Submodular Optimization Under Uncertainty

Roie Levin

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Contents

Chapter 1

Introduction

Consider the following combinatorial optimization problem. We are given a graph with edge capacities representing a local area network, along with a set of client vertices each with some positive flow demand. We wish to purchase the smallest number of servers to place in the network, such that each client can feasibly route its demand to some server, all while respecting the edge capacities of the graph. What is more, this is not a static scenario, instead the clients come and go over time. How should one approach solving such an involved problem?

As a first step, it is helpful to distill the core features of this problem. We have a ground set $\mathcal{N}$ of decisions we can make (here $\mathcal{N}$ is the set of candidate server locations), and a notion of coverage f for every subset of these locations ($f(S)$ is the total client demand that can be feasibly satisfied when S is chosen as the set of server locations). The goal is to pick a set S of minimum cost such that coverage offered by S is best possible (i.e. all demand is satisfied), which is certainly achieved when $S = \mathcal{N}$. Hence our goal is:

$$\boxed{\begin{array}{rl} \min\limits_{S \subseteq \mathcal{N}} & c(S) \\ \text{s.t.} & f(S) = f(\mathcal{N}) \end{array}}$$

This abstract min-cost subset-selection problem is intractable in general. Thankfully, in the case of the server problem, this function f satisfies the additional properties of nonnegativity, monotonicity and *submodularity* (proof due to [AGG⁺09], who refer to this problem as SourceLocation). We will give a formal definition of submodularity in Section 1.1. With these assumptions, the optimization problem is known as *Submodular Cover* (SubmodCover), which itself lies within a larger field as submodular optimization. Conveniently, once we have written our problem in this form, we can use known optimal algorithms for this general version [Wol82]. As it turns out, reducing the server problem to SubmodCover is the right approach, in the sense that the approximation guarantees for submodular cover are best possible in polynomial time, even for this special case [AGG⁺09].

This example highlights the expressive power of submodular functions, and indeed, these are ubiquitous in computer science and beyond. These functions, which are the natural discrete analog of convex/concave functions, strike a sweet spot between structure and generality. They are sufficiently well behaved that there are good algorithms for optimizing these functions in many settings. On the other hand, since many natural quantities have decreasing marginal returns, they still model an immense variety of applications. Beyond the function from the server example, functions commonly modeled as submodular are human utility functions for discrete goods in auctions

[Von08, LLN06, CCPV11], the coverage of a collection of sensors [WCZW15, RP15, ZPW+17, MW16] or network devices [AGG+09, LPS13, KN15, LRS18, CWJ18], the influence of a set of nodes in social networks [GBLV13, LG16, TWPD17, IIOW10], feature selection/document summarization objectives in machine learning [MKSK13, LB11a], diversity maximization criteria [AGHI09, ADF17], and more. An algorithm for submodular optimization is automatically an algorithm for any application that can be framed as submodular optimization; as a result, the field is a plentiful source of high impact algorithmic questions.

Yet this is not the end of our story, since the client set and their demands change with time. One way to model this is as an *online* problem where we only learn the full network traffic patterns gradually. Every time we learn a new client's demand, we must immediately and irrevocably decide which server(s) to open to satisfy the demand, and we want to minimize the total amount spent on servers over time. In the language of SubmodCover, suppose the constraint f is only revealed piecemeal (we detail what we mean by "reveal piecemeal" formally in Chapter 2); the goal is to maintain feasibility at every step while, even with limited information, still making decisions that are robust to the final realization of the problem. A special case of this adversarial online setting is the *stochastic* paradigm, where we assume the problem input is drawn from a distribution, known or unknown.

Another modeling option is to frame this as a *dynamic* problem. Whereas in the online setting we assume we discover an unknown but fixed constraint f over time, in the dynamic version we assume f actually changes over time. In the server example this means clients may not only arrive but also depart. We no longer require that decisions be irrevocable (one can show that no algorithm can be competitive if this were the case); instead our goal is to maintain a feasible and near optimal solution for the current function f, while also minimizing the amount to which our solution changes between steps. Avoiding changing the solution can be a huge boon in practice: in the server problem for example, setting up and tearing down servers is expensive and labor intensive in practice [LRS18].

In the big-data regime where the input to the problem is too large to store in memory all at once, algorithms must deal with another type of uncertainty, namely uncertainty about what is outside of main memory. Here a natural goal is to design *streaming algorithms* that can adequately solve problems with only limited space, after only one sequential pass (or few passes) over the data. Many submodular optimization problems are natural big-data questions, for example those inspired by ranking search results tasks.

In this thesis, we study problems relating to these themes, namely how to design algorithms for submodular optimization in scenarios of incomplete information. The idea of designing algorithms under uncertainty has become an important trend in modern computer science, since for many applications in practice it is unrealistic to expect complete and perfect information about the problem at hand. Prior to our work, no algorithms were known for several important instances of submodular optimization in uncertain environments; we improve known algorithms for others. In doing so, we develop techniques that we hope find uses elsewhere. We also propose avenues for future work, including expanding the scope of these algorithms to new incomplete information settings, as well as applying insights from these submodular optimization algorithms to solve problems that are not on their surface related to submodularity.

1.1 Preliminaries

A set function $f : 2^N \to \mathbb{R}^+$ is *submodular* if it respects the inequality

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) \qquad \text{for any } A \subseteq B \subseteq \mathcal{N},\ x \in \overline{B}$$

It is *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq N$.

In this thesis, 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 for the maximum and minimum marginals of f, which are commonly used parameters quantifying the "smoothness" of the submodular function:

$$f_{\max} := \max\{f(j \mid S) \mid j \in N,\ S \subseteq N\} \tag{1.1.1}$$
$$f_{\min} := \min\{f(j \mid S) \mid j \in N,\ S \subseteq N\} \tag{1.1.2}$$

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:

$$F(x) := \mathop{\mathbb{E}}_{S \sim x}[f(S)] = \sum_{S \subseteq N} \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j) f(S) \tag{1.1.3}$$

where $S \sim x$ means we add each element $j \in N$ independently to S with probability x_j. We can naturally extend definition (1.1.3) to contractions of functions: $F_T(x) := \mathbb{E}_{S \sim x}[f_T(S)]$.

All logarithms in this paper are taken to be base e. In the following definitions, let $x, y \in \mathbb{R}^n_+$ be vectors. The standard dot product between x and y is denoted $\langle x, y\rangle = \sum_{i=1}^n x_i y_i$. We use a weighted generalization of KL divergence. Given a weight function c, define

$$\mathrm{KL}_c(x \,||\, y) := \sum_{i=1}^n c_i \left[x_i \log\left(\frac{x_i}{y_i}\right) - x_i + y_i\right].$$

We introduce the *bag-of-functions* model for several online and dynamic variants of SubmodCover. In this model, we assume that the nonnegative, monotone, submodular function $f^{(t)}$ takes the form

$$f^{(t)}(S) = \sum_{g \in G^{(t)}} g(S)$$

for some set $G^{(t)}$ of nonnegative, monotone, submodular functions. We refer to $G^{(t)}$ as the *active set* functions at time t. For ease of notation we assume that for all $g \in \bigcup_t G^{(t)}$, the parameter $g(\mathcal{N})$ is the same.

