Submodular Maximization via Taylor Series Approximation

Gözde Özcan[1] Armin Moharrer[1] Stratis Ioannidis[1]

Abstract

We study submodular maximization problems with matroid constraints, in particular, problems where the objective can be expressed via compositions of analytic and multilinear functions. We show that for functions of this form, the so-called *continuous greedy* algorithm [1] attains a ratio arbitrarily close to $(1 - 1/e) \approx 0.63$ using a deterministic estimation via Taylor series approximation. This drastically reduces execution time over prior art that uses sampling.

1 Introduction.

Submodular functions are set functions that exhibit a diminishing returns property. They naturally arise in data summarization [2–4], facility location [5], recommendation systems [6], influence maximization [7], sensor placement [8], dictionary learning [9, 10], and active learning [11]. In these problems, the goal is to maximize a submodular function subject to matroid constraints. This is generally NP-hard, but a celebrated greedy algorithm [12] achieves a $1 - 1/e$ approximation ratio on uniform matroids. For general matroids this approximation ratio drops to $1/2$ [13]. The *continuous greedy* algorithm [1,14] improves this: it maximizes the *multilinear relaxation* of a submodular function in the continuous domain, guaranteeing a $1 - 1/e$ approximation ratio [1]. The fractional solution is then rounded to a feasible integral solution, e.g., via pipage rounding [15] or swap rounding [16]. The multilinear relaxation of a submodular function is its expected value under independent Bernoulli trials; however, computing this expectation is hard in general. The state of the art is to estimate the multilinear relaxation via sampling [1,14]. Nonetheless, the number of samples required in order to achieve the superior $1 - 1/e$ guarantee is quite high; precisely because of this, the resulting running time of continuous greedy is $O(N^8)$ in input size N [1].

Nevertheless, for some submodular functions, the multilinear relaxation can be computed efficiently. One well-known example is the *coverage function*, which we describe in Sec. 4; given subsets of a ground set, the coverage function computes the number of elements covered in the union of these subsets. The multilinear relaxation for coverage can be computed precisely, without sampling, in polynomial time. This is well-known, and has been exploited in several different contexts [15, 17, 18].

We extend the range of problems for which the multilinear relaxation can be computed efficiently. First, we observe that this property naturally extends to *multilinear functions*, a class that includes coverage functions. We then consider a class of submodular objectives that are a summation over non-linear functions of these multilinear functions. Our key observation is that the polynomial expansions of these functions are again multilinear; hence, compositions of multilinear functions with arbitrary *analytic* functions, that can be approximated by a Taylor series, can be computed efficiently. A broad range of problems, e.g., data summarization, influence maximization, facility location, and cache networks (c.f. Sec. 6), can be expressed in this manner and solved efficiently via our approach.

In summary, we make the following contributions:

- We introduce a class of submodular functions that can be expressed as weighted compositions of analytic and multilinear functions.
- We propose a novel polynomial series estimator for approximating the multilinear relaxation of this class of problems.
- We provide strict theoretical guarantees for a variant of the continuous greedy algorithm that uses our estimator. We show that the sub-optimality due to our polynomial expansion is bounded by a quantity that can be made arbitrarily small by increasing the polynomial order.
- We show that multiple applications, e.g., data summarization, influence maximization, facility location, and cache networks can be cast as instances of our framework.
- We conduct numerical experiments for multiple problem instances on both synthetic and real datasets. We observe that our estimator achieves 74% lower error, in 89% less time, in comparison with the sampling estimator.

We review related work and technical background in Sec. 2 and 3, respectively. We introduce multilinear functions in Sec. 4. We present our estimator and main results in Sec. 5, examples of cases that can be instances

Supported by NSF grant CCF-1750539.

[1]{gozcan, amoharrer, ioannidis}@ece.neu.edu, Electrical and Computer Engineering Department, Northeastern University, Boston, MA, USA.

of our problem in Sec. 6, and our numerical evaluation in Sec. 7. We conclude in Sec. 8.

2 Related Work.

We refer the reader to Krause and Golovin [5] for a thorough review of submodularity and its applications.

Accelerating Greedy. The seminal greedy algorithm proposed by Nemhauser et al. [12] provides a $1-1/e$ approximation ratio for submodular maximization problems subject to the uniform matroids. However, for general matroids this ratio deteriorates to $1/2$ [13]. Several works have introduced variants to greedy algorithm to accelerate it [19, 20], particularly for influence maximization [21, 22]. However, these accelerations do not readily apply to the continuous greedy algorithm.

Multilinear Relaxation. The continuous greedy algorithm was proposed by Vondrák [14] and Calinescu et al. [1]. Maximizing the multilinear relaxation of submodular functions improves the $1/2$ approximation ratio of the greedy algorithm [13] to $1-1/e$ [1] over general matroids. Beyond this, the multilinear relaxation has been used to obtain guarantees for non-monotone submodular maximization [23, 24], and pipage rounding [15]. All of these approaches resort to sampling; as we provide general approximation guarantees, our approach can be used to accelerate these algorithms as well.

DR-Submodularity. Submodular functions have also been studied in the continuous domain recently. Continuous functions that exhibit the diminishing returns property are termed *DR-submodular* functions [25–30], and arise in mean field inference [31], budget allocation [32], and non-negative quadratic programming [26]. DR-submodular functions are in general neither convex nor concave; however, gradient-based methods [25–27, 33] provide constant approximation guarantees. The multilinear relaxation is also a DR-submodular function; hence, obtaining fractional solutions to multilinear relaxation maximization problems, without rounding, is of independent interest. Our work can thus be used to accelerate precisely this process.

Stochastic Submodular Maximization. Stochastic submodular maximization, in which the objective is itself random, has attracted great interest recently [17,33–36], both in the discrete and continuous domains. A quintessential example is influence maximization [7], where the total number of influenced nodes is determined by random influence models. In short, when submodular or DR-submodular objectives are expressed as expectations, sampling in gradient-based methods has two sources of randomness (one for sampling the objective, and one for estimating the multilinear relaxation/sampling inputs); continuous greedy still comes with guarantees. Our work is orthogonal, in that it can be used to eliminate the second source of randomness. It can therefore be used in conjunction with stochastic methods whenever our assumptions apply.

Connection to Other Works. Our work is closest to, and inspired by, Mahdian et al. [37] and Karimi et al. [17]. To the best of our knowledge, the only other work that approximates the multilinear relaxation via a power series is [37]. The authors apply this technique to a submodular maximization problem motivated by cache networks. We depart by (a) extending this approach to more general submodular functions, (b) establishing formal assumptions under which this generalization yields approximation guarantees, and (c) improving upon earlier guarantees for cache networks by [37]. In particular, the authors assume that derivatives are bounded; we relax this assumption, that does not hold for any of the problems we study here.

