Monotone Closure of Relaxed Constraints in Submodular Optimization: Connections Between Minimization and Maximization

Rishabh Iyer
Dept. of Electrical Engineering
University of Washington
Seattle, WA-98175, USA

Stefanie Jegelka
Dept. of EECS
University of California, Berkeley
Berkeley, CA-94720, USA

Jeff Bilmes
Dept. of Electrical Engineering
University of Washington
Seattle, WA-98175, USA

Abstract

It is becoming increasingly evident that many machine learning problems may be reduced to submodular optimization. Previous work addresses generic discrete approaches and specific relaxations. In this work, we take a generic view from a relaxation perspective. We show a relaxation formulation and simple rounding strategy that, based on the monotone closure of relaxed constraints, reveals analogies between minimization and maximization problems, and includes known results as special cases and extends to a wider range of settings. Our resulting approximation factors match the corresponding integrality gaps. For submodular maximization, a number of relaxation approaches have been proposed. A critical challenge for the practical applicability of these techniques, however, is the complexity of evaluating the multilinear extension. We show that this extension can be efficiently evaluated for a number of useful submodular functions, thus making these otherwise impractical algorithms viable for real-world machine learning problems.

1 INTRODUCTION

Submodularity is a natural model for many real-world problems including many in the field of machine learning. Submodular functions naturally model aspects like cooperation, complexity, and attractive potentials in minimization problems, and also notions of diversity, coverage, and information in maximization problems. A function $f : 2^V \to \mathbb{R}$ on subsets of a ground set $V = \{1, 2, \ldots, n\}$ is *submodular* [37, 15] if for all subsets $S, T \subseteq V$, we have $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. The *gain* of an element $j \in V$ with respect to $S \subseteq V$ is defined as $f(j|S) \triangleq f(S \cup j) - f(S)$. Submodularity is equivalent to *diminishing gains*: $f(j|S) \geq f(j|T), \forall S \subseteq T, j \notin T$.

A large number of machine learning problems may be phrased as submodular minimization or maximization problems. In this paper, we address the following two very general forms of submodular optimization:

$$\text{Problem 1: } \min_{X \in \mathcal{C}} f(X), \qquad \text{Problem 2: } \max_{X \in \mathcal{C}} f(X)$$

Here, $\mathcal{C}$ denotes a family of feasible sets, described e.g., by cardinality constraints, or by combinatorial constraints insisting that the solution be a tree, path, cut, matching, or a cover in a graph.

Applications. Unconstrained submodular minimization occurs in machine learning and computer vision in the form of combinatorial regularization terms for sparse reconstruction and denoising, and MAP inference, e.g. for image segmentation [30]. Other applications are well modeled as constrained submodular minimization. For example, a rich class of models for image segmentation has been encoded as minimizing a submodular functions subject to cut constraints [28]. Similarly, [9] efficiently solves MAP inference in a sparse higher-order graphical model through submodular vertex cover, and [48] proposes to interactively segment images by minimizing a submodular function subject to connectivity constraints, i.e., the selected set of vertices must contain an s-t path. Moreover, bounded-complexity corpus construction [36] can be modeled as cardinality constrained submodular minimization. Constrained submodular maximization is a fitting model for problems such as optimal sensing [32], marketing [29], document summarization [35], and speech data subset selection [34].

Previous Work. Since most instances of Problems 1 and 2 are NP-hard, one must strive for approximations that have bounded error. Broadly speaking[1], the algorithms can be classified into discrete (*combinatorial*) and continuous *relaxation* based. The discrete approaches were initially proposed for certain *specific* constraints [17, 27, 47, 41, 12, 4, 3], but later made *general* and unified [25, 18, 24]. In the case of submodular minimization, the discrete approaches have been based on approximating the submodular function

[1] Emphasized words in this paragraph correspond to headings in Table 1, which also serves as a summary.

by tractable approximations [25, 18], while in the case of submodular maximization, they have been based on greedy and local search techniques [25, 41, 12, 4, 3]. Most of these algorithms are *fast* and scalable. The continuous relaxation techniques, on the other hand, have so far either been analyzed for very specific constraints, or when general, are too *slow* to use in practice. For example, in the case of minimization, they were studied only for the specific constraints of covers [20] and cuts [27], and in the case of maximization, the techniques though general have yet to show significant practical impact due to their prohibitive computational costs [6, 5]. Hence discrete algorithms are typically used in applications (e.g., [34]).

Constraints or Function	Operation (& speed)	Algorithm Approach	
		Combinatorial	Relaxation
Specific	Min (fast)	[17, 27]	[20, 27]
	Min (slow)	[47]	Unnecessary
	Max (fast)	[41, 12, 4, 3]	**This paper**
	Max (slow)	Unnecessary	[4, 5]
General	Min (fast)	[25]	**This paper**
	Min (slow)	[18]	Unnecessary
	Max (fast)	[25]	Open
	Max (slow)	Unnecessary	[6]

Table 1: Past work & our contributions (see text for explanation).

In the present paper, we develop a continuous relaxation methodology for Problems 1 and 2 that applies not only for multiple types of constraints but that even establishes connections between minimization and maximization problems. We summarize our contributions, in comparison to previous work, in Table 1, which lists one problem as being still open, and other problems as being unnecessary (given a "fast" approach, the corresponding "slow" approach is unnecessary). Our techniques are not only connective, but also fast and scalable. In the case of constrained minimization, we provide a formulation applicable for a large class of constraints. In the case of submodular maximization, we show how for a large class of submodular functions of practical interest, the generic slow algorithms can be made fast and scalable. We note, however, that it is still an open problem to provide a fast and scalable algorithmic framework (with theoretical guarantees) based on continuous relaxations for general submodular maximization.

The connections between minimization and maximization is based on the up- or down-monotonicity of the constraint set: up-monotone constraints are relevant for submodular minimization problems, and down-monotone constraints are relevant for submodular maximization problems. Our relaxation viewpoint, moreover, complements and improves on the bounds found in [25]. For example, where [25] may have an approximation bound of k, our results imply a bound of $n-k+1$, where $n = |V|$, so considering both [25] and our new work presented here, we obtain combined bounds of the form $\min(k, n-k+1)$ (more specifics are given in Table 2). This also holds for maximization – in certain cases discrete algorithms obtain suboptimal results, while relaxation techniques obtain improved, and sometimes optimal guarantees.