Chapter 2

Online Algorithms

We study the general problem of minimum cost subset selection subject to an evolving set of submodular constraints. Recall the SubmodCover problem from the introduction.

$$\boxed{\begin{aligned} \min_{S \subseteq \mathcal{N}} \;& c(S) \\ \text{s.t.} \;& f(S) = f(\mathcal{N}) \end{aligned}}$$

where f is nonnegative, monotone and submodular. The classic offline version of SubmodCover was introduced by Wolsey in [Wol82], where he showed that the greedy algorithm achieves an $\ln(f_{\max}/f_{\min})$ approximate solution (recall the definitions of $f_{\max}$ and $f_{\min}$ from Section 1.1). This is best possible unless $\mathsf{P} = \mathsf{NP}$ [Fei98, DS14a], even for the special case of SetCover.

We study this important special case, SetCover, in further depth. Given a set system $(U, \mathcal{S})$ (where U is a ground set of size n and $\mathcal{S}$ is a collection of subsets with $|\mathcal{S}| = m$), the goal is to select a minimum cost subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that the union of the sets in $\mathcal{S}'$ is U. This is an instance of SubmodCover where $\mathcal{N} = \mathcal{S}$ because the coverage function measuring the size of the union of sets selected is submodular; here $f_{\max} = \max_{S \in \mathcal{S}} |S| \leq n$ and $f_{\min} = 1$, so Wolsey's algorithm recovers the classic $\ln n$ approximation guarantees for SetCover (see e.g. [Chv79, Joh74, Lov75, WS11]).

We define the online variant of SubmodCover as follows. We are given a sequence of nonnegative, monotone, submodular functions $f^{(1)}, f^{(2)}, \cdots$, over a common universe $\mathcal{N}$ 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 $S_{t-1} \subseteq S_t$.

We ask the natural question:

> *Is it possible to maintain a competitive solution to an* OnlineSubmodCover *instance as f evolves over time?*

2.1 Results

2.1.1 Online Submodular Cover

Our main result in [GL20b] that under minimal monotonicity assumptions on the changes in f, the answer is yes.

Theorem 2.1.1. *For a "time-monotone"*[1] *sequence of nonnegative, monotone submodular functions* $f^{(1)}, \ldots, f^{(T)}$, *there exists an efficient randomized algorithm for the* OnlineSubmodCover *problem 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 bound generalizes the known $O(\log m \log n)$ competitive ratio for the special case of OnlineSetCover [AAA+09], which is tight unless $\mathsf{NP} \subseteq \mathsf{BPP}$ [Kor04]. This ports all applications of SubmodCover to the online setting where coverage requirements arrive over time, at the expense of only one additional log factor loss in the competitive ratio.

Along the way to this result, we introduce several potentially useful concepts and techniques. We follow the paradigm introduced in [AAA+09] of (i) maintaining a fractional solution to a continuous (namely linear programming) relaxation of the underlying combinatorial problem even as the constraint set evolves, and then (ii) rounding the fractional solution to true integral one, all while respecting the online "no take-back" rule. The LP we use (introduced by Wolsey [Wol82]) to our knowledge is the only LP relaxation known for submodular cover.

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

Known off-the-shelf methods fail for several reasons stemming from the fact that this LP has an exponential number of constraints:

- **Issue (a):** One cannot use known techniques [BFS19] for maintaining LP solutions online in polynomial time, since their running time scales with number of constraints. Furthermore, even identifying a violated constraint is difficult since the separation problem is itself NP hard.
- **Issue (b):** it is not a priori clear how to analyze the rounding phase, since the best bounds for rounding general covering LPs online (e.g. [GN14]) give bounds of $O(\log m)$, where m is the number of constraints, which would only yield a polynomial bound on the approximation ratio.

We circumvent issue (a) with a 'Round or Cut' procedure that combines rounding and separation; we show how to either identify a constraint violated by a large margin on which to make progress, or complete the rounding successfully (but not necessarily both). We argue that only a small number of large margin violations need to be fixed before the solution is necessarily feasible. Concretely, we show the following which we hope finds other independent applications:

Lemma 2.1.2. *There is an algorithm such that given a nonnegative, monotone function* $f : 2^N \to \mathbb{R}^+$ *with* $f(\emptyset) = 0$, *and* $x \in [0,1]^n$ *with the guarantee that* $F(x) < (e(1+\epsilon))^{-1} f(N)$, *with probability* $1 - \delta$ *the algorithm yields a subset* $C \subseteq N$ *such that:*

$$f(C) + \sum_{j \in N} f_C(j) x_j < \left(1 - \frac{\epsilon}{2}\right) \cdot f(N)$$

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

[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 since dropping it makes the problem intractable. On the other hand, we do not explicitly require that $f^{(t)}(S) \geq f^{(t-1)}(S)$. Note that this is a generalization of the bag-of-functions model (see Section 1.1) where the active set $G^{(t)}$ can only grow monotonically.

We solve issue (b) by relating the constraints of Wolsey's LP to the multilinear extension of submodular functions (see e.g. [Von07]), which exactly measures the coverage of a set obtained by randomized rounding with respect to a vector.

Our algorithm is interesting even when run offline, since it is the first randomized rounding based algorithm for submodular cover. Furthermore, our competitive analysis makes use of a quantity we call Mutual Coverage, that generalizes the classic notion of mutual information from information theory. This quantity was studied independently in [IKBA20] under the name Submodular Mutual Information.

Definition 2.1.3 (Mutual Coverage). *The* mutual coverage *and* conditional mutual coverage *with respect to a set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ are defined as:*

$$\mathcal{I}_f(A; B) := f(A) + f(B) - f(A \cup B), \tag{2.1.1}$$

$$\mathcal{I}_f(A; B \mid C) := f_C(A) + f_C(B) - f_C(A \cup B). \tag{2.1.2}$$

We may think of $\mathcal{I}_f(A; B \mid C)$ intuitively as being the amount of coverage B "takes away" from the coverage of A (or vice-versa, since the definition is symmetric in A and B), given that C was already chosen. This generalizes the notion of *mutual information* from information theory: if $\mathcal{N}$ is a set of random variables, and $S \subseteq \mathcal{N}$, and if $f(S)$ denotes the joint entropy of the random variables in the set S, then $\mathcal{I}$ is the mutual information.

Fact 2.1.4 (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. We demonstrate the use of this definition in the next section when we use it to also give algorithms for SubmodCover in the dynamic setting.