Karimi et al. [17] maximize stochastic *coverage functions* subject to matroid constraints, showing that many different problems can be cast in this setting. Some of the examples we consider (see Sec. 6) consist of compositions of analytic, non-linear functions with coverage functions; hence, our work can be seen as a direct generalization of [17].

3 Technical Preliminaries.

3.1 Submodularity and Matroids. Given a ground set $V = \{1, \dots, N\}$ of N elements, a set function $f : 2^V \to \mathbb{R}_+$ is submodular if and only if $f(B \cup \{e\}) - f(B) \leq f(A \cup \{e\}) - f(A)$, for all $A \subseteq B \subseteq V$ and $e \in V$. Function f is *monotone* if $f(A) \leq f(B)$, for every $A \subseteq B$.

Matroids. Given a ground set V, a matroid is a pair $\mathcal{M} = (V, \mathcal{I})$, where $\mathcal{I} \subseteq 2^V$ is a collection of *independent sets*, for which the following holds:

1. If $B \in \mathcal{I}$ and $A \subset B$, then $A \in \mathcal{I}$.
2. If $A, B \in \mathcal{I}$ and $|A| < |B|$, there exists $x \in B \setminus A$ s.t. $A \cup \{x\} \in \mathcal{I}$.

The *rank* of a matroid $r_{\mathcal{M}}(V)$ is the largest cardinality of its elements, i.e.: $r_{\mathcal{M}}(V) = \max\{|A| : A \in \mathcal{I}\}$. We introduce two examples of matroids:

1. **Uniform Matroids.** The uniform matroid with cardinality k is $\mathcal{I} = \{S \subseteq V, |S| \leq k\}$.
2. **Partition Matroids.** Let $\mathcal{B}_1, \dots, \mathcal{B}_m \subseteq V$ be a partitioning of V, i.e., $\bigcap_{\ell=1}^m \mathcal{B}_\ell = \emptyset$ and $\bigcup_{\ell=1}^m \mathcal{B}_\ell = V$. Let also $k_\ell \in \mathbb{N}, \ell = 1, \dots, m$, be a set of cardinalities. A partition matroid is defined as $\mathcal{I} = \{S \subseteq 2^V \mid |S \cap \mathcal{B}_\ell| \leq k_l, \text{ for all } \ell = 1, \dots, m\}$.

Change of Variables. A set function $f : 2^V \to \mathbb{R}_+$ can be interpreted as $f : \{0,1\}^N \to \mathbb{R}_+$ via $f(\mathbf{x}) \triangleq f(\mathtt{supp}(\mathbf{x}))$ where $\mathtt{supp}(\mathbf{x})$ is the support of $\mathbf{x} \in \{0,1\}^N$. We adopt this convention for the remainder of the

paper. We also treat matroids as subsets of $\{0,1\}^N$, defined consistently with this change of variables via

$$\mathcal{M} = \{\mathbf{x} \in \{0,1\}^N : \text{supp}(\mathbf{x}) \in \mathcal{I}\}. \tag{3.1}$$

For example, a partition matroid is:

$$\mathcal{M} = \left\{\mathbf{x} \in \{0,1\}^N \mid \bigcap_{\ell=1}^{m} \left(\sum_{i \in B_\ell} x_i \leq k_\ell\right)\right\}. \tag{3.2}$$

The *matroid polytope* $P(\mathcal{M}) \subseteq [0,1]^N$ is the convex hull of matroid $\mathcal{M}$, i.e., $P(\mathcal{M}) = \texttt{conv}(\mathcal{M})$.

3.2 Submodular Maximization Subject to Matroid Constraints. We consider the problem of maximizing a submodular function $f : \{0,1\}^N \to \mathbb{R}_+$ subject to matroid constraints $\mathcal{M}$:

$$\max_{\mathbf{x} \in \mathcal{M}} f(\mathbf{x}). \tag{3.3}$$

As mentioned in the introduction, the classic greedy algorithm achieves a $1/2$ approximation ratio over general matroids, while the continuous greedy algorithm [1] achieves a $1 - 1/e$ approximation ratio. We review the continuous greedy algorithm below.

3.3 Continuous Greedy Algorithm. The multilinear relaxation of a submodular function f is the expectation of f, assuming inputs x_i are independent Bernoulli random variables, i.e., $G : [0,1]^N \to \mathbb{R}_+$, and

$$G(\mathbf{y}) = \mathbb{E}_{\mathbf{x} \sim \mathbf{y}}[f(\mathbf{x})] = \sum_{\mathbf{x} \in \{0,1\}^N} f(\mathbf{x}) \prod_{i:x_i=1} y_i \prod_{i:x_i=0} (1 - y_i), \tag{3.4}$$

where $\mathbf{y} = [y_i]_{i=1}^N \in [0,1]^N$ is the vector of probabilities $y_i = \mathbb{P}[x_i = 1]$. The continuous greedy algorithm first maximizes G in the continuous domain, producing an approximate solution to:

$$\max_{\mathbf{y} \in P(\mathcal{M})} G(\mathbf{y}). \tag{3.5}$$

The algorithm initially starts with $\mathbf{y}_0 = \mathbf{0}$. Then, it proceeds in iterations, where in the k-th iteration, it finds a feasible point $\mathbf{m}_k \in P(\mathcal{M})$ which is a solution for the following linear program:

$$\max_{\mathbf{m} \in P(\mathcal{M})} \left\langle \mathbf{m}, \nabla G(\mathbf{y}_k) \right\rangle, \tag{3.6}$$

After finding $\mathbf{m}_k$, the algorithm updates the current solution $\mathbf{y}$ as follows:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \gamma_k \mathbf{m}_k, \tag{3.7}$$

where $\gamma_k \in [0,1]$ is a step size. We summarize the continuous greedy algorithm in Alg. 1.

The output of Alg. 1 is within a $1 - 1/e$ factor from the optimal solution $\mathbf{y}^* \in P(\mathcal{M})$ to (3.5) (see Thm. 3.1 below). This fractional solution can be rounded to produce a solution to (3.3) with the same approximation guarantee using, e.g., either the pipage rounding [15] or the swap rounding [1, 16] methods. Both are reviewed in detail in [38].

