The Online Submodular Cover Problem

Anupam Gupta*‡ Roie Levin†‡

Abstract

In the submodular cover problem, we are given a monotone submodular function $f : 2^N \to \mathbb{R}_+$, and we want to pick the min-cost set S such that $f(S) = f(N)$. This captures the set cover problem when f is a coverage function. Motivated by problems in network monitoring and resource allocation, we consider the submodular cover problem in an *online* setting. As a concrete example, suppose at each time t, a nonnegative monotone submodular function g_t is given to us. We define $f^{(t)} = \sum_{s \le t} g_s$ as the sum of all functions seen so far. We need to maintain a submodular cover of these submodular functions $f^{(1)}, f^{(2)}, \dots f^{(T)}$ in an online fashion; i.e., we cannot revoke previous choices. Formally, at each time t we produce a set $S_t \subseteq N$ such that $f^{(t)}(S_t) = f^{(t)}(N)$—i.e., this set S_t is a cover—such that $S_{t-1} \subseteq S_t$, so previously decisions to pick elements cannot be revoked. (We actually allow more general sequences $\{f^{(t)}\}$ of submodular functions, but this sum-of-simpler-submodular-functions case is useful for concreteness.)

We give polylogarithmic competitive algorithms for this online submodular cover problem. The competitive ratio on an input sequence of length T is $O(\ln n \ln(T \cdot f(N)/f_{\min}))$, where $f_{\min}$ is the smallest nonzero marginal for functions $f^{(t)}$, and $|N| = n$. For the special case of online set cover, our competitive ratio matches that of Alon et al. [AAA+09], which are best possible for polynomial-time online algorithms unless $\mathsf{NP} \subseteq \mathsf{BPP}$ [Kor04]. Since existing offline algorithms for submodular cover are based on greedy approaches which seem difficult to implement online, the technical challenge is to (approximately) solve the exponential-sized linear programming relaxation for submodular cover, and to round it, both in the online setting. Moreover, to get our competitiveness bounds, we define a (seemingly new) generalization of mutual information to general submodular functions, which we call *mutual coverage*; we hope this will be useful in other contexts.

*Department of Computer Science, New York University, New York, NY 10012. Email: anupam.g@nyu.edu.

†Department of Computer Science, Rutgers University, Piscataway, NJ 08854. Email: roie.levin@rutgers.edu.

‡The original version of this paper was written while the authors were affiliated with Carnegie Mellon University and supported by NSF awards CCF-1536002, CCF-1540541, and CCF-1617790.

1 Introduction

We consider the problem of maintaining a minimum cost submodular cover online. In the offline Submodular Cover problem (SubCov) first introduced in the seminal work of Wolsey [Wol82], we are given a monotone, submodular function f over a universe N along with a linear cost function c, and we wish to select a minimum cost set $S \subseteq N$ such that $f(S) = f(N)$. We may interpret f as modeling an abstract notion of coverage, and S as a subset of elements guaranteeing full coverage. For example, if N is a collection of subsets over a ground set and $f(S)$ computes the cardinality of the union of sets in S, SubCov exactly recovers the classical Set Cover problem. Another example is the partial cover problem and its capacitated version [Fuj00], in which f is the minimum of a coverage function and a threshold function.

Like in the case of set cover, for many applications it does not suffice to solve a single static instance of SubCov. Instead, the function f—the notion of coverage—changes over time, so we need to update our cover S. For example, in the Source Location problem [AGG+09], given a network equipped with flow demand at every vertex, we want to designate the minimum number of vertices as servers, such that every vertex can simultaneously route its demand from some server without violating edge capacities. [AGG+09] show that if $f(S)$ is the maximum demand satisfied by selecting S as the set of servers, then f is submodular and the problem can cleanly be solved using the SubCov framework. In the likely scenario that communication demand evolves over time, revoking prior decisions may be expensive or prohibitive, hence we want that our solution be monotone—i.e., we only add servers to S and not remove any. See the related works section for further examples in which avoiding revoking past decisions is useful in applications.

We ask the natural question: is it possible to maintain a competitive solution to an online SubCov instance as f changes over time? We show that under minimal monotonicity assumptions on the changes in f, the answer is yes. Formally, the online submodular cover (OSubCov) problem is the following:

> We are given a "time-monotone"[1] sequence of monotone submodular functions $f^{(1)}, f^{(2)}, \cdots$, over a common universe N that arrive online, as well as a fixed cost function c. Upon seeing $f^{(t)}$ at time t, our algorithm must output a set S_t such that $f^{(t)}(S_t) = f^{(t)}(N)$. Moreover, past decisions cannot be revoked, so we require $S_{t-1} \subseteq S_t$.

One example of this time-monotone setting was mentioned in the abstract: suppose each time t we get a new positive monotone submodular function g_t, and define $f^{(t)} := \sum_{s \le t} g_s$. It may be convenient for the reader to keep this example in mind since it captures most features of the general problem.

Theorem 1 (Main Theorem)**.** *There exists an efficient randomized algorithm for the OSubCov problem which guarantees that for each T, the expected cost of solution S_T is within $O(\ln n \cdot \ln(T \cdot f(N)/f_{\min}))$ of the optimal submodular cover solution for $f^{(T)}$.*

Here $f_{\max}$ and $f_{\min}$ are the largest and smallest marginals for any of the functions in the input sequence. When restricting to the case of online set cover, $f_{\max} = f_{\min} = 1$ and $T = f(N)$ is the number of elements of the set system, and hence this competitiveness guarantee matches that of the best current algorithm for that problem [AAA+09]. Moreover, if we restrict ourselves to efficient algorithms, the hardness result of Feige and Korman [Kor04] says that our result is the best possible up to constant factors unless NP $\subseteq$ BPP.

Our algorithm also is interesting in the *offline* setting: indeed, if we are given a single submodular function (i.e., $T = 1$), we can avoid the $O(\ln n)$ loss that comes from solving an LP relaxation online. So we get an $O(\ln(f(N)/f_{\min}))$-approximate solution via randomized rounding a fractional LP solution. This is the first LP-rounding algorithm known for SubCov, and it matches the (optimal) performance of the greedy algorithm (analyzed by dual-fitting). LP rounding approaches are powerful and versatile, since we can add side constraints to the LP and maintain them (approximately) during rounding, so we hope that our techniques will find applications even in the offline setting.

For a subclass of nonnegative, monotone, submodular functions known as 3-increasing functions, we show slightly improved results improving the $f(N)$ parameter to $f_{\max}$.

[1] This "time-monotonicity" means that any submodular cover for $f^{(t)}$ is also a cover for $f^{(t-1)}$; the functions cannot be completely unrelated to each other. This is a minimal requirement; we show in Section A that dropping it makes the problem intractable. On the other hand, we do not explicitly require that $f^{(t)}(S) \ge f^{(t-1)}(S)$.

Theorem 2. *There exists an efficient randomized algorithm for the* OSubCov *problem for* 3*-increasing functions which guarantees that for each* T*, the expected cost of solution* S_T *is within* $O(\ln n \cdot \ln(T \cdot f_{\max}/f_{\min}))$ *of the optimal submodular cover solution for* $f^{(T)}$*.*

This is analogous to improving the approximation ratio of set cover from log(number of elements) to log(maximum set size).

We note that the original conference version of this paper claimed the same $O(\ln n \cdot \ln(T \cdot f_{\max}/f_{\min}))$ guarantee for the class of all monotone submodular functions, but there was a gap in the argument. See Section 5 for a discussion of the changes in this version.

1.1 Our Techniques.

There are a few challenges to proving Theorem 1.1. A natural approach is to mimic the strategy of [AAA+09] for the online set cover problem. I.e., we can try to maintain a solution to an LP relaxation of the problem, and then perform randomized rounding online. Unfortunately, the only known LP relaxation of SubCov from [Wol82] is exponential-sized and it is not known how to solve it efficiently, nor how to analyze randomized rounding given a fractional solution. In this paper, we show that we can overcome these concerns and make the technique work.

We start in Section 2 with an $O(\ln n \ln(T \cdot f(N)/f_{\min}))$ competitive ratio algorithm for online SubCov with no efficient runtime guarantees. This allows us to illustrate the overall approach of solving the SubCov covering linear program online, in conjunction with a randomized-rounding algorithm for it. The rounding is the natural one, where we sample from the fractional solution multiple times (in an online fashion); as opposed to the set cover-style analysis which argues item-by-item, here we have to argue about the coverage as a whole. In particular, we exploit the relationship between the multilinear extension and the linearizations of submodular functions (Lemma 7) to reinterpret the constraints in the LP as statements about expected coverage under randomized rounding.