2.1.2 Random Order Set Cover

Can one improve upon the worst case guarantees for OnlineSubmodCover if we assume the order in which we learn the problem is random? In this section, we show that the answer is yes for the special case of OnlineSetCover.

In OnlineSetCover, an adversary reveals the elements of the ground set U one-by-one in an arbitrary order. On the arrival of every element, the algorithm must immediately pick some set $S \in \mathcal{S}$ to cover it, and the goal is to minimize the total cost of the sets chosen. The worst-case guarantee of $O(\log n \log m)$ achieved by the algorithm in Section 2.1.1 is best-possible, and generalizes [AAA+09, BN09]. In [GKL21], we answer the question:

> *What is the best competitive ratio possible for* OnlineSetCover *the adversary is constrained to reveal the elements in uniformly random order (RO)?*

We call this version ROSetCover. Note that the element set U is still adversarially chosen and unknown, and only the arrival order is random.

We show that with only this one additional assumption on the element arrival order, there is an efficient algorithm for ROSetCover (which we name LearnOrCover) with expected competitive ratio matching the best-possible offline approximation guarantee (at least in the regime where $m = \text{poly}(n)$).

Theorem 2.1.5. LearnOrCover *is a polynomial-time randomized algorithm for* ROSetCover *achieving competitive ratio $O(\log(mn))$.*

When run offline, our approach gives a new, asymptotically optimal algorithm for SetCover for $m = \text{poly}(n)$, which may be of independent interest. Indeed, given an estimate for the optimal *value* of the set cover, our algorithm makes a single pass over the elements (considered in random order), updating a fractional solution using a multiplicative-weights framework, and sampling sets as it goes. This simplicity, and the fact that it uses only $\tilde{O}(m)$ bits of memory, may make the algorithm useful for some low-space streaming applications. (Note that previous formulations of StreamingSetCover [SG09, DIMV14, HPIMV16, ER16] only consider cases where *sets* arrive in a stream.)

We show next that a suitable generalization of the same algorithm achieves the same competitive ratio for the *RO Covering Integer Program* problem (ROCIP), in which the rows of a pure covering integer program are revealed in random order, and the goal is to maintain a monotone solution that is feasible for the constraints revealed so far.

Theorem 2.1.6. LearnOrCoverCIP *is a polynomial-time randomized algorithm for* ROCIP *achieving competitive ratio $O(\log(mn))$.*

The algorithm LearnOrCover, true to its name, is based on the insight that one can make progress progress via either learning or covering. Specifically, we maintain a distribution $\{x_S\}_{S\in\mathcal{S}}$ on sets: for each arriving uncovered element, we first sample from this distribution x, then update x via a multiplicative weights rule. If the element remains uncovered, we buy the cheapest set covering it. For the analysis, we introduce the potential function

$$\boxed{\Phi(t) = C_1 \cdot \text{KL}_c\left(x^* \mid\mid x^t\right) + C_2 \cdot \beta \cdot \log\left(\frac{1}{\beta} \sum_{v \text{ uncovered}} \min_{S \ni v} c_S\right)} \tag{2.1.3}$$

where x^* is the (unknown) optimal LP solution, β is an estimate for the cost of x^* (which can be obtained via guess-and-double), and C_1, C_2 are constants. This potential Φ simultaneously measures the convergence of the distribution x to the optimal fractional solution, and the progress towards covering the ground set U. Crucially, this progress is measured in expectation over the random order, thereby circumventing lower bounds for the adversarial-order setting [Kor04].

We complement our main theorem with some lower bounds. For instance, we show that the algorithms of [AAA+09, BN09] have a performance of $\Theta(\log m \log n)$ even in RO, so a new algorithm is indeed needed. Moreover, we observe an $\Omega(\log n)$ lower bound on *fractional* algorithms for ROSetCover. This means we cannot pursue a two-phase strategy of maintaining a good monotone fractional solution and then randomly rounding it (as was done in prior works) without losing $\Omega(\log^2 n)$. Interestingly, our algorithm *does* maintain a (non-monotone) fractional solution while rounding it online, but does so in a way that avoids extra losses.

Finally we show lower bounds for several generalizations. Our starting point is a lower bound for the batched version of the problem where the input is specified by a partition of the ground set U into batches $B_1, B_2, \ldots B_b$. We assume all batches have the same size s. The batches are revealed one-by-one in their entirety to the algorithm, in uniform random order. After the arrival of a batch, the algorithm must select sets to buy to cover all the elements of the batch.

Theorem 2.1.7. *The competitive ratio of any polynomial time randomized algorithm on batched* ROSetCover *with b batches of size s is $\Omega(\log b \log s)$ unless* $\mathsf{NP} \subseteq \mathsf{BPP}$.

We derive as a first corollary a lower bound for the random order version of SubmodCover. We show that removing a log from the bound is not possible in general.

Corollary 2.1.8. *The competitive ratio of any polynomial time randomized algorithm against* ROSubmodCover *is* $\Omega(\log n \cdot \log(f(N)/f_{\min}))$ *unless* $\mathsf{NP} \subseteq \mathsf{BPP}$.

Finally, our second corollary addresses whether it is possible to weaken the RO assumption. One may wonder whether uniformly random order is truly necessary, or whether one can achieve a competitive ratio bound that decays with the entropy of the element arrival distribution. Such a question was recently studied by [ADK21] for the prophet secretary problem. We answer this question negatively by showing that there exist distributions with almost full entropy on which any algorithm is $\Omega(\log^2 n)$ competitive.

Corollary 2.1.9. *For any* $\epsilon \in (0, 1/2)$*, the competitive ratio of any polynomial time randomized algorithm for* ROSetCover *when the input distribution has entropy* $(1-\epsilon)\cdot n \log n - O(n^{1-\epsilon})$ *is* $\Omega(\epsilon \log^2 n)$ *unless* $\mathsf{NP} \subseteq \mathsf{BPP}$.

2.2 Future Work

2.2.1 Random Order Models

A natural research program is to study the large body of follow up work to the original paper on online set cover [AAA+09] and understand to what extent LearnOrCover can be adapted to improve known results when constraints are revealed in random order.

A first immediate question which we leave open in [GKL21] is whether the technique can be extended to handle covering IPs with box constraints.

Question 2.2.1. *Can one achieve a competitive ratio of* $O(\log mn)$ *for* ROCIP with box constraints*?*

The proof fails to generalize for this case for somewhat technical reasons; at a high level, the addition of box constraints can cause the expected progress as measured by the second term of the potential to be too small. Intuitively it seems that we suffer a loss due to the fact that the simplest LP relaxation of CIPs with box constraints has an unbounded integrality gap (see e.g. [CFLP00]). It is known that one can overcome this gap by strengthening the LP with so-called *knapsack cover* (KC) inequality constraints. A promising direction for the question above is to try to incorporate information from these KC inequalities when updating the fractional solution.

