Scenario Submodular Cover

Nathaniel Grammel[*] NGRAMMEL@NYU.EDU
Department of Computer Science and Engineering
NYU Tandon School of Engineering
Brooklyn, NY 11201

Lisa Hellerstein[*] LISA.HELLERSTEIN@NYU.EDU
Department of Computer Science and Engineering
NYU Tandon School of Engineering
Brooklyn, NY 11201

Devorah Kletenik[*] KLETENIK@SCI.BROOKLYN.CUNY.EDU
Department of Computer and Information Science
Brooklyn College, City University of New York
2900 Bedford Avenue
Brooklyn, NY 11210

Patrick Lin[*] PLIN15@ILLINOIS.EDU
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL

Abstract

Many problems in Machine Learning can be modeled as submodular optimization problems. Recent work has focused on stochastic or adaptive versions of these problems. We consider the Scenario Submodular Cover problem, which is a counterpart to the Stochastic Submodular Cover problem studied by Golovin and Krause (2011). In Scenario Submodular Cover, the goal is to produce a cover with minimum expected cost, where the expectation is with respect to an empirical joint distribution, given as input by a weighted sample of realizations. In contrast, in Stochastic Submodular Cover, the variables of the input distribution are assumed to be independent, and the distribution of each variable is given as input. Building on algorithms developed by Cicalese et al. (2014) and Golovin and Krause (2011) for related problems, we give two approximation algorithms for Scenario Submodular Cover over discrete distributions. The first achieves an approximation factor of $O(\log Qm)$, where m is the size of the sample and Q is the goal utility. The second, simpler algorithm achieves an approximation bound of $O(\log QW)$, where Q is the goal utility and W is the sum of the integer weights. (Both bounds assume an integer-valued utility function.) Our results yield approximation bounds for other problems involving non-independent distributions that are explicitly specified by their support.

[*] Partially Supported by NSF Grant 1217968

1. Introduction

Many problems in Machine Learning can be modeled as submodular optimization problems. Recent work has focused on stochastic or adaptive versions of submodular optimization problems, which reflect the need to make sequential decisions when outcomes are uncertain.

The Submodular Cover problem generalizes the classical NP-complete Set Cover problem and is a fundamental problem in submodular optimization. Adaptive versions of this problem have applications to a variety of machine learning problems that require building a decision tree, where the goal is to minimize expected cost. Examples include problems of entity identification (exact learning with membership queries), classification (equivalence class determination), and decision region identification (cf. Golovin and Krause (2011); Golovin et al. (2010); Bellala et al. (2012); Javdani et al. (2014)). Other applications include reducing prediction costs for learned Boolean classifiers, when there are costs for determining attribute values (Deshpande et al. (2014)).

Previous work on the *Stochastic* Submodular Cover problem assumes that the variables of the input probability distribution are independent. Optimization is performed with respect to this distribution. We consider a new version of the problem that we call Scenario Submodular Cover, that removes the independence assumption. In this problem, optimization is performed with respect to an input distribution that is given explicitly by its support (with associated probability weights). We give approximation algorithms solving the Scenario Submodular Cover problem over discrete distributions.

Before describing our contributions in more detail, we give some background. 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 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 utility function $g\colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$. It assigns a non-negative integer value to each subset of the items and the value can depend on the states of the items.[1] 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 must 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 the 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 a random variable, and these variables are assumed to be independent. The distributions of the variables are given as input. Golovin and Krause introduced a simple greedy algorithm for this problem, called Adaptive Greedy, that achieves an approximation factor of $O(\log Q)$. A dual greedy algorithm for the problem, called Adaptive Dual Greedy, was presented and analyzed by Deshpande et al. (2014). These greedy algorithms have been useful in solving other stochastic optimization

1. The definitions of the terms "monotone" and "submodular," for state-dependent utility functions, has not been standardized. We define these terms in Section 2. In the terminology used by Golovin and Krause Golovin and Krause (2011), g is *pointwise* monotone and pointwise submodular.

problems, which can be reduced to Stochastic Submodular Cover through the construction of appropriate utility functions (e.g., Javdani et al. (2014); Chen et al. (2015a); Deshpande et al. (2014); Golovin et al. (2010)).

The problem we study in this paper, *Scenario Submodular Cover* (Scenario SC), is also a distributional, adaptive submodular cover problem. The distribution is given by a weighted sample, which is provided as part of the input to the problem. 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, in a way that minimizes the expected cost incurred. However, because many of the proofs of results for the Stochastic Submodular Cover problem rely on the independence assumption, the proofs do not apply to the Scenario SC problem.

Results

We present an approximation algorithm for the Scenario SC problem that we call *Mixed Greedy*. It uses two different greedy criteria. It is a generalization of an algorithm by Cicalese et al. (2014) for the Equivalence Class Determination problem (which has also been called the Group Identification problem and the Discrete Function Evaluation problem).

The approximation factor achieved by Mixed Greedy for the Scenario SC problem is $O\left(\frac{1}{\rho}\log Q\right)$, where ρ is a quantity that depends on the utility function g. In the case of the utility function constructed for the Equivalence Class Determination Problem, ρ is constant, but this is not true in general.

We describe a modified version of Mixed Greedy that we call *Scenario Mixed Greedy*. It works by first constructing a new monotone, submodular utility function g_S from g and the sample, for which ρ is constant. It then runs Mixed Greedy on g_S with goal value Qm, where m is the size of the sample. We show that Scenario Mixed Greedy achieves an $O(\log Qm)$ approximation factor for any Scenario SC problem.

Mixed Greedy is very similar to the algorithm of Cicalese et al., and we use the same basic analysis. However, at the heart of their analysis is a technical lemma with a lengthy proof bounding a quantity that they call the "sepcost". The proof applies only to the particular utility function used in the Equivalence Class Determination problem. We replace this proof with an entirely different proof that applies to the general Scenario SC problem. Our proof is based on the work of Streeter and Golovin (2009) for the Min-Sum Submodular Cover problem.

In addition to presenting and analyzing Mixed Greedy, we also present another algorithm for the Scenario SC problem that we call *Scenario Adaptive Greedy*. It is a modified version of the Adaptive Greedy algorithm of Golovin and Krause. Scenario Adaptive Greedy is simpler and more efficient than Mixed Greedy, and is therefore likely to be more useful in practice. However, the approximation bound proved by Golovin and Krause for Adaptive Greedy depends on the assumption that g and the distribution defined by the sample weights jointly satisfy the *adaptive submodularity* property. This is not the case for general instances of the Scenario SC problem. We extend the approach used in constructing g_S to

give a simple, generic method for constructing a modified utility function g_W, with goal utility QW, from g, which incorporates the weights on the sample. We prove that utility function g_W and the distribution defined by the sample weights jointly satisfy adaptive submodularity. This allows us to apply the Adaptive Greedy algorithm, and to achieve an approximation bound of $O(\log QW)$ for the Scenario SC problem, where W is the sum of the weights.

Our constructions of g_S and g_W are similar to constructions used in previous work on Equivalence Class Determination and related problems (cf. Golovin et al. (2010); Bellala et al. (2012); Chen et al. (2015a,b)). Our proof of adaptive submodularity uses the same basic approach as used in previous work (see, e.g., Golovin et al. (2010); Chen et al. (2015a,b)), namely showing that the value of a certain function is non-decreasing along a path between two points; however, we are addressing a more general problem and the details of our proof are different.

We believe that our work on Adaptive Greedy should make it easier to develop efficient approximation algorithms for sample-based problems in the future. Previously, using ordinary Adaptive Greedy to solve a sample-based problem involved the construction of a utility function g, and a proof that g, together with the distribution on the weighted sample, was adaptive submodular. The proof was usually the most technically difficult part of the work (see, e.g., Golovin et al. (2010); Bellala et al. (2012); Javdani et al. (2014); Chen et al. (2015b)). Our construction of g_W, and our proof of adaptive submodularity, make it possible to achieve an approximation bound using Adaptive Greedy after proving only submodularity of a constructed g, rather than adaptive submodularity of g and the distribution. Proofs of submodularity are generally easier because they do not involve distributions and expected values. Also, the standard OR construction described in Section 2 preserves submodularity, while it does not preserve Adaptive Submodularity (Chen et al. (2015a)).

Given a monotone, submodular g with goal value Q, we can use the algorithms in this paper to immediately obtain three approximation results for the associated Scenario SC problem: running Mixed Greedy with g yields an $O\left(\frac{1}{\rho}\log Q\right)$ approximation, running Mixed Greedy with g_S yields an $O(\log Qm)$ approximation, and running Adaptive Greedy with g_W yields an $O(\log QW)$ approximation. By the results of Golovin and Krause (2011), running Adaptive Greedy with g yields an $O(\log Q)$ approximation for the associated Stochastic SC problem.

Applications

Our results on Mixed Greedy yield approximation bounds for other problems. For example, we can easily obtain a new bound for the Decision Region Identification problem studied by Javdani et al. (2014), which is an extension of the Equivalence Class Determination problem. Javdani et al. construct a utility function whose value corresponds to a weighted sum of the hyperedges cut in a certain hypergraph. We can define a corresponding utility function whose value is the *number* of hyperedges cut. This utility function is clearly monotone and submodular. Using Mixed Greedy with this utility function yields an approximation bound of $O(k \log m)$, where k is a parameter associated with the problem, and m is the size

of the input sample for this problem. In contrast, the bound achieved by Javdani et al. is $O\left(k \log\left(\frac{W}{w_{min}}\right)\right)$, where w_{min} is the minimum weight on a assignment in the sample.

We can apply our greedy algorithms to Scenario BFE (Boolean Function Evaluation) problems, which we introduce here. These problems are a counterpart to the Stochastic BFE problems[2] that have been studied in AI, operations research, and in the context of learning with attribute costs (see e.g., Ünlüyurt (2004); Deshpande et al. (2014); Kaplan et al. (2005)). In a Scenario BFE problem, we are given a Boolean function f. For each $i \in \{1, \ldots, n\}$, we are also given a cost $c_i > 0$ associated with obtaining the value of the ith bit of an initially unknown assignment $a \in \{0,1\}^n$. Finally, we are given a weighted sample $S \subseteq \{0,1\}^n$. The problem is to compute a (possibly implicit) decision tree computing f, such that the expected cost of evaluating f on $a \in \{0,1\}^n$, using the tree, is minimized. The expectation is with respect to the distribution defined by the sample weights.

Deshpande et al. (2014) gave approximation algorithms for some Stochastic BFE problems that work by constructing an appropriate 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 approximation results for analogous Scenario BFE problems. For example, using Mixed Greedy, we can show that the Scenario BFE problem for k-of-n functions has an approximation algorithm achieving a factor of $O(k \log n)$ approximation, independent of the size of the sample. Details are in Appendix B. Bounds for other functions follow easily using Scenario Mixed Greedy and Scenario Adaptive Greedy. For example, Deshpande et al. (2014) presented an algorithm achieving an $O(\log t)$ approximation for the Stochastic BFE problem for evaluating decision trees of size t. Substituting Scenario Mixed Greedy for Adaptive Greedy in this algorithm yields an $O(\log tm)$ approximation for the associated Scenario BFE problem.

