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

Rishabh Iyer
Dept. of Electrical Engineering
University of Washington
Seattle, WA-98195, 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-98195, USA

Abstract

It is becoming increasingly evident that many machine learning problems may be reduced to some form of 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. The results in this paper complement, in a sense explained in the paper, related discrete gradient based methods [30], and are particularly useful given the ever increasing need for efficient submodular optimization methods in very large-scale machine learning. 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 many 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* [43, 18] 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, clustering [47], and MAP inference, e.g. for image segmentation [36]. 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 function subject to cut constraints [32]. Similarly, [12] efficiently solves MAP inference in a sparse higher-order graphical model through submodular vertex cover, and [56] 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 [42] can be modeled as cardinality constrained submodular minimization. In operations research, a number of power assignment and transportation problems have been modeled as submodular minimization over spanning trees [59] or paths [2]. Similarly, constrained submodular maximization is a fitting model for problems such as optimal sensing [38], marketing [35], document summarization [41], and speech data subset selection [40].

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 [21, 31, 55, 48, 15, 6, 5], but later made *general* and unified [30, 22, 29]. In the case of submodular minimization, the discrete approaches have been based on approximating the submodular function by tractable approximations [30, 22], while in the case of submodular maximization, they have been based on greedy and local search techniques [30, 48, 15, 6, 5]. 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 [25] and cuts [31], and in the case of maximization, the techniques though general have yet to show significant practical impact due to their prohibitive computational costs [9, 7]. Hence discrete algorithms are typically used in applications (e.g., [40]).

Constraints or Function	Operation (& speed)	Algorithm Approach	
		Combinatorial	Relaxation
Specific	Min (fast)	[21, 31]	[25, 31]
	Min (slow)	[55]	Unnecessary
	Max (fast)	[48, 15, 6, 5]	**This paper**
	Max (slow)	Unnecessary	[6, 7]
General	Min (fast)	[30]	**This paper**
	Min (slow)	[22]	Unnecessary
	Max (fast)	[30]	Open
	Max (slow)	Unnecessary	[9]

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 [30]. For example, where [30] may have an approximation bound of k, our results imply a bound of $n-k+1$, where $n = |V|$, so considering both [30] 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 (i.e., characteristic vectors), 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, and finally round the obtained fractional solution to a discrete one. 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 [57], we instead use the *multilinear extension* (see Eqn. (2)) that behaves like a concave function in certain directions [9, 7]. 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. 14. We show that many important constraints, including matroid, cardinality, covers, cuts, paths, matchings, etc. can be expressed as Eq. 14. As a corollary to our main result (Theorem 1), we obtain known results (like covers [25] and cuts [31]), and also novel ones (for spanning trees, cardinality constraints, paths, matchings etc.). We also show lower 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).

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

2 CONTINUOUS RELAXATIONS

Convex relaxation. The Lovász extension [43] 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 ($\sigma(1)$ is the largest element), 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:

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

The Lovász extension is unique (despite possibly non-unique orderings if y has duplicate entries), and convex if and only if f is submodular. An alternative, related view on the Lovász extension is via the submodular polyhedron $\mathcal{P}_f = \{x \in \mathbb{R}^n : x(S) = \sum_{j \in S} x(j) \leq f(X)\}$. The Lovász extension can be expressed as $\breve{f}(y) = \max_{x \in \mathcal{P}_f} \langle y, x \rangle$ for $y \in [0,1]^n$.

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' greedy algorithm [13]:

$$h^f_{\sigma_y}(\sigma_y(j)) = f(\Sigma_j^y) - f(\Sigma_{j-1}^y), \quad \forall j \in \{1, 2, \cdots, n\}$$

The Lovász extension has also found applications in defining norms for structured sparsity [3] and divergences for rank aggregation [28].

It is instructive to consider an alternative representation of the Lovász extension. Let $\emptyset = Y_0 \subset Y_1 \subset Y_2 \subset \cdots \subset Y_k$ denote the unique chain corresponding to the point y, such that $y = \sum_{j=1}^{k} \lambda_j 1_{Y_j}$. Note that in general $k \leq n$ with equality only if y is totally ordered. Then the Lovász extension can also be expressed as [18]: $\breve{f}(y) = \sum_{j=1}^{k} \lambda_j f(Y_j)$.

Multilinear and Concave relaxations. For maximization problems, the relaxation of choice has frequently been the multilinear extension [15]

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

where f is any set function. Since Eqn. (2) has an exponential number of terms, its evaluation is in general computationally expensive, or requires approximation. The multilinear extension has particularly nice properties when the set function f is submodular. In particular, $\frac{\partial \tilde{f}}{\partial x_i} \geq 0$ iff f is monotone and $\frac{\partial^2 \tilde{f}}{\partial x_i \partial x_j} \leq 0$ iff f is submodular. This implies that for a non-decreasing set function, $\tilde{f}$ is increasing along any positive direction, and for a submodular function, $\tilde{f}$ is concave along any non-negative direction. These properties are key in providing efficient optimization algorithms and rounding schemes.