Box constraints seem to be a prerequisite for studying further generalizations, but assuming they can be handled, a subsequent project is to understand if the fully general SubmodCover case can be improved in random order.

Question 2.2.2. *Can one achieve a competitive ratio of* $O(\log(T \cdot n)\log(f(N)/f_{\min}))$ *for* OnlineSubmodCover *in the bag-of-functions setting when the functions are revealed in random order?*

Note that $f(N)/f_{\min}$ translates to $a_{\min}^{-1}$ for CIPs (assuming constraints of the CIP have the form $\sum_j a_{ij}x_j \geq 1$, and $a_{\min} = \min_{i,j} a_{ij}$), so this bound translates to $O(\log(mn) \cdot \log(a_{\min}^{-1}))$ for the CIP case. Corollary 2.1.8 implies that the extra $\log(f(N)/f_{\min})$ cannot be removed for this fully general problem.

Finally, we propose extending the LearnOrCover framework to network design questions.

Question 2.2.3. *Can one remove log factors in the competitive ratios for online network design problems in random order settings? For example, can one improve the* $O(\log^3 n \log k)$ *bound due to [AAA+06] for online* GroupSteinerTree*?*

Network design problems pose additional challenges on top of box constraints, since their LP formulations usually involve cut-covering constraints. Hence the arrival of a single connectivity constraint may introduce an exponential size batch of cut-covering constraints. Theorem 2.1.7 implies that in general LearnOrCover loses its advantage if constraints arrive in batches, even if the batches arrive in random order; it is an interesting question whether batches arising from cut constraints are sufficiently structured to allow for an improvement over the adversarial order. An intermediate question is whether one can beat $O(\log m \log n)$ for online-non metric facility location, since this last problem generalizes SetCover but is a special case of GroupSteinerTree.

2.2.2 Other Online Models

Given our success studying online SubmodCover problems, we propose studying other variations on the online arrival model.

There has been a recent surge of interest in algorithms that use machine learning and predictions about the input to improve upon worst case guarantees. [GGL+08] studied the following problem, which we call Stochastic Set Cover (StochSetCover). The input is a collection of sets and a distribution on elements from those sets. The goal is to solve the SetCover problem on the instance given by k independent draws from the element distribution. What is more, the algorithm of [GGL+08] has the special property that it commits a priori to a deterministic mapping from elements to sets that it uses to decide which set to use for each arriving element. Such an algorithm is known as a *universal* algorithm. To our knowledge, there is no work on extending this type of result to the more general submodular cover setting.

Question 2.2.4. *Is there a universal algorithm for* StochSubmodCover *that achieves an* $O(\log(f(N)/f_{\min}))$ *competitive ratio?*

A naive extension of the analysis of [GGL+08] only gives very coarse $O(g(N)/f_{\min} \cdot \log(f(\mathcal{N})/f_{\min}))$ approximation guarantees.

Recently, [KNR20, KNR21] proposed a model in which the algorithm is allowed to first sample a random fraction of the input upfront, before solving the online problem on the remaining fraction of the input; they study the secretary problem and online matching respectively in this setting. A natural question is to wonder whether any non-trivial gains are possible in this model for online set cover.

Question 2.2.5. *Suppose an adversary chooses an* OnlineSetCover *instance, but we are allowed to see a random fraction of the elements (say half) before starting the game. After this period, the remaining elements are revealed in* adversarial *order. Can one improve the competitive ratio of* $\log m \log n$ *in this setting?*

Our prior work suggests several candidate approaches. A first is to pretend the first half was an online sequence revealed in random order, run LearnOrCover, and then to continue running LearnOrCover on the adversarial half as well (ignoring the fact that the order is not random). Improving the analysis boils down to understanding whether the adversary has sufficient leeway in the adversarial half to confuse the algorithm. Another option is to run LearnOrCover on the first random half of the input, and then to use the fractional solution learned as a warm start for the algorithm of [BN09] designed for the worst case. Finally a third approach may be to run a variant the algorithm of [GGL+08], where instead of an explicit prior over instances, we use an empirical estimate of the distribution obtained during the sample phase.

Several papers have studied the following stochastic probing variant of submodular cover [GK11, EKM19, GGN21] which we call SPSubmodCover[2]. Suppose each item of the ground set is in a random state, and the global submodular constraint is determined by the realizations of all the item states. The algorithm may only observe the state of an element by buying it. The challenge is to adaptively decide which elements to buy such as to pay the minimum cost. The competitive ratio is measured with respect to the optimal adaptive policy. [EKM19, GGN21] showed that the *greedy policy* which picks the element maximizing the expected coverage to cost ratio is $O(\log(n \cdot f(\mathcal{N})/f_{\min}))$ competitive. However, this analysis does not quite match the $\log(f_{\max}/f_{\min})$ of Wolsey's greedy algorithm; in fact, this gap can be arbitrarily large since $f_{\max}$ need to not be related to $f(N)$.

Question 2.2.6. *Is there an algorithm for* SPSubmodCover *that achieves an* $O(\log(f_{\max}/f_{\min}))$ *(as opposed to* $O(\log(f(N)/f_{\min}))$*) competitive ratio?*

The study of SPSubmodCover has so far focused on analyzing the greedy policy, presumably because the greedy algorithm was the only known algorithm for SubmodCover prior to our work [GL20b]. We propose studying whether the randomized rounding algorithm we develop can be used to improve the known results.

Finally, another direction of research is to generalize techniques for *deterministic* online set cover to the more general submodular case. In the set cover setting, the ground set of the set system is known in advance, and an adversary reveals a subset of these online. A natural generalization is the bag-of-functions model in which some superset of the final active set is known in advance.

Question 2.2.7. *Is there a* poly log *competitive deterministic algorithm for online* SubmodCover*?*

The proof in [AAA+09] uses a potential function to argue about the competitive ratio. We hope that ideas from our randomized algorithm can be used to inform the right generalization to the submodular case. We note that this goal seems ambitious because algorithms are not even known for simple generalizations of set cover (e.g. for set multi-cover, only bicriteria algorithms are known that cover most of the elements [AAG09]).

[2] [GK11, EKM19, GGN21] use the name StochSubmodCover for this problem, but we use SPSubmodCover in order to distinguish it from the submodular generalization of the problem in [GGL+08].

Chapter 3

Dynamic Algorithms

In some settings, requiring that an algorithm be allowed no "take-backs" may be too stringent. Here we study the question, what if a bounded number of updates to the solution (a.k.a. recourse) is allowed. The natural question is, what does the achievable recourse/approximation tradeoff look like?

We focus our attention on the bag-of-functions setting (see Section 1.1) in which for all time steps t, we assume $G^{(t)} \triangle G^{(t+1)} = 1$. In words, in every round t a single nonnegative, monotone, submodular function g_t is added or removed from an active set $G^{(t)}$. We wish to maintain a competitive solution to SubmodCover for $f^{(t)}$, the sum of all active functions, and we wish to do so with low recourse.