The idea of our relaxation strategy is as follows: the submodular function $f(S)$, which is defined on the vertices of the n-dimensional hypercube, is extended to a function defined on $[0,1]^n$. The two functions valuate identically if the vector $x \in [0,1]^n$ is the characteristic vector of a set. We then solve a continuous optimization problem subject to linear constraints. For minimization, the convex *Lovász extension* defined in Eqn. (1) is a suitable extension of f. Appropriately rounding the resulting optimal continuous solutions leads to a number of approximation guarantees. For maximization, ideally we could utilize a concave extension. Since the tightest concave extension of a submodular function is hard to characterize [49], we instead use the *multilinear extension* (see Eqn. (2)) that behaves like a concave function in certain directions [6, 5]. Our resulting algorithms often achieve better bounds than discrete greedy approaches.

Paper Roadmap. For constrained minimization (Sec. 3), we provide a generic approximation factor (Theorem 1), for the general class of constraints defined in Eq. 4. We show that many important constraints, including matroid, cardinality, covers, cuts, paths, matchings, etc. can be expressed as Eq. 4. As a corollary to our main result (Theorem 1), we obtain known results (like covers [20] and cuts [27]), and also novel ones (for spanning trees, cardinality constraints, paths, matchings etc.). We also show bounds on integrality gaps for constrained submodular minimization, which to our knowledge is novel. In the context of maximization (Sec. 4), we provide closed form multi-linear extensions for several submodular functions useful in applications. We also discuss the implications of these algorithmically. Note that this is particularly important, given that many optimal algorithms for several submodular maximization problems are based on the multilinear extension. Lastly, we extend our techniques to minimize the difference between submodular functions, and provide efficient optimization and rounding techniques for these problems (Sec. 5).

2 CONTINUOUS RELAXATIONS

Convex relaxation. The Lovász extension [37] reveals an important connection between submodularity and convexity, and is defined as follows. For each $y \in [0,1]^n$, we obtain a permutation σ_y by ordering its elements in non-increasing order, and thereby a chain of sets $\Sigma_0^y \subseteq \ldots \subseteq \Sigma_n^y$, with $\Sigma_j^y = \{\sigma_y(1), \cdots, \sigma_y(j)\}$ for $j \in \{1, 2, \ldots, n\}$. The Lovász extension $\breve{f}$ of f is a weighted sum of the ordered entries of y:

$$\breve{f}(y) = \sum_{j=1}^{n} y[\sigma_y(j)] \left(f(\Sigma_j^y) - f(\Sigma_{j-1}^y) \right) \qquad (1)$$

The Lovász extension is unique (despite possibly non-unique orderings if y has duplicate entries), and convex if and only if f is submodular. Since it agrees with f on the vertices of the hypercube, i.e., $f(X) = \breve{f}(1_X)$, for all $X \subseteq V$ (where 1_X is the characteristic vector of X, i.e., $1_X(j) = I(j \in X)$), $\breve{f}$ is a natural convex extension of a submodular function. The Lovász extension is a non-smooth (piece-wise linear) convex function for which a subgradient $h^f_{\sigma_y}$ at y can be computed efficiently via Edmonds's' greedy algorithm [10]:

$$h^f_{\sigma_y}(\sigma_y(j)) = f(\Sigma^y_j) - f(\Sigma^y_{j-1}), \;\; \forall j \in \{1, 2, \cdots, n\}$$

The Lovász extension has also found applications in defining norms for structured sparsity [1] and divergences for rank aggregation [23].

Multilinear relaxations. For maximization problems, the relaxation of choice has frequently been the multilinear extension [12]

$$\tilde{f}(x) = \sum_{X \subseteq V} f(X) \prod_{i \in X} x_i \prod_{i \notin X} (1 - x_i), \qquad (2)$$

where f is any set function. Since Eqn. (2) has an exponential number of terms, its evaluation in general computationally expensive, or requires approximation.

One may define at least two types of gradients for the multilinear extension. The first, "standard" gradient is

$$\nabla_j \tilde{f}(x) = \partial \tilde{f} / \partial x_j = \tilde{f}(x \vee e_j) - \tilde{f}(x \vee e_j - e_j).$$

where $e_j = 1_{\{j\}}$, and $\{x \vee y\}(i) = \max(x(i), y(i))$. A second gradient is $\nabla^a_j \tilde{f}(x) = \tilde{f}(x \vee e_j) - \tilde{f}(x)$. The two gradients are related component-wise as $\nabla_j \tilde{f}(x) = (1 - x_j)\nabla^a_j \tilde{f}(x)$, and both can be computed in $O(n)$ evaluations of $\tilde{f}$.

Optimization. Relaxation approaches for submodular optimization follow a two-stage procedure:

1. Find the optimal (or approximate) solution $\hat{x}$ to the problem $\min_{x \in \mathcal{P}_\mathcal{C}} \breve{f}(x)$ (or $\max_{x \in \mathcal{P}_\mathcal{C}} \tilde{f}(x)$).
2. Round the continuous solution $\hat{x}$ to obtain the discrete indicator vector of set $\hat{X}$.

Here, $\mathcal{P}_\mathcal{C}$ denotes the polytope corresponding to the family $\mathcal{C}$ of feasible sets – i.e., their convex hull or its approximation, which is a "continuous relaxation" of the constraints $\mathcal{C}$. The final approximation factor is then $f(\hat{X})/f(X^*)$, where X^* is the exact optimizer of f over $\mathcal{C}$.

An important quantity is the *integrality gap* that measures – over the class $\mathcal{S}$ of all submodular (or monotone submodular) functions – the largest possible discrepancy between the optimal discrete solution and the optimal continuous solution. For minimization problems, the integrality gap is defined as:

$$\mathcal{I}^\mathcal{S}_\mathcal{C} \triangleq \sup_{f \in \mathcal{S}} \frac{\min_{X \in \mathcal{C}} f(X)}{\min_{x \in \mathcal{P}_\mathcal{C}} \breve{f}(x)} \geq 1. \qquad (3)$$

For maximization problems, we would take the supremum over the inverse ratio. In both cases, $\mathcal{I}^\mathcal{S}_\mathcal{C}$ is defined only for non-negative functions. The integrality gap largely depends on the specific formulation of the relaxation. Intuitively, it provides a lower bound on our approximation factor: we usually cannot expect to improve the solution by rounding, because any rounded discrete solution is also a feasible solution to the relaxed problem. One rather only hopes, when rounding, to not worsen the cost relative to that of the continuous optimum. Indeed, integrality gaps can often be used to show tightness of approximation factors obtained from relaxations and rounding [7]. For a detailed discussion on this connection, see [26].

3 SUBMODULAR MINIMIZATION

For submodular minimization, the optimization problem in Step 1 is a convex optimization problem, and can be solved efficiently if one can efficiently project onto the polytope $\mathcal{P}_\mathcal{C}$. Our second ingredient is rounding. To round, a surprisingly simple thresholding turns out to be quite effective for a large number of constrained and unconstrained submodular minimization problems: choose an appropriate $\theta \in (0, 1)$ and pick all elements with "weights" above θ, i.e., $\hat{X}_\theta = \{i : \hat{x}(i) \geq \theta\}$. We call this procedure the *θ-rounding procedure*. In the following sections, we first review relaxation techniques for unconstrained minimization (which are known), and afterwards phrase a generic framework for constrained minimization. Interestingly, both constrained and unconstrained versions essentially admit the same rounding strategy and algorithms.

