b,i}\}$.

Intuitively, when the state of item i is discovered, the distance between the utility achieved and the goal utility is reduced by some fraction (possibly zero). The fraction can vary depending on item state. Parameter ρ equals the smallest possible value for the fraction associated with the next-to-worst case state, starting from any partial realization, and considering any item i whose state is about to be discovered.

An instance of the Scenario SC problem is a tuple (g, Q, S, w, c), where $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ is an integer-valued, monotone submodular utility function with goal value $Q > 0$, $S \subseteq \Gamma^n$, $w : S \to \mathbb{Z}^n_{>0}$ assigns a weight to each realization $a \in S$, and $c \in \mathbb{R}^n_{>0}$ is a *cost vector*. We consider a setting where we select items without repetition from the set of items N, and the states of the items correspond to an initially unknown realization $a \in \Gamma^n$. Each time we select an

item, the state a_i of the item is revealed. The selection of items can be adaptive, in that the next item chosen can depend on the states of the previous items. We continue to choose items until $g(b) = Q$, where b is the partial realization representing the states of the chosen items.

The Scenario SC problem asks for an adaptive order in which to choose the items (i.e. a *strategy*), until goal value Q is achieved, such that the expected sum of costs of the chosen items is minimized. The expectation is with respect to the distribution on Γ^n that is proportional to the weights on the assignments in the sample: $\Pr[a] = 0$ if $a \notin S$, and $\Pr[a] = \frac{w(a)}{W}$ otherwise, where $W = \sum_{a \in S} w(a)$. We call this the *sample distribution* defined by S and w and denote it by $\mathcal{D}_{S,w}$.

The strategy corresponds to a decision tree. The internal nodes of the tree are labeled with items $i \in N$, and each such node has $|\Gamma|$ children, one for each state $\gamma \in \Gamma$. We refer to the child corresponding to state γ as the γ-child. Each root-leaf path in the tree is associated with a partial realization b such that for each consecutive pairs of nodes v and v' on the path, if i is the label of v, and v' is the γ-*child* of v, then $b_i = \gamma$. If i does not label any node in the path, then $b_i = *$. The tree may be output in an implicit form (for example, in terms of a greedy rule), specifying how to determine the next item to choose, given the previous items chosen and their states. Although realizations $a \notin S$ do not contribute to the expected cost of the strategy, we require the strategy to achieve goal value Q on *all* realizations $a \in \Gamma^n$.

We will use an existing "OR construction," a method for taking the OR of two utility functions [5,9]. It is a method for combining two monotone submodular utility functions g_1 and g_2 defined on $(\Gamma \cup \{*\})^n$, and values Q_1 and Q_2, into a new monotone submodular utility function g. For $b \in (\Gamma \cup \{*\})^n$,

$$g(b) = Q_1 Q_2 - (Q_1 - g_1(b))(Q_2 - g_2(b)) \tag{2}$$

If for all $a \in \Gamma^n$, $g_1(a) = Q_1$ or $g_2(a) = Q_2$, then $g(a) = Q_1 Q_2$ for all $a \in \Gamma^n$.

3 Mixed Greedy

Mixed Greedy is a generalization of the approximation algorithm developed by Cicalese et al. for the Equivalence Class Determination problem [4]. That algorithm solves the Scenario Submodular Cover problem for a particular "Pairs" utility function associated with Equivalence Class Determination. In contrast, Mixed Greedy can be used on any monotone, submodular utility function g. Following Cicalese et al., we present Mixed Greedy as outputting a tree. If the strategy is only to be used on one realization, it is not necessary to build the entire tree.

3.1 Algorithm

The Mixed Greedy algorithm builds a decision tree for a Scenario SC instance (g, Q, S, w, c). The tree is built top-down, and is structured as described at the end of Sect. 2.