Algorithm 1 the Continuous Greedy algorithm

Input: $G : P(\mathcal{M}) \to \mathbb{R}_+$, $0 < \gamma \leq 1$
$\mathbf{y}_0 \leftarrow 0$, $t \leftarrow 0$, $k \leftarrow 0$
while $t < 1$ **do**
  $\mathbf{m}_k \leftarrow \arg\max_{\mathbf{m} \in P(\mathcal{M})} \langle \mathbf{v}, \nabla G(\mathbf{y}_k) \rangle$
  $\gamma_k \leftarrow \min(\gamma, 1 - t)$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k + \gamma_k \mathbf{m}_k$, $t \leftarrow t + \gamma_k$, $k \leftarrow k + 1$
end while
return $\mathbf{y}_k$

Sample Estimator. The gradient ∇G is needed to perform step (3.6); computing it directly via (3.4), involves a summation over 2^N terms. Instead, Calinescu et al. [1] estimate it via sampling. First, observe that function G is affine w.r.t a coordinate y_i. As a result,

$$(\partial G(\mathbf{y})/\partial y_i) = \mathbb{E}_{\mathbf{x} \sim \mathbf{y}}[f([\mathbf{x}]_{+i})] - \mathbb{E}_{\mathbf{x} \sim \mathbf{y}}[f([\mathbf{x}]_{-i})], \tag{3.8}$$

where $[\mathbf{x}]_{+i}$ and $[\mathbf{x}]_{-i}$ are equal to the vector $\mathbf{x}$ with the i-th coordinate set to 1 and 0, respectively. The gradient of G can thus be estimated by (a) producing T random samples $\mathbf{x}^{(l)}$, for $l \in \{1, \ldots, T\}$ of the random vector $\mathbf{x}$, consisting of independent Bernoulli coordinates with $\mathbf{P}(x_i = 1) = y_i$, and (b) computing the empirical mean of the r.h.s. of (3.8), yielding:

$$\widehat{\partial G(\mathbf{y})}/\partial y_i = \sum_{l=1}^{T} (f([\mathbf{x}^{(l)}]_{+i}) - f([\mathbf{x}^{(l)}]_{-i}))/T. \tag{3.9}$$

This estimator yields the following guarantee:

THEOREM 3.1. *[Calinescu et al. [1]] Consider Algorithm 1, with $\nabla G(\mathbf{y}_k)$ replaced by $\widehat{\nabla G}(\mathbf{y}_k)$ given by (3.9). Set $T = \frac{10}{\delta^2}(1 + \ln |V|)$, where $\delta = \frac{1}{40 d^2 |V|}$ and $d = r_{\mathcal{M}}(V)$ is the rank of the matroid. The algorithm terminates after $K = \frac{1}{\delta}$ steps and, w.h.p.,*

$$G(\mathbf{y}_K) \geq (1 - (1 - \delta)^{\frac{1}{\delta}}) G(\mathbf{y}^*) \geq (1 - \frac{1}{e}) G(\mathbf{y}^*) \tag{3.10}$$

where $\mathbf{y}^*$ is an optimal solution to (3.5).

4 Multilinear Functions.

In practice, estimating G (and, through (3.8), its gradient) via sampling poses a considerable computational burden. Attaining the guarantees of Thm. 3.1 requires the number of samples per estimate to grow as $N^2 d^4$, that can quickly become prohibitive.

In some cases, however, the multilinear relaxation $G(\mathbf{y})$ has a polynomially-computable closed form. A prominent example is the coverage function, that arises in several different contexts [15, 17]. Let $U = \{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ be a collection of subsets of some ground set $V = \{1, \ldots, N\}$. The coverage $f : \{0,1\}^N \to \mathbb{R}_+$ is:

$$f(\mathbf{x}) = \sum_{\ell=1}^{n} \left(1 - \prod_{i \in \mathcal{J}_\ell} (1 - x_i)\right). \tag{4.11}$$

It is easy to confirm that:

$$\begin{aligned} G(\mathbf{y}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{y}}[f(\mathbf{x})] &= \mathbb{E}_{\mathbf{x}\sim\mathbf{y}}\left[\sum_{\ell=1}^{n} \left(1 - \prod_{i \in \mathcal{J}_\ell} (1 - x_i)\right)\right] \\ &= \sum_{\ell=1}^{n} \left(1 - \prod_{i \in \mathcal{J}_\ell} (1 - \mathbb{E}_{\mathbf{x}\sim\mathbf{y}}[x_i])\right) = f(\mathbf{y}). \end{aligned} \tag{4.12}$$

In other words, the multilinear relaxation evaluated over $\mathbf{y} \in [0,1]^N$ is actually equal to $f(\mathbf{y})$, when the latter has form (4.11). Therefore, computing it does not require sampling; crucially, (4.11) is $O(nN)$, i.e., polynomial in the input size.

This clearly has a computational advantage when executing the continuous greedy algorithm. In fact, (4.12) generalizes to a broader class of functions: it holds as long as the objective f is, itself, multilinear. Formally, a function, $f : \mathbb{R}^N \to \mathbb{R}$ is multilinear if it is affine w.r.t. each of its coordinates. Multilinear functions can be written as:

$$g(\mathbf{x}) = \sum_{\ell \in \mathcal{I}} c_\ell \prod_{i \in \mathcal{J}_\ell} x_i, \tag{4.13}$$

where $c_\ell \in \mathbb{R}$ for ℓ in some index set $\mathcal{I}$, and subsets $\mathcal{J}_\ell \subseteq V$.[1] Clearly, both the coverage function (4.11) and the multilinear relaxation (3.4) are multilinear in their respective arguments.

Eq. (4.12) generalizes to *any multilinear function*:

LEMMA 4.1. *Let $f : \mathbb{R}^N \to \mathbb{R}_+$ be a multilinear function and let $\mathbf{x} \in \{0,1\}^N$ be a random vector of independent Bernoulli coordinates parameterized by $\mathbf{y} \in [0,1]^N$. Then, $G(\mathbf{y}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{y}}[f(\mathbf{x})] = f(\mathbf{y})$.*

The proof can be found in [38]. Lem. 4.1 immediately implies that all polytime-computable, submodular multilinear functions behave like the coverage function: computing their multilinear relaxation *does not require sampling*. Hence, continuous greedy admits highly efficient implementations in this setting. Our main contribution is to extend this to a broader class of functions, by leveraging Taylor series approximations. We discuss this in detail in the next section.

5 Main Results.

In this section, we show that Eq. (4.12) can be extended to submodular objectives that can be expressed via compositions of analytic functions and multilinear functions.