3.1 Results

In [GL20a] we show the following theorem:

Theorem 3.1.1. *There is a deterministic algorithm that maintains an $e^2 \cdot (1 + \log(f_{\max}/f_{\min}))$-competitive solution to* SubmodCover *in the fully-dynamic setting where functions arrive/depart over time. This algorithm has total recourse:*

$$O\bigg(\sum_t \frac{g_t(\mathcal{N})}{f_{\min}} \ln\Big(\frac{c_{\max}}{c_{\min}}\Big)\bigg),$$

where $g_t(\mathcal{N})$ is the value of the function considered at time t, and $c_{\max}, c_{\min}$ are the maximum and minimum element costs.

Qualitatively, this result simultaneously shows that with a small amount of recourse one can extend online submodular cover to the fully-dynamic setting, and furthermore, that this small amount of recourse saves the additional log factor loss that is unavoidable in the strict online setting. We both greatly generalize and simplify prior work on dynamic SetCover [GKKP17], a special case of submodular cover. Once again, this also ports all black box uses of SubmodCover to the dynamic setting.

The main contribution of this work is a general template for converting offline greedy algorithms to online dynamic algorithms. The dynamic algorithm maintains an "execution history" of the decisions made by an auxiliary offline algorithm, and performs local search on this history. To analyze the convergence of this search, we make use of a potential function inspired by Tsallis Entropy [Tsa88], which is a parametrized family of functions that generalize the classical Shannon Entropy[1].

Formally, we define the execution history as a permutation π on the elements of the ground set. The *coverage vector* $\mathfrak{F}_\pi$ is indexed by elements of the universe $\mathcal{N}$, and the value of each coordinate is the marginal coverage (according to permutation π) of the corresponding element.

$$\mathfrak{F}_\pi(\pi_i) := f(\pi_i \mid \pi_{1:i-1})$$

In the unit cost setting, we define the potential to be the (scaled and shifted) Tsallis entropy of this vector (parametrized by $\alpha \in (0,1)$).

$$\Phi_\alpha(f,\pi) := \sum_{i \in N} \left(\mathfrak{F}_\pi(\pi_i)\right)^\alpha .$$

The general cost setting is slightly more involved.

Entropy is often used as a potential function but usually in a qualitatively different way. Many algorithms follow the maximum-entropy principle and seek *high entropy* solutions subject to feasibility constraints; the potential function then tracks the convergence to this goal. On the other hand, in our setting the cost function is the support size of the coverage vector (at least for the unit cost setting), and minimizing this roughly corresponds to *low entropy*. We show entropy decreases sufficiently quickly with every change performed during the local search, and increases by a controlled amount with insertion/deletion of each constraint to the problem, thus proving our recourse bounds. Our proofs make use of the definition of Mutual Coverage from [GL20b] which we hope demonstrates its potential in analyzing other submodular optimization algorithms.

We show how to remove the $\log(c_{\max}/c_{\min})$ dependence and achieve optimal recourse for a structured family of monotone, submodular functions: the class of 3-*increasing (monotone, submodular) set functions* [FH05].

Theorem 3.1.2. *There is a deterministic algorithm that maintains an $O(\log f(N)/f_{\min})$-competitive solution to* SubmodCover *in the fully-dynamic setting where functions arrive/depart over time, and each function is* 3*-increasing in addition to monotone and submodular. Furthermore, this algorithm has total recourse:*

$$O\left(\frac{\sum_t g_t(\mathcal{N})}{f_{\min}}\right).$$

These are set functions, all of whose discrete mixed third derivatives are nonnegative everywhere. Submodular functions in this class include *measure coverage functions*, which generalize set coverage functions, as well as entropies for distributions with positive interaction information (see, e.g., [Jak05] for a discussion). Since this class includes SetCover, this recovers the optimal $O(1)$ recourse bound of [GKKP17]. For this result we use a more interesting generalization of the potential function that reweights the coverage of elements in the permutation non-uniformly as a function of their mutual coverage with other elements of the permutation.

Though the dynamic source location problem does not technically fit the model of Theorem 3.1.1 (since the submodular function in question is not additive over clients), we show that one can adapt the technique to give an algorithm for this problem as well.

Theorem 3.1.3. *There is a deterministic algorithm that maintains an $O\left(\log\left(\sum_t D(c_t)\right)\right)$-competitive solution to dynamic* SourceLocation *in the fully-dynamic setting where clients c_t arrive/depart over time, and client demands $D(c_t)$/edge capacities are integral. Furthermore, this algorithm has total recourse $O\left(\sum_t D(c_t)\right)$.*

[1]The α-Tsallis Entropy of a vector $x \in \mathbb{R}_n^+$ is defined to be $\frac{1}{\alpha-1}(1-\sum_i x_i^\alpha)$, which becomes Shannon Entropy in the limit as $\alpha \to 1$.

3.2 Future Work

3.2.1 Dynamic Submodular Cover

One clear opportunity for future work is to improve our analysis.

Question 3.2.1. *Is there an algorithm for fully-dynamic* SubmodCover *that maintains a competitive solution and uses only $O(g_t(N)/f_{\min})$ recourse time per time step (as opposed to the $O(g_t(N)/f_{\min} \cdot \log(c_{\max}/c_{\min}))$ we show in Theorem 3.1.1)? Is the extra dependence on the cost ratio in our bound just an analysis artifact?*

If indeed the extra $\log(c_{\max}/c_{\min})$ is an analysis artifact, one approach is to use a better potential function. To this end, it is tempting to use potential functions that are amenable to the same proof template. One can write the recourse bound we prove using our proof template as the objective value of a functional optimization problem over the choice of the potential function, which involves inequality constraints on the derivatives. Solving this optimization problem seems challenging.

An interesting orthogonal line of research is the computational, update-time question.

Question 3.2.2. *Is there an algorithm for fully-dynamic* SubmodCover*, in the bag-of-functions model where each function is an r-junta, that maintains a competitive solution and only $\text{poly}(r) \cdot g_t(N)/f_{\min} \cdot \text{poly}\log(f(N)/f_{\min}, c_{\max}/c_{\min})$ computation time per time step (assuming value oracle access to f)?*

A natural approach here is to reuse the local search technique we develop for the recourse bound, and show that there is a data structure that can maintain the permutation π and identify local search moves quickly. The naive implementation of our local search requires polynomial time, since it must greedily search for local moves, as well as preform bubble sort between any two such moves. Since known algorithms for dynamic SetCover with update time bounds ([GKKP17]) explicitly maintain an assignment of elements to sets, it is conceivable that one may need more than black box access to a value oracle. This question is interesting even for special cases of submodular cover such as partial cover, or the SourceLocation problem we discuss in Chapter 1.