In Section 3, we improve the $O(\ln n \ln(T \cdot f(N)/f_{\min}))$-competitiveness to an $O(\ln n \ln(T \cdot f_{\max}/f_{\min}))$-competitiveness for 3-increasing functions, which is much tighter for the case where the functions f are "smoother" and each element has smaller marginal value compared to $f(N)$. The intuition behind this refined analysis is that we charge each element selected by the algorithm to the elements of Opt that "cover the same part of the space".[2] To capture the overlap between two elements, we study the *mutual coverage* $\mathcal{I}_f(A;B) := f(A) + f(B) - f(A \cup B)$, which is natural generalization of mutual information from information theory, and inherits properties from that literature, such as the chain-rule. It turns out that mutual coverage is the right abstract quantity to focus on for our analyses. We also have to strengthen the submodular cover LP slightly, and use a modified rounding-with-alterations scheme to get the tighter result. When run offline, our rounding algorithm recovers the $O(\ln f_{\max}/f_{\min})$ approximation ratio of Wolsey's algorithm.

In Section 4, we make the above algorithms run in polynomial time. The difficulty is two-fold: firstly, we do not know how to solve the submodular cover LP even in the offline setting, since the separation problem itself is APX-hard. Secondly, even if we could separate in the offline setting, naïvely solving the exponential-sized LP online requires us to give exponentially-many constraints one at a time to the online primal-dual LP-solving framework. To remedy this, we use a "round-or-separate" approach: we use the rounding algorithm itself as an approximate separation oracle. Indeed, we give an approximate separation oracle that given an arbitrary solution promises to either round it to a feasible integer solution with a similar objective value, or else to output a constraint violated by the solution. This approach to blur the line between rounding and separation may prove useful for other problems in online algorithms, where we are faced with an exponential number of new constraints at each step. Furthermore, we give a new exponential-clock based sampling procedure that finds constraints violated by a large margin with high probability. Our analysis extends a technique of [Von07] that approximates independent random sampling by a continuous process. We use a potential function argument to conclude that we only need a polynomial number of calls to the separation oracle in the worst case.

[2] The analogue in the case of set cover is an improvement from $O(\log \text{universe-size})$ to $O(\log \text{max-set-size})$. However, there is a clear notion in set cover of *items* that are covered by the same two sets (and the analysis proceeds item-by-item), whereas in SubCov it is not at all obvious what it means for the coverage of two elements to overlap.

1.2 Related Work

Submodular Cover. Wolsey [Wol82] showed that the natural greedy algorithm gives a $1 + \ln(f_{\max}/f_{\min})$ approximation for SubCov; this guarantee is tight unless P = NP even for the special case of set cover [DS14]. Fujito [Fuj00] gave a dual greedy algorithm that generalizes the $F-$approximation for set cover [Hoc82]. Neither approach seems portable to the online setting (even for set cover), since changing f to f' changes the greedy/primal-dual solution in non-smooth ways. One can also solve SubCov via repeated sequential use of submodular maximization subject to a cardinality/knapsack constraint (e.g. [Svi04] or the survey [BF17]), but once again it is not clear how to do this online when the function f changes.

SubCov has been used in many applications to resource allocation and placement problems, by showing that the coverage function is monotone and submodular, and then applying Wolsey's greedy algorithm as a black box. We port these applications to the online setting where coverage requirements change with time. E.g., in selecting influential nodes to disseminate information in social networks [GBLV13, LG16, TWPD17, IIOW10], exploration for robotics planning problems [KMGG08, JCMP17, BMKB13], placing sensors [WCZW15, RP15, ZPW+17, MW16], and other physical resource allocation objectives [YCDW15, LCL+16, TRPJ16]. There has been much recent interest in SubCov from the networking community, as SubCov models network function placement tasks [AGG+09, LPS13, KN15, LRS18, CWJ18]. E.g., [LRS18] want to place middleboxes in a network incrementally, and point out that avoiding closing extant boxes is a huge boon in practice. We extend this application to the case where the coverage function f changes.

There are previous definitions of online submodular cover, but similarity to our work is mainly in the name. In one variant inspired by regret-minimization in online learning [SG09, GB11], a submodular function $f^{(t)}$ arrives at each time step t, and we must output a permutation of the universe elements that minimizes the (average or maximum) cover time. The approach compares the algorithm's objective value to the single best *fixed* cover in hindsight. Hence, there is no notion of time-monotonicity, and of element selection being irrevocable. Another distantly related line of work ([GK11, DHK16, HK18, GHKL16, AAK19]) explores a stochastic variant of SubCov, where there is a fixed submodular function f, but each element of the universe is active only with a certain probability. The goal is to output an ordering of the elements minimizing the expected cover time.

Online Coverage Problems. In online set cover, "items" arrive online one by one; when an item arrives we find out which sets cover this item. [AAA+09, BN09] give an online algorithm with a competitive ratio of $O(\log n \log T)$, where there are n sets and T items. This is based on solving the LP relaxation via the online primal-dual framework, followed by randomized rounding. Our OSubCov problem generalizes the online set cover problem; if we identify N with the sets, and $f^{(t)}$ is the coverage function induced by the set system restricted to the first t items. [BN09, GN14] extend this result to a wide class of covering and packing LPs/IPs, with competitiveness $O(\log n \log T)$ for solving $\{0,1\}$ covering IPs with n variables and T constraints, and better results for row/column sparse instances.

[GTW14] maintain a matroid base (or spanning set) under changing costs. Given a fixed matroid $\mathcal{M}$, each matroid element is given holding and and acquisition costs at each time t, after which we must output a matroid base (spanning set). We want to minimize the sum of acquisition plus holding costs over time. In the spanning set case, when holding costs are zero, acquisition costs are fixed and the matroid is additionally changing over time, this is the very special case of our OSubCov problem where the submodular functions are matroid rank functions. Recall that matroid rank functions are a very restricted class of submodular functions: for instance, we can efficiently find a min-weight matroid base, while no efficient algorithms exist for finding a minimum weight submodular cover unless P = NP. [BCN14] study the online matroid caching problem, where one must maintain an independent set in a matroid subject to the constraints that the set must contain certain elements at specified points in time.

1.3 Notation and Preliminaries.

A set function $f : 2^N \to \mathbb{R}^+$ is *submodular* if $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$ for any $A, B \subseteq N$. It is *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq N$. We assume access to a *value oracle* for f that computes $f(T)$ given $T \subseteq N$. The *contraction* of $f : 2^N \to \mathbb{R}^+$ onto $N \setminus T$ is defined as $f_T(S) = f(S \mid T) := f(S \cup T) - f(T)$.

If f is submodular then f_T is also submodular for any $T \subseteq N$. We use the following notation:

$$f_{\max} := \max\left\{ f^{(t)}(j \mid S) \middle| j \in N,\ t \in [T],\ S \subseteq N \right\} \tag{1.1}$$

$$f_{\min} := \min\left\{ f^{(t)}(j \mid S) \middle| j \in N,\ t \in [T],\ S \subseteq N,\ f^{(t)}(j \mid S) \neq 0 \right\} \tag{1.2}$$

Note that if $f^{(t)} = \sum_{s \leq t} g_s$ is the prefix sum of monotone submodular functions arriving online, then $f_{\max}, f_{\min}$ are at most T times maximum and minimum marginals of these g_s functions. Also we let $c_{\max}$ and $c_{\min}$ denote the largest and smallest costs of elements respectively.

The well-known *multilinear extension* of f is defined as:

$$F(x) := \mathop{\mathbb{E}}_{S \sim x}[f(S)] = \sum_{S \subseteq N} \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j) f(S) \tag{1.3}$$

where $S \sim x$ means we add each element $j \in N$ independently to S with probability x_j. We can naturally extend definition (1.3) to contractions of functions: $F_T(x) := \mathbb{E}_{S \sim x}[f_T(S)]$.

Time-monotonicity generalizes the notion of an insertion-only arrival model for coverage functions, and is consistent with the (insertion-only) formulation of online set cover as defined in [AAA+09].

Definition (Time-monotone)**.** A sequence of monotone submodular functions $f^{(0)}, f^{(1)}, f^{(2)}, \ldots, f^{(T)}$ is *time-monotone* if for all t and S, we have $f^{(t)}(S) = f^{(t)}(N) \implies f^{(t-1)}(S) = f^{(t-1)}(N)$.

1.4 Higher Order Monotonicity and Mutual Coverage.

To describe some of our results, we require some definitions which generalize the idea of monotonicity and submodularity.