[1] By convention, if $\mathcal{J}_\ell = \emptyset$, we set $\prod_{i \in \mathcal{J}_\ell} x_i = 1$.

Table 1: Notation Summary

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of non-negative real numbers
$G(V, E)$	Graph G with nodes V and edges E
V	Ground set of N elements
f	A monotone, submodular set function
$\mathcal{I}$	Collection of independent sets in 2^V
$\mathcal{M}$	Matroid denoting the $(V, \mathcal{I})$ pair
$\texttt{conv}(\cdot)$	Convex hull of a set
k	Cardinality constraint of a uniform matroid
$\mathbf{x}$	Global item placement vector of x_i's in $\{0,1\}^N$
$[\mathbf{x}]_{+i}$	Vector $\mathbf{x}$ with the ith coordinate set to 1
$[\mathbf{x}]_{-i}$	Vector $\mathbf{x}$ with the ith coordinate set to 0
y_i	Probability of $i \in S$
$\mathbf{y}$	Vector of marginal probabilities y_i's in $[0,1]^N$
$G(\mathbf{y})$	Multilinear extension with marginals $\mathbf{y}$
h_i	An analytic function
g_i	A multilinear function
w_i	Weights in $\mathbb{R}$
$\hat{h}_L$	Polynomial estimator of h_i of degree L
$R_{i,L}$	Residual error of the estimator $\hat{h}_L$
$\hat{f}_L(\mathbf{x})$	Polynomial estimator of $f(\mathbf{x})$ of degree L
$R_L(\mathbf{x})$	Residual error vector of the polynomial estimator $\hat{f}_L(\mathbf{x})$
$\epsilon_{i,L}(\mathbf{y})$	Residual error of the estimator $\widehat{\partial G(\mathbf{y})/\partial y_i}$
$\varepsilon(L)$	Bias of the estimator $\widehat{\nabla G(\mathbf{y})}$
	Influence Maximization
M	Number of cascades
	Facility Location
V	Number of facilities
M	Number of customers
	Summarization
M	Number of partitions

In a nutshell, our approach is based on two observations: (a) when restricted to binary values, polynomials of multilinear functions are themselves multilinear functions, and (b) analytic functions are approximated at arbitrary accuracy via polynomials. Exploiting these two facts, we approximate the multilinear relaxation of an arbitrary analytic function via an appropriate Taylor series; the resulting approximation is multilinear and, hence, directly computable without sampling.

5.1 Motivation and Intuition.

We first establish that polynomials of multilinear functions are themselves multilinear, when restricted to binary values:

LEMMA 5.1. *The set of multilinear functions restricted over the domain $\{0,1\}^N$ is closed under addition, multiplication, and multiplication with a scalar.*

The proof can be found in [38]. It is important to note that multilinear functions are closed under multiplication only when restricted to domain $\{0,1\}^N$. The general set of multilinear functions $f : [0,1]^N \to \mathbb{R}_+$ is *not* closed under multiplication.

Lem. 5.1 has the following implication. Consider a submodular function $f : \{0,1\} \to \mathbb{R}_+$ of the form $f(\mathbf{x}) = h(g(\mathbf{x}))$ where $g : \mathbb{R}^N \to \mathbb{R}$ is a multilinear func-

tion, and $h : \mathbb{R} \to \mathbb{R}_+$ is an analytic function (e.g., log, exp, sin, etc.). As h is analytic, it can be approximated by a polynomial $\hat{h}$ around a certain value in its domain. This gives us a way to estimate the multilinear relaxation of f without sampling. First, we approximate f by replacing h with $\hat{h}$, getting $\hat{f} = \hat{h}(g)$. As $\hat{f}$ is the polynomial of a multilinear function restricted to $\{0,1\}^N$, by Lem. 5.1, *$\hat{f}$ can also be expressed as a multilinear function.* Thus, G can be estimated *without sampling* via the estimator $\hat{G}(\mathbf{y}) \triangleq \hat{f}(\mathbf{y})$.

In the remainder of this section, we elaborate further on construction, slightly generalizing the setup, and providing formal approximation guarantees.

5.2 Assumptions. Formally, we consider set functions $f : \{0,1\}^N \to \mathbb{R}_+$ that satisfy two assumptions:

ASSUMPTION 1. *Function $f : \{0,1\}^N \to \mathbb{R}_+$ is monotone and submodular.*

ASSUMPTION 2. *Function $f : \{0,1\}^N \to \mathbb{R}_+$ has form*

$$f(\mathbf{x}) = \textstyle\sum_{j=1}^{M} w_j h_j(g_j(\mathbf{x})), \tag{5.14}$$

for some $M \in \mathbb{N}$, and $w_j \in \mathbb{R}$, $h_j : [0,1] \to \mathbb{R}_+$, and $g_j : [0,1]^N \to [0,1]$, for $j \in \{1,\dots,M\}$. Moreover, for every $j \in \{1,\dots,M\}$, the following holds:

1. *Function $g_j : [0,1]^N \to [0,1]$ is multilinear.*
2. *There exists a polynomial $\hat{h}_L : [0,1] \to \mathbb{R}$ of degree L for $L \in \mathbb{N}$, such that $|h_j(s) - \hat{h}_L(s)| \leq R_{j,L}(s)$, where $\lim_{L\to\infty} R_{j,L}(s) = 0$, for all $s \in [0,1]$.*

Asm. 2 implies that f can be written as a linear combination of compositions of analytic functions h_j with multilinear functions g_j. The former can be arbitrarily well approximated by polynomials of degree L; any residual error from this approximation converges to zero as the degree of the polynomial increases.

Tab. 2 summarizes several problems that satisfy Asm. 1 and 2. We review each of these problems in more detail in Sec. 6; in the remainder of this section, we provide approximation guarantees for objectives that satisfy these two assumptions.