A final question asks whether the amortization is truly necessary:

Question 3.2.3. *Can one provide similar recourse guarantees as [GL20a] but in the worst case?*

Here it seems that the potential function approach is fundamentally not sufficient and new ideas are needed.

3.2.2 Dynamic Fractional Covering

Linear programming is one of the central primitives of algorithm design, and in online algorithms in particular. The classic approach to solving combinatorial optimization problems when the constraints are revealed online, is to approximately solve the LP relaxation and then to round the solution online such that the marginal probabilities of elements entering the solution are consistent with the LP solution. For example, this was the first approach found to give a tight competitive ratio for the OnlineSetCover problem in [AAA+09]. However, there is no analog of the online LP solver when constraints may not only arrive but also depart online (a.k.a. the fully-dynamic setting); crucially, the LP solutions used in the online setting are monotone, and in the fully dynamic setting no monotone solution can be competitive in general. Hence LPs play a less central role in the design of combinatorial optimization algorithms in fully-dynamic settings. It is natural to wonder, is there a useful counterpart to online LP solvers in fully-dynamic settings?

A concrete setting to consider is a dynamic version of *fractional* set cover. Constraints arrive and depart from an active set, and we wish to maintain a low recourse competitive fractional solution (here recourse means the ℓ_1 distance between solutions of successive time steps).

Question 3.2.4. *Is there an algorithm that maintains an $O(1)$ competitive algorithm for dynamic fractional* SetCover *as the constraints arrive/depart over time, with at most* $\operatorname{poly}\log(m, n, c_{\max}/c_{\min})$ *recourse measured as ℓ_1 movement?*

Note that if relax the requirement to $O(\log n)$ competitive solutions, one can just use known algorithms for the integral version (e.g. [GKKP17] or our result [GL20a]).

We suggest two approaches. The first is to adapt our framework for converting greedy-like algorithms to dynamic algorithms to the fractional case. One candidate greedy-like algorithm is the multiplicative weights algorithm for pure covering LPs (see e.g. [AHK12]), which can be thought of as follows. Let each element have a weight initialized to 1, and let the weight of a set be the sum of its elements' weights. For a sufficiently large number of iterations, pick the set of largest weight, then halve the weights of all elements covered by that set. Finally, take the unweighted average of the sets chosen this way. The list of sets picked in order by the algorithm seems like a good analog of the execution history in our previous work [GL20a]. Unfortunately, due to the multiplicative decay in element weights over the course of this process, the weights of sets at the tail of this history are exponentially smaller than those at the head. Hence a naive version of our potential function will not decrease sufficiently quickly during local search moves which only affect the tail.

The second candidate approach is to use a projection based method onto the feasible region of the set cover constraints together with the cost constraint $\sum c_S x_S \leq c(\textsc{Opt})$. Here we hope to use insights from our work on ROSetCover [GKL21]. We show that there is a constant $c < 1$ such that so long as an element v is fractionally covered to extent less than c, increasing the sets $S \ni v$ multiplicatively and renormalizing so that the solution costs $c(\textsc{Opt}_{LP})$ decreases the KL divergence to the optimal LP solution by at least constant. Repeating this operation until all elements are covered to extent c and scaling gives an $O(1/c)$ competitive algorithm. Can the KL divergence serve as a potential function, and can this be used to argue that this algorithm has bounded recourse? This strategy seems to require an upper bound on the increase in the KL divergence with the arrival/departure of elements.

A future and more ambitious goal is to understand what class of LPs beyond SetCover (or even what class of more general convex programs) can be solved dynamically with low recourse.

3.2.3 Beyond Submodularity

Can ideas from our work on dynamic algorithms be used for applications beyond submodular optimization? As an example, in our paper [GL20a], we show that one can use the dynamic submodular cover analysis to bound the recourse of an algorithm for the dynamic SteinerTree problem. This network design problem (where the goal is to maintain a tree that connects a changing set of terminal nodes in the graph) is not a case of submodular optimization in any obvious sense. Nevertheless, we show that one can view one particular greedy algorithm for SteinerTree as approximately solving a sequence of minimum cost matroid base problems for a particular sequence of matroids, each of which *is* an instance of submodular cover.

Theorem 3.2.5. *There is a deterministic algorithm for fully-dynamic metric* SteinerTree *that achieves a competitive ratio of $O(1)$ and an amortized recourse bound of $O(\log D)$, where D is the* *ratio of the maximum to minimum distance in the metric.*

Given that we use the Tsallis Entropy potential for the analysis, a tantalizing related problem is to give algorithms for the more general dynamic SteinerForest problem. In this problem, the input is a set of pairs of vertices called terminals, and the algorithm must decide on the minimum cost subgraph that connects each terminal to its mate (this generalizes SteinerTree, in that the solution is allowed to consist of a collection of disconnected trees, i.e. a forest). Famously, [AKR95, GW95] showed a 2-approximation for this problem. However, other than a small body of work [GGK+18, GK15, KLS05], there is a relative dearth of other algorithms for this problem. In particular, there is no known low-recourse dynamic algorithm.

Question 3.2.6. *Is there an* $O(1)$*-competitive algorithm for fully-dynamic* SteinerForest *that uses only* poly log *recourse?*

A natural approach is to use a greedy algorithm [GK15], and combine it with our framework for converting greedy algorithms to low-recourse dynamic algorithms. Alternatively, one can imagine using a different local search algorithm for SteinerForest (e.g. [GGK+18]) and then use our potential function to analyze the convergence. Unforunately, a priori, known algorithm may require a large amount of recourse per local move; this suggests that such costs must either be amortized, or that a new algorithm is needed altogether.

The answer to this question is challenging even known for the very special case of the line metric, where the offline optimal solution is trivial (it is the union of the intervals defined by the terminal pairs). The difficulty arises in situations when a terminal pair arrives and most of the distance between the pair is spanned by the current solution, but to bridge the gap requires purchasing many edges and hence a large amount of recourse. The algorithm must choose: suffer this loss in the recourse, or instead buy a direct edge between the endpoints and suffer loss in the approximation factor? In the first case, the adversary might toggle the request on and off to incur unbounded recourse, in the second case it might request similar copies of the same terminal pair and incur unbounded loss in the approximation factor. In light of this, it may be useful to assume the adversary is oblivious to the random decisions of the algorithm.

Finally, the improved bound we give for 3-increasing functions suggests that better algorithms may be possible for other submodular optimization problems when functions are from this class.