Algorithm 1

Procedure `MixedGreedy`(g, Q, S, w, c, b)

If $g(b) = Q$ **then return** a single (unlabeled) leaf l
Let T be an empty tree
$N' \leftarrow \{i : b_i = *\}$
For $i \in N'$, $\sigma_i \leftarrow \arg\min_{\gamma \in \Gamma} \Delta g(b, i, \gamma)$
Define $g' \colon 2^{N'} \to \mathbb{Z}_{\geq 0}$ such that for all $U \subseteq N'$, $g'(U) = g(b_U) - g(b)$, where b_U is the extension of b produced by setting $b_i = \sigma_i$ for all $i \in U$.
$B \leftarrow$ `FindBudget`(N', g', c), $spent \leftarrow 0$, $spent_2 \leftarrow 0$, $k \leftarrow 1$
$I \leftarrow \{i \in N' | c_i \leq B\}$
For all $R \subseteq I$, define $D_R := \{a \in S | a \succeq b$ and $a_i \neq \sigma_i$ for some $i \in R\}$
Define $h : 2^I \to \mathbb{Z}_{\geq 0}$ such that for all $R \subseteq I$, $h(R) = \sum_{a \in D_R} w(a)$
$R \leftarrow \emptyset$
repeat
  Let i be an item which maximizes $\frac{h(R \cup \{i\}) - h(R)}{c_i}$ among all items $i \in I$
  Let t_k be a new node labeled with item i
  If $k = 1$ **then** make t_1 the root of T
  else make t_k the σ_j-child of t_{k-1}
  $j \leftarrow i$
  for every $\gamma \in \Gamma$ such that $\gamma \neq \sigma_i$ **do**
    $T^\gamma \leftarrow$ `MixedGreedy`$(g, Q, S, w, c, b_{i \leftarrow \gamma})$
    Attach T^γ to T by making the root of T^γ the γ-child of t_k
  $b_i \leftarrow \sigma_i$, $R \leftarrow R \cup \{i\}$, $I \leftarrow I - \{i\}$, $spent \leftarrow spent + c_i$, $k \leftarrow k + 1$
until $spent \geq B$
repeat
  Let i be an item which maximizes $\frac{\Delta g(b, i, \sigma_i)}{c_i}$ among all items $i \in I$
  Let t_k be a node labeled with item i
  Make t_k the σ_j-child of t_{k-1}
  $j \leftarrow i$
  for every $\gamma \in \Gamma$ such that $\gamma \neq \sigma_i$ **do**
    $T^\gamma \leftarrow$ `MixedGreedy`$(g, Q, S, w, c, b_{i \leftarrow \gamma})$
    Attach T^γ to T by making the root of T^γ the γ-child of t_k
  $b_i \leftarrow \sigma_i$, $I \leftarrow I - \{i\}$, $spent_2 \leftarrow spent_2 + c_i$, $k \leftarrow k + 1$
until $spent_2 \geq B$ **or** $I = \emptyset$
$T' \leftarrow$ `MixedGreedy`(g, Q, S, w, c, b); Attach T' to T by making the root of T' the σ_j-child of t_{k-1}
Return T

Mixed Greedy works by calling recursive function `MixedGreedy`, which we present in Algorithm 1. In the initial call, $b = (*, \ldots, *)$. Only the value of parameter b changes between recursive calls. Each call constructs a subtree of the full tree for g, rooted at a node v of that tree. In the call building the subtree rooted at v, b is the partial realization corresponding to the path from the root to v in the full tree: $b_i = \gamma$ if the path includes a node labeled i and its γ-child, and $b_i = *$ otherwise.

The algorithm of Cicalese et al. [4] is essentially the same as Mixed Greedy in the special case where g is equal to their "Pairs" function. Like their algorithm, Mixed Greedy uses a subroutine, `FindBudget`, that relies on a greedy algorithm

of Wolsey for Budgeted Submodular Cover [17]. `FindBudget` is presented in the full version [8] of this paper and is omitted here due to space constraints.

If $g(b) = Q$, then `MixedGreedy` returns an (unlabeled) single node, which will be a leaf of the full tree for g. Otherwise, `MixedGreedy` constructs a tree T. It does so by computing a special realization called σ, and then iteratively using σ to construct a path descending from the root of this subtree, which is called the *backbone*. It uses recursive calls to build the subtrees "hanging" off the backbone. The backbone has a special property: for each node v' in the path, the successor node in the path is the σ_i-child of v', where i is the item labeling node v'.

The backbone is constructed as follows. Using `FindBudget`, `MixedGreedy` computes a lower bound B on the minimum cost required to achieve a fraction of approximately $\frac{1}{3}$ of the goal value Q, assuming we start with partial realization b (Step 6).