3.1 UNCONSTRAINED MINIMIZATION

Continuous relaxation techniques for unconstrained submodular minimization have been well studied [1, 15]. In this case, $\mathcal{P}_\mathcal{C} = [0, 1]^n$, and importantly, the approximation factor and integrality gap are both 1.

Lemma 1. *[15] For any submodular function f, it holds that $\min_{X \subseteq V} f(X) = \min_{x \in [0,1]^n} \breve{f}(x)$. Given a continuous minimizer $x^* \in \operatorname{argmin}_{x \in [0,1]^n} \breve{f}(x)$, the discrete minimizers are exactly those obtained by θ-rounding x^*, for any $\theta \in (0, 1)$.*

Since the Lovász extension is a non-smooth convex function, it can be minimized up to an additive accuracy of ϵ in $O(1/\epsilon^2)$ iterations of the subgradient method. This accuracy directly transfers to the discrete solution if we choose the best set obtained with any $\theta \in (0, 1)$ [1]. For special cases, such as submodular functions derived from concave functions, smoothing techniques yield a convergence rate of $O(1/t)$ [45].

3.2 CONSTRAINED MINIMIZATION

We next address submodular minimization under constraints, where rounding affects the accuracy of the discrete solution. By appropriately formulating the problem, we show that θ-rounding applies to a large class of problems. We assume that the family $\mathcal{C}$ of feasible solutions can be expressed by a (low-order) polynomial number of linear inequalities, or at least that linear optimization over $\mathcal{C}$ can be done efficiently, as is the case for matroid polytopes [10].

A straightforward relaxation of $\mathcal{C}$ is the convex hull $\mathcal{P}_{\mathcal{C}} = \text{conv}(1_X, X \in \mathcal{C})$ of $\mathcal{C}$. Often however, it is not possible to obtain a decent description of the inequalities determining $\mathcal{P}_{\mathcal{C}}$, even in cases when minimizing a linear function over $\mathcal{C}$ is easy (two examples are the s-t cut and s-t path polytopes [44]). In those cases, we relax $\mathcal{C}$ to its *up-monotone* closure $\widehat{\mathcal{C}} = \{X \cup Y \mid X \in \mathcal{C} \text{ and } Y \subseteq V\}$. With $\widehat{\mathcal{C}}$, a set is feasible if it is in $\mathcal{C}$ or a superset of a set in $\mathcal{C}$. The convex hull of $\widehat{\mathcal{C}}$ is the up-monotone extension of $\mathcal{P}_{\mathcal{C}}$ within the hypercube, i.e. $\mathcal{P}_{\widehat{\mathcal{C}}} = \hat{\mathcal{P}}_{\mathcal{C}} = (\mathcal{P}_{\mathcal{C}} + \mathbb{R}^n_+) \cap [0,1]^n$, which is often easier to characterize than $\mathcal{P}_{\mathcal{C}}$ [26]. If $\mathcal{C}$ is already up-monotone, then $\hat{\mathcal{P}}_{\mathcal{C}} = \mathcal{P}_{\mathcal{C}}$.

Optimization. The relaxed minimization problem $\min_{x \in \hat{\mathcal{P}}_{\mathcal{C}}} \breve{f}(x)$ is non-smooth and convex with linear constraints, and therefore amenable to, e.g., projected subgradient methods. We here assume that the submodular function f is monotone nondecreasing (which often holds in applications), and extend our results to non-monotone functions in [26].

For projected (sub)gradient methods, it is vital that the projection on $\hat{\mathcal{P}}_{\mathcal{C}}$ can be done efficiently. Indeed, this holds with the above assumptions that we can efficiently solve a linear optimization over $\hat{\mathcal{P}}_{\mathcal{C}}$. In this case, e.g. Frank-Wolfe [13] methods apply. The projection onto matroid polyhedra can also be cast as a form of unconstrained submodular function minimization and is hence polynomial time solvable [15]. To apply splitting methods such as the alternating directions method of multipliers (ADMM) [2], we write the problem as $\min_{x,y:x=y} \breve{f}(x) + I(y \in \hat{\mathcal{P}}_{\mathcal{C}})$. ADMM needs a projection oracle onto the constraints – discussed above – and the proximal operator of f. Computing the proximal operator of the Lovász extension is equivalent to unconstrained submodular minimization, or to solving the minimum norm point problem. In special cases, faster algorithms apply [39, 45].

Rounding. Once we have obtained a minimizer $\hat{x}$ of $\breve{f}$ over $\widehat{\mathcal{P}}_{\mathcal{C}}$, we apply simple θ-rounding. Whereas in the unconstrained case, $\hat{X}_\theta$ is feasible for any $\theta \in (0,1)$, we must now ensure $\hat{X}_\theta \in \widehat{\mathcal{C}}$. Hence, we pick the largest threshold θ such that $\hat{X}_\theta \in \widehat{\mathcal{C}}$, i.e., the smallest $\hat{X}_\theta$ that is feasible. This is always possible since $\widehat{\mathcal{C}}$ is up-monotone and contains V. The threshold θ can be found using $O(\log n)$ checks among the sorted entries of the continuous solution $\hat{x}$. The following lemma states how the threshold θ determines a worst-case approximation:

Lemma 2. *For a monotone submodular f and any $\hat{x} \in [0,1]^V$ and $\theta \in (0,1)$ such that $\hat{X}_\theta = \{i \mid \hat{x}_i \geq \theta\} \in \hat{\mathcal{C}}$, it holds that $f(\hat{X}_\theta) \leq \frac{1}{\theta}\breve{f}(\hat{x})$. If, moreover, $\breve{f}(\hat{x}) \leq \beta \min_{x \in \hat{\mathcal{P}}_{\mathcal{C}}} \breve{f}(x)$, then it holds that $f(\hat{X}_\theta) \leq \frac{\beta}{\theta} \min_{X \in \mathcal{C}} f(X)$.*

The set $\hat{X}_\theta$ is in $\widehat{\mathcal{C}}$ and therefore guaranteed to be a superset of a solution $\hat{Y}_\theta \in \mathcal{C}$. As a final step, we prune down $\hat{X}_\theta$ to $\hat{Y}_\theta \subseteq \hat{X}_\theta$. Since the objective function is nondecreasing, $f(\hat{Y}_\theta) \leq f(\hat{X}_\theta)$, Lemma 2 holds for $\hat{Y}_\theta$ as well. If, in the worst case, $\theta = 0$, then the approximation bound in Lemma 2 is unbounded. Fortunately, in most cases of interest we obtain polynomially bounded approximation factors.