Question 3.2.7. *What natural submodular functions (beyond coverage functions) are* 3*-increasing, and are there submodular optimization settings where this assumption allows for better approximation algorithms than those for general functions? What about higher order monotonicities?*

We note that [IKBA20] gives a few interesting examples of 3-increasing submodular functions beyond coverage functions, however it is far from a complete characterization. This may provide an interesting alternative characterization of the complexity of submodular functions, beyond the commonly used notion of curvature. Another recent complexity measure due to [BS20] is the number of parts in the decomposition of the submodular function as a *partition submodular function.*

Chapter 4

Streaming Algorithms

We turn to the study of constrained submodular maximization in the streaming setting. In particular, we are interested in the Maximum Submodular Matching problem (MSM), where we wish to find a matching in a graph that maximizes a global submodular notion of utility. Beyond theoretical appeal, this problem models important applications in machine translation [LB11b], internet advertising [DSSX19, KMZ18], combinatorial auctions [Von08, LLN06, CCPV11], and diversity maximization [AGHI09, ADF17].

Since many of these uses are perfect for big-data settings, it is natural to study the streaming variant of the problem where the edges of the graph appear in a stream and the algorithm is allowed only $\tilde{O}(n)$ bits of memory. Indeed, there has been a recent flurry of work on related streaming submodular maximization problems [BMKK14, KMZ$^+$19, AEF$^+$20, CGQ15, FKK18, MJK18]. Constant factor approximations for this particular problem were previously known in the streaming setting [CK15, CGQ15].

4.1 Results

We give improved bounds for both the case where the function is monotone, and more generally when it is not assumed to be monotone. We show:

Theorem 4.1.1. *For every $C > 1$ there is a semi-streaming algorithm that for a monotone* MSM *instance outputs a matching M of value*

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geq f(\textsc{Opt}).$$

This is optimized by taking $C = 1 + \frac{1}{\sqrt{2}}$, resulting in an approximation ratio of $3 + 2\sqrt{2} \approx 5.828$.

This improves upon the previous state-of-the-art of 7.75. When the function is not monotone:

Theorem 4.1.2. *For every $C > 1$ there is a semi-streaming that for a non-monotone* MSM *instance outputs a matching M of value*

$$\left(\frac{4C^2-1}{2C-2}\right) \cdot f(M) \geq f(\textsc{Opt}).$$

This is optimized by taking $C = 1 + \frac{\sqrt{3}}{2}$, resulting in an approximation ratio of $4 + 2\sqrt{3} \approx 7.464$. Moreover, the same algorithm is $2C + C/(C-1)$ approximate for monotone MSM.

This improves upon the previous state-of-the-art of 9.90. Note, even our non-monotone algorithm beats the previous state-of-the-art algorithm for the easier, monotone setting.

Our algorithm is a variant of the (randomized) primal-dual technique applied to a continuous-extension based LP relaxation of the integral problem. We show that one can maintain a low-space dual solution in the stream, and that we need only compare each incoming edge against this dual solution (as opposed to all edges seen so far) to construct a good quality solution. As far as we know, this is the first use of the primal dual method in the subfield of streaming submodular maximization. We also show how to reinterpret older results for this problem under the primal-dual lens, and hence we hope that this is a step towards unifying the theory of streaming submodular maximization.

We also improve the lower bound for this problem to 1.91 assuming ETH.

Theorem 4.1.3. *Assuming ETH, there exists a distribution over* MSM *instances such that any deterministic algorithm achieving an* 1.914 *approximation with constant probability must use either* $n^{1+\Omega(1/\log\log n)}$ *space or* $\Omega\left(2^{(\log n)^{10}}\right)$ *time.*

Our lower bound instance makes use of two sources of hardness: computational hardness under ETH ([Fei98, DS14b]) and information-theoretic hardness resulting form the algorithm not knowing the contents or order of the stream in advance ([GKK12]). In particular, our proof embeds a hard SetCover instance in the submodular values of the edges of [GKK12] (which otherwise has only linear edge weights), and hence exploits the submodularity in the MSM objective. Interestingly, our lower bound of 1.914 is higher than any convex combination of the previous hardness results we make use of, both of which imply a lower bound no higher than $e/(e-1)$.

4.2 Future Work

Very recently, [FNFSZ21] improved our lower bound to approximately 2.69 and removed the ETH assumption. This definitively separates the submodular and *weighted* matching problems, since a $2+\epsilon$ approximation is known for the weighted case [PS18]. Furthermore, assuming no algorithm better than $2+\epsilon$ is possible for the weighted case, [FNFSZ21] shows a lower bound of 3 for MSM. A clear future direction is to close the extant gaps between upper and lower bounds which remain frustratingly large.

Question 4.2.1. *Can one improve the approximation ratios of Theorems 4.1.1 and 4.1.2 for streaming* MSM*? Or can one instead improve the lower bound?*

To improve the lower bound, one idea is to reuse our technique of splicing a hard instance for streaming max weight matching with a hard instance of set cover (thereby reintroducing the ETH assumption), but to use the new construction in the new result of [FNFSZ21] instead of the instance in [GKK12] which we use.

Improving the upper bound seems more ambitious. In our paper, we complement our algorithm with an instance on which the analysis is tight, indicating that to make further progress will require new ideas. Very recently [HS21] gave a slight variant of our algorithm and proved the same guarantees as ours; though they show that there are instances on which their algorithm behaves differently, their algorithm behaves identically to ours on the tight instance, and hence their analysis is also tight. This is further evidence that the instance is a significant barrier.

Another important direction is to extend our primal-dual framework to capture more general constraint families studied in the streaming submodular maximization (SubmodMax) literature:

Question 4.2.2. *Can one frame existing algorithms for streaming* SubmodMax *(e.g. under p-matroids, p-matchoids, or p-exchange systems constraints) as primal dual algorithms? Does the primal dual lens yield new algorithms for these settings? Does this give improved bounds?*

Since the publication of our paper [LW20], [GJS21] showed how to extend our results to achieve the same ratios of $3 + 2\sqrt{2}$ and $4 + 2\sqrt{3}$ for monotone and nonmonotone submodular optimization subject to a matroid intersection constraint in the streaming setting. Their result is based on a suitable generalization of the vertex potentials which make up our dual solutions to the matroid intersection case.

The constraints studied by [GJS21] and our paper, matroid intersection and non-bipartite matching respectively, are two incomparable generalizations of bipartite matching constraints. However, they are both special cases of 2-matchoid constraints. A 2-matchoid is defined by a collection of matroids over subsets of the ground set such that no ground set element belongs to more than 2 of these matroids. For example, given a non-bipartite graph, we can represent matching constraints as uniform matroids of rank 1 for every vertex, where the members of each matroid are the adjacent edges of that vertex. A concrete subproblem to attack within the question above is to try to lift our results to general 2-matchoids constraints by combining the generalized vertex potentials of [GJS21] with the other ideas of our paper.

Finally, it is natural to wonder what guarantees are possible when the element stream is observed in random order; indeed, in big data applications one often has the ability to randomize the stream order. [NFTM+18] show that under this assumption, there is a semi-streaming algorithm for SubmodMax algorithm and a (very small) constant $\epsilon > 0$ such that this algorithm achieves a $2 - \epsilon$ approximation in random order. The authors did not seem to have fully optimized the parameter ϵ, which suggests the following question.