After calculating B, `MixedGreedy` constructs the backbone in two stages, using a different greedy criterion in each to determine which item i to place in the current node. In the first stage, corresponding to the first repeat loop, the goal is to remove weight (probability mass) from the backbone, as cheaply as possible. That is, consider an $a \in \Gamma^n$ to be removed from the backbone (or "covered") if i labels a node in the backbone and $a_i \neq \sigma_i$; removing a from the backbone results in the loss of weight $w(a)$ from the backbone. The greedy choice used in the first stage in Step 12 follows the rule of maximizing *bang-for-the-buck*: the algorithm chooses i such that the amount of weight removed from the backbone, divided by c_i, is maximized. In making this choice, it only considers items that have cost at most B. The first stage ends as soon as the total cost of the items in the chosen sequence is at least B. For each item i chosen during the stage, b_i is set to σ_i.

In the second stage, corresponding to the second repeat loop, the goal is to increase utility as measured by g, under the assumption that we already have b, and that the state of each remaining item i is σ_i. The algorithm again uses a bang-for-the-buck rule, choosing the i that maximizes the increase in utility, divided by c_i (Step 23). In making this choice, it again considers only items with cost at most B. The stage ends when the total cost of the items in the chosen sequence is at least B. For each item i chosen during the stage, b_i is set to σ_i.

In Sect. 2, we defined ρ. The way B is chosen guarantees that the updates to b during the two greedy stages cause the value of $Q - g(b)$ to shrink by at least a fraction $\min\{\rho, \frac{1}{9}\}$ before each recursive call. We use this fact to prove the following theorem. The proof can be found in the full version of the paper [8].

Theorem 1. *Mixed Greedy is an approximation algorithm for Scenario Submodular Cover that achieves an approximation factor of* $O(\frac{1}{\rho} \log Q)$.

4 Scenario Mixed Greedy

We now use the Scenario-OR modification to obtain a modified version of Mixed Greedy, called Scenario Mixed Greedy, that eliminates the dependence on ρ in

the approximation bound in favor of a dependence on m, the size of the sample. Rather than running Mixed Greedy with g, it first combines g and the sample to produce a new utility function g_S, and then runs Mixed Greedy with g_S, rather than with g. Utility function g_S is produced by combining g with another utility function h_S, using the OR construction described at the end of Sect. 2. Here $h_S\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$, where $h_S(b) = m - |\{a \in S : a \succeq b\}|$ and $m = |S|$. Thus $h_S(b)$ is the total number of assignments that have been eliminated from S because they are incompatible with the partial state information in b. Utility m for h_S is achieved when all assignments in S have been eliminated. Clearly, h_S is monotone and submodular.

When the OR construction is applied to combine g and h_S, the resulting utility function g_S reaches its goal value Qm when all possible realizations of the sample have been eliminated or when goal utility is achieved for g.

In an on-line setting, Scenario Mixed Greedy uses the following procedure to determine the sequence of items to choose on an initially unknown a. We note that the third step in the procedure is present because goal utility Q must be reached even for realizations a not in S.

Scenario Mixed Greedy:

1. Construct utility function g_S by applying the OR construction to g and utility function h_S.
2. Adaptively choose a sequence of items by running Mixed Greedy for utility function g_S with goal value Qm, with respect to the sample distribution $\mathcal{D}_{S,w}$.
3. After goal value Qm is achieved, if the final partial realization b computed by Mixed Greedy does not satisfy $g(b) = Q$, then choose the remaining items in N in a fixed but arbitrary order until $g(b) = Q$.

Theorem 2. *Scenario Mixed Greedy approximates Scenario Submodular Cover with an approximation factor of $O(\log Qm)$, where m is the size of sample S.*

Proof. Scenario Mixed Greedy achieves utility value Q for g when run on any $a \in \Gamma^n$, because the b computed by Mixed Greedy satisfies $a \succeq b$, and the third step ensures Q is reached. Let $c(g)$ and $c(g_S)$ denote the expected cost of the optimal strategies for Scenario SC problems on g and g_S respectively, with respect to sample distribution $\mathcal{D}_{S,w}$. Let τ be an optimal strategy for g achieving expected cost $c(g)$. It is also a valid strategy for the problem on g_S, since it achieves utility Q for g on all realizations, and hence achieves goal utility Qm for g_S on all realizations. Thus $c(g_S) \leq c(g)$.