In the following, we will see that our $\widehat{\mathcal{P}}_{\mathcal{C}}$ provides the basis for relaxation schemes under a variety of constraints, and that these, together with θ-rounding, yield bounded-factor approximations. We assume that there exists a family $\mathcal{W} = \{W_1, W_2, \dots\}$ of sets $W_i \subseteq V$ such that the polytope $\widehat{\mathcal{P}}_{\mathcal{C}}$ can be described as

$$\hat{\mathcal{P}}_{\mathcal{C}} = \Big\{x \in [0,1]^n \;\Big|\; \sum_{i \in W} x_i \geq b_W \text{ for all } W \in \mathcal{W}\Big\}. \quad (4)$$

Analogously, this means that $\hat{\mathcal{C}} = \{X \mid |X \cap W| \geq b_W, \text{ for all } W \in \mathcal{W}\}$. In our analysis, we do not require $\mathcal{W}$ to be of polynomial size, but a linear optimization over $\widehat{\mathcal{P}}_{\mathcal{C}}$ or a projection onto it should be possible at least within a bounded approximation factor. This is the case for s-t paths and cuts, covering problems, and spanning trees.

The following main result (proven in [26]) states approximation bounds and integrality gaps for the class of problems described by Equation (4).

Theorem 1. *The θ-rounding scheme for constraints $\mathcal{C}$ whose relaxed polytope $\hat{\mathcal{P}}_{\mathcal{C}}$ can be described by Equation (4) achieves a worst case approximation bound of* $\max_{W \in \mathcal{W}} |W| - b_W + 1$.

We also show [26] that this factor matches the integrality gap for the constraints considered in this paper.

A result similar to Theorem 1 was shown in [31] for a different, greedy algorithmic technique. While their result also holds for a large class of constraints, for the constraints in Equation (4) they obtain a factor of $\max_{W \in \mathcal{W}} |W|$, which is worse than Theorem 1 if $b_W > 1$. This is the case, for instance, for matroid span constraints, cardinality constraints, trees and multiset covers.

Pruning. The final piece of the puzzle is the pruning step, where we reduce the set $\hat{X}_\theta \in \hat{\mathcal{C}}$ to a final solution $\hat{Y}_\theta \subseteq \hat{X}_\theta$ that is feasible: $\hat{Y}_\theta \in \mathcal{C}$. This is important when the true constraints $\mathcal{C}$ are not up-monotone, as is the case for cuts or paths. Since we have assumed that the function f is monotone, pruning can only reduce the objective value. The pruning step means finding *any* subset of $\hat{X}_\theta$ that is in $\mathcal{C}$, which is often not hard. We propose the following heuristic for this: if $\mathcal{C}$ admits (approximate) linear optimization, as is the case for all the constraints considered here, then we may improve over a given rounded subset by assigning additive weights: $w(i) = \infty$ if $i \notin \hat{X}_\theta$, and otherwise use either uniform ($w(i) = 1$) or non-uniform ($w(i) = 1 - \hat{x}(i)$) weights. We then solve $\hat{Y}_\theta \in \operatorname{argmin}_{Y \in \mathcal{C}} \sum_{i \in Y} w(i)$. Uniform weights lead to the solution with minimum cardinality, and non-uniform weights will give a bias towards elements with higher certainty in the continuous solution. Truncation via optimization works well for paths, cuts, matchings or matroid constraints. In the extended version [26], we discuss how to handle non-monotone submodular functions and down-monotone constraints.

To demonstrate the utility of Theorem 1, we apply it to a variety of problems. We state only the results, all proofs are in [26]. Many of the constraints below are based on a graph $G = (\mathcal{V}, \mathcal{E})$, and in that case the ground set is the set $\mathcal{E}$ of graph edges. When the context is clear, we overload notation and refer to $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Results are summarized in Table 2.

3.2.1 MATROID CONSTRAINTS

An important class of constraints are matroid span or base constraints, with cardinality constraints (uniform matroids) and spanning trees (graphic or cycle matroids) as special cases. A matroid $\mathcal{M} = (\mathcal{I}_\mathcal{M}, r_\mathcal{M})$ is defined by its down-monotone family of independent sets $\mathcal{I}_\mathcal{M}$ or its rank function $r_\mathcal{M} : 2^V \to \mathbb{R}$. A set Y is a *spanning set* if its rank is that of V: $r_M(Y) = r_M(V)$. It is a *base* if $|Y| = r_M(Y) = r_M(V)$. Hence, the family of all spanning sets is the up-monotone closure of the family of all bases (e.g., supersets of spanning trees of a graph in the case of a graphic matroid). See [44] for more details on matroids. Let $\mathcal{S}_\mathcal{M}$ denote the spanning sets of matroid $\mathcal{M}$, and set $k = r_\mathcal{M}(V)$. It is then easy to see that with $\mathcal{C} = \mathcal{S}_\mathcal{M}$, the polytope $\mathcal{P}_\mathcal{C}$ is the matroid span polytope, which can be described as $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^n, x(S) \geq r_\mathcal{M}(V) - r_\mathcal{M}(V \backslash S), \forall S \subseteq V\}$ [44]. This is clearly in the form of Eqn. 4. Although this polytope is described via an exponential number of inequalities, it can be projected onto efficiently via submodular minimization [15].

Corollary 1. *Let $\hat{Y}_\theta$ be the rounded and pruned solution obtained from minimizing the Lovász extension over the span polytope. Then $f(\hat{Y}_\theta) \leq (n-k+1) f(X^*)$. The integrality gap is also $n-k+1$.*

In general, the rounding step will only provide an $\hat{X}_\theta$ that is a spanning set, but not a base. We can prune it to a base by greedily finding a maximum weight base among the elements of $\hat{X}_\theta$. The worst-case approximation factor of $n-k+1$ complements other known results for this problem [25, 18]. The semi-gradient framework of [25] guarantees a bound of k, while more complex (and less practical) approximations [18] yield factors of $O(\sqrt{n})$. The factor k of [25] is the best for small k, while our continuous relaxation works well when k is large.

Cardinality Constraints. This is a special class of a matroid, called the uniform matroid. Since it suffices to analyze monotone submodular functions, the constraint of interest is $\mathcal{C} = \{X : |X| = k\}$. In this case, the corresponding polytope takes a very simple form: $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^n : \sum_i x_i = k\}$. Furthermore, the rounding step in this context is very intuitive. It corresponds to choosing the elements with the k largest entries in $\hat{x}$.