We note that our Scenario BFE problem differs from the function evaluation problem by Cicalese et al. (2014). In their problem, the computed decision tree need only compute f correctly on assignments $a \in \{0,1\}^n$ that are in the sample, while ours needs to compute f correctly on all $a \in \{0,1\}^n$. To see the difference, consider the problem of evaluating the Boolean OR function, for a sample S consisting of only $a \in \{0,1\}^n$ with at least one 1. If the tree only has to be correct on $a \in S$, a one-node decision tree that immediately outputs 1 is valid, even though it does not compute the OR function. Also, in Scenario BFE we assume that the function f is given with the sample, and we consider particular types of functions f.

Organization

We begin with definitions in Section 2. In Section 3, we present the overview of the Mixed Greedy algorithm. Finally, we present Scenario Mixed Greedy in Section 4, followed by Scenario Adaptive Greedy in Section 5.

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

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* of the items 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 of items $I = \{i \mid b_i \neq *\}$ where each item $i \in I$ has state b_i. For $\gamma \in \Gamma$, the quantity $b_{i \leftarrow \gamma}$ denotes the partial realization that is identical to b except that $b_i = \gamma$. For partial realizations $b, b' \in (\Gamma \cup \{*\})^n$, 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 $b' \neq b$.

Let $g \colon (\Gamma \cup \{*\})^n \to \mathbb{Z}_{\geq 0}$ be a utility function. Utility 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 the definitions of monotonicity and submodularity to (state-dependent) utility 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 minimizing state 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:

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

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

Intuitively, right before the state of an item i is discovered, there is a certain distance from the current utility achieved to the goal utility. When the state of that item is discovered, the distance to goal is reduced by some fraction (or possibly by zero). The size of that fraction can vary depending on the state of the item. In the definition of ρ, we are concerned with the value of that fraction, not for the worst-case state in this case (leading to the smallest fraction), but for the next-to-worst case state. The parameter ρ is the smallest possible value for this fraction, 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 *s*trategy), until goal value Q is achieved, such that the expected sum of the 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 one child for each state $\gamma \in \Gamma$. 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), specifyng 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 make frequent use of a construction that we call the *standard OR construction* (cf. Guillory and Bilmes (2011); Deshpande et al. (2014)). 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 $b \in (\Gamma \cup \{*\})^n$,

$$g(b) = Q_1 Q_2 - (Q_1 - g_1(b))(Q_2 - g_2(b))$$

Suppose that on any $a \in \Gamma^n$, $g_1(a) = Q_1$ or $g_2(a) = Q_2$. Then, $g(a) = Q_1 Q_2$ for all $a \in \Gamma^n$.

3. Mixed Greedy

The Mixed Greedy algorithm is a generalization of the approximation algorithm developed by Cicalese et al. for the Equivalence Class Determination problem. That algorithm effectively 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 decision tree. If the strategy is only to be used on one realization, it is not necessary to build the entire tree. While Mixed Greedy is very similar to the algorithm of Cicalese et al, we describe it fully here so that our presentation is self-contained.

3.1. Algorithm

The Mixed Greedy algorithm builds a decision tree for Scenario SC instance (g, Q, S, w, c). The tree is built top-down. It has approximately optimal expected cost, with respect to the sample distribution $\mathcal{D}_{S,w}$ defined by S and w. Each internal node of the constructed tree has $|\Gamma|$ children, one corresponding to each state $\gamma \in \Gamma$. We refer to the child corresponding to γ as the γ-child.

The Mixed Greedy algorithm works by calling the recursive function `MixedGreedy`, whose pseudocode we present in Algorithm 1. In the initial call to `MixedGreedy`, b is set to be equal to $(*, \ldots, *)$. Only the value of b changes between the recursive calls; the other values remain fixed. Each call to `MixedGreedy` constructs a subtree of the full tree for g, rooted at a node v of that tree. In the recursive call that builds 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. for the Equivalence Class Determination problem is essentially the same as our Mixed Greedy algorithm, for g equal to their "Pairs" utility function. (There is one small difference – in their algorithm, the first stage ends right before the greedy step in which the budget B would be exceeded, whereas we allow the budget to be exceeded in the last step.) Like their algorithm, our Mixed Greedy algorithm relies on a greedy algorithm for the Budgeted Submodular Cover problem due to Wolsey. We describe Wolsey's algorithm in detail in Appendix A.1.

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 construction of the backbone is done as follows. Using subroutine `FindBudget`, `MixedGreedy` first computes a lower bound B on the minimum additional cost required in order to achieve a portion α of the goal value Q, assuming we start with partial realization b (Step 6). This computation is done using the Greedy algorithm of Wolsey (1982) described in Section A.1 in the Appendix.

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 of the pseudocode, the goal is to remove weight (probability mass) from the backbone, as cheaply and as soon as possible. That is, consider a realization $a \in \Gamma^n$ to be removed from the backbone (or "covered") if

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

Procedure `FindBudget`(I, f, c)

Let $\alpha = 1 - e^{-\chi} \approx 0.35$
Do a binary search in the interval $[0, \sum_{i \in I} c_i]$ to find the smallest B such that Wolsey's greedy algorithm for maximizing a submodular function within a budget of B, applied to f and the items in I, returns a set of items with utility at least $\alpha f(I)$
Return B

i labels a node in the spine 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 standard rule of maximizing *bang-for-the-buck*; the algorithm chooses i such that the amount of probability mass removed from the backbone, divided by the cost c_i, is maximized. However, in making this greedy 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 the bang-for-the-buck rule, choosing the i that maximizes the increase in utility, divided by the cost c_i (Step 23). In making this greedy choice, it again considers only items that have cost at most B. The 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 Section 2, we defined the value ρ. The way the value 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 ρ before each recursive call. In Appendix A, we prove this fact and use it to prove the following theorem.

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

4. Scenario Mixed Greedy

We now present a variant of Mixed Greedy that eliminates the dependence on ρ in the approximation bound in favor of a dependence on m, the size of the sample. We call this variant Scenario Mixed Greedy.

Scenario Mixed Greedy works by first modifying g to produce a new utility function g_S, and then running Mixed Greedy with g_S, rather than g. Utility function g_S is produced by combining g with another utility function h_S, using the standard OR construction described at the end of Section 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 adaptive sequence of items to choose on an initially unknown realization a.

Scenario Mixed Greedy:

1. Construct utility function g_S by applying the standard 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 $g(b) = Q$.

The third step in the procedure is present because goal utility Q must be reached for g even on realizations a that are not in S.

Theorem 2 *Scenario Mixed Greedy is an approximation algorithm for the Scenario Submodular Cover problem that achieves 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 realization $a \in \Gamma^n$, because the b computed by Mixed Greedy is such that $a \succeq b$, and the third step ensures that Q is reached.

Let $c(g)$ and $c(g_S)$ denote the expected cost of the optimal strategies for the Scenario SC problems on g and g_S respectively, with respect to the 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 goal utility Q for g on all realizations, and hence achieves goal utility Qm for g_S on all realizations. Thus $c(g_S) \leq c(g)$.

The two functions, g and h_S, are monotone and submodular. Since the function g_S is produced from them using the standard OR construction, g_S is also 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 Cicalese et al. (2014), 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 the 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, it is not possible for both of them to have size greater than $\frac{C_b}{2}$. It follows that $\Delta h_S(b, i, \gamma) \geq \frac{C_b}{2}$ or $\Delta h_S(b, i, \gamma_{b,i}) \geq \frac{C_b}{2}$ or both. By the construction of g_S, it immediately follows 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 given in Theorem 1. ■

5. Scenario Adaptive Greedy

Scenario Adaptive Greedy works by first constructing a utility function g_W, produced by applying the standard 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 that have been eliminated from S because they are incompatible with the partial

state information in b. Utility W is achieved for h_W when all assignments in S have been eliminated. It is obvious that 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 adaptive sequence of items to choose on an initially unknown realization a.

Scenario Adaptive Greedy:

1. Construct modified utility function g_W by applying the standard 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 $g(b) = Q$.

In Appendix C, we prove the following lemma.

Lemma 3 *Utility function g_W is adaptive submodular with respect to sample distribution $\mathcal{D}_{S,w}$.*

The consequence of Lemma 3 is that we may now use any algorithm designed for adaptive submodular utility functions. This gives us Theorem 4.

Theorem 4 *Scenario Adaptive Greedy is an approximation algorithm for the Scenario Adaptive Submodular Cover problem that achieves 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 3, 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 $O(\log QW)$. ■

Acknowledgments

L. Hellerstein thanks Andreas Krause for useful discussions at ETH, and especially for directing our attention to the bound of Streeter and Golovin for min-sum submodular cover.

References

G. Bellala, S. Bhavnani, and C. Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Transactions on Information Theory*, 2012.

Y. Ben-Dov. Optimal testing procedure for special structures of coherent systems. *Management Science*, 1981.

M.-F. Chang, W. Shi, and W. K. Fuchs. Optimal diagnosis procedures for k-out-of-n structures. *IEEE Transactions on Computers*, 39(4):559–564, April 1990.

Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha S. Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3511–3518, 2015a.

Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Andrew Bagnell, Siddhartha S. Srinivasa, and Andreas Krause. Submodular surrogates for value of information (long version). 2015b. URL http://las.ethz.ch/files/chen15submsrgtvoi-long.pdf.

Ferdinando Cicalese, Eduardo Laber, and Aline Medeiros Saettler. Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In *Proceedings of The 31st International Conference on Machine Learning*, pages 414–422, 2014.

A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *Symposium on Discrete Algorithms*, 2014.

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

D. Golovin, A. Krause, and D. Ray. Near-optimal Bayesian active learning with noisy observations. In *24th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 766–774, 2010.

Andrew Guillory and Jeff A. Bilmes. Simultaneous learning and covering with adversarial noise. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 369–376, 2011.

Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha S. Srinivasa. Near optimal bayesian active learning for decision making. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, pages 430–438, 2014.

H. Kaplan, E. Kushilevitz, and Y. Mansour. Learning with attribute costs. In *Symposium on the Theory of Computing*, pages 356–365, 2005.

S. Salloum. *Optimal testing algorithms for symmetric coherent systems.* PhD thesis, University of Southern California, 1979.

S. Salloum and M. Breuer. An optimum testing algorithm for some symmetric coherent systems. *Journal of Mathematical Analysis and Applications*, 101(1):170 – 194, 1984. ISSN 0022-247X. doi: 10.1016/0022-247X(84)90064-7. URL http://www.sciencedirect.com/science/article/pii/0022247X84900647.

Martin Skutella and David P. Williamson. A note on the generalized min-sum set cover problem. *Operations Research Letters*, 39(6):433 – 436, 2011.

Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.

Tonguç Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004.

Laurence Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.

Appendix A. Proof of Bound for Mixed Greedy

We first discuss the algorithm of Wolsey used in `FindBudget`.

A.1. Wolsey's Greedy Algorithm for Budgeted Submodular Cover

The Budgeted Submodular Cover problem takes as input a finite set N of items, a positive integer $B > 0$ called the *budget*, a monotone submodular set function $f : 2^N \to \mathbb{Z}_{\geq 0}$, and a vector c indexed by the items in N, such that $c_i \in \mathbb{R}_{\geq 0}$ for all $i \in N$. The problem is to find a subset $R \subseteq N$ such that $\sum_{i \in R} c_i \leq B$, and $f(R)$ is maximized.