Following the notation of [FH05], we define the *derivative of a set function* f as:

$$\frac{df}{dx}(S) = f(S \cup \{x\}) - f(S \backslash \{x\}).$$

We notate the m-th order derivative of f with respect to the subset $A = \{i_1, \ldots i_m\}$ as df/dA, and this quantity has the following concise expression:

$$\frac{df}{dA}(S) = \sum_{B \subseteq A} (-1)^{|B|} f((S \cup A) \backslash B).$$

Definition (m-increasing [FH05])**.** We say that a set function is *m-increasing* if all its m-th order derivatives are nonnegative, and *m-decreasing* if they are nonpositive. We denote by $\mathcal{D}_m^+$ and $\mathcal{D}_m^-$ the classes of m-increasing and m-decreasing functions respectively.

Note that $\mathcal{D}_1^+$ is the class of monotone set functions, and $\mathcal{D}_2^-$ is the class of submodular set functions. When f is the joint entropy set function, the m-th derivative above is also known as the *interaction information*, which generalizes the usual mutual information for two sets of variables, to m sets of variables. We now focus on the usual $m = 2$ case, but relax f to be a general nonnegative, monotone, submodular function.

Definition (Mutual Coverage)**.** The *mutual coverage* and *conditional mutual coverage* with respect to a set function $f : 2^N \to \mathbb{R}^+$ are defined as:

$$\mathcal{I}_f(A; B) := f(A) + f(B) - f(A \cup B) \tag{1.4}$$

$$\mathcal{I}_f(A; B \mid C) := f_C(A) + f_C(B) - f_C(A \cup B) \tag{1.5}$$

Independently, [IKBA21] defined and studied the same quantity under the slightly different name *submodular mutual information*.

We write $\mathcal{I}_f^{(A)}(B) := \mathcal{I}_f(A; B)$ and $\mathcal{I}_f^{(A|C)}(B) := I(A; B \mid C)$ if we want to view mutual coverage as a function of B for fixed A and C. We may think of $\mathcal{I}_f^{(A|C)}(B)$ intuitively as the amount of coverage B takes away from the coverage of A given that C was already chosen (or vice-versa, since the definition is symmetric in A and B).

Fact 3 (Chain Rule). *Mutual coverage respects the identity:*

$$\mathcal{I}_f(A; B_1 \cup B_2 \mid C) = \mathcal{I}_f(A; B_1 \mid C) + \mathcal{I}_f(A; B_2 \mid C \cup B_1)$$

This neatly generalizes the chain rule for mutual information. The expression $\mathcal{I}_f^{(A|C)}(B)$ is monotone in B by the submodularity of f, but not submodular in general (see Section B).

We will soon show improved algorithms for the class $\mathcal{D}_3^+$, that is the class of 3-increasing set functions. The following derivation gives some intuition for these functions.

$$\frac{df}{d\{x,y,z\}}(S) = \sum_{B \subseteq \{x,y,z\}} (-1)^{|B|} f((S \cup \{x,y,z\}) \backslash B)$$
$$= f(x \mid S) - f(x \mid S \cup \{y\}) - (f(x \mid S \cup \{z\}) - f(x \mid S \cup \{y,z\})) = \mathcal{I}_f(x, y \mid S) - \mathcal{I}_f(x, y \mid S \cup \{z\}).$$

Thus a function f in contained in $\mathcal{D}_3^+$ if and only if mutual coverage decreases after conditioning, in other words mutual coverage is submodular in each of its arguments.

2 An $O\left(\ln n \ln\left(T \cdot f(N)/f_{\min}\right)\right)$ competitive algorithm

In this section, we show how to get our first algorithm for OSubCov: we use the approach of solving covering LPs online to maintain a competitive fractional solution x_t at every time t, with the special property that $x_1, x_2, \ldots, x_T$ are monotonically increasing, i.e., $x_t \geq x_{t-1}$. To get an integer solution online, we use randomized rounding with the method of alterations: we round the increments in variables at each step, and make corrections in the case of failures by greedily selecting elements until our solution is feasible. Our algorithm always outputs a feasible solution; the challenge is to bound its expected cost: the crucial technical component is a relationship between the submodular cover LP and the multilinear relaxation, which allows us to show that a logarithmic number of rounds suffice.

2.1 An LP for Submodular Cover.

How can we frame submodular cover as a linear program? We use the well known formulation used by Wolsey [Wol82] (see also [NWF78, FNW78]), but first we define some notation for convenience. For $S \subseteq N$, define $f^{\#S} : [0,1]^{|N|} \to \mathbb{R}^+$ as:

$$f^{\#S}(x) := f(S) + \textstyle\sum_{j \in N} f_S(j) x_j \tag{2.1}$$

We define the *covering extension* of f as:

$$f^*(x) := \min_{S \subseteq N} f^{\#S}(x) \tag{2.2}$$

When f is submodular, the function f^* is a continuous extensions of f: i.e., if χ_T is the characteristic vector of a set $T \subseteq N$, then $f^*(\chi_T) = F(\chi_T) = f(T)$. We will also sometimes need to extend definitions (2.1) and (2.2) to contractions of functions.

$$f_T^{\#S}(x) := f_T(S) + \sum_{j \in N} f_{S \cup T}(j) x_j$$
$$f_T^*(x) := \min_{S \subseteq N} f_T^{\#S}(x)$$

Now we can write the submodular cover LP as:

$$\begin{array}{lll} & \min \sum_j c(j)\, x_j & \\ \text{s.t.} & \sum_{j \in N} f_S(j)\, x_j \geq f_S(N) & \forall S \subseteq N \\ & 1 \geq x_j \geq 0 & \forall j \end{array}$$

or in our notation:

$$(P) \qquad \begin{array}{lll} & \min \sum_j c(j)\, x_j & \\ \text{s.t.} & f^*(x) \geq f(N) & \\ & 1 \geq x_j \geq 0 & \forall j \end{array}$$

The integer solutions of this LP are precisely the solutions to SubCov:

Lemma 4 (Proposition 2 of [Wol82])**.** *A set T has $f(T) = f(N)$ if and only if χ_T, the characteristic vector of T, is a feasible integer solution to (P).*

Let $(P^{(t)})$ refer to the program (P) for $f^{(t)}$. We would like to argue that we can maintain one large LP for OSubCov. At time t, we feed in the constraints of $(P^{(t)})$, and update the solution accordingly. We use the following online LP solver from [BN09, Theorem 4.2]:

Theorem 5. *For the setting where the rows of a covering LP $\min\{c^\intercal x \mid Ax \geq b\}$ with $A \in \mathbb{R}_+^{m \times n}$, $b \in \mathbb{R}_+^m$, $c \in \mathbb{R}_+^n$ arrive online, there is an algorithm that achieves a competitive ratio of $O(\ln n)$. Furthermore, the sequence of solutions produced $x_1, x_2, \ldots, x_T$ is monotonically increasing.*

Each of the linear programs $(P^{(t)})$ is indeed a covering LP with box constraints that is a feasible relaxation for the function $f^{(t)}$. However it is not *a priori* clear that the constraints for the linear program $(P^{(t')})$ for some $t' < t$ are also valid for the linear program $(P^{(t)})$. We prove a lemma which circumvents this issue. While we do not guarantee that for all $t' < t$ the constraints of $(P^{(t')})$ will be valid for the LP $(P^{(t)})$ itself, we show that they are valid for the *integer* solutions of $(P^{(t)})$. This suffices to ensure that the union of all constraints seen so far is a relaxation for OSubCov.