Spanning Trees. Here, the ground set $V = \mathcal{E}$ is the edge set in a graph and $\mathcal{C}$ is the set of all spanning trees. The corresponding polytope $\mathcal{P}_\mathcal{C}$ is then the spanning tree polytope. Our bound in this setting is $m - n + 1$. The discrete algorithms of [25, 17] achieve a complementary bound of $|\mathcal{V}| = n$. For dense graphs, the discrete algorithms admit better worst case guarantees, while for sparse graphs (e.g., embeddable into r-regular graphs for small r), our guarantees are better.

3.2.2 SET COVERS

A fundamental family of constraints are set covers. Given a universe $\mathcal{U}$, and a family of sets $\{S_i\}_{i \in V}$, the task is to find a subset $X \subseteq V$ that covers the universe, i.e., $\bigcup_{i \in X} S_i = \mathcal{U}$, and has minimum cost as measured by a submodular function $f : 2^\mathcal{S} \to \mathbb{R}$. The set cover polytope is up-monotone, constitutes the set of fractional covers, and is easily represented by Eqn. (4) as $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^{|V|} \mid \sum_{i: u \in S_i} x(i) \geq 1, \forall u \in \mathcal{U}\}$. The following holds for minimum submodular set cover:

Corollary 2. *The approximation factor of our algorithm, and the integrality gap for the minimum submodular set cover problem, is $\gamma = \max_{u \in \mathcal{U}} |\{i : u \in S_i\}|$.*

The approximation factor in Corollary 2 (without the integrality gap) was first shown in [20]. The quantity γ corresponds to the maximum frequency of the elements in $\mathcal{U}$.

A generalization of set cover is the multi-set cover problem [43], where every element u is to be covered multiple (c_u) times. The multi-cover constraints can be formalized as $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^{|\mathcal{S}|} \mid \sum_{i: u \in S_i} x(i) \geq c_u, \forall u \in \mathcal{U}\}$.

Corollary 3. *The approximation factor and integrality gap of the multi-set cover problem is $\max_{u \in \mathcal{U}} |\{i : u \in S_i\}| - c_u + 1$.*

This result also implies the bound for set cover (with $c_u = 1$). Since the rounding procedure above yields a solution that is already a set cover (or a multi set cover), a subsequent pruning step is not necessary.

[2]These results were shown in [17, 20, 47]

	Matroid Constraints		Set Covers		Paths, Cuts and Matchings		
	Cardinality	Trees	Vertex Covers	Edge Covers	Cuts	Paths	Matchings
CR.	$n-k+1$	$m-n+1$	2	$deg(G) \leq n$	$P_{max} \leq n$	$C_{max} \leq m$	$deg(G) \leq n$
SG	k	n	$\|VC\| \leq n$	$\|EC\| \leq n$	$C_{max} \leq m$	$P_{max} \leq n$	$\|M\| \leq n$
EA	$\sqrt{n}$	$\sqrt{m}$	$\sqrt{n}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$	$\sqrt{m}$
Integrality Gaps	$\Omega(n-k+1)$	$\Omega(m-n+1)$	2	$\Omega(n)$	$\Omega(n)$	$\Omega(m)$	$\Omega(n)$
Hardness[2]	$\Omega(\sqrt{n})$	$\Omega(n)$	$2-\epsilon$	$\Omega(n)$	$\Omega(\sqrt{m})$	$\Omega(n^{2/3})$	$\Omega(n)$

Table 2: Comparison of the results of our framework (CR) with the semigradient framework of [25] (SG), the Ellipsoidal Approximation (EA) algorithm of [18], hardness [17, 20, 47], and the integrality gaps of the corresponding constrained submodular minimization problems. Note the complementarity between CR and SG.

Vertex Cover. A vertex cover is a special case of a set cover, where $\mathcal{U}$ is the set of edges in a graph, V is the set of vertices, and S_v is the set of all edges incident to $v \in V$. Corollary 2 implies a 2-approximation for submodular vertex cover, which matches the integrality gap and the lower bound in [17]. The 2-approximation for vertex cover was also shown in [17, 20].

Edge Cover. In the Edge Cover problem, $\mathcal{U}$ is the set of vertices in a graph, V is the set of edges and S_v contains the two vertices comprising edge v. We aim to find a subset of edges such that every vertex is covered by some edge in the subset. It is not hard to see that the approximation factor we obtain is the maximum degree of the graph $deg(G)$, which is upper bounded by $|\mathcal{V}|$, but is often much smaller. The algorithm in [25] has an approximation factor of the size of the edge cover $|EC|$, which is also upper bounded by $O(|\mathcal{V}|)$. These factors match the lower bound shown in [17].

3.2.3 CUTS, PATHS AND MATCHINGS

Even though Eqn. (4) is in the form of covering constraints, it can help solve problems with apparently very different types of constraints. The covering generalization works if we relax $\mathcal{C}$ to its up-monotone closure: $\widehat{\mathcal{C}}$ demands that a feasible set must contain (or "cover") a set in $\mathcal{C}$. To go from $\widehat{\mathcal{C}}$ back to $\mathcal{C}$, we prune in the end.

Cuts and Paths. Here, we aim to find an edge set $X \subseteq \mathcal{E}$ that forms an s-t path (or an s-t cut), and that minimizes the submodular function f. Both the s-t path and s-t cut polytopes are hard to characterize. However, their up-monotone extension $\hat{\mathcal{P}}_{\mathcal{C}}$ can be easily described. Furthermore, both these polytopes are intimately related to each other as a blocking pair of polyhedra (see [44]). The extended polytope for s-t paths can be described as a *cut-cover* [44] (i.e., any path must hit every cut at least once): $\hat{\mathcal{P}}_{\mathcal{C}} = \{x \in [0,1]^{|\mathcal{E}|} \mid \sum_{e \in C} x(e) \geq 1$, for every s-t cut $C \subseteq \mathcal{E}\}$. The closure of the s-t path constraint (or the cut-cover) is also called s-t connectors [44]. Conversely, the extended s-t cut polytope can be described as a *path-cover* [44, 27]: $\hat{\mathcal{P}}_{\mathcal{C}} = \{x \in [0,1]^{|\mathcal{E}|} \mid \sum_{e \in P} x(e) \geq 1$, for every s-t path $P \subseteq \mathcal{E}\}$.

Corollary 4. *The relaxation algorithm yields an approximation factor of $P_{max} \leq |\mathcal{V}|$ and $C_{max} \leq |\mathcal{E}|$ for minimum submodular s-t path and s-t cut respectively (P_{max} and C_{max} refer to the maximum size simple s-t path and s-t cut respectively). These match the integrality gaps for both problems.*

While the description of the constraints as covers reveals approximation bounds, it does not lead to tractable algorithms for minimizing the Lovász extension. However, the extended cut and the extended path polytopes can be described exactly by a linear number of inequalities [42, 44]. The pruning step for paths and covers becomes a shortest path or minimum cut problem, respectively. As in the other cases, the approximations obtained from relaxations complement the bounds of P_{max} for paths and C_{max} for cuts shown in [25].