5.3 A Polynomial Estimator. Given a function f that satisfies Asm. 2, we construct the *polynomial estimator of $f(\mathbf{x})$ of degree L* via

$$\hat{f}_L(\mathbf{x}) \triangleq \textstyle\sum_{j=1}^{M} w_j \hat{h}_L(g_j(\mathbf{x})). \tag{5.15}$$

By Lem. 5.1, function $\hat{f}_L : \{0,1\}^N \to \mathbb{R}$ can be expressed as a multilinear function. We define an estimator $\widehat{\nabla G_L}$ of the gradient of the multilinear relaxation G as follows: for all $i \in V$,

$$\begin{aligned}(\widehat{\partial G_L}/\partial y_i)\big|_{\mathbf{y}} &= \mathbb{E}_{\mathbf{y}}[\hat{f}_L([\mathbf{x}]_{+i})] - \mathbb{E}_{\mathbf{y}}[\hat{f}_L([\mathbf{x}]_{-i})] \\ &\overset{\text{Lem. 4.1}}{=} \hat{f}_L([\mathbf{y}]_{+i}) - \hat{f}_L([\mathbf{y}]_{-i}).\end{aligned} \tag{5.16}$$

We characterize the quality of this estimator via the following theorem, whose proof is in [38]:

THEOREM 5.1. *Assume that function f satisfies Asm. 2. Let $\widehat{\nabla G_L}$ be the estimator of the multilinear relaxation given by (5.16), and define $R_L(\mathbf{x}) \triangleq \sum_j |w_j||R_{j,L}(g_j(\mathbf{x}))|$ for $\mathbf{x} \in \{0,1\}^N$. Then,*

$$\left\|\nabla G(\mathbf{y}) - \widehat{\nabla G_L}(\mathbf{y})\right\|_2 \leq \|\epsilon_L(\mathbf{y})\|_2 \tag{5.17}$$

where $\epsilon_L(\mathbf{y}) = [\epsilon_{i,L}(\mathbf{y})]_{i=1}^N \in \mathbb{R}^N$ and $\epsilon_{i,L}(\mathbf{y}) \triangleq \mathbb{E}_{\mathbf{y}}[R_L([\mathbf{x}]_{+i})] + \mathbb{E}_{\mathbf{y}}[R_L([\mathbf{x}]_{-i})]$. Moreover, $\lim_{L\to\infty} \|\epsilon_L(\mathbf{y})\|_2 = 0$, uniformly on $[0,1]^N$.

The theorem implies that, under Asm. 2, we can approximate ∇G arbitrarily well, uniformly over all $\mathbf{y} \in [0,1]^N$. This approximation can be used in continuous greedy, achieving the following guarantee:

THEOREM 5.2. *Assume a function $f : \{0,1\}^N \to \mathbb{R}_+$ satisfies Assumptions 1 and 2. Then, consider Alg. 1, in which $\nabla G(\mathbf{y}_K)$ is estimated via the polynomial estimator given in (5.16). Then,*

$$G(\mathbf{y}_K) \geq (1 - 1/e)\, G(\mathbf{y}^*) - D\, \varepsilon(L) - P/2K,$$

where $K = (1/\gamma)$ is the number of iterations, $\mathbf{y}^*$ is an optimal solution to (3.5), $D = \max_{\mathbf{y}\in P(\mathcal{M})} \|\mathbf{y}\|_2$ is the diameter of the polymatroid, $\varepsilon(L) = \max_k \|\epsilon_L(\mathbf{y}_k)\|_2$ is the bias of the estimator, and $P = 2\max_{\mathbf{x}\in\mathcal{M}} f(\mathbf{x})$.

The proof can be found in [38]. Uniform convergence in Thm. 5.1 implies that the estimator bias $\varepsilon(L)$ converges to zero. Hence, Thm. 5.2 implies that we can obtain an approximation arbitrarily close to $1 - 1/e$, by setting L and K appropriately.

We note that Thm. 5.2 provides a tighter guarantee than the one achieved by Mahdian et al. [37] (see [38] for a detailed comparison).

5.4 Time Complexity. For all examples in Tab. 2, the error $\varepsilon(L)$ decays exponentially with L. Hence, to achieve an approximation $1 - 1/e + \varepsilon$, we must have $L = \Theta\left(\log\left(\frac{1}{\varepsilon}\right)\right)$. Hence, if multilinear functions g_j, $j \in \{1,\dots,M\}$ are polynomially computable w.r.t N (as is the case for our examples), the total number of terms in $\widehat{f}_L$ will be polynomial in both N and $\frac{1}{\varepsilon}$. We further elaborate on complexity issues in [38].

Table 2: Summary of problems satisfying Assumptions 1 & 2.

	Input	$g_j : \{0,1\}^{\lvert V\rvert} \to [0,1]$ $\mathbf{x} \to g_j(\mathbf{x})$	$h_j : [0,1] \to \mathbb{R}_+$ $s \to h_j(s)$	$f : \{0,1\}^{\lvert V\rvert} \to \mathbb{R}_+$ $\mathbf{x} \to f(\mathbf{x})$	Bias $\varepsilon(L)$
SM	Partitions $\bigcup_{j=1}^{M}\{P_j\} = V$ weights $\mathbf{r} \in \mathbb{R}_+^N$, and $\sum_{i=1}^{N} r_i = 1$	$\sum_{i \in P_j} r_i x_i$	$\log(1+s)$	$\sum_{j=1}^{M} h(s_j)$	$\frac{M\sqrt{N}}{(L+1)2^L}$
IM	Instances $G = (V, E)$ of a directed graph, partitions $\{P_v^j\}_{j=1}^{N} \subset V$	$\sum_{i \in V} \frac{1}{N}\Big(1 - \prod_{u \in P_i^j}(1 - x_u)\Big)$	$\log(1+s)$	$\frac{1}{M}\sum_{j=1}^{M} h(s_j)$	$\frac{\sqrt{N}}{(L+1)2^L}$
FL	Complete weighted bipartite graph $G = (V \cup V')$ weights $w_{i_\ell, j} \in [0,1]^{N \times M}$	$\sum_{\ell=1}^{N}(w_{i_\ell,j} - w_{i_{\ell+1},j})\left(1 - \prod_{k=1}^{\ell}(1 - x_{i_k})\right)$	$\log(1+s)$	$\frac{1}{M}\sum_{j=1}^{M} h(s_j)$	$\frac{\sqrt{N}}{(L+1)2^L}$
CN	Graph $G = (V, E)$, service rates $\mu \in \mathbb{R}_+^M$, requests $r \in \mathcal{R}$, P_j path of r, arrival rates $\lambda \in \mathbb{R}_+^{\lvert\mathcal{R}\rvert}$	$\frac{1}{\mu_j}\sum_{r \in \mathcal{R}: j \in p^r} \lambda^r \prod_{k'=1}^{k_{p^r}(v)}(1 - x_{p^r_{k'}, i^r})$	$\frac{s}{1-s}$	$\sum_{j=1}^{M} h(s_0) - \sum_{j=1}^{M} h(s_j)$	$2M\sqrt{\lvert V\rvert\lvert\mathcal{C}\rvert}\frac{\bar{s}^{L+1}}{1-\bar{s}}$

6 Examples.

In this section, we list three problems that can be tackled through our approach, also summarized in Tab. 2; we also review cache networks (CN) in [38].