Functions g and h_S are monotone and submodular. Since g_S is produced from them using the OR construction, g_S is monotone and submodular. Let ρ_S be the value of parameter ρ for the function g_S. By the bound in Theorem 1, running Mixed Greedy on g_S, for the sample distribution $\mathcal{D}_{S,w}$, has expected cost that is at most a $O(\frac{1}{\rho_S} \log Qm)$ factor more than $c(g_S)$. Its expected cost is thus also within an $O(\frac{1}{\rho_S} \log Qm)$ factor of $c(g)$. Making additional choices on realizations not in S, as done in the last step of Scenario Mixed Greedy, does not affect the expected cost, since these realizations have zero probability.

Generalizing an argument from [4], we now prove that ρ_S is lower bounded by a constant fraction. Consider any $b \in (\Gamma \cup \{*\})^n$ and $i \in N$ such that $b_i = *$, and any $\gamma \in \Gamma$ where $\gamma \neq \gamma_{b,i}$. Let $C_b = |S| - h_S(b) = |\{a \in S \mid a \succeq b\}|$. Since sets $\{a \in S \mid a \succeq b \text{ and } a_i = \gamma\}$ and $\{a \in S \mid a \succeq b \text{ and } a_i = \gamma_{b,i}\}$ are disjoint, both cannot have size greater than $\frac{C_b}{2}$. Thus $\Delta h_S(b, i, \gamma) \geq \frac{C_b}{2}$ or $\Delta h_S(b, i, \gamma_{b,i}) \geq \frac{C_b}{2}$ or both. By the construction of g_S (recall the definition of the OR construction in (2)), we have that $\Delta g_S(b, i, \gamma) \geq \frac{(Q-g(b))C_b}{2}$ or $\Delta g_S(b, i, \gamma_{b,i}) \geq \frac{(Q-g(b))C_b}{2}$ or both. Since $\gamma_{b,i}$ is the "worst-case" setting for b_i with respect to g_S, it follows that $\Delta g_S(b, i, \gamma) \geq \Delta g_S(b, i, \gamma_{b,i})$, and so in all cases $\Delta g_S(b, i, \gamma) \geq \frac{(Q-g(b))C_b}{2}$. Also, $(Q - g(b))C_b = Qm - g_S(b)$. Therefore, $\rho_S \geq \frac{1}{2}$. The theorem follows from the bound in Theorem 1. □

5 Scenario Adaptive Greedy

Scenario Adaptive Greedy works by first constructing a utility function g_W, produced by applying the OR construction to g and utility function h_W. Here $h_W \colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$, where $h_W(b) = W - \sum_{a \in S : a \succeq b} w(a)$. Intuitively, $h_W(b)$ is the total weight of assignments eliminated from S because they are incompatible with the information in b. Utility W is achieved for h_W when all assignments in S have been eliminated. Clearly h_W is monotone and submodular. The function g_W reaches its goal value QW when all possible realizations of the sample have been eliminated or when goal utility is achieved for g. Once g_W is constructed, Scenario Adaptive Greedy runs Adaptive Greedy on g_W.

In an on-line setting, Scenario Adaptive Greedy uses the following procedure to determine the sequence of items to choose on an initially unknown a.

Scenario Adaptive Greedy:

1. Construct modified utility function g_W by applying the OR construction to g and utility function h_W.
2. Run Adaptive Greedy for utility function g_W with goal value QW, with respect to sample distribution $\mathcal{D}_{S,w}$, to determine the choices to make on a.
3. After goal value QW is achieved, if the partial realization b representing the states of the chosen items of a does not satisfy $g(b) = Q$, then choose the remaining items in N in arbitrary order until $g(b) = Q$.

The analysis of Scenario Adaptive Greedy is based on the following lemma.

Lemma 1. *Utility function g_W is adaptive submodular with respect to sample distribution $\mathcal{D}_{S,w}$.*

The proof of Lemma 1 can be found in the full version of the paper.