Perfect Matchings. Given a graph $G = (\mathcal{V}, \mathcal{E})$, the goal is to find a set of edges $X \subseteq \mathcal{E}$, such that X is a perfect matching in G and minimizes the submodular function f. For a bipartite graph, the polytope $\hat{\mathcal{P}}_{\mathcal{C}}$ can be characterized as $\mathcal{P}_{\mathcal{C}} = \{x \in [0,1]^{|\mathcal{E}|} \mid \sum_{e \in \delta(v)} x(e) = 1$ for all $v \in \mathcal{V}\}$, where $\delta(v)$ denotes the set of edges incident to v. Similar to the case of Edge Cover, Theorem 1 implies an approximation factor of $deg(G) \leq |\mathcal{V}|$, which matches the lower bound shown in [17, 24].

4 SUBMODULAR MAXIMIZATION

To relax submodular maximization, we use the multilinear extension. We first show that this extension can be efficiently computed for a large subclass of submodular functions (deferring detailed derivations to [26]). As above, $\mathcal{C}$ denotes the family of feasible sets, and $\mathcal{P}_{\mathcal{C}}$ the polytope corresponding to $\mathcal{C}$. For maximization, it makes sense to consider $\mathcal{C}$ to be down-monotone (particularly when the function is monotone). Such a down-monotone $\mathcal{C}$ could represent for example, matroid independence constraints, or upper bounds on the cardinality $\mathcal{C} = \{X : |X| \leq k\}$. Analogous to the case of minimization [26], an approximation algorithm for down-monotone constraints can be extended to up-monotone constraints, by using $f'(X) = f(V \backslash X)$.

The relaxation algorithms use the multilinear extension (Eqn. (2)) which in general requires repeated sampling and can be very expensive to compute. In the below, we show how this can be computed efficiently and exactly for many practical and useful submodular functions.

Weighted Matroid Rank functions. A common class of submodular functions are sums of weighted matroid rank functions, defined as: $f(X) = \sum_i \max\{w_i(A) | A \subseteq X, A \in \mathcal{I}_i\}$, for linear weights $w_i(j)$. These functions form a rich class of coverage functions for summarization tasks [34]. The multilinear extension can be efficiently computed for a number of specific instances of this function. One such special case is the facility location objective [34]: $f(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$, for pairwise similarities s_{ij}. The facility location function admits a nice representation of the multilinear extension (the full derivation is in [26]): $\tilde{f}(x) = \sum_{i \in V} \sum_{l=1}^{n} s_{ij_i^l} x_{ij_i^l} \prod_{m=1}^{l} (1 - x_{ij_i^l})$, where for each $i \in V$, the indices $j_i^1, j_i^2, \cdots, j_i^n$ denote in sorted order the elements closest to i (in terms of similarity s_{ij}). We can similarly obtain closed form expressions with other forms of matroids, including uniform matroids or partition matroids. Due to limited space, detailed derivations are all given in [26].

Set Cover function: This is another important function, capturing notions of coverage [34]. Given a set of sets $\{\mathcal{S}_1, \cdots, S_n\}$ and the universe $\mathcal{U} = \cup_i \mathcal{S}_i$, define $f(X) = w(\cup_{i \in X} \mathcal{S}_i)$, where w_j denotes the weight of item $j \in \mathcal{U}$. This setup can alternatively be expressed via a neighborhood function $\Gamma : 2^V \to 2^{\mathcal{U}}$ such that $\Gamma(X) = \cup_{i \in X} \mathcal{S}_i$. Then $f(X) = w(\Gamma(X))$. Let $\Gamma^{-1}(j) = \{i \in V : j \in \Gamma(i)\}$. Then the multilinear extension has a simple form: $\tilde{f}(x) = \sum_{j \in \mathcal{U}} w_j [1 - \prod_{i \in \Gamma^{-1}(j)} (1 - x_i)]$. Again, for full derivations see [26].

Probabilistic Coverage Functions. This is a generalization of the set cover function above, and has been used in a number of models for summarization problems [11]. This provides a probabilistic notion to the set cover function, and can be defined as $f(X) = \sum_{i \in \mathcal{U}} w_i [1 - \prod_{j \in X} (1 - p_{ij})]$. We get back the set cover function, when p_{ij} is a binary vector (either i covers j or not). This function also has an efficiently computable multilinear extension [26]: $\tilde{f}(x) = \sum_{i \in \mathcal{U}} w_i [1 - \prod_{j \in V} (1 - p_{ij} x_j)]$.

Graph Cut related functions Graph cuts are a widely used class of functions. Their multilinear extension also a admits closed form representation. The function and its multilinear extension can be written as: $f(X) = \sum_{i \in X, j \notin X} s_{ij}$, $\tilde{f}(x) = \sum_{i,j \in V} s_{ij} x_i (1 - x_j)$. A related function is a similarity penalizing function: $f(X) = -\sum_{i,j \in X} s_{ij}$. This function has been used for encouraging diversity [35, 34]. Its multilinear extension is $\tilde{f}(x) = -\sum_{i,j \in V} s_{ij} x_i x_j$. The detailed derivations of both these expressions are in [26].

Sparse Pseudo-Boolean functions. For graphical models, in particular in computer vision, set functions are often written as polynomials [19]. Any set function can be written as a polynomial, $p_f(x) = \sum_{T \subseteq V} \alpha_T \prod_{i \in T} x_i$, where $x \in \{0, 1\}^n$ is the characteristic vector of a set. In other words, $f(S) = \sum_{T \subseteq S} \alpha_T$. Submodular functions are a subclass of these polynomials. This representation directly gives the multilinear extension as the same polynomial, $\tilde{f}(x) = \sum_{T \subseteq V} \alpha_T \prod_{i \in T} x_i$, and is efficiently computable if the polynomial is *sparse*, i.e., has few nonzero coefficients α_T [26]. This is the case for graph cut like functions above and for the functions considered in [46, 19]. We have been unable to find the above result elsewhere in the literature, so we formalize it as follows:

Proposition 1. *The polynomial representation is the multilinear extension:* $\tilde{f}(x) = p_f(x)$

Spectral functions. Diversity can also be encouraged via spectral regularizers [8]. Given a positive definite matrix $S \in \mathbb{R}^{n \times n}$, define S_X to be the $|X| \times |X|$ sub-matrix of the rows and columns indexed by X. Any scalar function ψ whose derivative is operator-antitone defines a submodular function, $f(X) = \sum_{i=1}^{|X|} \psi(\lambda_i(S_X))$, by applying it to the eigenvalues of S_X [14]. The resulting class of submodular functions includes the log determinants occurring in DPP inference [16], and, more generally, a smoothed log-determinant function $f(X) = \log\det(S_X + \delta I_X) = \sum_{i=1}^{|X|} \log(\lambda_i(S_X) + \delta)$. It is monotone for $\delta \geq 1$, and has an efficiently computable soft-max extension that is similar to the multilinear extension [16]. A related function that encourages diversity is $f(X) = -\sum_{i=1}^{|X|} (\lambda_i(S_X) - 1)^2$ [8]. It has a surprisingly simple multilinear extension: $\tilde{f}(x) = -\sum_{i,j \in V} s_{ij}^2 x_i x_j + \sum_{i \in V} (2s_{ii} + 1)$. For the detailed derivation of this, see [26].