Wolsey (1982) developed a greedy approximation algorithm for this problem. We present the pseudocode for this algorithm here, together with Wolsey's approximation bound.

Procedure `WolseyGreedy`(N, f, c, B)

$spent \leftarrow 0$, $R \leftarrow \emptyset$, $k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  Let i_k be the $i \in N$ that minimizes $\frac{f(R \cup \{i\}) - f(R)}{c_i}$ among all $i \in N$ with $c_i \leq B$
  $N \leftarrow N - \{i\}$, $spent \leftarrow spent + c_i$, $R \leftarrow R \cup \{i_k\}$
until $spent > B$ or $N = \emptyset$
if $f(\{i_k\}) \geq f(R - \{i_k\})$ **then**
  return $\{i_k\}$
else
  return $R - \{i_k\}$

Lemma 5 (Wolsey (1982)) *Let R^* be the optimal solution to the Budgeted Submodular Cover problem on instance (N, f, c, B). Let $R = \{i_1, \ldots, i_k\}$ be the set of items chosen by running Wolsey-Greedy(N, f, c, B). Let e be the base of the natural logarithm, and let χ be the solution to $e^\chi = 2 - \chi$. Then $f(R) \geq (1 - e^{-\chi}) f(R^*)$.*

A.2. Analysis of Mixed Greedy

Consider a Scenario SC instance (g, Q, S, w, c), and a partial realization $b \in (\Gamma \cup \{*\})^n$. We now consider $\texttt{MixedGreedy}(g, Q, S, w, c, b)$. It constructs a tree for the Scenario SC instance *induced by* b. In this induced instance, the item set is $N' = \{i \mid b_i = *\}$. Without loss of generality, assume that $N' = \{1, \ldots, n'\}$ for some n'. For $d \in (\Gamma \cup \{*\})^n$ such that $d \succeq b$, define $\nu(d)$ be the restriction of d to the items in N'. For $d' \in (\Gamma \cup \{*\})^{n'}$, $\nu^{-1}(d')$ denotes the extension $d \succeq d'$ to all elements in N such that $d_i = d'_i$ for $i \in N'$ and $d_i = b_i$ otherwise.

The utility function $g' : (\Gamma \cup \{*\})^{n'} \to \mathbb{Z}_{\geq 0}$ for the instance induced by b is a function on partial realizations d' of the items in N'. Specifically, for $d' \in (\Gamma \cup \{*\})^{n'}$, $g'(d') = g(\nu^{-1}(d'))$. The sample S' in the induced instance consists of the restrictions of the realizations in $\{a \in S \mid a \succeq b\}$ to the items in N'. That is, $S' = \{\nu(a) \mid a \in S, a \succeq b\}$. Note that each realization in S' corresponds to a unique realization in S. The weight function w' for the induced instance is such that for all $d' \in S'$, $w'(d') = w(\nu^{-1}(d'))$. The goal value for the induced instance is Q.

If $g(b) = Q$, then $\texttt{MixedGreedy}(g, Q, S, w, c, b)$ returns the optimal tree for the instance induced by b, which is a single (unlabeled) leaf with expected cost 0. Assume $g(b) < Q$.

For any decision tree τ for the induced instance and any realization a defined over the item set N' (or over any superset of N'), let $\kappa(\tau, a) = \sum_{i \in M} c_i$, where M is the set of items labeling the nodes on the root-leaf path followed in τ on realization a. That is, $\kappa(\tau, a)$ is the cost incurred when using tree τ on realization a.

Let τ^* be a decision tree that is an optimal solution for the induced instance. Let $C^* = \mathbb{E}[\kappa(\tau^*, a)]$ where a is a random realization drawn from $D_{S',w'}$. Thus C^* is the expected cost of an optimal solution to the induced instance. Let τ^G denote the tree output by running $\texttt{MixedGreedy}(g, Q, S, w, c, b)$.

Let $\sigma \in \Gamma^{n'}$ be such that for $i \in N'$, $\sigma_i = \arg\min_{\gamma \in \Gamma} g(b_{i \leftarrow \gamma})$. Thus, σ is the realization whose entries are computed in Step 4 of $\texttt{MixedGreedy}$.

For each node v in the tree τ^G, let $\tilde{p}(v)$ denote the probability that node v will be reached when using τ^G on a random realization a drawn from $D_{S',w'}$. Let $c_v = c_i$ where i is the item labeling node v. Consider the backbone constructed during the call to $\texttt{MixedGreedy}(g, Q, S, w, c, b)$. The backbone consists of the nodes created during the two repeat loops in this call, excluding the recursive calls. Let Y be the set of nodes in the backbone. Let $c_Y = \sum_{v \in Y} \tilde{p}(v) c_v$. Thus c_Y is the contribution of the nodes in the backbone to the expected cost of tree τ^G. The following lemma says that this contribution is no more than a constant times the expected cost of the optimal tree τ^*.

Lemma 6 $c_Y \leq 24 C^*$.

Lemma 6 is the key technical lemma in our analysis, and it is the proof of this lemma that constitutes the major difference between our analysis and the analysis in Cicalese et al. (2014). We defer the proof of this lemma to Section A.3. Using this lemma, it is easy to generalize the rest of the analysis of Cicalese et al. to obtain the proof of Theorem 1. The proofs in the remainder of this section closely follow the proofs in Cicalese et al. We present them so that this paper will be self-contained.

Let B be the budget that is computed in Line 6, with `FindBudget`, when running `MixedGreedy`(g, Q, S, w, c, b). Recall the constant α defined in `FindBudget`, based on the bound on Wolsey's Greedy algorithm (Lemma 5).

Lemma 7 *The condition at the end of the first repeat loop (spent $\geq B$) will be satisfied. Also, $\kappa(\tau^*, \sigma) \geq B$.*

Proof Trees τ^G and τ^* must achieve utility $Q - g(b)$ on realization σ. The binary search procedure in `FindBudget` finds the least budget B allowing Wolsey's greedy algorithm to achieve a total increase in utility of at least $\alpha(Q - g(b))$, on realization σ. It follows from the bound on Wolsey's greedy algorithm (Lemma 5) that on realization σ, an increase of $\alpha(Q - g(b))$ could not be achieved with a budget smaller than B. Thus, $\kappa(\tau^*, \sigma) \geq B$. ■

The next lemma clearly holds because in the two repeat loops, we only consider items of cost at most B, and we continue choosing items of cost at most B until a budget of B is met or exceeded.

Lemma 8 $\sum_{v \in Y} c_v \leq 4B$.

Let b^{final} denote the final value of b in the last recursive call, in Line 32, when running `MixedGreedy`(g, Q, S, w, c, b).

Lemma 9 $g(b^{final}) \geq g(b) + \frac{1}{9}(Q - g(b))$.

Proof Recall that $N' = \{1, \ldots, n'\}$. For any $D \subseteq N'$, let $\hat{\sigma}^D$ denote the extension of b, to $(\Gamma \cup \{*\})^n$, such that $\hat{\sigma}^D_i = \sigma_i$ (as specified in line 4 of `MixedGreedy`(g, Q, S, w, c, b)) for $i \in D$, and $\hat{\sigma}^D_i = b_i$ otherwise.

It follows from the way that B was computed in `FindBudget`, and the fact that the value of g is Q on any (full) realization of the items in N, that there is a subset $L \subseteq N'$ such that $\sum_{i \in L} c_i = B$ and $g(\hat{\sigma}^L) \geq \alpha(Q - g(b)) + g(b)$.

Let Y_1 and Y_2 be the set of items i chosen in the first and second repeat loops respectively. Thus $b^{final} = \hat{\sigma}^{Y_1 \cup Y_2}$.

Let $d_1 = g(\hat{\sigma}^{Y_1}) - g(b)$ represent the utility gained in the first repeat loop. Let $d_2 = g(\hat{\sigma}^{Y_1 \cup L}) - g(\hat{\sigma}^{Y_1})$ represent the additional utility that the items in $L \setminus Y_1$ would provide. Since $g(\hat{\sigma}^L) \geq \alpha(Q - g(b)) + g(b)$ and g is monotone, $g(\hat{\sigma}^{Y_1 \cup L}) \geq g(\hat{\sigma}^L)$, and thus $g(\hat{\sigma}^{Y_1 \cup L}) \geq \alpha(Q - g(b)) + g(b)$. So $d_1 + d_2 \geq \alpha(Q - g(b))$. At the end of the first repeat loop the items in Y_1 have been chosen. If we were to add the items in $L \setminus Y_1$ to those in Y_1, it would increase the utility by $d_2 \geq \alpha(Q - g(b)) - d_1$. Since the items in the second repeat loop are chosen greedily with respect to g (and c) until budget B is met or exceeded, or goal value Q is attained, it follows by the approximation bound on Wolsey's algorithm (Lemma 5) that the amount of additional utility added during the second repeat loop is at least α times the amount of additional utility that would be added by instead choosing the items in $L \setminus Y_1$. We thus have $g(\hat{\sigma}^{Y_1 \cup Y_2}) - g(\hat{\sigma}^{Y_1}) \geq \alpha d_2$. Adding d_1 to both sides, from the definition of d_1 we get $g(\hat{\sigma}^{Y_1 \cup Y_2}) - g(b) \geq d_1 + \alpha d_2$. We know from above that $d_2 \geq \alpha(Q - g(b)) - d_1$ so we have $g(\hat{\sigma}^{Y_1 \cup Y_2}) - g(b) \geq d_1 + \alpha\left(\alpha\left(Q - g(b)\right) - d_1\right) \geq d_1 + \alpha^2\left(Q - g(b)\right) - \alpha d_1 \geq \alpha^2\left(Q - g(b)\right)$. The lemma follows because the constant α^2 is greater than $\frac{1}{9}$. ■

We can now give the proof of Theorem 1, stating that the Mixed Greedy algorithm achieves an approximation factor of $O(\frac{1}{\rho}\log Q)$.

Proof of Theorem 1 The Mixed Greedy algorithm solves the Scenario SC instance (g, Q, S, w, c) by running recursive function `MixedGreedy`(g, Q, S, w, c, b). In the initial call, b is set to $*^n$.

Let τ^G denote the tree that is output by running `MixedGreedy`(g, Q, S, w, c, b). Let τ^* denote the optimal tree for the Scenario SC instance induced by b.

The expected cost of τ^G can be broken into the part that is due to costs incurred on items in the backbone in the top-level call to the `MixedGreedy` function, and costs incurred in the subtrees built in the recursive calls to `MixedGreedy`. The recursive calls in Steps 18 and 28 build subtrees of τ^G that are rooted at a γ-child of a node labeled i, such that $\gamma \neq \sigma_i$. It follows from the definition of ρ that the value of the partial realization used in each of these recursive calls, $b_{i\leftarrow\gamma}$ is such that $g(b_{i\leftarrow\gamma})-g(b) \geq \rho(Q-g(b))$, so $g(b_{i\leftarrow\gamma}) \geq \rho(Q-g(b))+g(b)$,

The remaining recursive call is performed on b^{final}, and by Lemma 9, $g(b^{final}) \geq \frac{1}{9}(Q - g(b))$.