Question 4.2.3. *What is the best possible approximation ratio for streaming* SubmodMax *under cardinality constraints, when the ground set is streamed in random order? For* MSM*? Other constraint families?*

We propose thinking about this special case of coverage functions under cardinality constraints first, since the definitive answer (beyond $2 - \epsilon$) to the question above is not known even for this case. The algorithm of [NFTM+18] is analyzed with a fairly involved case analysis; we wonder if there is a natural parameter that interpolates between these cases, and if the analysis can be performed more cleanly with respect to such a parameter. We hope that insights from our work on ROSetCover may prove fruitful here. However, potential-based proofs using KL divergence seem difficult to employ, since these tend to lose logarithmic factors in the approximation ratio.

Bibliography

[AAA$^+$06] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006.

[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.

[AAG09] Noga Alon, Yossi Azar, and Shai Gutner. Admission control to minimize rejections and online set cover with repetitions. *ACM Transactions on Algorithms (TALG)*, 6(1):1–13, 2009.

[ADF17] Faez Ahmed, John P Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. In *26th*, pages 35–41, 2017.

[ADK21] Makis Arsenis, Odysseas Drosis, and Robert Kleinberg. Constrained-order prophet inequalities. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2034–2046. SIAM, 2021.

[AEF$^+$20] Naor Alaluf, Alina Ene, Moran Feldman, Huy L Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[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.

[AGHI09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *2nd*, pages 5–14, 2009.

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[AKR95] Ajit Agrawal, Philip Klein, and Ramamoorthi Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[BFS19] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. *ACM Trans. Algorithms*, 15(3), June 2019.

[BMKK14] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[BN09] Niv Buchbinder and Joseph Naor. *The design of competitive online algorithms via a primal-dual approach*. Now Publishers Inc, 2009.

[BS20] Eric Balkanski and Yaron Singer. A lower bound for parallel submodular minimization. In *Proceedings of the 52nd annual ACM SIGACT symposium on theory of computing*, pages 130–139, 2020.

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

[CFLP00] Robert D Carr, Lisa K Fleischer, Vitus J Leung, and Cynthia A Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 106–115, 2000.

[CGQ15] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pages 318–330. Springer, 2015.

[Chv79] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[CK15] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[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.

[DIMV14] Erik D Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. On streaming and communication complexity of the set cover problem. In *International Symposium on Distributed Computing*, pages 484–498. Springer, 2014.

[DS14a] 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.

[DS14b] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *46th*, pages 624–633, 2014.

[DSSX19] John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Balancing relevance and diversity in online bipartite matching via submodularity. In *53rd*, volume 33, pages 1877–1884, 2019.

[EKM19] Hossein Esfandiari, Amin Karbasi, and Vahab Mirrokni. Adaptivity in adaptive submodularity. *arXiv preprint arXiv:1911.03620*, 2019.

[ER16] Yuval Emek and Adi Rosén. Semi-streaming set cover. *ACM Transactions on Algorithms (TALG)*, 13(1):1–22, 2016.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[FH05] Stephan Foldes and Peter L. Hammer. Submodularity, supermodularity, and higher-order monotonicities of pseudo-Boolean functions. *Mathematics of Operations Research*, 30(2):453–461, 2005.

[FKK18] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. *arXiv preprint arXiv:1802.07098*, 2018.

[FNFSZ21] Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Streaming submodular maximization with matroid and matching constraints. *arXiv preprint arXiv:2107.07183*, 2021.

[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.

[GGK$^+$18] Martin Groß, Anupam Gupta, Amit Kumar, Jannik Matuschke, Daniel R Schmidt, Melanie Schmidt, and José Verschae. A local-search algorithm for steiner forest. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[GGL$^+$08] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 347–356. IEEE, 2008.

[GGN21] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. The power of adaptivity for stochastic submodular cover. In *International Conference on Machine Learning*, pages 3702–3712. PMLR, 2021.

[GJS21] Paritosh Garg, Linus Jordan, and Ola Svensson. Semi-streaming algorithms for submodular matroid intersection. In *Integer Programming and Combinatorial Optimization - 22nd International Conference, IPCO 2021, Proceedings*, Lecture Notes in Computer Science. Springer, 2021.

[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.

[GK15] Anupam Gupta and Amit Kumar. Greedy algorithms for steiner forest. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 871–878, 2015.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 468–485. SIAM, 2012.

[GKKP17] Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and Dynamic Algorithms for Set Cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 537–550, New York, NY, USA, 2017. ACM.

[GKL21] Anupam Gupta, Gregory Kehne, and Roie Levin. Random order online set cover. 2021.

[GL20a] Anupam Gupta and Roie Levin. Fully-dynamic submodular cover with bounded recourse. In *2020 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2020.

[GL20b] Anupam Gupta and Roie Levin. The Online Submodular Cover Problem. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1525–1537. SIAM, 2020.

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

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

[HPIMV16] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards tight bounds for the streaming set cover problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 371–383, 2016.

[HS21] Chien-Chung Huang and FranÃğois Sellier. Semi-streaming algorithms for submodular function maximization under b-matching constraint, 2021.

[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.

[IKBA20] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. *arXiv preprint arXiv:2006.15412*, 2020.

[Jak05] Aleks Jakulin. *Machine learning based on attribute interactions.* PhD thesis, Univerza v Ljubljani, 2005.

[Joh74] David S Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.

[KLS05] Jochen Könemann, Stefano Leonardi, and Guido Schäfer. A group-strategyproof mechanism for steiner forests. In *SODA*, volume 5, pages 612–619, 2005.

[KMZ18] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing*, 47(3):1056–1086, 2018.

[KMZ$^+$19] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *36th*, pages 3311–3320, 2019.

[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.

[KNR20] Haim Kaplan, David Naori, and Danny Raz. Competitive analysis with a sample and the secretary problem. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2082–2095. SIAM, 2020.

[KNR21] Haim Kaplan, David Naori, and Danny Raz. Online weighted bipartite matching with a sample. *CoRR*, abs/2104.05771, 2021.

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

[LB11a] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520, 2011.

[LB11b] Hui Lin and Jeff Bilmes. Word alignment via submodular maximization over matroids. In *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 170–175, 2011.

[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.

[LLN06] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

[Lov75] László Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

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

[LW20] Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. *arXiv preprint arXiv:2008.10062*, 2020.

[MJK18] Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[MKSK13] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[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.

[NFTM$^+$18] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *35th*, pages 3829–3838, 2018.

[PS18] Ami Paz and Gregory Schwartzman. A $(2+\varepsilon)$-approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG)*, 15(2):1–15, 2018.

[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] Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proceedings of the 2009 siam international conference on data mining*, pages 697–708. SIAM, 2009.

[Tsa88] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487, 1988.

[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.

[Von08] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *40th*, pages 67–74, 2008.

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

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

[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.