Given expressions for the functions above, we can also handle weighted combinations $f(X) = \sum_i \lambda_i f_i(X)$, since its multilinear extension is $\tilde{f}(x) = \sum_i \lambda_i \tilde{f}_i(x)$. In the following sections, we briefly describe relaxation algorithms and rounding schemes for maximization.

4.1 MONOTONE MAXIMIZATION

We first investigate monotone submodular maximization subject to matroid independence constraints $\mathcal{I}$. The technique for maximizing the multilinear extension is the continuous greedy algorithm [50], which is a slight modification of the Frank-Wolfe algorithm [13], with a fixed step size. In each iteration, the algorithm takes a step $x^{t+1} = x^t + \delta h^t$ (with step size $\delta = 1/n^2$) in the direction $h^t = \mathrm{argmax}_{h' \in \mathcal{P}_\mathcal{C}} \langle h', \nabla^a \tilde{f}(x^t) \rangle$ best aligned with the alternate gradient. This *continuous greedy* procedure terminates in $O(n^2)$ iterations, after which we are guaranteed to obtain a point x such that $\tilde{f}(x) \geq (1 - 1/e)\tilde{f}(x^*)$ [5, 49]. Moreover, using the pipage rounding technique (in particular, the deterministic variant [50]) ensures that we can round the continuous solution to a set in $O(n^2)$ function calls.

[3]This extends to top-k facility location as well.

[4]This is for soft-max extension [16].

	Fac. Location [3]	Set Cover	Graph Cuts	Diversity I/ II	Concave over card.	log-Det [4]
Multilinear Closed form	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$
Multilinear Sampling	$O(n^7 \log n)$	$O(n^6)$	$O(n^7)$	$O(n^7)$	$O(n^6)$	$O(n^8)$
Gradient Closed form	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$
Gradient Sampling	$O(n^7 \log n)$	$O(n^7)$	$O(n^8)$	$O(n^8)$	$O(n^7)$	$O(n^9)$

Table 3: Complexity of evaluating the multilinear extensions and their gradients for both the optimized closed forms given in this paper and for sampling at high accuracy.

Unfortunately, naïve computation of the multilinear extension or its gradient takes exponential time. To compute these in polynomial time, we can apply sampling techniques. To obtain an accuracy better than $1/n^2$, we need $O(n^5)$ samples for the multilinear extension or for each coordinate of its gradient [50, 49]. This implies a complexity of $O(n^6)$ function evaluations for the gradient and $O(n^5)$ function evaluations for the extension itself, thus implying the algorithm's complexity as $O(n^8 T_{\nabla f})$, where $T_{\nabla f}$ is the time of evaluating the gain of f. For facility location, this means a running time of $O(n^9 \log n)$, and for set cover functions $O(n^9)$.

But these high complexities are for using a sampling approximation of the generic expression for the multilinear extension (Eqn. (2)). The specialized expressions in Section 4 lead to algorithms that run several orders of magnitude faster. With $O(n^2)$ iterations, the time becomes $O(n^2 T_{\nabla \tilde{f}})$, where $\nabla \tilde{f}$ is the time to compute the gradient of $\tilde{f}$. Table 3 compares the function evaluation times for some useful submodular functions. Moreover, we can also use mixtures of these submodular functions, each with efficiently computable multilinear extensions, and compute the resulting multilinear extension also efficiently. While this is still slower than the accelerated greedy [38], it gains power for more complex constraints, such as matroid independence constraints, where the discrete greedy algorithm only achieves an approximation factor of $1/2$, whereas the continuous greedy obtains at least a $1 - 1/e$ factor. Similarly, the continuous greedy algorithm achieves a $1 - 1/e$ approximation guarantee for multiple knapsack constraints [33], while the discrete greedy techniques do not have such guarantees. Hence, the formulations above make it possible to use the optimal theoretical results with a more manageable running time.

4.2 NON-MONOTONE MAXIMIZATION

In the non-monotone setting, we must find a local optimum of the multilinear extension. We could use, for example, a Frank-Wolfe style algorithm [13] and run it until it converges to a local optimum. It is easy to see that at convergence x satisfies $\langle \nabla \tilde{f}(x), y - x \rangle \leq 0, \forall y \in \mathcal{P}_{\mathcal{C}}$ and is a local optimum. Practically, this would mean checking if $\text{argmax}_{y \in \mathcal{P}_{\mathcal{C}}} \langle y, \nabla \tilde{f}(x) \rangle = x$. For simple or no constraints, we could also use a method like L-BFGS. Running this procedure twice, we are guaranteed to obtain a 0.25 approximate solution [6]. This procedure works for any down-monotone constraint $\mathcal{C}$. Moreover, this procedure with a slightly different extension has been successfully applied in practice to MAP inference with determinantal point processes [16].

A generic rounding strategy for submodular maximization problems was given by [6], and works for a large class of constraints (including matroid, knapsack constraints, and a combination thereof). Without constraints, this amounts to sampling a set by a distribution based on the continuous solution x — it will satisfy $\mathbf{E}_{X \sim x} f(X) = \tilde{f}(x)$. In practice, however, this may not work well. Since the multilinear extension is linear in any coordinate (holding the other ones fixed), a simpler co-ordinate ascent scheme of choosing the better amongst 0 or 1 for any fractional co-ordinate will guarantee a deterministic procedure of obtaining an integral solution no worse than the continuous one.

The above algorithms and rounding techniques offer a general and optimal framework, even for many complex constraints. Moreover, many of the best algorithms for non-monotone submodular maximization are based on the multilinear extension. For example, the best known algorithm for cardinality constrained non-monotone submodular maximization [4] uses a continuous double greedy algorithm on the multilinear extension. However, the practical utility of those algorithms is heavily impaired by computational complexity. In fact, non-monotone functions even require $O(n^7)$ samples [6]. For DPPs, [16] used an extension that is practical and close to the multilinear extension. Since they do not use the multilinear extension, the above rounding schemes do not imply the same approximation bounds as for the multilinear extension, leaving the worst-case approximation quality unknown. The expressions we show above use the multilinear extension and maintain its benefits, demonstrating that for many functions of practical interest, sampling, and hence extremely high complexity, is not necessary. This observation is a step from theory into practice, and allows for the improved approximations to occur in practice.