6.1 Data Summarization (SM) [2, 6]. In data summarization, ground set V is a set of tokens, representing, e.g., sentences in a document or documents in a corpus. The goal is to select a "summary" $S \subseteq V$ that is representative of V. We present here the diversity reward function proposed by Lin and Bilmes [2]. Assume that each token i has a value $r_i \in [0,1]$, where $\sum_i r_i = 1$. The summary S should contain tokens of high value, but should simultaneously be diverse. The authors achieve this by partitioning V to sets $\{P_j\}_{j=1}^{M}$, where each set $P_j \subset V$ contains tokens that are similar. They then seek a summary that maximizes

$$f(S) = \sum_{j=1}^{M} h\left(\sum_{i \in P_j \cap S} r_i\right), \tag{6.18}$$

where $h : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing concave function (e.g., $h(s) = \log(1+s)$, $h(s) = s^\alpha$, where $\alpha < 1$, etc.). Intuitively, the use of h suppresses the selection of similar items (in the same P_j), even if they have high values, thereby promoting diversity.

Objective (6.18) is clearly of form (5.14). For example, for $h = \log(1+s)$, f is monotone and submodular [2], and is the sum of compositions of h with multilinear functions $g_j(\mathbf{x}) = \sum_{i \in P_j} r_i x_i$, as illustrated in Tab. 2. Moreover, h is analytic and can be approximated within arbitrary accuracy by its L^{th}-order Taylor approximation around 1/2, given by:

$$\hat{h}_L(s) = \sum_{\ell=0}^{L} \frac{h^{(\ell)}(1/2)}{\ell!}(s - 1/2)^\ell. \tag{6.19}$$

We show in [38] that this estimator ensures that f indeed satisfies Asm. 2. Moreover, the estimator bias appearing in Thm. 5.2 is also bounded:

THEOREM 6.1. *Assume a diversity reward function $f : \{0,1\}^N \to \mathbb{R}_+$ that is given by (6.18), with $h(s) = \log(1+s)$. Then, consider the estimator $\widehat{\nabla G}(\mathbf{y}_K)$ given in (5.16) using $\hat{h}_L(\mathbf{x})$, the L^{th} Taylor polynomial of $f(\mathbf{x})$ around 1/2, given by (6.19). Then, the bias of the estimator satisfies $\varepsilon(L) \leq \frac{M\sqrt{N}}{(L+1)2^L}$.*

The proof can be found in [38]. Our work directly allows for the optimization of such objectives over matroid constraints. For example, a partition matroid (distinct from $\{P_j\}_{j=1}^{M}$) could be used to enforce that no more than k_ℓ sentences come from ℓ-th paragraph, etc.

6.2 Influence Maximization (IM) [7, 39]. Influence maximization problems can be expressed as weighted coverage functions (see, e.g., [17]). In short, given a directed graph $G = (V, E)$, we wish to maximize the expected fraction of nodes reached if we infect a set of nodes $S \subseteq V$ and the infection spreads via the Independent Cascade (IC) model [7]. In our notation this objective can be written as

$$f(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N}\sum_{v \in V}\left(1 - \prod_{i \in P_v^j}(1 - x_i)\right), \tag{6.20}$$

where $P_v^j \subseteq V$ is the set of nodes reachable from v in a random simulation of the IC model. This is a multilinear function. Our approach allows us to extend this to maximizing the expectation of *analytic functions* h of the fraction of infected nodes. For example, for $h(s) = \log(1+s)$, we get:

$$g_j(\mathbf{x}) = \sum_{v \in V}\frac{1}{N}\big(1 - \prod_{i \in P_v^j}(1 - x_i)\big), \tag{6.21}$$

for $j = 1, \ldots, M$, and

$$f(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} h\left(g_j(\mathbf{x})\right). \tag{6.22}$$

Functions $g_j : [0,1]^N \to [0,1]$ are multilinear, monotone submodular, and $O(N^2)$ computable, while $h : [0,1] \to$

$\mathbb{R}$ is non-decreasing and concave. As a result, (6.22) satisfies Asm. 1. Again, h can be approximated within arbitrary accuracy by its L^{th}-order Taylor approximation around 1/2, given by (6.19). This again ensures that f indeed satisfies Asm. 2. Moreover, we bound the estimator bias appearing in Thm. 5.2 as follows:

THEOREM 6.2. *For function $f : \{0,1\}^N \to \mathbb{R}_+$ that given by (6.22), consider the estimator $\widehat{\nabla G}$ given in (5.16) using $\hat{h}_L$, the L^{th}-order Taylor approximation of h around 1/2, given by* (6.19). *Then, the bias of estimator $\widehat{\nabla G}$ satisfies $\varepsilon(L) \leq \frac{\sqrt{N}}{(L+1)2^L}$.*

The proof can be found in [38]. Partition matroid constraints could be used in this setting to bound the number of seeds from some group (e.g., males/females, people in a zip code, etc.).

6.3 Facility Location (FL) [5, 34, 40]. Given a weighted bipartite graph $G = (V \cup V')$ and weights $w_{v,v'} \in [0,1]$, $v \in V$, $v' \in V'$, we wish to maximize:

$$f(S) = \frac{1}{M} \sum_{j=1}^{M} \max_{i \in S} w_{i,j} \,. \tag{6.23}$$

Intuitively, V and V' represent facilities and customers respectively and $w_{v,v'}$ is the utility of facility v for customer v'. The goal is to select a subset of facility locations $S \subset V$ to maximize the total utility, assuming every customer chooses the facility with the highest utility in the selection S. This too becomes a coverage problem by observing that $\max_{i \in S} w_{i,j}$ equals [17]:

$$g_j(\mathbf{x}) = \sum_{\ell=1}^{N} (w_{i_\ell,j} - w_{i_{\ell+1},j}) \big(1 - \prod_{k=1}^{\ell} (1 - x_{i_k})\big), \tag{6.24}$$

where, for a given $j \in V'$, weights have been pre-sorted in a descending order as $w_{i_1,j} \geq \ldots \geq w_{i_n,j}$ and $w_{i_{n+1},j} \triangleq 0$. We can again extend this problem to maximizing analytic functions h of the utility of a user. For example, for $h(s) = \log(1+s)$, we can maximize

$$f(\mathbf{x}) = \frac{1}{M} \sum_{j=1}^{M} \log\left(1 + g_j(\mathbf{x})\right). \tag{6.25}$$