Let $\eta = \min\{\rho, \frac{1}{9}\}$. Let $b^1, \ldots, b^t$ denote the partial realizations on which the recursive calls are made, and for which the value of g on the partial realization is strictly less than Q. These are the recursive calls which result in the construction of non-trivial subtrees, with non-zero cost. Note that $b^1, \ldots, b^t$ may include b^{final}. For all $j \in \{1, \ldots, t\}$, $g(b^j) \geq \eta(Q - g(b)) + g(b)$, or equivalently

$$Q - g(b^j) \leq (1 - \eta)(Q - g(b)) \tag{1}$$

For $j \in \{1, \ldots, t\}$, let τ_j^G denote the tree returned by the recursive call on b^j.

Let S' be the sample for the Scenario SC instance induced by b, so $S' = \{\nu(a) \mid a \in A\}$. Let w' be the weight function for that induced instance. Let $A_j = \{\nu(a) \mid a \in S, a \succeq b^j\}$. Let μ_j^* denote an optimal decision tree for the Scenario SC instance induced by b^j. Consider the optimal decision tree τ^* for the instance induced by b, and use it to form a decision tree τ_j^* for the instance induced by b^j as follows: for each item i such that $b_i = *$ and $b_i^j \neq *$, fix i to have state b_i^j in the tree. That is, for any node in the tree labeled i, delete all its children except the one corresponding to state b_i^j, and then delete the node, connecting the parent of the node to its one remaining child. Since μ_j^* is optimal for the induced problem, τ_j^* cannot have lower expected cost for this problem. It follows that $\sum_{a\in A_j} w'(a)\kappa(\tau_j^*, a) \geq \sum_{a\in A_j} w'(a)\kappa(\mu_j^*, a)$. Further, since $\kappa(\tau^*, a) \geq \kappa(\tau_j^*, a)$ for any $a \in A_j$,

$$\sum_{a\in A_j} w(a)\kappa(\tau^*, a) \geq \sum_{a\in A_j} w'(a)\kappa(\mu_j^*, a). \tag{2}$$

From the description of `MixedGreedy`, it is easy to verify that the A_j are disjoint subsets of S'. Therefore,

$$\sum_{a\in S'} w'(a)\kappa(\tau^*, a) = \sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau^*, a)$$

Let $W = \sum_{a\in S'} w'(a)$. For $a \in S'$, let $p(a)$ be the probability assigned to a by distribution $D_{S',w'}$, so $p(a) = w'(a)/W$. Let c_Y be the sum of the costs incurred on

the backbone of τ^G as in Lemma 6. Taking expectations with respect to $D_{S',w'}$, we have $\mathbb{E}[\kappa(\tau^G, a)] = c_Y + \sum_{j=1}^{t} \sum_{a \succeq b^j} p(a)\kappa(\tau_j^G, a)$. We can now bound the ratio between $G = \mathbb{E}[\kappa(\tau^G, a)]$ and $C^* = \mathbb{E}[\kappa(\tau^*, a)]$.

$$\begin{aligned}
\frac{G}{C^*} &= \frac{\sum_{a\in S'} w'(a)\kappa(\tau^G, a)}{\sum_{a\in S'} w'(a)\kappa(\tau^*, a)} \\
&= \frac{Wc_Y + \sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau_j^G, a)}{\sum_{a\in S'} w'(a)\kappa(\tau^*, a)} \\
&= \frac{Wc_Y}{\sum_{a\in S'} w'(a)\kappa(\tau^*, a)} + \frac{\sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau_j^G, a)}{\sum_{a\in S'} w'(a)\kappa(\tau^*, a)} \\
&\leq 24 + \frac{\sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau_j^G, a)}{\sum_{a\in S'} w'(a)\kappa(\tau^*, a)} && \text{by Lemma 6} \\
&= 24 + \frac{\sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau_j^G, a)}{\sum_{j=1}^{t} \sum_{a\in A_j} w'(a)\kappa(\tau^*, a)} \\
&\leq 24 + \max_j \frac{\sum_{a\in A_j} w'(a)\kappa(\tau_j^G, a)}{\sum_{a\in A_j} w'(a)\kappa(\mu_j^*, a)}
\end{aligned}$$

In the last line, we substitute $\kappa(\tau^*, a)$ with $\kappa(\mu_j^*, a)$ because of (2), and we use the max because of the fact that $\frac{\sum x_i}{\sum y_i} \leq \max_i \frac{x_i}{y_i}$ for $x_i, y_i > 0$.

As described above, for each j, the recursive call to `MixedGreedy` on $b = b^j$ constructs a tree τ_j^G for a Scenario SC instance I' induced by b^j, with goal value $Q - g(b^j)$. The tree μ_j^* is an optimal tree for instance I'. It follows that the ratio $\frac{\sum_{a\in A_{j'}} w(a)\kappa(\tau_j^G, a)}{\sum_{a\in A_j} w(a)\kappa(\mu_j^*, a)}$ is equal to $\frac{G_j}{C_j^*}$, where G_j and C_j^* are the values of C^* and G for the induced instance I'. Thus we have $\frac{G}{C^*} \leq 24 + \max_j \frac{G_j}{C_j^*}$.

We now prove that $\frac{G}{C^*} \leq 1 + 24\frac{1}{\eta}\ln(Q - g(b))$, when $g(b) < Q$, by induction on the total number of items $n = |N|$. The base case $n = 1$ clearly holds. Assume inductively that $\frac{G}{C^*} \leq 1 + 24\frac{1}{\eta}\ln(Q - g(b))$ when the number of items is less than n, where Q is the goal value. Then for n items, we have $\frac{G}{C^*} \leq 24 + (1 + 24\frac{1}{\eta}(\ln(Q - g(b^j))))$ for the j maximizing $\frac{G_j}{C_j^*}$. By (1), $Q - g(b^j) \leq (1 - \eta)(Q - g(b))$ so

$$\begin{aligned}
\frac{G}{C^*} &\leq 24 + \left(1 + 24\frac{1}{\eta}\ln((1-\eta)(Q - g(b))\right) \\
&\leq 1 + 24\left(1 + \frac{1}{\eta}\ln((1-\eta)(Q - g(b))\right) \\
&= 1 + 24\left(1 + \frac{1}{\eta}\ln(1-\eta) + \frac{1}{\eta}\ln(Q - g(b))\right)
\end{aligned}$$

$$\leq 1 + 24\frac{1}{\eta}\ln(Q - g(b))$$

where the last inequality holds because $1-\eta \leq e^{-\eta}$ so $\log(1-\eta) \leq -\eta$ and thus $\frac{1}{\eta}\ln(1-\eta) \leq -1$.

Since $Q \geq Q - g(b)$, the expected cost of the greedy tree τ^G constructed by the Mixed Greedy algorithm is within an $O(\frac{1}{\eta}\ln Q)$ factor of the expected cost of the optimal tree. Also, since $\eta = \min\{\rho, \frac{1}{9}\}$, we know that $\frac{1}{\eta}$ is either constant or it is equal to $\frac{1}{\rho}$. We therefore have that the expected cost of τ^G is within an $O(\frac{1}{\rho}\log Q)$ factor of the expected cost of the optimal tree. ■

A.3. Proof of Lemma 6

We now present our proof bounding the expected cost incurred on the backbone of the greedy tree. Our proof relies heavily on the work of Streeter and Golovin (2009) on the Min-Sum Submodular Cover problem. We use some of their terminology and definitions in our proof.

A.3.1. Definitions

We begin by defining a discrete version of the Min-Sum Submodular Cover problem. Let $N = \{1, \ldots, n\}$ be a set of items, and let $c \in \mathbb{Z}^n_{\geq 0}$ be a non-negative integer vector of "times" associated with those items. Let $f : 2^N \to \mathbb{Z}_{\geq 0}$ be a monotone, submodular utility function and let $Q = f(N)$. We define a *schedule* to be a finite sequence $S = \langle (i_1, \tau_1), \ldots, (i_m, \tau_m)\rangle$ of pairs in $N \times \mathbb{R}_{\geq 0}$ and refer to τ_j as the *time to process item* i_j.

For a schedule S, we define $\ell(S) = \sum_{j\geq 1}\tau_j$ to be the sum of the times spent on all items in S. Given a schedule $S = \langle (v_1, \tau_1), (v_2, \tau_2), \ldots\rangle$, we define $S_{\langle t\rangle}$ to be the schedule such that for $t \leq \ell(S)$,

$$S_{\langle t\rangle} = \langle (v_1, \tau_1), (v_2, \tau_2), \ldots, (v_k, \tau_k), (v_{k+1}, t - \textstyle\sum_{i=1}^{k}\tau_i)\rangle$$

where $k = \max\{j : \sum_{i=1}^{j}\tau_i < t\}$. For $t > \ell(S)$, we let $S_{\langle t\rangle} = S$. We refer to $S_{\langle t\rangle}$ as S *truncated at time* t.

Let f^c denote the function defined on schedules S such that $f^c(S) = \frac{1}{f(N)}f(\{i \mid (i, c_i) \in S\})$. Thus, the only pairs (i, τ) in the schedule that contribute to the value of f^c are those for which $\tau = c_i$. Where c is understood, we will omit the superscript and use f to denote both the original utility function on 2^N, and the function f^c which is defined on schedules.

We define the *cost of schedule* S, with respect to f and c, to be

$$\text{cost}(f^c, S) = \int_{t=0}^{\ell(S)} 1 - f^c(S_{\langle t\rangle})dt \tag{3}$$

We define the *Discrete Min-Sum Submodular Cover* Problem on f and c to be the problem of finding a schedule S that achieves $f^c(S) = 1$ with minimum cost.

Streeter and Golovin presented a greedy algorithm for the general Min-Sum Submodular Cover problem. In Discrete Min-Sum Submodular Cover, a pair (i, τ) can only contribute to the utility of a schedule if $\tau = c_i$. The general problem studied by Streeter and Golovin does not have this restriction.

A.3.2. Standard Greedy Algorithm for Discrete Min-Sum Submodular Cover

The algorithm of Streeter and Golovin for the general Min-Sum Submodular Cover problem uses a standard greedy approach. It adds pairs (i, τ) iteratively to the end of an initially empty schedule, using the greedy rule of choosing the pair that will result in the largest increase in utility per unit time. We call this algorithm *Standard Greedy*.

We restrict our attention to the Discrete Min-Sum Submodular Cover problem. Applied to this problem, Standard Greedy uses the greedy rule of choosing the pair (i, c_i) that will result in the largest increase in utility as measured by f^c, per unit time. The algorithm ends when the constructed schedule S satisfies $f^c(S) = 1$.

More formally, Standard Greedy uses the greedy rule below to construct a greedy schedule $G = \langle (g_1, \tau_1), (g_2, \tau_2), \dots \rangle$, where each $g_j = i$ for some $i \in N$, and $\tau_i = c_i$. Since each τ_i is determined by g_i, we drop the τ_i from the description of the schedule, and consider G to be simply a list of actions $g = \langle g_1, g_2, \dots, \rangle$.