Theorem 3. *Scenario Adaptive Greedy is an approximation algorithm for Scenario Submodular Cover achieving an approximation factor of $O(\log QW)$, where W is the sum of the weights on the realizations in S.*

Proof. Since g_W is produced by applying the OR construction to g and h_W, which are both monotone, so is g_W. By Lemma 1, g_W is adaptive submodular with respect to the sample distribution. Thus by the bound of Golovin and Krause on Adaptive Greedy, running that algorithm on g_W yields an ordering of choices with expected cost that is at most a $O(\log QW)$ factor more than the optimal expected cost for g_W. By the analogous argument as in the proof of Theorem 2, it follows that Scenario Adaptive Greedy solves the Scenario Submodular Cover problem for g, and achieves an approximation factor of $O(\log QW)$. □

Acknowledgements. The work in this paper was supported by NSF Grant 1217968. L. Hellerstein thanks Andreas Krause for useful discussions at ETH, and for directing our attention to the bound of Streeter and Golovin for min-sum submodular cover. We thank an anonymous referee for suggesting the Kosaraju trick.

References

1. Bellala, G., Bhavnani, S., Scott, C.: Group-based active query selection for rapid diagnosis in time-critical situations. IEEE Trans. Inf. Theor. **58**, 459–478 (2012)
2. Chen, Y., Javdani, S., Karbasi, A., Bagnell, J.A., Srinivasa, S.S., Krause, A.: Submodular surrogates for value of information. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, USA, 25–30 January 2015, pp. 3511–3518 (2015)
3. Chen, Y., Javdani, S., Karbasi, A., Bagnell, J.A., Srinivasa, S.S., Krause, A.: Submodular surrogates for value of information (long version) (2015). http://las.ethz.ch/files/chen15submsrgtvoi-long.pdf
4. Cicalese, F., Laber, E., Saettler, A.M.: Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In: Proceedings of the 31st International Conference on Machine Learning, pp. 414–422 (2014)
5. Deshpande, A., Hellerstein, L., Kletenik, D.: Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In: Symposium on Discrete Algorithms (2014)
6. Golovin, D., Krause, A.: Adaptive submodularity: theory and applications in active learning and stochastic optimization. J. Artif. Intell. Res. **42**, 427–486 (2011)
7. Golovin, D., Krause, A., Ray, D.: Near-optimal Bayesian active learning with noisy observations. In: 24th Annual Conference on Neural Information Processing Systems (NIPS), pp. 766–774 (2010)
8. Grammel, N., Hellerstein, L., Kletenik, D., Lin, P.: Scenario submodular cover. CoRR abs/1603.03158 (2016). http://arxiv.org/abs/1603.03158
9. Guillory, A., Bilmes, J.A.: Simultaneous learning and covering with adversarial noise. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, 28 June–2 July 2011, pp. 369–376 (2011)
10. Gupta, A., Nagarajan, V., Ravi, R.: Approximation algorithms for optimal decision trees and adaptive TSP problems. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 690–701. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14165-2_58
11. Javdani, S., Chen, Y., Karbasi, A., Krause, A., Bagnell, D., Srinivasa, S.S.: Near optimal bayesian active learning for decision making. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, 22–25 April 2014, pp. 430–438 (2014)

12. Kaplan, H., Kushilevitz, E., Mansour, Y.: Learning with attribute costs. In: Symposium on the Theory of Computing, pp. 356–365 (2005)
13. Kosaraju, S.R., Przytycka, T.M., Borgstrom, R.: On an optimal split tree problem. In: Dehne, F., Sack, J.-R., Gupta, A., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 157–168. Springer, Heidelberg (1999). doi:10.1007/3-540-48447-7_17
14. Navidi, F., Kambadur, P., Nagarajan, V.: Adaptive submodular ranking. CoRR abs/1606.01530 (2016). http://arxiv.org/abs/1606.01530
15. Streeter, M., Golovin, D.: An online algorithm for maximizing submodular functions. In: Advances in Neural Information Processing Systems, pp. 1577–1584 (2009)
16. Ünlüyurt, T.: Sequential testing of complex systems: a review. Discrete Appl. Math. **142**(1–3), 189–205 (2004)
17. Wolsey, L.: Maximising real-valued submodular functions: primal and dual heuristics for location problems. Math. Oper. Res. **7**(3), 410–425 (1982)