Lemma 6. *Any feasible integer solution to $(P^{(t+1)})$ is a feasible integer solution to $(P^{(t)})$.*

Proof. Suppose x is a feasible integer solution to $(P^{(t+1)})$. Then by Lemma 4, x is the characteristic vector of a set S such that $f^{(t+1)}(S) = f^{(t+1)}(N)$. By the time-monotonicity of the sequence $f^{(0)}, f^{(1)}, \ldots, f^{(T)}$, it holds that $f^{(t)}(S) = f^{(t)}(N)$ as well. Now again using Lemma 4, x is feasible to $(P^{(t)})$. □

2.2 Rounding Fractional Solutions.

The last remaining step in our plan is to understand the behavior of randomized rounding for (P). Since the multilinear extension G is defined as the value of g on a set obtained by randomized rounding, we can use the following relationship between G and g^* [Von07, Lemma 3.8]:

Lemma 7. *For any monotone, nonnegative set function g, it holds that $G(x) \geq (1 - e^{-1})g^*(x)$.*

Observe that Lemma 7 does not require that g be submodular, only that it be monotone and nonnegative. (There exist functions for which the bound is tight. If g is also submodular, we also get the bound $G(x) \leq g^*(x)$ [Von07, Lemma 3.7].) The next lemma helps us analyze random rounding:

Lemma 8 (Rounding Lemma)**.** *Let g be a monotone, nonnegative set function such that $g(\emptyset) = 0$, and let k be a positive integer. Let $R_1, \ldots R_k$ be a sequence of random sets and denote $R_{1:\ell} := \bigcup_{i \in [\ell]} R_i$. Suppose the following condition holds for all $\ell \in [k]$ and all $R \subseteq N$:*

$$\mathop{\mathbb{E}}_{R_\ell}\left[g(R_\ell \mid R_{1:\ell-1}) \mid R_{1:\ell-1} = R\right] \geq (1-\gamma)g(N \mid R) \tag{2.3}$$

Then:

$$\mathop{\mathbb{E}}_{R_{1:k}}\left[g(N \mid R_{1:k})\right] \leq \gamma^k \cdot g(N) \tag{2.4}$$

Proof. By definition:

$$\begin{aligned} \mathop{\mathbb{E}}_{R_{1:\ell}}\left[g(N \mid R_{1:\ell})\right] &= \mathop{\mathbb{E}}_{R_{1:\ell-1}}\left[g(N) - g(R_{1:\ell-1}) - \mathop{\mathbb{E}}_{R_\ell}\left[g(R_{1:\ell}) - g(R_{1:\ell-1})\right]\right] \\ &\leq \mathop{\mathbb{E}}_{R_{1:\ell-1}}\left[g(N \mid R_{1:\ell-1}) - (1-\gamma)(g(N \mid R_{1:\ell-1}))\right] \end{aligned} \tag{2.5}$$

$$= \gamma \cdot \mathop{\mathbb{E}}_{R_{1:\ell-1}} [g(N) - g(R_{1:\ell-1})]$$

Note that (2.5) holds by assumption (2.3). Claim (2.4) holds by induction on ℓ and the fact that $g(\emptyset) = 0$. □

Lemma 9. *Let $x \in [0,1]^n$ be a feasible solution to (P). Let R be a set obtained by performing k rounds of randomized rounding according to x. Then: $\mathbb{E}_R[f_R(N)] \leq e^{-k} f(N)$.*

Proof. Since x is feasible to (P), for all $S \subseteq N$, we have $f_S^*(x) \geq f_S(N)$. By Lemma 7 applied to f_S, it holds that $F_S(x) \geq (1 - e^{-1}) f_S^*(x)$. Together these imply:

$$F_S(x) \geq (1 - e^{-1}) f_S(N) \tag{2.6}$$

Thus x, f and R together satisfy the conditions of the Rounding Lemma with $\gamma = e^{-1}$, and the claim follows directly. □

2.3 The Analysis.

To summarize, our first algorithm for OSubCov is the following: maintain a fractional solution x, and two sets $R, G \subseteq N$ initially empty. At time t, feed the batch of constraints of $(P^{(t)})$ to [BN09] and update x accordingly. Set $k := \ln(t^2 \cdot f(N)/f_{\min})$, and for every $j \in N$, add j to R with probability Δ_j, where Δ_j is calculated such that the total probability that j gets added to R over the course of the algorithm is $1 - (1 - x_j)^k$. In a subsequent greedy phase, iteratively select the cheapest element $j \in N$ with non-zero marginal value and add it to the set G until no such elements remain. Return $R \cup G$.

Given our tools from the previous section, we can now analyze the expected cost.

Theorem 10. *There exists a randomized algorithm for the OSubCov problem which guarantees that for each T, the expected cost of solution S_T is within $O(\ln n \cdot \ln(T \cdot f(N)/f_{\min}))$ of the optimal submodular cover solution for $f^{(T)}$.*

Proof. By construction, the algorithm always produces a feasible solution to the submodular cover problem. Thus it suffices to relate the cost of the output to the cost of the optimal solution, $c(\textsc{Opt})$.

To bound the cost of the sampling phase, remark that at time t, by construction we can view R as the result of k rounds of independent random rounding. The expected cost after one round of rounding is precisely $c(x)$. By linearity of expectation, the total expected cost of R is $\ln\left(t^2 \cdot f(N)/f_{\min}\right) \cdot c(x)$.

Next we bound the cost of the greedy phase. At time t, the solution x is feasible to $(P^{(t)})$. By Lemma 9 with parameter $k = \ln\left(t^2 \cdot f(N)/f_{\min}\right)$:

$$\mathop{\mathbb{E}}_{R}[f_R(N)] \leq t^{-2} \left(\frac{f_{\min}}{f(N)}\right) f(N) = t^{-2} f_{\min}$$

This gives a bound on the expected cost of alterations at time t. Let G_t denote the elements added in the greedy phase at time t:

$$\mathop{\mathbb{E}}_{R}[c(G_t)] \leq c(\textsc{Opt}) \cdot \mathop{\mathbb{E}}_{R}[|G_t|] \tag{2.7}$$

$$\leq t^{-2} c(\textsc{Opt}) \tag{2.8}$$

(2.7) comes from the fact that each element must be cheaper than $c(\textsc{Opt})$, the second step (2.8) from the fact that for every element $h \in G$, both $f_R(h) \geq f_{\min}$ and $\mathbb{E}_R[f_R(N)] \leq t^{-2} f_{\min}$. By linearity of expectation, the expected cost of cumulative alterations over all time steps $t \in [T]$ is $O(c(\textsc{Opt}))$. In conclusion, the final solution has expected cost at most $\ln(T \cdot f(N)/f_{\min})\, c(x) + O(c(\textsc{Opt}))$. By Theorem 2.1, $c(x) = O(\ln n) \cdot c(\textsc{Opt})$, hence the algorithm outputs a solution with competitive ratio $O(\ln n \ln(T \cdot f(N)/f_{\min})) \cdot c(\textsc{Opt})$. □

In the next section, we show how to replace the $f(N)$ term in the competitiveness by an $f_{\max}$ term, which is often much smaller. In order to do the finer analysis, we will also show how to use the ideas of mutual coverage. We then make these algorithms efficient in the subsequent section.

3 An $O\left(\ln n \ln\left(T \cdot f_{\max}/f_{\min}\right)\right)$ competitive algorithm for $\mathcal{D}_3^+$

In this section, we show how to recover the finer $O\left(\ln n \ln\left(T \cdot f_{\max}/f_{\min}\right)\right)$ approximation for the special case of 3-increasing functions. The main algorithmic change is to perform $\ln\left(T \cdot f_{\max}/f_{\min}\right)$ rounds of rounding instead of $\ln\left(T \cdot f(N)/f_{\min}\right)$, and then compensate with more greedy alterations. The intuition behind the analysis is a fractional charging scheme. Using mutual coverage as a notion of contribution, we argue that:

(i) every greedily chosen element is cheaper than the elements in Opt to which it contributes.

(ii) every greedily chosen element contributes at least a minimum amount overall.

(iii) for every element in Opt, the total contribution it receives from greedily chosen elements is bounded above.

We show that these statements together imply a good bound on the expected cost of alterations. However, this proof strategy requires that the mutual coverage between pairs of elements decrease sufficiently after randomized rounding. In Section B, we show that the constraints of the LP (P) do not necessarily guarantee that this is the case; hence we introduce additional constraints that enforce the desired property.

3.1 A Stronger Formulation.

Recall our definition of mutual coverage $\mathcal{I}_f(A;B) = f(A) + f(B) - f(A \cup B)$ from Section 1.3. We begin by strengthening LP (P) with additional constraints as follows:

$$
(Q) \qquad \boxed{\begin{aligned} &\min \sum_j c(j)x_j \quad \text{s.t.} \\ &\sum_{j \in N} \mathcal{I}_f^{(v|S)}(j)x_j \geq \mathcal{I}_f^{(v|S)}(N) \qquad \forall v \in N, \forall S \subseteq N \quad (*) \\ &1 \geq x_j \geq 0 \qquad\qquad\qquad\qquad \forall j \end{aligned}}
$$

Note that by assumption, that $f \in$, the function $\mathcal{I}_f^{(\{v\})}$ is submodular, and hence the program above is the intersection of several submodular cover LPs (one for each v).