We define $G_j = \langle g_1, g_2, \dots g_{j-1} \rangle$, where $G_1 = \langle\, \rangle$. The action g_j chosen using the greedy rule is as follows (using $\oplus$ to represent the concatenation of two schedules):

$$g_j = \underset{(i,c_i)|i \in N}{\arg\max} \left\{ \frac{f(G_j \oplus \langle (i, c_i) \rangle) - f(G_j)}{c_i} \right\} \tag{4}$$

The following theorem of Streeter and Golovin shows that the schedule constructed by Standard Greedy has a cost that is within a factor of 4 of the cost achieved by any schedule (including the optimal schedule).

Theorem 10 (Streeter and Golovin (2009)) *Let I be an instance of the Discrete Min-Sum Submodular Cover problem with time vector c, monotone submodular utility function f, and item set N. Let $\mathcal{S}$ denote the set of all schedules S for item set N and cost vector c that satisfy $f^c(S) = 1$. Let G be the schedule constructed by running Standard Greedy algorithm on instance I. Then for all $S \in \mathcal{S}$, $\text{cost}(f^c, G) \leq 4\,\text{cost}(f^c, S)$.*

A.3.3. Bound on Cost of MixedGreedy

We now return to our analysis of `MixedGreedy`(g, Q, S, w, c, b). As part of our analysis, we will prove a result similar to Theorem 10.

Without loss of generality, assume that $b = *^n$.

Recall that $c_Y = \sum_{v \in Y} \tilde{p}(v) c_v$, where Y is the set of nodes in the backbone, $\tilde{p}(v)$ is the probability that a random realization will reach node v, and c_v is the cost of the item labeling node v. Let $S^Y = \langle (i_1, c_{i_1}), \dots, (i_{k-1}, c_{i_{k-1}}) \rangle$ be the schedule such that $i_1, \dots, i_{k-1}$ is the sequence of items labeling the nodes in the backbone, from the top of the backbone and moving downwards.

Define a utility function $h_p : 2^N \to \mathbb{R}_{\geq 0}$ such that for $R \in 2^N$, $h_p(R) = 1 - \sum_{a \succeq \sigma^R} p(a)$, where σ^R is the realization in Γ^n such that $\sigma^R_i = \sigma_i$ for $i \in R$, and $\sigma^R_i = *$ otherwise. The function h_p is clearly monotone and submodular. Additionally, we can see that $\sum_{v \in Y} \tilde{p}(v) c_v$ is the cost of schedule S^Y with respect to utility function utility function h_p.

Recall that τ^* denotes the optimal strategy solving the Scenario Submodular Cover instance on g and c. Consider the sequence $j_1, \dots, j_t$ of items chosen by τ^* on realization σ.

Let $S^* = \langle (j_1, c_{j_1}), \ldots, (j_t, c_{j_t}) \rangle$. The schedule S^Y created by `MixedGreedy` is constructed greedily, using the same type of greedy rule as in (4). However, S^Y is constructed in two stages: the first stage greedily chooses with respect to h_p, and the second chooses greedily with respect to an entirely different utility function. We therefore cannot directly apply Theorem 10 to bound the cost of schedule S^Y. We deal with this by using an approach analogous to one used by Cicalese et al. (2014) (in the analysis of their Equivalence Class Determination algorithm) that allows us to concentrate only on the cost of the portion of the schedule constructed during the first stage.

To do this, we note that schedule S^Y can be expressed as the concatenation of two schedules, S^1 and S^2, where S^1 contains the i_j chosen during the first repeat loop, with their costs, and S^2 contains the i_j chosen during the second, also with their costs. Recall that $\sum_{v \in Y} \tilde{p}(v) c_v$ is the cost of schedule S^Y with respect to h_p. We can express this cost as follows:

$$\sum_{v \in Y} \tilde{p}(v) c_v = \int_{t=1}^{\ell(S^1)} 1 - h_p(S^1_{\langle t \rangle}) dt + \int_{t=0}^{\ell(S^2)} 1 - h_p(S^1 \oplus S^2_{\langle t \rangle}) dt$$

Note that $\ell(S^2) \leq 2B$, since we have assumed that each $c_i \leq B$, and the second repeat loop of `MixedGreedy` ends as soon as the last item added causes the length of S^2 to exceed B. Since h_p is monotone, the value of the second integral is at most $2B(1 - h_p(S^1))$, and the value of the first integral is at least $B(1 - h_p(S^1))$ because $\ell(S^1) \geq B$. It follows that the value of the second integral is at most twice the value of the first, so we have

$$\sum_{v \in Y} \tilde{p}(v) c_v \leq 3 \int_{t=0}^{\ell(S^1)} 1 - h_p(S^1_{\langle t \rangle}) dt$$

which yields the following inequality, allowing us to bound the total cost of S^Y by analyzing the cost of S^1.

$$\operatorname{cost}(h_p, S^Y) \leq 3 \operatorname{cost}(h_p, S^1) \tag{5}$$

Therefore, to prove Lemma 6, it suffices to bound $\int_{t=0}^{\ell(S^1)} 1 - h_p(S^1_{\langle t \rangle}) dt$, which is the cost of schedule S_1 with respect to h_p.

Schedule S^1 selects items greedily with respect to h_p. However, we cannot apply Theorem 10 to bound the cost of S_1 in terms of the cost of S^*, because only items of cost at most B are considered in greedily forming S^1, while items of cost greater than B may be included in S^*.

We will instead bound the cost of S^1 in terms of the cost of the truncated schedule $S^*_{\langle B \rangle}$. To do this, we will prove a lemma that is similar to Theorem 10. We defer its proof to the next section, since it is somewhat technical and is similar to the proof of Theorem 10. The definitions of G_j and d are as given in the previous section.

The statement of the lemma is as follows.

Lemma 11 *Let I be an instance of the Discrete Min-Sum Submodular Cover problem with time vector c, utility function f, and item set N. Let $\mathcal{S}$ denote the set of all schedules S for item set N and cost vector c satisfying $f^c(S) = f(N)$. Let $G = \langle g_1, g_2, \ldots \rangle$ be the schedule*

constructed by running Standard Greedy on instance I and let $G_j = \langle g_1, g_2, \dots g_{j-1} \rangle$, where $G_1 = \langle\rangle$. Let $B \in \mathbb{R}$ be such that $\ell(G) \geq B$ and let d be the maximum j such that $\ell(G_j) < B$. For any schedule $S \in \mathcal{S}$, $\text{cost}(f, G_d) \leq 4\,\text{cost}(f, S_{\langle B \rangle})$. Further, $\text{cost}(f, G_{d+1}) \leq 8\,\text{cost}(f, S_{\langle B \rangle})$.

We now show how to use Lemma 11 to prove Lemma 6.

Let m be such that $S^*_{\langle B \rangle} = \langle (j_1, c_{j_1}), \dots, (j_{m-1}, c_{j_{m-1}}), (j_m, \tau_{j_m}) \rangle$. By the definition of schedule truncation, $\tau_{j_m} \leq c_{j_m}$. Since the length of $S^*_{\langle B \rangle}$ is B, each of $c_{j_1}, \dots, c_{j_{m-1}}$ is at most B, but it is possible that $c_{j_m} > B$.

Consider a restricted version I' of our current Min-Sum Submodular Cover instance I in which we include only those items $i \in N$ such that $c_i \leq B$. Let N' be the set of those items. Let S' be the schedule that results from concatenating $\langle (j_1, c_{j_1}), \dots, (j_{m-1}, c_{j_{m-1}}) \rangle$ with an arbitrary sequence of pairs (i, c_i) with $i \in N'$, such that $h_p(S') = h_p(N')$. Let ℓ' denote $\ell(\langle (j_1, c_{j_1}), \dots, (j_{m-1}, c_{j_{m-1}} \rangle))$. Comparing $S'_{\langle B \rangle}$ to $S^*_{\langle B \rangle}$, both have the same first $m-1$ elements. Schedule $S^*_{\langle B \rangle}$ then has (j_m, τ_{j_m}) where $\tau_{j_m} = B - \ell'$, whereas schedule $S'_{\langle B \rangle}$ may then have multiple elements in $N' \times \mathbb{Z}_{\geq 0}$ which together have length $B - \ell'$. Because h_p is monotone, and the cost of h_p on schedule $S^*_{\langle B \rangle}$ is $\text{cost}(h_p, S^*_{\langle B \rangle}) = \int_{t=0}^{B} 1 - h_p(S^*_{\langle t \rangle}) dt$, and analogously for $S'_{\langle B \rangle}$, it immediately follows that

$$\text{cost}(h_p, S'_{\langle B \rangle}) \leq \text{cost}(h_p, S^*_{\langle B \rangle}) \tag{6}$$

Now consider the schedule S^1 that is computed during the the first stage of running `MixedGreedy`. Let G' be the greedy schedule produced by running the Greedy algorithm on instance I', with utility function h_p and times c. Because only items i with $c_i \leq B$ are considered when S^1 is constructed, and items are chosen greedily with respect to h_p, S^1 is a prefix of G'.

Let d be such that $S^1 = \langle (i_1, c_{i_1}), \dots, (i_d, c_{i_d}) \rangle$. Thus, $S^1 = G'_{d+1}$. in particular, we have that $\ell(S^1) \geq B$ and $\ell(\langle (i_1, c_{i_1}), \dots, (i_{d-1}, c_{i_{d-1}}) \rangle) < B$. It follows from (6) and from Lemma 11 that

$$\text{cost}(h_p, S^1) \leq 8\,\text{cost}(h_p, S'_{\langle B \rangle}) \leq 8\,\text{cost}(h_p, S^*_{\langle B \rangle}) \tag{7}$$

and therefore

$$\text{cost}(h_p, S^1) \leq 8\,\text{cost}(h_p, S^*) \tag{8}$$

We have that $\text{cost}(h_p, S^Y) \leq 3\,\text{cost}(h_p, S^1)$. We also have that $c_Y = \text{cost}(h_p, S^Y)$ and $C^* = \text{cost}(h_p, S^*)$. Therefore, we have

$$c_Y = \text{cost}(h_p, S^Y) \leq 3\,\text{cost}(h_p, S^1) \leq 24\,\text{cost}(h_p, S^*) = 24C^*$$

■

A.4. Proof of Lemma 11, approximation bounds for truncated schedules

We prove Lemma 11, which states that the following two properties hold:

Property 1: $\text{cost}(f, G_d) \leq 4\,\text{cost}(f, S_{\langle B \rangle})$

Property 2: $\text{cost}(f, G_{d+1}) \leq 8\,\text{cost}(f, S_{\langle B \rangle})$

The proof is similar to the proof of Streeter and Golovin for Theorem 10. [3] We will assume that $f : 2^N \to [0, 1]$. We can transform any $f : 2^N \to \mathbb{R}_{\geq 0}$ into a function of this type by scaling f so that for all $S \in 2^N$, the scaled version of $f(S)$ is equal to $\frac{f(S)-f(\emptyset)}{f(N)-f(\emptyset)}$.