4.3 INTEGRALITY GAPS

Surprisingly, the multilinear extension has an integrality gap of 1 for a number of constraints including the matroid and cardinality constraints, since it is easy to round it exactly (using say, the pipage rounding or contention resolution schemes [5, 6]).

5 DIFFERENCE OF SUBMODULAR (DS) FUNCTIONS

Finally, we investigate minimizing the differences between submodular functions. Given submodular functions f and g, we consider the following minimization problem: $\min_{X \in \mathcal{C}} \big(f(X) - g(X)\big)$. In fact, any set function can be represented as a difference between two non-negative monotone submodular functions [40, 21]. In the unconstrained setting, $\mathcal{C} = 2^V$. A natural continuous relaxation (not necessarily convex) is $\tilde{h}(x) = \breve{f}(x) - \breve{g}(x)$. The continuous relaxation problem is a DC programming problem, and can be addressed (often very efficiently) using the convex-concave procedure [51]. Moreover, thanks to the special structure of the Lovász extension, there exists a simple rounding scheme for the unconstrained version.

Lemma 3. *Given submodular functions f and g, and a continuous vector x, there exists a $\theta \in (0, 1)$ such that $f(X_\theta) - g(X_\theta) \geq \breve{f}(x) - \breve{g}(x)$, where $X_\theta = \{x \geq \theta\}$. Moreover, the integrality gap of $\tilde{h}(x)$ (in the unconstrained setting) is equal to 1.*

6 DISCUSSION

We have provided a unifying view to continuous relaxation methods for submodular optimization. For minimization problems with various constraints, we provide a generic rounding strategy with new approximation bounds and matching integrality gaps. For maximization, we provide efficiently computable expressions for many practically interesting submodular functions. This is a useful step towards transferring optimal theoretical results to real-world applications. An interesting question remains whether there exist improved sampling schemes for cases where the multilinear extension is too complex. Also recently, [22] investigated forms of submodular minimization and maximization, with submodular constraints. The proposed algorithms there were all discrete, and it will be interesting if our framework could extend to their setting as well.

Acknowledgments: We thank Bethany Herwaldt, Karthik Narayanan, Kai Wei and the rest of the submodular group at UW for discussions. This material is based upon work supported by the National Science Foundation under Grant No. (IIS-1162606), and is also supported by a Google, a Microsoft, and an Intel research award. SJ's work is supported by the Office of Naval Research under contract/grant number N00014-11-1-0688, and gifts from Amazon Web Services, Google, SAP, Blue Goji, Cisco, Clearstory Data, Cloudera, Ericsson, Facebook, General Electric, Hortonworks, Intel, Microsoft, NetApp, Oracle, Samsung, Splunk, VMware and Yahoo!.

References

[1] F. Bach. *Learning with Submodular functions: A convex Optimization Perspective*, volume 6 of *Foundations and Trends in Machine Learning*. 2013.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[3] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight (1/2) linear-time approximation to unconstrained submodular maximization. *In FOCS*, 2012.

[4] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. *In SODA*, 2014.

[5] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[6] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *STOC*, 2011.

[7] E. Chlamtac and M. Tulsiani. Convex relaxations and integrality gaps. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 139–169. Springer, 2012.

[8] A. Das, A. Dasgupta, and R. Kumar. Selecting diverse features via spectral regularization. In *NIPS*, 2012.

[9] A. Delong, O. Veksler, A. Osokin, and Y. Boykov. Minimizing sparse high-order energies by submodular vertex-cover. In *NIPS*, 2012.

[10] J. Edmonds. Submodular functions, matroids and certain polyhedra. *Combinatorial structures and their Applications*, 1970.

[11] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *KDD*, 2009.

[12] U. Feige, V. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM J. COMPUT.*, 40(4): 1133–1155, 2007.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 1956.

[14] S. Friedland and S. Gaubert. Submodular spectral functions of principal submatrices of a hermitian matrix, extensions and applications. *Linear Algebra and its Applications*, 2011.

[15] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[16] J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal MAP inference for determinantal point processes. In *NIPS*, 2012.

[17] G. Goel, C. Karande, P. Tripathi, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *FOCS*, 2009.

[18] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544, 2009.

[19] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *CVPR*, 2009.

[20] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *In FOCS*, pages 671–680. IEEE, 2009.

[21] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *In UAI*, 2012.

[22] R. Iyer and J. Bilmes. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *NIPS*, 2013.

[23] R. Iyer and J. Bilmes. The Lovász-Bregman Divergence and connections to rank aggregation, clustering and web ranking. In *UAI*, 2013.

[24] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions . In *NIPS*, 2013.

[25] R. Iyer, S. Jegelka, and J. Bilmes. Fast Semidifferential based Submodular function optimization. In *ICML*, 2013.

[26] R. Iyer, S. Jegelka, and J. Bilmes. Fast Algorithms for Submodular Optimization based on Continuous Relaxations and Rounding: Extended Version, 2014.

[27] S. Jegelka and J. A. Bilmes. Approximation bounds for inference using cooperative cuts. In *ICML*, 2011.

[28] S. Jegelka and J. A. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

[29] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, 2003.

[30] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 26(2):147–159, 2004.

[31] C. Koufogiannakis and N. Young. Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. *Algorithmica*, 2013.

[32] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

[33] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, 2009.

[34] H. Lin. *Submodularity in Natural Language Processing: Algorithms and Applications*. PhD thesis, University of Washington, Dept. of EE, 2012.

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

[36] H. Lin and J. A. Bilmes. Optimal selection of limited vocabulary speech corpora. In *Interspeech*, Florence, Italy, 2011.

[37] L. Lovász. Submodular functions and convexity. *Mathematical Programming*, 1983.

[38] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[39] K. Nagano and Y. Kawahara. Structured convex optimization under submodular constraints. In *Proc. UAI*, 2013.

[40] M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*, 2005.

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

[42] C. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.

[43] S. Rajagopalan and V. Vazirani. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525–540, 1998.

[44] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Verlag, 2003.

[45] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

[46] P. Stobbe and A. Krause. Learning fourier sparse set functions. In *AISTATS*, 2012.

[47] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706, 2008.

[48] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proc. CVPR*, 2008.

[49] J. Vondrák. *Submodularity in combinatorial optimization*. PhD thesis, Charles University, 2007.

[50] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74. ACM, 2008.

[51] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.