To see that the new constraints imply the constraints of (P), consider any set S. Order the elements of N arbitrarily $v_1, v_2 \ldots, v_n$ and define $S_i := S \cup \{v_1, \ldots v_{i-1}\}$ with $S_1 := S$. If we sum the $(*)$ constraints corresponding to all pairs (v_i, S_i), we obtain:

$$
\begin{aligned}
&\sum_{i \in [n]} \sum_{j \in N} \mathcal{I}_f(v_i; j \mid S_i) \cdot x_j \geq \sum_{i \in [n]} \mathcal{I}_f(v_i; N \mid S_i) \\
&\Rightarrow \sum_{j \in N} \mathcal{I}_f(N; j \mid S) \cdot x_j \geq \mathcal{I}_f(N; N \mid S) \qquad (3.1) \\
&\Rightarrow \sum_{j \in N} f_S(j) \cdot x_j \geq f_S(N)
\end{aligned}
$$

where we used the Chain Rule for mutual coverage for step (3.1).

In order to treat our problem as an online covering LP and use [BN09] like we did before, we again need to make sure the new constraints do not eliminate any integer solutions.

Lemma 11. *(Q) is a relaxation of* SubCov.

Proof. Consider a feasible solution T to SubCov and its corresponding characteristic vector χ_T. Order the elements of T arbitrarily $j_1, j_2 \ldots, j_k$ and define $S_i := S \cup \{j_1, \ldots j_{i-1}\}$ with $S_1 := S$. Then for any $S \subseteq N$ and $v \in N - S$:

$$
\sum_{j \in N} \mathcal{I}_f(v, j \mid S)(\chi_T)_j = \sum_{j \in T} \mathcal{I}_f(v, j \mid S) \geq \sum_{i=1}^{k} \mathcal{I}_f(v, j_i \mid S_i) \qquad (3.2)
$$

$$
= \mathcal{I}_f(v, T \mid S) = \mathcal{I}_f(v, N \mid S). \qquad (3.3)
$$

Above, inequality (3.2) used the 3-increasing property, which implies the submodularity of mutual coverage. Step (3.3) used the Chain Rule. □

As we will soon see, the purpose of the new $(*)$ constraints is to ensure that, just like $f_R(N)$, the expected remaining mutual coverage decreases geometrically with the number of rounds.

3.2 The Improved Analysis.

Our second algorithm for OSubCov resembles the first; it is the analysis that differs. Indeed, at time t, feed the batch of constraints of $(Q^{(t)})$ (defined as (Q) for the function $f^{(t)}$) to [BN09] and update the fractional solution x accordingly. In the rounding phase, perform $k = \ln(t^2 \cdot f_{\max}/f_{\min})$ rounds of randomized rounding using the appropriate probability vector $\Delta \in [0,1]^n$ (recall Δ_j is calculated such that the total probability that j gets added to R over the course of the algorithm is $1-(1-x_j)^k$) and add the new elements to the set R. In a subsequent greedy phase, iteratively add the cheapest element $j \in N$ with non-zero marginal value to set G, until none remain. Return $R \cup G$.

It is clear that the algorithm is correct, so it remains to bound the expected cost. The main idea is to keep track of the amount of coverage that the t^{th} greedy element in the augmentation procedure contributes to the coverage achieved by the s^{th} element of the optimal solution; the (conditional) mutual coverage between the two is the quantity of interest here.

Theorem 12. *There exists a randomized algorithm for the OSubCov problem when $f \in \mathcal{D}_3^+$ which guarantees that for each T, the expected cost of solution S_T is within $O(\ln n \cdot \ln(T \cdot f_{\max}/f_{\min}))$ of the optimal submodular cover solution for $f^{(T)}$.*

Proof. The algorithm always outputs a feasible solution, and it suffices to relate the cost of $R \cup G$ to the cost of the optimal solution. As before, at every time $t \in [T]$, the probability that j is in R is $1-(1-x_j)^k$ and we may view R as the result of k rounds of randomized rounding. Thus the expected cost of R is $\ln(t^2 \cdot f_{\max}/f_{\min}) \cdot c(x)$. In the remainder, we analyze the greedy phase.

At time t, the fractional solution x is feasible to $Q^{(t)}$. If S_{t-1} is the solution the algorithm produced for time $t-1$, define $f := f^{(t)}_{S_{t-1}}$. Then x is also feasible to (Q) for the contracted function f. By Lemma 9 with $k = \ln(t^2 \cdot f_{\max}/f_{\min})$:

$$\mathop{\mathbb{E}}_{R}[\mathcal{I}_f(v; N \mid R)] \le t^{-2}\left(\frac{f_{\min}}{f_{\max}}\right)\mathcal{I}_f(v;N) \tag{3.4}$$

$$= t^{-2}\left(\frac{f_{\min}}{f_{\max}}\right) f(v) \tag{3.5}$$

Let $G_1, \ldots, G_p$ denote the elements added by the greedy phase in increasing order of cost, and let $G_{1:t} = \{G_1, \ldots, G_t\}$ be the prefix of the first t elements. Also order the elements of Opt arbitrarily $O_1, \ldots, O_q$ and similarly let $O_{1:s} = \{O_1, \ldots, O_s\}$. For convenience, let $G_{1:0} = O_{1:0} = \emptyset$.

We say an element G_t *contributes* to O_s if conditioned on having selected $G_{1:t-1}$, and $O_{1:s-1}$, selecting G_t changes the marginal value of O_s. Let $C(t,s)$ be the contribution of G_t to O_s, or the amount by which this marginal value changes. Formally:

$$C(t,s) := \mathcal{I}_f(G_t; O_s \mid G_{1:t-1} \cup O_{1:s-1} \cup R)$$

We will need the three claims we alluded to in the opening of the section to proceed.

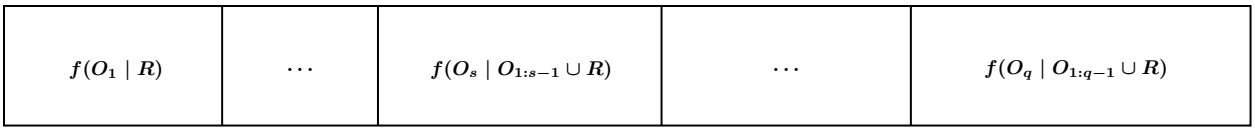
Claim 13. *If G_t contributes to O_s, i.e. $C(t,s) > 0$, then $c(G_t) \le c(O_s)$.*

Claim 14. *For any index $t \in [p]$ we have: $\sum_s C(t,s) \ge f_{\min}$.*

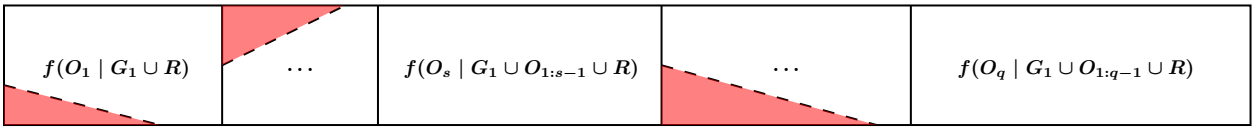
Claim 15. *For any index $s \in [q]$ we have: $\mathbb{E}_R\left[\sum_t C(t,s)\right] \le t^{-2} f_{\min}$.*

Before we prove these claims, let us show how they imply the statement of the theorem.

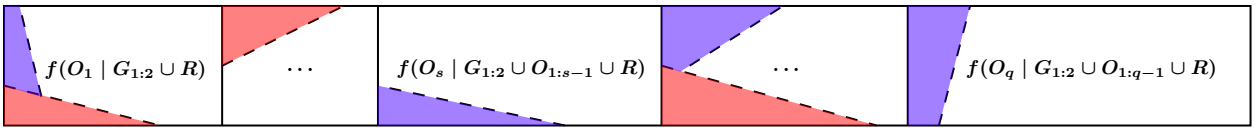
$$\mathbb{E}\left[\sum_t c(G_t)\right] = \mathbb{E}\left[\sum_t \left(\sum_s \frac{C(t,s)}{\sum_{s'} C(t,s')}\right) c(G_t)\right]$$



(a) Before the greedy algorithm



(b) After G_1 is selected



(c) After G_2 is selected

Figure 1: Illustration of the charging scheme. The white areas initially represent the marginal values of each element of Opt conditioned on the elements that come before it in the ordering. The red areas represent the contribution of G_1 to the elements of Opt, or the amount by which the marginal value of each element in Opt changes after conditioning on G_1. Likewise, the purple areas represent the contribution of G_2 to the elements of Opt.

$$\leq \mathbb{E}\left[\sum_s \left(\sum_t \frac{C(t,s)}{\sum_{s'} C(t,s')}\right) c(O_s)\right] \tag{3.6}$$

$$\leq \sum_s \frac{\mathbb{E}\left[\sum_t C(t,s)\right]}{f_{\min}} \cdot c(O_s) \tag{3.7}$$

$$\leq t^{-2} c(\textsc{Opt}) \tag{3.8}$$

Above (3.6) is due to Claim 13, (3.7) due to Claim 14 and (3.8) due to Claim 15. Thus the expected cost of cumulative alterations over all time steps $t \in [T]$ is $O(c(\textsc{Opt}))$. To wrap up, Theorem 2.1 implies that $c(x) = O(\ln n) \cdot c(\textsc{Opt})$, hence:

$$\mathbb{E}\left[c(R \cup G)\right] = \ln\left(T \cdot \frac{f_{\max}}{f_{\min}}\right) \cdot c(x) + O(c(\textsc{Opt})) = O\left(\ln n \ln\left(T \cdot \frac{f_{\max}}{f_{\min}}\right)\right) \cdot c(\textsc{Opt})$$

It remains to prove the three claims.