Recall that f^c is the function defined on schedules S such that $f^c(S) = \frac{1}{f(N)} f(\{i \mid (i, c_i) \in S\})$. We call f^c a *job*. We refer to a pair $(i, \tau) \in N \times \mathbb{R}_{\geq 0}$ as an *action* and to τ as the *time* taken by that action.

As in Section A.3, let $G = \langle (g_1, \tau_1), (g_2, \tau_2), \dots \rangle$, denote the schedule computed by the Greedy algorithm on I and let $G_j = \langle g_1, g_2, \dots g_{j-1} \rangle$. et S be an arbitrary schedule for the instance with $f(S) = f(N)$. Let d be the maximum j such that $\ell(G_j) < B$.

We may assume without loss of generality that for every (i, τ) in S, $\tau = c_i$, since f^c does not gain any value from pairs (i, τ) with $\tau \neq c_i$. As before, we will generally omit the superscript on f^c and simply write $f(S)$.

We begin by showing that Property 1 implies Property 2.

Property 1 $\Rightarrow$ Property 2: We define $f_{G_d}(S)$, a new function defined on schedules that is derived from f. Intuitively, G_d completes some portion of the job f; we wish to consider the portion of the job that remains to be completed after the actions in G_d have been performed. The function $f_{G_d}(S)$ is defined to be the portion of the job completed by first executing schedule G_d and then executing schedule S. We express this as $f_{G_d}(S) = f(G_d \oplus S)$. Note that f_{G_d} still satisfies the essential conditions for a job as it is monotone and submodular. It should be noted, however, that unless $f(G_d) = 0$, then $f_{G_d}(\langle\rangle) \neq 0$ (equivalently, due to monotonicity, there is no schedule S for which $f_{G_d}(S) = 0$).

It is easy to show that the $\text{cost}(f_{G_d}, S)$ represents the additional cost incurred by schedule S on job f after the schedule G_d has already been executed.

$$\begin{aligned}
\text{cost}(f_{G_d}, S) &= \int_{t=0}^{\ell(S)} \left(1 - f_{G_d}(S_{\langle t \rangle})\right) dt \\
&= \int_{t=0}^{\ell(S)} \left(1 - f(G_d \oplus S_{\langle t \rangle})\right) dt \\
&= \int_{t=\ell(G_d)}^{\ell(G_d \oplus S)} \left(1 - f\left((G_d \oplus S)_{\langle t \rangle}\right)\right) dt \\
&= \int_{t=0}^{\ell(G_d \oplus S)} \left(1 - f\left((G_d \oplus S)_{\langle t \rangle}\right)\right) dt - \int_{t=0}^{\ell(G_d)} \left(1 - f(G_{d\,\langle t \rangle})\right) dt
\end{aligned}$$

Therefore, we have

$$\text{cost}(f, G_d) + \text{cost}(f_{G_d}, S) = \text{cost}(f, G_d \oplus S) \tag{9}$$

Property 1 asserts that $\text{cost}(f, G_d) \leq 4\,\text{cost}(f, S_{\langle B \rangle})$ for any schedule $S \in \mathcal{S}$. STOPPED HERE Also from this assumption, the greedy schedule for f_{G_d} is within a factor of 4 of any other schedule for f_{G_d}. Additionally, if we look at only the first action of the greedy

3. Although we give a proof only for Discrete Min-Sum Submodular Cover, the proof can easily be adapted to give the same result for the more general Min-Sum Submodular Cover problem considered by Streeter and Golovin.

schedule for f_{G_d} (i.e. action g_d), the cost incurred by this one action is less than that of the entire greedy schedule for f_{G_d}, which in turn is less than 4 times any other schedule for f_{G_d}. Thus, we also have that $\text{cost}(f_{G_d}, \langle g_d \rangle) \leq 4\,\text{cost}(f_{G_d}, S^*_{\langle B \rangle})$. Therefore, we have

$$\begin{aligned}\text{cost}(f, G_{d+1}) &= \text{cost}(f, G_d) + \text{cost}(f_{G_d}, \langle g_d \rangle) && \text{(by (9))}\\ &\leq 4\,\text{cost}(f, S^*_{\langle B \rangle}) + 4\,\text{cost}(f_{G_d}, S^*_{\langle B \rangle})\\ &\leq 8\,\text{cost}(f, S^*_{\langle B \rangle})\end{aligned}$$

since, by the monotonicity of f, $\text{cost}(f_{G_d}, S^*_{\langle B \rangle}) \leq \text{cost}(f, S^*_{\langle B \rangle})$.

Proof of Property 1: We first define a few values. The quantity $R_j = 1 - f(G_j)$ represents how much of our task remains to be completed before the jth item of the greedy schedule is chosen. We define s_j to be the "bang for the buck" earned from that item. That is, $s_j = \frac{(R_j - R_{j+1})}{\tau_j}$. Then, let $p_j = \frac{R_j}{s_j}$ for all $j \leq d$, and $p_j = 0$ for $j > d$. Let $x_j = \frac{p_j}{2}$ and let $y_j = \frac{R_j}{2}$. Also, let $\psi(x) = 1 - f(S^*_{\langle x \rangle})$.

In order to prove the theorem, we wish to show

$$\int_{t=0}^{B} \left(1 - f(S^*_{\langle t \rangle})\right) dt = \int_{x=0}^{B} \psi(x)\, dx \geq \frac{1}{4}\,\text{cost}(f, G_d)$$

We need to integrate $\psi(x)$ only up to $x = B$. When $x = B$, $\psi(x) = \psi(B)$ is the amount of the task that remains to be completed at time B under schedule the optimal schedule S^*. We associate with this amount a y_k, corresponding to the greedy schedule, where $k = \min\{\, j : y_j \leq \psi(B) \,\}$. This can be seen in Figure 1, where $x = B$ and $y = \psi(B)$ are shown as dotted lines, with y_k being the first y_j appearing below the dotted line $y = \psi(B)$.

We first present an important fact. For any schedule S, any positive integer $j \leq d$, and any $t >= 0$,

$$f(S_{\langle t \rangle}) \leq f(G_j) + t \cdot s_j \tag{10}$$

This is a consequence of the monotonicity and submodularity of f, together with the fact that the greedy algorithm always chooses the item with the best "bang for the buck". It is shown in Streeter and Golovin (2009) as Fact 1.

Using this fact, we have

$$f(S^*_{\langle x_j \rangle}) \leq f(G_j) + x_j s_j = f(G_j) + \frac{R_j}{2}$$

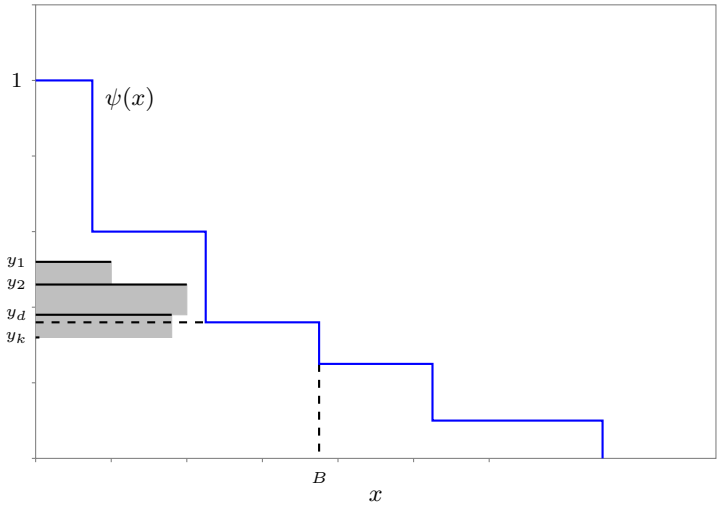
So, for $j \leq d$ we have

$$\psi(x_j) = 1 - f(S^*_{\langle x_j \rangle}) \geq 1 - f(G_j) - \frac{R_j}{2} = R_j - \frac{R_j}{2}$$

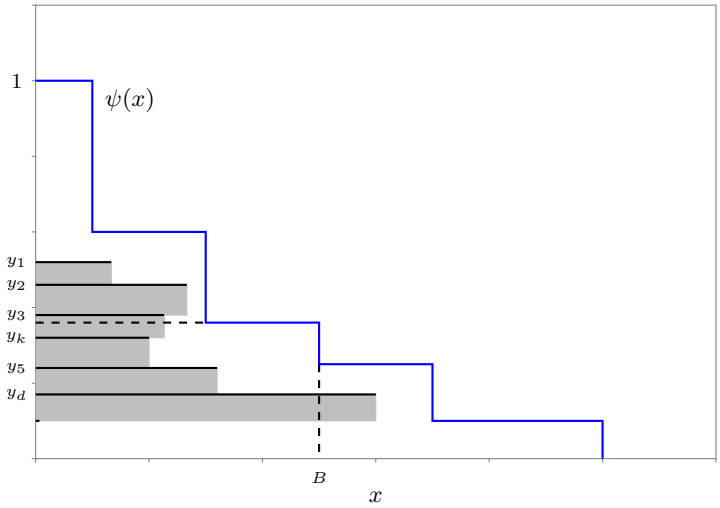
and therefore

$$\psi(x_j) \geq y_j \tag{11}$$

Note that (11) holds for $j > d$ as well, since $x_j = 0$ by definition; thus, $\psi(x_j) = \psi(0) = 1 \geq y_j$.



(a) The case where $d < k$. The area of the gray bars below y_k is 0.



(b) The case where $d > k$. The area of the gray bars below y_k is nonzero. Note that the bars below y_k may extend past $x = B$.

Figure 1: Above y_k, the gray bars fit entirely inside the area of integration of $\psi - y_k$ (the area below the $\psi(x)$ graph and above y_k). Below y_k, the total area of the gray bars is still less than the remaining area of integration for ψ (that is, the rectangle bounded above by y_k and on the right by the dotted line $x = B$)

The cost of the greedy schedule is $\sum_{j=1}^{d} R_j\tau_j$. The quantity $R_j\tau_j$ is the contribution of action j to the cost of the greedy schedule. We can think of this quantity as charging τ_j per unit of R_j. We can rewrite the contribution by instead dividing the charge per unit of utility change, $R_j - R_{j+1}$. That is, we can rewrite $R_j\tau_j$ as the product of $R_j\tau_j/(R_j - R_{j+1})$ and $R_j - R_{j+1}$. It follows from the definitions that $R_j\tau_j/(R_j - R_{j+1}) = \frac{R_j}{s_j}$ and therefore

$$x_j(y_j - y_{j+1}) = \frac{1}{4}R_j\tau_j \tag{12}$$

Since $\text{cost}(f, G_d) = \sum_{j=1}^{d} R_j\tau_j$, we now have

$$\frac{1}{4}\,\text{cost}(f, G_d) = \sum_{j=1}^{d} x_j(y_j - y_{j+1}) \tag{13}$$

The lemma now follows immediately from the following claim:

Claim 1 $\sum_{j=1}^{d} x_j(y_j - y_{j+1}) \leq \int_{x=0}^{B} \psi(x)dx$.

To prove this claim, we note that for each j, we have a pair (x_j, y_j). Figure 1 shows two histograms (represented by gray bars). For any given j, we have a gray bar such that the top of the bar is at y_j, the bottom is at y_{j+1}, and the length of the bar is x_j.