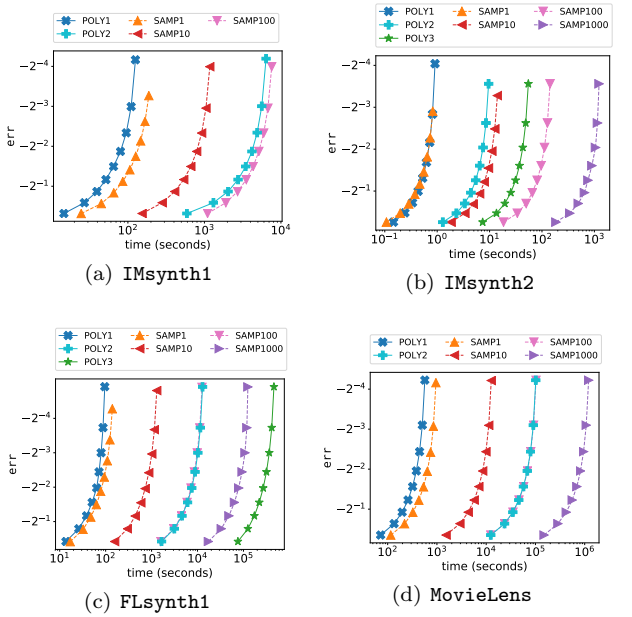
In a manner similar to Sec 6.2, we can show that this function again satisfies Asm. 1 and 2, using the L^{th}-order Taylor approximation of g, given by (6.19). Moreover, as in Thm. 6.2, the corresponding estimator bias is again $\varepsilon(L) \leq \frac{\sqrt{N}}{(L+1)2^L}$. We can again optimize such an objective over arbitrary matroids, which can enforce, e.g., that no more than k facilities are selected from a geographic area or some other partition of V.

7 Experimental Study.

7.1 Experiment Setup. We execute Alg. 1 with sampling and polynomial estimators over 6 different graph settings and 3 different problem instances, summarized in Tab. 3. Our code is publicly available.[2]

instance	dataset	M	N	$\sum_{j=1}^{M} \mathcal{I}$	$\bar{\mathcal{J}}$	m	k	f^*
IM	`IMsynth1`	1	200	200	5.2	10	3	0.3722
IM	`IMsynth2`	1	200	200	5.1	10	3	0.6031
FL	`FLsynth1`	200	200	40000	4.3	10	5	0.5197
FL	`MovieLens`	100	100	10000	4.6	10	4	0.5430
IM	`Epinions`	10	100	1000	3.2	2	2	0.5492
SM	`SMsynth1`	5	200	200	7.4	2	10	0.7669

Table 3: Datasets and Experiment Parameters.



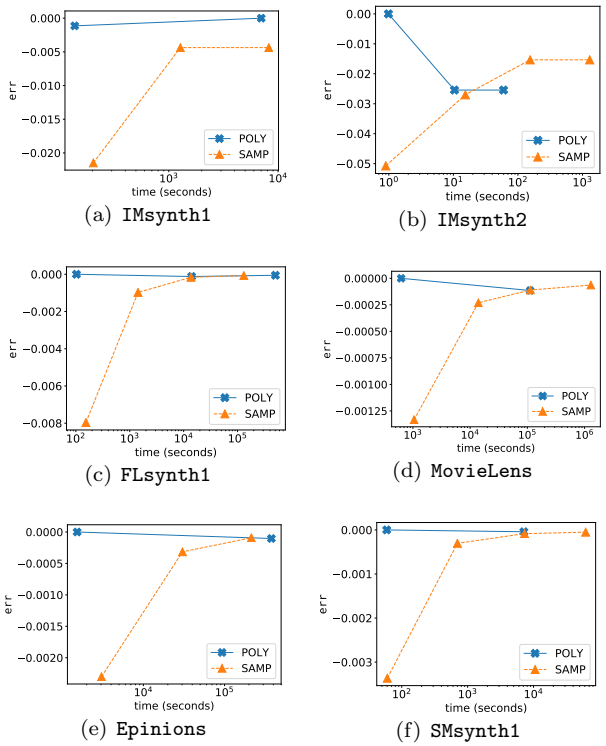
(a) `IMsynth1` (b) `IMsynth2` (c) `FLsynth1` (d) `MovieLens`

Figure 1: Trajectory of the FW algorithm. Utility of the function at the current $\mathbf{y}$ as a function of time is marked for every 10th iteration.

Influence Maximization. We experiment on two synthetic datasets and one real dataset. For synthetic data, we generate two bipartite graphs with $|V_1| = |V_2| = 100$, $|E| = 400$ and $M = 1$. Seeds are always selected from V_1. We select the edges across V_1 and V_2 u.a.r. (`IMsynth1`) or by a power law distribution (`IMsynth2`). We construct a partition matroid of $m = 10$ equal-size partitions of V_1 and set $k = 3$. The real dataset is the `Epinions` dataset [41] on SNAP [42]. We use the subgraph induced by the top $N = 100$ nodes with the largest out-degree and use the IC model [7] with $M = 10$ cascades. The probability for each node to influence its neighbors is set to $p = 0.02$. We construct a matroid of $m = 2$ equal-size partitions and set $k = 5$.

Facility Location. We experiment on one synthetic and one real dataset. We generate a bipartite graph with $N = M = 200$, $|E| = 800$ and

[2] `https://github.com/neu-spiral/WDNFFunctions`



(a) IMsynth1 (b) IMsynth2 (c) FLsynth1 (d) MovieLens (e) Epinions (f) SMsynth1

Figure 2: Comparison of different estimators on different problems. Blue lines represent the POLY estimators and the marked points correspond to POLY1, POLY2, POLY3 respectively. Orange lines represent the SAMP estimators and the marked points correspond to SAMP1, SAMP10, SAMP100, SAMP1000 respectively.

select the edges across V and V' u.a.r (FLsynth1). Weights of the edges $(w_{i,j})$ are selected randomly from $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. We construct a matroid of $m = 10$ equal-size partitions and set to $k = 4$. The real one is a subgraph of the MovieLens 1M dataset with the top $N = 100$ users who rated the most movies and the $M = 100$ movies chosen u.a.r. among the movies rated by the user who rated the most movies [43]. In this problem, we treat movies as facilities, users as customers, and ratings as $w_{i,j}$. We construct a matroid of $m = 10$ partitions by dividing movies according to their genres. We consider the first genre name listed if a movie belongs to multiple genres and we set $k = 2$.