Proof of Claim 13. The greedy algorithm repeatedly selects the cheapest element that has nonzero marginal value. If for some element O_s it is the case that $c(O_s) < c(G_t)$, then by the time G_t is picked, O_s must already have been selected, which means $C(t,s) = 0$. □

Proof of Claim 14. This is immediate from the definitions:

$$\sum_{s \in [q]} C(t,s) = \sum_{s \in [q]} \mathcal{I}_f(G_t; O_s \mid G_{1:t-1} \cup O_{1:s-1} \cup R) = \mathcal{I}_f(G_t; N \mid G_{1:t-1} \cup R) \tag{3.9}$$

$$= f_{G_{1:t-1} \cup R}(G_t) \geq f_{\min} \tag{3.10}$$

where step (3.9) used the Chain Rule, and the final step (3.10) used the definition of $f_{\min}$. □

Proof of Claim 15. Again by definition:

$$\sum_t C(t,s) = \sum_{t \in [p]} \mathcal{I}_f(G_t; O_s \mid G_{1:t-1} \cup O_{1:s-1} \cup R) = \mathcal{I}_f(N; O_s \mid O_{1:s-1} \cup R) \leq \mathcal{I}_f(N; O_s \mid R)$$

Where the inequality follows from the submodularity of f and $\mathcal{I}_f(N; O_s \mid O_{1:s-1} \cup R) = f(O_s \mid O_{1:s-1} \cup R)$. Taking expectation over R:

$$\mathop{\mathbb{E}}_{R}\left[\sum_t C(t,s)\right] \leq \mathop{\mathbb{E}}_{R}\left[\mathcal{I}_f(O_s; N \mid R)\right] \leq t^{-2}\left(\frac{f_{\min}}{f_{\max}}\right) f(O_s) \leq t^{-2} f_{\min}$$

The second inequality follows from (3.5), and the third from the definition of $f_{\max}$. □

Given the claims, the proof of Theorem 3.1 is complete. □

It remains to show that the algorithm of this section and the previous one can be made efficient; this we do in the following section.

4 Polynomial Runtime

The algorithms in the two previous sections naïvely run in exponential time, and here are some of the barriers to making them efficient. For one, the algorithm of [BN09] runs in time linear in the number of constraints added, and naively both $(P^{(t)})$ and $(Q^{(t)})$ add an exponential number of constraints at every step. The standard method for solving LPs with many constraints is to construct an efficient separation oracle that, given a point as input, either confirms that it is a feasible solution or outputs a violated constraint. Given such an oracle, we could check if our current solution is feasible, if so output the solution, if not feed the violated constraint to [BN09] and repeat.

However, a polynomial-time separation oracle is unlikely to exist since evaluating f^* is APX-hard [Von07, Section 3.7.2]. Moreover, even assuming access to a separation oracle, we would additionally need to argue that we can make do with a polynomial number of calls to the oracle. However, the algorithm of [BN09] makes no a priori guarantees about how many constraints it will need to fix before it produces a feasible solution (in contrast to the ellipsoid algorithm).

Our solution is to avoid solving the fractional LP directly. Since we merely want to find a feasible integer solution with bounded expected cost, we combine rounding and separation. We show that every LP solution either guarantees random rounding will make progress, or violates an efficiently computable constraint by a large margin. Finally, we show that fixing the large-margin violations requires a large change to the LP solution; by a potential function argument this can only be done a polynomial number of times before the solution is necessarily feasible.

We describe this strategy in the context of the simpler LP $P^{(t)}$. Since $Q^{(t)}$ (when f is 3-increasing) is the intersection of several of the simpler LP's constraints, one for each submodular function $\mathcal{I}_f^{(\{v\})}$, this also implies an efficient algorithm for the second setting, because we can search for a large margin violation in *any* of the submodular cover LPs we are intersecting, and round if none is found.

4.1 Finding Violated Constraints.

Lemma 7 implies that for monotone, nonnegative functions g, the multilinear extension G (which is precisely the expected coverage of a set obtained by randomized rounding according to x) cannot be smaller than $(1 - e^{-1})g^*$. This means that if $G(x)$ is much smaller than $g(N)$, then so is $g^*(x)$, and there must exist C such that $g^{\#C}(x) < g(N)$. Call such a set C *violated*, since we will soon use this as a mechanism for finding violated constraints of (P). Lemma 7 does not immediately imply a constructive algorithm for finding a set C witnessing the violation. We show that the following procedure, inspired by the proof of Lemma 7 (Lemma 3.8 of [Von07]), allows us find such a C.

The input is a fractional LP solution x. For every element $i \in N$, sample an exponential random variable h_i with parameter x_i, and order all the elements $i_1, \ldots i_n$ in increasing order of their corresponding expontential h_i. Of all prefixes of this ordering, consider the prefix $Q = i_1, \ldots i_z$ that minimizes $g^{\#Q}(x)$. If $g^{\#Q}(x) < g(N)$, then say that Q is a violated set. Repeat these steps $3/\epsilon \cdot \ln(1/\delta)$ times or until a violated set is found. If none is found, return "no constraint found".

Lemma 16. *There is an algorithm such that given a nonnegative, monotone function $g : 2^N \to \mathbb{R}^+$ with $g(\emptyset) = 0$, and $x \in [0,1]^n$ with the guarantee that $G(x) < (e(1+\epsilon))^{-1} g(N)$, with probability $1-\delta$ the algorithm yields a subset $C \subseteq N$ such that*

$$g^{\#C}(x) < \left(1 - \frac{\epsilon}{2}\right) \cdot g(N).$$

Furthermore, the procedure runs in time poly $\left(\frac{1}{\epsilon} \ln \frac{1}{\delta}\right)$.

Proof. Consider the set $C(t) := \{i \in N \ : \ h_i < t\}$. Since $\Pr[j \in C(1)] = 1 - e^{-x_j} \leq x_j$, we have by monotonicity of g that $G(x) = \mathbb{E}_{S \sim x}[g(S)] \geq \mathbb{E}[g(C(1)]$. Now for any C,

$$\begin{aligned}
&\mathbb{E}\left[g_{C(t)}(C(t+dt)) \mid C(t) = C\right] = \sum_{D \subseteq N} \Pr\left[C(t+dt) = D \mid C(t) = C\right] \cdot g_C(D) \\
&= \sum_{D \subseteq N \setminus C} g_C(D) \cdot \prod_{j \in D} (x_j\, dt) \prod_{j' \notin D} (1 - x_{j'}\, dt) = \sum_{j \in N} g_C(j) x_j dt + O(dt^2) \\
&= (g^{\#C}(x) - g(C))dt + O(dt^2).
\end{aligned}$$

Dividing by dt and taking an expectation over C,

$$\frac{1}{dt}\mathbb{E}\left[g(C(t+dt)) - g(C(t))\right] \geq \mathbb{E}\left[g^{\#C(t)}(x)\right] - \mathbb{E}\left[g(C(t))\right] + O(dt).$$

Now define $\phi(t) = \mathbb{E}[g(C(t))]$. Taking the limit of the last expression as $dt \to 0$ we get that $\frac{d\phi}{dt} = \mathbb{E}\left[g^{\#C(t)}(x)\right] - \phi(t)$. Using e^t as an integrating factor, for any $t \geq 0$

$$e^t \phi(t) = \int_0^t \frac{d(e^z \phi(z))}{dz} dz = \int_0^t e^z\, \mathbb{E}\left[g^{\#C(t)}(x)\right] dz.$$

By the mean value theorem, for some value $t^* \in [0,t]$ it holds that

$$e^t \phi(t) = t e^{t^*}\, \mathbb{E}\left[g^{\#C(t^*)}(x)\right] \geq t\, \mathbb{E}\left[g^{\#C(t^*)}(x)\right].$$