Proving the claim is equivalent to showing that the total area of the gray bars does not exceed the integral of $\psi(x)$ up to $x = B$. Combining (11) with the fact that ψ is non-increasing, it follows that for a gray bar extending to a length of x_j, the gray bar has a height no more than y_j and thus is below the graph of ψ. This allows us to conclude that the gray bars fit entirely inside of the graph of ψ. However, since we are integrating ψ only up to $x = B$, there may be some gray bars which, although they are within the graph of ψ, fall outside of the area of integration of ψ. These are the values x_j such that $x_j > B$. We note the following important fact:

Fact 1 *For all $j < k$, $x_j \leq B$.*

The justification for this fact is as follows: For any $x_j > B$, we know that $y_j \leq \psi(x_j)$ from (11) and $\psi(x_j) \leq \psi(B)$ since ψ is nonincreasing. So, since $x_j > B$ implies that $y_j \leq \psi(B)$, we know that $y_j > \psi(B)$ implies that $x_j \leq B$. For all $j < k$, by the definition of k, we know that $y_j > \psi(B)$, and thus $x_j \leq B$.

In order to show that the area of the histogram defined by the (x_j, y_j) pairs is no larger than the integral up to $x = B$ of $\psi(x)$, we will break the integral into two parts:

$$\int_{x=0}^{B} \psi(x)dx = \int_{x=0}^{B} (\psi(x) - y_k)dx + \int_{x=0}^{B} y_k dx$$

and analyze each part. We note that the first part of the integral consists of the area above the line $y = y_k$. Above this line, the reasoning follows the same reasoning as in Streeter and Golovin (2009): Due to (11), we see that each bar is contained entirely inside the graph of ψ, and since $j < k$, the bar is entirely inside the area of integration.

The second part of the integral consists of the area below $y = y_k$, where the bars are still inside the graph of ψ, but may extend past $x = B$ and thus fall outside the area of

integration. We must use different reasoning to show that the area of the bars below $y = y_k$ do not exceed the area of ψ below $y = y_k$ and left of $x = B$.

Let $B' = \ell(G_d)$. We have $B' = \sum_j \tau_j \geq \sum_{j \geq k} \tau_j$. Therefore, using (12), and the fact that $R_j \leq R_k$ for $j \geq k$,

$$\sum_{j=k}^{d} x_j(y_j - y_{j+1}) = \sum_{j=k}^{d} \frac{1}{4}\tau_j R_j \leq \frac{1}{4} R_k B' \tag{14}$$

This holds true even when $d < k$, as in this case the sum is simply 0.

Using this fact, combined with the fact that $\psi(x_j) \geq y_j$ for $j < d$, we can prove the claim. We have that

$$\int_{x=0}^{B} y_k dx = y_k B = \frac{1}{2} R_k B > \frac{1}{4} R_k B' \geq \sum_{j=k}^{d} x_j(y_j - y_{j+1}) \tag{15}$$

where the last inequality follows from (14). If we look once again at Figure 1, we see that for each j, we have a gray bar with area $x_j(y_j - y_{j+1})$. From (11), we know that the gray bars fit entirely inside $\psi(x)$, and so the area of the gray bars above y_k is not more than the area under $\psi(x)$ and above y_k. That is,

$$\int_{x=0}^{B} (\psi(x) - y_k) dx \geq \sum_{j=1}^{k-1} x_j \left(y_j - y_{j+1}\right) \tag{16}$$

By using (15) and (16), we now have

$$\begin{aligned} \int_{x=0}^{B} \psi(x)\, dx &= \int_{x=0}^{B} \left(\psi(x) - y_k\right) dx + \int_{x=0}^{B} y_k dx \\ &\geq \sum_{j=1}^{d} x_j \left(y_j - y_{j+1}\right) \end{aligned} \tag{17}$$

as desired, thus proving Claim 1. By proving Claim 1, we have therefore also proven Property 1, and thus Lemma 11. ■

Appendix B. $O(k \log n)$-approximation for Scenario k-of-n function evaluation

Let $k \in \{0, \ldots, n\}$ and let $f : \{0,1\}^n \to \{0,1\}$ be the *Boolean k-of-n function* where $f(x) = 1$ iff at least k bits of f are equal to 1. To determine the value of this f on an unknown $a \in \{0,1\}^n$, we need to determine whether f has at least k ones, or at least $n - k + 1$ zeros. There is an elegant polynomial-time exact algorithm solving the Stochastic BFE problem for Boolean k-of-n functions (cf. Salloum (1979); Salloum and Breuer (1984); Ben-Dov (1981); Chang et al. (1990)).

Here we consider the Scenario BFE problem for k-of-n functions. Following techniques used in a reduction of Deshpande et al. (2014) for Stochastic BFE, we reduce this problem

to a Scenario SC problem, through the construction of an appropriate utility function g for the state set $\Gamma = \{0, 1\}$. We obtain g by combining two other functions g_0, and g_1, with respective goal values $n - k + 1$ and k respectively, using the standard OR construction described in Section 2. Function $g_1 : \{0, 1, *\}^n \to \mathbb{Z}_{\geq 0}$ is such that for all $b \in \{0, 1, *\}^n$, $g_1(b) = \min\{k, |\{i \mid b_i = 1\}|\}$. Similarly, $g_0(b) = \min\{n - k + 1, |\{i \mid b_i = 0\}|\}$. Combining g_0 and g_1, and their goal values using the OR construction yields the new function $g : \{0, 1, *\}^n \to \mathbb{Z}_{\geq 0}$ such that for $b \in \{0, 1, *\}^n$, $g(b) = k(n - k + 1) - ((n - k + 1) - g_0(b))(k - g_1(b))$. The new goal value is $Q = k(n - k + 1)$. For $b \in \{0, 1, *\}^n$, $g(b) = Q$ iff b either contains at least $(n - k + 1)$ 0's or at least k 1's, and thus determining the value of f on initially unknown a is equivalent to achieving goal value for g.

We now lower bound the value of parameter ρ for this g. For $b \in \{0, 1, *\}^n$ where $g(b) < Q$, and i such that $b_i = *$, $\Delta_g(b, i, 1) \geq (n-k+1-g_0(b))$ and $\Delta_g(b, i, 0) \geq (k-g_1(b))$. Thus $\frac{\Delta_g(b,i,1)}{Q-g(b)} \geq \frac{n-k+1-g_0(b)}{(n-k+1-g_0(b))(k-g_1(b))} = \frac{1}{k-g_1(b)}$ and $\frac{\Delta_g(b,i,0)}{Q-g(b)} \geq \frac{k-g_1(b)}{(n-k+1-g_0(b))(k-g_1(b))} \geq \frac{1}{k}$. The larger of these is at least $\frac{1}{k}$, and hence the value of ρ for g is at least $\frac{1}{k}$. It follows that running Mixed Greedy on g with respect to the sample distribution, gives an $O(k \log n)$ approximation algorithm for our Scenario Boolean k-of-n function evaluation problem. The bound $O(k \log n)$ has no dependence on the sample size or on the weights. For constant k, this bound is $O(\log n)$.

Our Scenario k-of-n function evaluation problem has some similarities to the Generalized Min-Sum Set Cover problem, which has a constant-factor approximation algorithm (see, e.g., Skutella and Williamson (2011)). However, in the Generalized Min-Sum Set Cover problem, the goal is to find a *non-adaptive* strategy of minimum cost. Further, the sample is unweighted, and the covering requirements are different for different assignments in the input sample.

Appendix C. Adaptive Submodularity of g_W

Proof of Lemma 3 Let $w(b) = \sum_{a \in S : a \succeq b} w(a)$ be the sum of the weights of realizations in the sample S that are extensions of b. Then, we can write $h_W(b) = W - w(b)$.

The OR construction gives us

$$g_W(b) = QW - (Q - g(b))(W - h_W(b))$$

By the properties of the standard OR construction, because g and h_W are monotone and submodular, so is g_W.

Let $b, b' \in (\Gamma \cup \{*\})^n$ such that $b' \succ b$, and $i \in N$ where $b_i = b'_i = *$. To show that g_W is adaptive submodular with respect to distribution $\mathcal{D}_{S,w}$, we must show that $\mathbb{E}[\Delta g_W(b, i, \gamma)] \geq \mathbb{E}[\Delta g_W(b', i, \gamma)]$ with respect to $\mathcal{D}_{S,w}$.

We start by finding $\Delta g_W(b, i, \gamma)$ for any $b \in (\Gamma \cup \{*\})^n$, $\gamma \in \Gamma$, and $i \in N$ such that $b_i = *$:

$$\begin{aligned}
\Delta g_W(b, i, \gamma) &= QW - (Q - g(b_{i \leftarrow \gamma}))\,(W - h_W(b_{i \leftarrow \gamma})) \\
&\quad - QW + (Q - g(b))\,(W - h_W(b)) \\
&= Qh_W(b_{i \leftarrow \gamma}) + Wg(b_{i \leftarrow \gamma}) - g(b_{i \leftarrow \gamma})h_W(b_{i \leftarrow \gamma}) \\
&\quad - Qh_W(b) - Wg(b) + g(b)h_W(b) \\
&= Q\Delta h_W(b, i, \gamma) + W\Delta g(b, i, \gamma) + g(b)h_W(b) - g(b_{i \leftarrow \gamma})h_W(b_{i \leftarrow \gamma})
\end{aligned}$$

By adding and subtracting the same quantity, $g(b)h_W(b_{i\leftarrow\gamma})$, to the expression on the last line, we get

$$\begin{aligned}
\Delta g_W(b,i,\gamma) &= Q\Delta h_W(b,i,\gamma) + W\Delta g(b,i,\gamma) + g(b)h_W(b) - g(b_{i\leftarrow\gamma})h_W(b_{i\leftarrow\gamma}) \\
&\quad + g(b)h_W(b_{i\leftarrow\gamma}) - g(b)h_W(b_{i\leftarrow\gamma}) \\
&= Q\Delta h_W(b,i,\gamma) + W\Delta g(b,i,\gamma) \\
&\quad - g(b)(h_W(b_{i\leftarrow\gamma}) - h(b)) - h_W(b_{i\leftarrow\gamma})(g(b_{i\leftarrow\gamma}) - g(b)) \\
&= Q\Delta h_W(b,i,\gamma) + W\Delta g(b,i,\gamma) \\
&\quad - \Delta h_W(b,i,\gamma)g(b) - \Delta g(b,i,\gamma)h_W(b_{i\leftarrow\gamma}) \\
&= \Delta h_W(b,i,\gamma)(Q - g(b)) + \Delta g(b,i,\gamma)(W - h_W(b_{i\leftarrow\gamma}))
\end{aligned}$$

We next recall that, by definition, $h_W(b) = W - w(b)$. Thus, we have that $W - h_W(b_{i\leftarrow\gamma}) = w(b_{i\leftarrow\gamma})$, and we can simplify further:

$$\Delta g_W(b,i,\gamma) = \Delta h_W(b,i,\gamma)(Q - g(b)) + \Delta g(b,i,\gamma)w(b_{i\leftarrow\gamma})$$