Summarization. We generate a synthetic dataset with $N = 200$ nodes (SMsynth1). We assign a reward r_i to each node i u.a.r between $[0, 1]$ and divide each r_i with $\sum_i r_i$. We divide the nodes into $M = 5$ equal-size P_j. We construct a matroid of $m = 2$ equal-size partitions and set $k = 10$.

Algorithms. We compare the performance of different estimators. These estimators are: (a) sampling estimator (SAMP) with $T = 1, 10, 100, 1000$ and (b) polynomial estimator (POLY) with $L = 1, 2, 3$.

Metrics. We measure the performance of the estimators via $\texttt{err} = (f(\mathbf{y}) - f^*)/f^*$, where $f^* = \max f(\mathbf{y})$ is the maximum utility achieved using the best estimator for a given setting, and execution time. f^* values are reported on Tab. 3.

7.2 Results. The trajectory of the normalized difference between the utility obtained at each iteration of the continuous greedy algorithm (err) is shown as a function of time in Fig. 1. In Fig. 1(a), we see that both POLY1 and POLY2 outperforms sampling estimators. Moreover, POLY1 is almost 60 times faster than SAMP100. In Fig. 1(b), POLY1 runs as fast as SAMP1 and outperforms all estimators. It is important to note that POLY3 runs 2.5 times faster than SAMP1000. In Fig. 1(c), POLY1 visibly outperforms SAMP1 and in Fig. 1(d) polynomial estimators give comparable results to sampler estimators. Note that, even though small number of samples give comparable results, setting $T \leq 100$, is below the value needed to attain the theoretical guarantees of the continuous-greedy algorithm. This can be explained by the $1/2$ approximation guarantee of the greedy algorithm.

The err of the final results of the estimators are reported as a function of time in Figure 2. In all figures except Fig. 2(a), POLY1 outperforms other estimators in terms of time and/or utility whereas in Fig. 2(a) POLY2 is the best performer. As the number of samples increases, the quality of the sampling estimators increases and they catch up with the polynomial estimators. However, considering the running time, POLY1 still remains the better choice.

8 Conclusion.

We have shown that polynomial estimators can replace sampling of the multilinear relaxation. Our approach applies to other tasks, including rounding (see [38]) and stochastic optimization methods [17]. For example, sampling terms of the polynomial approximation can extend our method to even larger problems.

References

[1] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SICOMP*, 2011.

[2] H. Lin and J. Bilmes, "A class of submodular functions for document summarization," in *ACL*, 2011.

[3] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions," in *NAACL*, 2010.

[4] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *CVPR*, 2015.

[5] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*, Cambridge University Press, 2014.

[6] B. Mirzasoleiman, A. Badanidiyuru, and A. Karbasi, "Fast constrained submodular maximization: Personalized data summarization.," in *ICML*, 2016.

[7] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.

[8] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *JMLR*, 2008.

[9] Z. Jiang, G. Zhang, and L. S. Davis, "Submodular dictionary learning for sparse coding," in *CVPR*, 2012.

[10] F. Zhu, L. Shao, and M. Yu, "Cross-modality submodular dictionary learning for information retrieval," in *CIKM*, 2014.

[11] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, "Streaming submodular maximization: Massive data summarization on the fly," in *KDD*, 2014.

[12] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of operations research*, 1978.

[13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical programming*, 1978.

[14] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *STOC*, 2008.

[15] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, 2004.

[16] C. Chekuri, J. Vondrak, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *FOCS*, 2010.

[17] M. Karimi, M. Lucic, H. Hassani, and A. Krause, "Stochastic submodular maximization: The case of coverage functions," in *NeurIPS*, 2017.

[18] Y. Singer, "How to win friends and influence people, truthfully: influence maximization mechanisms for social networks," in *WSDM*, 2012.

[19] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques*, Springer, 1978.

[20] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy," in *AAAI*, 2015.

[21] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*, 2014.

[22] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*, 2015.

[23] M. Feldman, J. Naor, and R. Schwartz, "A unified continuous greedy algorithm for submodular maximization," in *FOCS*, 2011.

[24] C. Chekuri, J. Vondrák, and R. Zenklusen, "Submodular function maximization via the multilinear relaxation and contention resolution schemes," *SICOMP*, 2014.

[25] A. Bian, K. Levy, A. Krause, and J. M. Buhmann, "Continuous dr-submodular maximization: Structure and algorithms," in *NeurIPS*, 2017.

[26] A. A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *AISTATS*, 2017.

[27] C. Chekuri, T. Jayram, and J. Vondrák, "On multiplicative weight updates for concave and submodular function maximization," in *ITCS*, 2015.

[28] F. Bach, "Submodular functions: from discrete to continuous domains," *Mathematical Programming*, 2019.

[29] R. Niazadeh, T. Roughgarden, and J. Wang, "Optimal algorithms for continuous non-monotone submodular and dr-submodular maximization," in *NeurIPS*, 2018.

[30] T. Soma and Y. Yoshida, "Non-monotone dr-submodular function maximization," in *AAAI*, 2017.

[31] Y. Bian, J. Buhmann, and A. Krause, "Optimal continuous dr-submodular maximization and applications to provable mean field inference," in *ICML*, 2019.

[32] M. Staib and S. Jegelka, "Robust budget allocation via continuous submodular functions," in *ICML*, 2017.

[33] H. Hassani, M. Soltanolkotabi, and A. Karbasi, "Gradient methods for submodular maximization," in *NeurIPS*, 2017.

[34] A. Mokhtari, H. Hassani, and A. Karbasi, "Conditional gradient method for stochastic submodular maximization: Closing the gap," in *AISTATS*, 2018.

[35] A. Mokhtari, H. Hassani, and A. Karbasi, "Stochastic conditional gradient methods: From convex minimization to submodular maximization," *JMLR*, 2020.

[36] A. Asadpour, H. Nazerzadeh, and A. Saberi, "Stochastic submodular maximization," in *WINE*, 2008.

[37] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly cache networks," *IEEE/ACM Transactions on Networking*, 2020.

[38] G. Özcan, A. Moharrer, and S. Ioannidis, "Submodular maximization via Taylor series approximation," 2021. `https://arxiv.org/abs/2101.07423`.

[39] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD*, 2009.

[40] G. Cornuejols, M. Fisher, and G. Nemhauser, "Location of bank accounts of optimize float: An analytic study of exact and approximate algorithm," *Management Science*, 1977.

[41] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *ISWC*, 2003.

[42] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," June 2014.

[43] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *TiiS*, 2015.