Now setting $t = 1$ and using the fact that $G(x) \geq \phi(1)$, we have $\mathbb{E}\left[g^{\#C(t^*)}(x)\right] \leq e \cdot G(x)$. By construction, $Q = C(t')$ for some $t' = \operatorname{argmin}_t g^{\#C(t)}(x)$. The inequality $g^{\#Q}(x) \leq g^{\#C(t^*)}(x)$ holds regardless of the realizations of $h_1, \ldots, h_n$, hence

$$\mathbb{E}\left[g^{\#Q}(x)\right] \leq \mathbb{E}\left[g^{\#C(t^*)}(x)\right] \leq e \cdot G(x).$$

By a Markov bound, with probability at least $1 - (1+\epsilon/2)^{-1} \geq \epsilon/3$:

$$g^{\#Q}(x) \leq \left(1 + \frac{\epsilon}{2}\right) e \cdot G(x) < \frac{\left(1 + \frac{\epsilon}{2}\right) e}{(1+\epsilon)e} \cdot g(N) \leq \left(1 - \frac{\epsilon}{2}\right) \cdot g(N).$$

Since we repeat this procedure $k = (3/\epsilon) \ln 1/\delta$ times, the probability that after k attempts no violated constraint is found is: $(1 - \epsilon/3)^k \leq \exp(3/\epsilon \cdot \epsilon/3 \ln \delta) = \delta$. □

4.2 Implementing the Algorithm Efficiently.

We modify the proof of Theorem 10 to exploit the approximate separation oracle. This time we do not ever solve (*P*) explicitly. Instead, start with a fractional solution $x = 0$. At time t, set $f := f^{(t)}_{S_{t-1}}$ (where S_{t-1} is the solution produced for the previous time step), and perform the following procedure which proceeds in rounds.

In each round, search for violated constraints using the procedure from the last section, with $g = f$ and $\delta = 6 \cdot c_{\min}/(\pi^2 k t^2 \cdot c(N))$. If one is found, feed it to [BN09], update x accordingly and restart the current

round. If none is found, perform sample every $i \in N$ with probability x_i (i.e. perform one iteration of randomized rounding).

In total, perform $k := (\ln 1/\gamma)^{-1} \ln(t^2 \cdot f(N)/f_{\min})$ rounds of rounding with respect to the appropriate $\Delta \in [0,1]^n$ (where $\gamma := 1 - (e(1+\epsilon))^{-1}$) and add the result to set R. To finish, again greedily add to set G the cheapest element $j \in N$ with non-zero marginal value, until none remain. Return $R \cup G$.

Crucially, we argue that we do not need to fix too many constraints over the course of the algorithm.

Lemma 17. *The algorithm above finds a violated constraint at most $O(n/\epsilon)$ times.*

Proof. Consider a round in which the algorithm finds a violated constraint indexed by the set C. By Lemma 16,

$$f_R(C) + \sum_{j \in N} f_{R \cup C}(j) \cdot x_j < \left(1 - \frac{\epsilon}{2}\right) \cdot f_R(N).$$

In order to fix this constraint in this round, [BN09] must increase x by a vector Δx such that

$$\sum_{j \in N} \frac{f_{R \cup C}(j)}{f_R(N)} \cdot (\Delta x)_j > \frac{\epsilon}{2}.$$

Since $f_{R \cup C}(j)/f_R(N) \leq 1$, it follows that

$$\sum_{j \in N} (\Delta x)_j \geq \sum_{j \in N} \frac{f_{R \cup C}(j)}{f_R(N)} (\Delta x)_j > \frac{\epsilon}{2}.$$

Since $x \leq 1$, the total sum $\sum_j x_j$ is bounded by n, and thus we can update x at most $2n/\epsilon$ times. □

We can finally conclude with our main theorem.

Theorem 1 (Main Theorem)**.** *There exists an efficient randomized algorithm for the* OSubCov *problem which guarantees that for each T, the expected cost of solution S_T is within $O(\ln n \cdot \ln(T \cdot f(N)/f_{\min}))$ of the optimal submodular cover solution for $f^{(T)}$.*

Proof. The algorithm always produces a feasible solution and we need to bound its cost. Since we perform $(\ln 1/\gamma)^{-1} \ln(t^2 \cdot f(N)/f_{\min})$ rounds of rounding, and $(\ln 1/\gamma)^{-1} \leq e(1+\epsilon)$, the expected cost of R is at most $e(1+\epsilon) \cdot \ln(t^2 \cdot f(N)/f_{\min}) \cdot c(x)$. Next we bound the cost of the greedy phase. Let G_t denote the elements added to G in iteration t.

Let $R_{1:\ell}$ denote the state of R at the end of round $\ell \in [k]$ and let R_ℓ denote the set of elements sampled in round ℓ. By the contrapositive of Lemma 16, for every $v \in N$ and every $\ell \in [k]$, with probability $1 - 6 \cdot c_{\min}/(\pi^2 k t^2 \cdot c(N))$ it holds that:

$$\mathop{\mathbb{E}}_{R_\ell \sim x} \left[f_{R_{1:\ell-1}}(R_\ell) \right] \geq \frac{1}{e(1+\epsilon)} f_{R_{1:\ell-1}}(N) \tag{4.1}$$

Let $\mathcal{E}$ be the event that (4.1) holds for every round of rounding over all time steps $t \in [T]$. By a union bound and the fact that $\sum t^{-2} \leq \pi^2/6$, event $\mathcal{E}$ holds with probability at least $1 - c_{\min}/c(N)$. Conditioned on this being the case, the triple x, f and $R_{1:k}$ together satisfies the conditions of the Rounding Lemma with $\gamma = 1 - (e(1-\epsilon))^{-1}$. Hence:

$$\begin{aligned} \mathop{\mathbb{E}}_{R} [f_R(N)] &\leq t^{-2} \left(\frac{f_{\min}}{f(N)} \right) f(N) \\ &= t^{-2} f_{\min} \end{aligned}$$

With this, as in the proof of Theorem 10, we have $\mathbb{E}[c(G_t) \mid \mathcal{E}] \leq t^{-2} c(\textsc{Opt})$, and cumulatively $\mathbb{E}\left[\sum_{t \in [T]} c(G_t) \mid \mathcal{E}\right] \leq c(\textsc{Opt})$. If $\mathcal{E}$ does not hold, we incur a cost of at most $c(N)$. Hence the total expected cost of cumulative alterations is:

$$\mathbb{E}[c(G)] = \mathbb{E}[c(G) \mid \mathcal{E}] \cdot \Pr[\mathcal{E}] + \mathbb{E}\left[c(G) \mid \overline{\mathcal{E}}\right] \cdot \Pr\left[\overline{\mathcal{E}}\right] \leq \frac{c_{\min}}{c(N)} \cdot c(N) + O(c(\textsc{Opt})) = O(c(\textsc{Opt}))$$

Finally, Theorem 2.1 implies that $c(x) = O(\ln n) \cdot c(\textsc{Opt})$. Setting $\epsilon = 1$, we conclude:

$$\mathbb{E}\left[c(R \cup G)\right] = e(1+\epsilon)\ln\left(T^2 \cdot \frac{f(N)}{f_{\min}}\right) c(x) + O(c(\textsc{Opt})) \leq O\left(\ln\left(T \cdot \frac{f(N)}{f_{\min}}\right)\right) c(\textsc{Opt})$$

This completes the proof of correctness. The polynomial runtime follows from Lemma 17. □

5 Note on Changes since Publication

The original version of this paper that appeared at SODA 2020 claimed Theorem 12 without assuming $f \in \mathcal{D}_3^+$. We would like to thank Jonny Gal for pointing out to us an error with Lemma 11, which is remedied by assuming $f \in \mathcal{D}_3^+$. The assumption further simplifies the proofs of Section 3, since the strengthened LP (Q) is the intersection of several LPs (P). The remaining proofs remain essentially unchanged, though we have also simplified the exposition in Section 4.2. Showing Theorem 12 for the class of all nonnegative monotone submodular functions remains an interesting open problem.

References

[AAA+09] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[AAK19] Arpit Agarwal, Sepehr Assadi, and Sanjeev Khanna. Stochastic submodular cover with limited adaptivity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 323–342. SIAM, 2019.

[AGG+09] Konstantin Andreev, Charles Garrod, Daniel Golovin, Bruce Maggs, and Adam Meyerson. Simultaneous source location. *ACM Trans. Algorithms*, 6(1):16:1–16:17, December 2009.

[BCN14] Niv Buchbinder, Shahar Chen, and Joseph Seffi Naor. Competitive algorithms for restricted caching and matroid caching. In *European Symposium on Algorithms*, pages 209–221. Springer, 2014.