We define for any partial realization d, the function $\hat{Q}(d) = Q - g(d)$ to represent the amount of utility remaining to be achieved by d. Also, let $U_\gamma = \Delta g(b,i,\gamma)$ represent the utility gained in g by observing state γ for item i in partial realization b. Let $W_\gamma = w(b_{i\leftarrow\gamma})$ represent the weight of all realizations in S consistent with $b_{i\leftarrow\gamma}$, referred to as the *total weight of state* γ. Let $\overline{W}_\gamma = \sum_{\gamma'\neq\gamma} W_{\gamma'}$ represent the total weight of all states which are not γ. It is clear that $\Delta h_W(b,i,\gamma) = \overline{W}_\gamma$. That is, the change in utility in h_W (or conceptually, the amount of weight *eliminated*) is equal to the total weight of states which are not the observed state, γ. We can now substitute these new values in the above equation and get:

$$\Delta g_W(b,i,\gamma) = \overline{W}_\gamma\,\hat{Q}(b) + U_\gamma W_\gamma$$

We now consider the calculation of the expected value of Δg_W. For a realization $a \in \Gamma^n$ drawn from $\mathcal{D}_{S,w}$, we have that $Pr[a_i = \gamma \mid a \succeq b] = \frac{w(b_{i\leftarrow\gamma})}{w(b)} = \frac{W_\gamma}{w(b)}$. Then, the expected increase in utility is

$$\begin{aligned}
\mathbb{E}[\Delta g_W(b,i,\gamma)] &= \sum_\gamma \frac{W_\gamma}{w(b)} \Delta g_W(b,i,\gamma) \\
&= \sum_\gamma \frac{W_\gamma}{w(b)} \left(\hat{Q}(b)\overline{W}_\gamma + U_\gamma W_\gamma\right) \\
&= \frac{\sum_\gamma W_\gamma \overline{W}_\gamma \hat{Q}(b) + U_\gamma W_\gamma^2}{w(b)} \\
&= \frac{\sum_\gamma W_\gamma \overline{W}_\gamma \hat{Q}(b) + U_\gamma W_\gamma^2}{\sum_\gamma W_\gamma}
\end{aligned}$$

The last equality is true since $W_\gamma = w(b_{i\leftarrow\gamma})$ and the sum of $w(b_{i\leftarrow\gamma})$ for all γ is equal to $w(b)$.

We now consider the partial realization b'. The expected value on partial realization b' is analogous to the above expected value on b:

$$\mathbb{E}[\Delta g_W(b',i,\gamma)] = \frac{\sum_\gamma W'_\gamma \overline{W}'_\gamma \hat{Q}(b') + U'_\gamma W'^2_\gamma}{\sum_\gamma W'_\gamma}$$

where $W'_\gamma = w(b'_{i\leftarrow\gamma})$, $\overline{W}'_\gamma = \sum_{\gamma'\neq\gamma} W'_{\gamma'}$, and $U'_\gamma = \Delta g(b', i, \gamma)$.

Next, let $\mathbf{W} = (W_{\gamma_1}, W_{\gamma_2}, \dots)$ be the tuple containing all of the weights of the possible states with respect to b, and let $\mathbf{U} = (U_{\gamma_1}, U_{\gamma_2}, \dots)$ be the tuple containing all of the U_γ values for each of the possible states. We also let $\mathbf{W'} = (W'_{\gamma_1}, W'_{\gamma_2}, \dots)$ and $\mathbf{U'} = (U'_{\gamma_1}, U'_{\gamma_2}, \dots)$.

It follows from the submodularity of g that $\hat{Q}(b') \leq \hat{Q}(b)$ and $U'_\gamma \leq U_\gamma$. Clearly $W'_\gamma \leq W_\gamma$. Finally, since g is monotone, and the maximum value of g on its domain is Q, $U_\gamma \leq \hat{Q}(b)$ and $U'_\gamma \leq \hat{Q}(b')$.

Now let $r = |\Gamma|$. We will use $w_{\gamma_1}, w_{\gamma_2}, \dots, w_{\gamma_r}$ to represent variables for a new function which we will define. Similarly, we will use $u_{\gamma_1}, u_{\gamma_2}, \dots, u_{\gamma_r}$ to represent variables of the same function. We will also let $\overline{w}_\gamma = \sum_{\gamma'\neq\gamma} w_{\gamma'}$ to simplify the definition of the function. The w_γ and u_γ variables are analogous to the W_γ and U_γ in the expression for expected value above. We now define our function $f\colon \mathbb{R}^{2|\Gamma|+1} \to \mathbb{R}$ such that

$$f(w_{\gamma_1}, \dots, w_{\gamma_r}, u_{\gamma_1}, \dots, u_{\gamma_r}, q) = \frac{\sum_\gamma q\overline{w}_\gamma w_\gamma + w_\gamma^2 u_\gamma}{\sum_\gamma w_\gamma}$$

Note that this function is analogous to the formula for expected value above. Specifically, we consider the point $(\mathbf{W'}, \mathbf{U'}, \hat{Q}(b'))$ and the point $(\mathbf{W}, \mathbf{U}, \hat{Q}(b))$. It should be noted that $f(\mathbf{W'}, \mathbf{U'}, \hat{Q}(b')) = \mathbb{E}[\Delta g_W(b, i, \gamma)]$ and $f(\mathbf{W}, \mathbf{U}, \hat{Q}(b)) = \mathbb{E}[\Delta g_W(b', i, \gamma)]$. Let P be the path from the first point to the second point, which increases q continuously from $\hat{Q}(b')$ to $\hat{Q}(b)$, then increases each u_{γ_i} continuously from U'_{γ_i} to U_{γ_i} for $i = 1, 2, \dots, r$, and finally increases each w_{γ_i} continuously from W'_{γ_i} to W_{γ_i} for $i = 1, 2, \dots, r$. We show that for every point along the path P, the partial derivatives of f are non-negative, and therefore the value of f is nondecreasing along the path. This proves that $f(\mathbf{W}, \mathbf{U}, \hat{Q}(b)) \geq f(\mathbf{W'}, \mathbf{U'}, \hat{Q}(b'))$. This implies that $\mathbb{E}[\Delta g_W(b, i, \gamma)] \geq \mathbb{E}[\Delta g_W(b', i, \gamma)]$, and thus g_W is adaptive submodular with respect to distribution $\mathcal{D}_{S,w}$.

We let $K = \sum_\gamma w_\gamma$. Then, we start by taking the partial derivative with respect to q:

$$\frac{\partial f}{\partial q} = \frac{\sum_\gamma (w_\gamma \overline{w}_\gamma)}{K} \geq 0$$

for all points on P since all weights are nonnegative and thus $w_\gamma \geq 0$.

We also examine the partial derivative with respect to each u_γ. Given any γ, the partial derivative is

$$\frac{\partial f}{\partial u_\gamma} = \frac{w_\gamma^2}{K} \geq 0$$

because K is positive since all weights are nonnegative (i.e. $w_\gamma \geq 0$ for all w_γ).

Finally, for each w_γ, we will use the fact that $\overline{w}_\gamma = \sum_{\gamma'\neq\gamma} w_{\gamma'}$. This means that for any γ, we can express the sum of all weights as $K = w_\gamma + \sum_{\gamma'\neq\gamma} w_{\gamma'} = w_\gamma + \overline{w}_\gamma$. This fact is

used several times in the following. We have

$$\begin{aligned}\frac{\partial f}{\partial w_\gamma} &= \frac{\left(\overline{w}_\gamma q + 2w_\gamma u_\gamma + \sum_{\gamma' \neq \gamma} w_{\gamma'} q\right) K - \left(\sum_{\gamma'} \left[w_{\gamma'} \overline{w}_{\gamma'} q + w_{\gamma'}^2 u_{\gamma'}\right]\right)}{K^2} \\ &= \frac{\left(\overline{w}_\gamma q + 2w_\gamma u_\gamma + \overline{w}_\gamma q\right)\left(w_\gamma + \overline{w}_\gamma\right) - \sum_{\gamma'} \left(w_{\gamma'} \overline{w}_{\gamma'} q + w_{\gamma'}^2 u_{\gamma'}\right)}{K^2} \\ &= \frac{\left(2\overline{w}_\gamma w_\gamma q + 2w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma\right) - \sum_{\gamma'} \left(w_{\gamma'} \overline{w}_{\gamma'} q + w_{\gamma'}^2 u_{\gamma'}\right)}{K^2}\end{aligned}$$

In the summation in the numerator, we look at the term for which $\gamma' = \gamma$ and we can simplify the numerator:

$$\frac{\partial f}{\partial w_\gamma} = \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - \sum_{\gamma' \neq \gamma} \left(w_{\gamma'} \overline{w}_{\gamma'} q + w_{\gamma'}^2 u_{\gamma'}\right)}{K^2}$$

Then, we can find a lower bound on this expression for all points on P. We note that initially, $u_\gamma \leq q$ since $U_\gamma \leq \hat{Q}(b')$ for all γ. We first increase q continuously to $\hat{Q}(b)$. Then we increase each u_γ continuously from U'_γ to U_γ. We also note that $U'_\gamma \leq \hat{Q}(b)$, and so after we have increased each u_γ we still have that $u_\gamma \leq q$. So at all points on the path we have that $u_\gamma \leq q$, and we can replace in the summation in the numerator each $u_{\gamma'}$ by q to produce our lower bound:

$$\begin{aligned}\frac{\partial f}{\partial w_\gamma} &\geq \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - \sum_{\gamma' \neq \gamma} \left(w_{\gamma'} \left(\overline{w}_{\gamma'} q + w_{\gamma'} q\right)\right)}{K^2} \\ &= \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - \sum_{\gamma' \neq \gamma} \left(q w_{\gamma'} \left(\overline{w}_{\gamma'} + w_{\gamma'}\right)\right)}{K^2} \\ &= \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - q \sum_{\gamma' \neq \gamma} \left(w_{\gamma'} K\right)}{K^2} \\ &= \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - qK \sum_{\gamma' \neq \gamma} \left(w_{\gamma'}\right)}{K^2}\end{aligned}$$

Then we note that, by definition, $\overline{w}_\gamma = \sum_{\gamma' \neq \gamma} w_{\gamma'}$, and simplify further:

$$\begin{aligned}
&= \frac{w_\gamma \overline{w}_\gamma q + w_\gamma^2 u_\gamma + 2\overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - qK\overline{w}_\gamma}{K^2} \\
&= \frac{\overline{w}_\gamma q\left(\overline{w}_\gamma + w_\gamma\right) + w_\gamma^2 u_\gamma + \overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - qK\overline{w}_\gamma}{K^2} \\
&= \frac{\overline{w}_\gamma qW + w_\gamma^2 u_\gamma + \overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma - qK\overline{w}_\gamma}{K^2} \\
&= \frac{w_\gamma^2 u_\gamma + \overline{w}_\gamma^2 q + 2w_\gamma \overline{w}_\gamma u_\gamma}{K^2} \\
&\geq 0
\end{aligned}$$

for all points on P because w_γ and u_γ are nonnegative on P.

Thus, f is nondecreasing along path P, and g_W is adaptive submodular with respect to the distribution $\mathcal{D}_{S,w}$. ■