[BF17] Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. *Handbook of Approximation Algorithms and Metaheuristics*, 1:753–788, 2017.

[BMKB13] Maximilian Beinhofer, Jörg Müller, Andreas Krause, and Wolfram Burgard. Robust landmark selection for mobile robot navigation. In *IROS*, pages 3637–2643, 2013.

[BN09] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

[CWJ18] Yang Chen, Jie Wu, and Bo Ji. Virtual network function deployment in tree-structured networks. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 132–142. IEEE, 2018.

[DHK16] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation algorithms for stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. *ACM Trans. Algorithms*, 12(3):42:1–42:28, April 2016.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. ACM.

[FH05] Stephan Foldes and Peter L. Hammer. Submodularity, supermodularity, and higher-order monotonicities of pseudo-boolean functions. *Math. Oper. Res.*, 30(2):453–461, 2005.

[FNW78] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. *An analysis of approximations for maximizing submodular set functions—II*, pages 73–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978.

[Fuj00] Toshihiro Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Transactions on Information and Systems*, 83(3):480–487, 2000.

[GB11] Andrew Guillory and Jeff A Bilmes. Online submodular set cover, ranking, and repeated active learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1107–1115. Curran Associates, Inc., 2011.

[GBLV13] Amit Goyal, Francesco Bonchi, Laks V. S. Lakshmanan, and Suresh Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3(2):179–192, Jun 2013.

[GHKL16] Nathaniel Grammel, Lisa Hellerstein, Devorah Kletenik, and Patrick Lin. Scenario submodular cover. In *International Workshop on Approximation and Online Algorithms*, pages 116–128. Springer, 2016.

[GK11] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

[GN14] Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. *Mathematics of Operations Research*, 39(4):998–1011, 2014.

[GTW14] Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing bases: Multistage optimization for matroids and matchings. In *International Colloquium on Automata, Languages, and Programming*, pages 563–575. Springer, 2014.

[HK18] Lisa Hellerstein and Devorah Kletenik. Revisiting the approximation bound for stochastic submodular cover. *Journal of Artificial Intelligence Research*, 63:265–279, 2018.

[Hoc82] D. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[IIOW10] Tomoko Izumi, Taisuke Izumi, Hirotaka Ono, and Koichi Wada. Approximability and inapproximability of the minimum certificate dispersal problem. *Theoretical Computer Science*, 411(31):2773 – 2783, 2010.

[IKBA21] Rishabh K. Iyer, Ninad Khargoankar, Jeff A. Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, volume 132 of *Proceedings of Machine Learning Research*, pages 722–754. PMLR, 2021.

[JCMP17] Stefan Jorgensen, Robert H Chen, Mark B Milam, and Marco Pavone. The risk-sensitive coverage problem: Multi-robot routing under uncertainty with service level and survival constraints. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 925–932. IEEE, 2017.

[KMGG08] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9(Dec):2761–2801, 2008.

[KN15] Guy Kortsarz and Zeev Nutov. Approximating source location and star survivable network problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 203–218. Springer, 2015.

[Kor04] Simon Korman. On the use of randomization in the online set cover problem. *Master's thesis, Weizmann Institute of Science, Rehovot, Israel*, 2004.

[LCL+16] Z. Liu, A. Clark, P. Lee, L. Bushnell, D. Kirschen, and R. Poovendran. Mingen: Minimal generator set selection for small signal stability in power systems: A submodular framework. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4122–4129, Dec 2016.

[LG16] Grigorios Loukides and Robert Gwadera. Limiting the diffusion of information by a selective pagerank-preserving approach. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 90–99. IEEE, 2016.

[LPS13] Seungjoon Lee, Manish Purohit, and Barna Saha. Firewall placement in cloud data centers. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 52:1–52:2, New York, NY, USA, 2013. ACM.

[LRS18] Tamás Lukovszki, Matthias Rost, and Stefan Schmid. Approximate and incremental network function placement. *Journal of Parallel and Distributed Computing*, 120:159 – 169, 2018.

[MW16] A. D. Mafuta and T. Walingo. Spatial relay node placement in wireless sensor networks. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2016.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, Dec 1978.

[RP15] Mohammad Amin Rahimian and Victor M Preciado. Detection and isolation of failures in directed networks of lti systems. *IEEE Trans. Control of Network Systems*, 2(2):183–192, 2015.

[SG09] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1577–1584. Curran Associates, Inc., 2009.

[Svi04] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[TRPJ16] Vasileios Tzoumas, Mohammad Amin Rahimian, George J Pappas, and Ali Jadbabaie. Minimal actuator placement with bounds on control effort. *IEEE Transactions on Control of Network Systems*, 3(1):67–78, 2016.

[TWPD17] Guangmo Tong, Weili Wu, Panos M Pardalos, and Ding-Zhu Du. On positive-influence target-domination. *Optimization Letters*, 11(2):419–427, 2017.

[Von07] Jan Vondrák. Submodularity in combinatorial optimization. 2007.

[WCZW15] Pan Wu, Guihai Chen, Xiaojun Zhu, and Xiaobing Wu. Minimizing receivers under link coverage model for device-free surveillance. *Computer Communications*, 63:53 – 64, 2015.

[Wol82] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, Dec 1982.

[YCDW15] Y. Yang, L. Chen, W. Dong, and W. Wang. Active base station set optimization for minimal energy consumption in green cellular networks. *IEEE Transactions on Vehicular Technology*, 64(11):5340–5349, Nov 2015.

[ZPW+17] Zhenzhe Zheng, Yanqing Peng, Fan Wu, Shaojie Tang, and Guihai Chen. Trading data in the crowd: Profit-driven data acquisition for mobile crowdsensing. *IEEE Journal on Selected Areas in Communications*, 35(2):486–501, 2017.

A Necessity of Time-Monotonicity

We note briefly that if we drop the time-monotonicity condition, then no algorithm can achieve a bounded competitive ratio. Consider the sequence $f^{(0)}, f^{(1)}, f^{(2)}$ such that $f^{(0)} \equiv f^{(2)} \equiv 0$, and $f^{(1)}(S) = \mathbf{1}(j \in S)$ for an abitrary choice of $j \in N$. In this case the optimal solution at time 2 is the empty set, whereas any online algorithm must output a set containing j.

B Mutual Coverage is not Submodular

We give one example which shows that mutual coverage is in general **not** submodular as a function of one of its arguments. Furthermore, the example demonstrates that the condition $F_S(x) \geq \alpha f_S(N)$ for all subsets $S \subseteq N$ does not guarantee that $\mathbb{E}_{R\sim x}\left[\mathcal{I}_f^{(y|S)}(R)\right] \geq \alpha \cdot \mathcal{I}_f^{(y|S)}(N)$. This justifies our use of the stronger LP (Q) instead of (P) in section Section 3.

Consider the following function on $\{a, b, y\}$:

$$\begin{array}{lll} f(\emptyset) = 0 & f(b) = 1 & f(b+y) = 2 \\ f(a+b+y) = 10 & f(y) = 1 & f(a+b) = 10 \\ & f(a) = 9 & f(a+y) = 10 \end{array}$$

The function f is submodular, but $\mathcal{I}_f^{(y)}(a) - \mathcal{I}_f^{(y)}(\emptyset) < \mathcal{I}_f^{(y)}(b+a) - \mathcal{I}_f^{(y)}(b)$, since:

$$\begin{aligned} &\mathcal{I}_f^{(y)}(a) - \mathcal{I}_f^{(y)}(\emptyset) \\ &= f(a) - f(y+a) - f(\emptyset) + f(y) \\ &= 0 \\ &\mathcal{I}_f^{(y)}(b+a) - \mathcal{I}_f^{(y)}(b) \\ &= f(b+a) - f(a+b+y) - f(b) + f(b+y) \\ &= 1 \end{aligned}$$

Now consider the vector $x = [x_a, x_b, x_y] = [3/4, 1/2, 0]$. One can check that $F_S(x) \geq 1/2 \cdot f_S(N)$ for all $S \subseteq N$ but:

$$\begin{aligned} &\mathop{\mathbb{E}}_{R\sim x}\left[\mathcal{I}_f^{(y)}(R)\right] \\ &= \frac{3}{8}\mathcal{I}_f^{(y)}(a+b) + \frac{3}{8}\mathcal{I}_f^{(y)}(a) + \frac{1}{8}\mathcal{I}_f^{(y)}(b) + \frac{3}{8}\mathcal{I}_f^{(y)}(\emptyset) \\ &= \frac{3}{8} < \frac{1}{2}\mathcal{I}_f^{(y)}(N) \end{aligned}$$