The multilinear extension may be seen as $\tilde{f}(x) = \sum_{X \subseteq V} p_x(X) f(X)$, where $p_x = \prod_{i \in X} x_i \prod_{i \notin X}(1 - x_i)$ is the product distribution over x. Alternatively, instead of taking a particular distribution, define a (different) continuous extension as the supremum over all valid distributions,

$$\hat{f}(x) = \max\Big\{ \sum_{X \subseteq V} p(X) f(X),\ p \in \Delta_x \Big\} \qquad (3)$$

where $\Delta_x = \{p \in [0,1]^{2^V} : \sum_X p_X = 1, \forall j \in V, \sum_{X : j \in X} p_X = x_j\}$. The resulting function $\hat{f}(x)$ is concave and a valid continuous extension, and hence a concave extension of f [57]. Unfortunately, this extension is NP-hard to evaluate, making the multilinear extension the preferred candidate.

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

$$\begin{aligned} \nabla_j \tilde{f}(x) &= \partial \tilde{f} / \partial x_j \\ &= \tilde{f}(x \vee e_j) - \tilde{f}(x \vee e_j - e_j). \end{aligned} \qquad (4)$$

where $e_j = 1_{\{j\}}$, and $\{x \vee y\}(i) = \max(x(i), y(i))$. A second gradient is

$$\nabla_j^a \tilde{f}(x) = \tilde{f}(x \vee e_j) - \tilde{f}(x). \qquad (5)$$

The two gradients are related component-wise as $\nabla_j \tilde{f}(x) = (1 - x_j) \nabla_j^a \tilde{f}(x)$, and both can be computed in $O(n)$ evaluations of $\tilde{f}$.

In terms of complexity, the multilinear extension (and its gradient) still has an exponential number of terms in the sum. One possibility is to approximate this sum via sampling. However, the number of samples needed for (theoretically) sufficient accuracy is polynomial but in many cases still prohibitively large for practical applications [7] (We discuss this further in Section 4). Below, we show that some practically useful submodular functions have alternative, low-complexity formulations of the multilinear extension that circumvent sampling entirely.

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:

$$\mathcal{I}_{\mathcal{C}}^{\mathcal{S}} \triangleq \sup_{f \in \mathcal{S}} \frac{\min_{X \in \mathcal{C}} f(X)}{\min_{x \in \mathcal{P}_{\mathcal{C}}} \breve{f}(x)} \geq 1. \tag{6}$$

For maximization problems, we would take the supremum over the inverse ratio. In both cases, $\mathcal{I}_{\mathcal{C}}^{\mathcal{S}}$ is defined only for non-negative functions. We may also consider the integrality gap $\mathcal{I}_{\mathcal{C}}^{\mathcal{L}}$, computed over the class $\mathcal{L}$ of all modular 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 [10].

One way to see the relationship between the integrality gap and the approximation factor obtained by rounding is as follows. Let ROPT denote the optimal relaxed value, while DOPT denotes the optimal discrete solution. The integrality gap measures the gap between DOPT and ROPT, i.e $\mathcal{I} =$ DOPT/ROPT. Let RSOL denote the rounded solution obtained from the relaxed optimum. The way one obtains bounds in a rounding scheme is by bounding the gap between RSOL and ROPT, which naturally is an upper bound on the approximation factor (which is the gap between DOPT and RSOL. However, notice that the gap between RSOL and ROPT is lower bounded by the integrality gap. Hence the integrality gap captures the tightness of the rounding scheme, and bounds on the integrality gap show bounds on the hardness.

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 [23, 52, 3, 18]. In this case, $\mathcal{P}_{\mathcal{C}} = [0,1]^n$, and importantly, the approximation factor and integrality gap are both 1.

Lemma 1. *[18] 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 $x^* \in \mathrm{argmin}_{x \in [0,1]^n} \breve{f}(x)$, the discrete minimizers are exactly the maximal chain of sets $\emptyset \subset X_{\theta_1} \subset \dots X_{\theta_k}$ obtained by θ-rounding x^*, for $\theta_j \in (0,1)$.*

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 $\theta \in (0,1)$ [3]. For special cases, such as submodular functions derived from concave functions, smoothing techniques yield a convergence rate of $O(1/t)$ [53].

It can often be faster to instead solve the unconstrained regularized problem $\min_{x \in \mathbb{R}^n} \breve{f}(x) + \frac{1}{2}\|x\|_2^2$. A motivation for this approach is that the problem, $\min_{x \in [0,1]^n} \breve{f}(x)$, can be seen as a specific form of barrier function $\min_{x \in \mathbb{R}^n} \breve{f}(x) + \delta_{[0,1]^n}(x)$ [52]. The level sets of the optimal solution to the regularized problem are the solutions of the entire regularization path of $\min_{X \subseteq V} f(X) + \theta|X|$ [3], and therefore a simple rounding at 0 gives the optimal solution. The dual problem $\min_{y \in \mathcal{B}_f} \|x\|_2^2$ is amenable to the Frank-Wolfe algorithm (or conditional gradient) [16]—with a convergence rate of $O(1/\sqrt{t})$, or an improved fully corrective version known as the minimum norm point algorithm [19]. The complexity of the improved method is still open. Moreover, regularizers other than the ℓ_2 norm are possible [52]. For decomposable functions, reflection methods can also be very effective [34].

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 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 [13].

A straightforward relaxation of $\mathcal{C}$ is the convex hull $\mathcal{P}_{\mathcal{C}} = \mathrm{conv}(1_X, X \in \mathcal{C})$ 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 [51]). 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 is 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 $\mathcal{P}_\mathcal{C}$. The following proposition formalizes this equivalence.

Proposition 1. *For any family $\mathcal{C}$ of feasible sets, the relaxed hull $\widehat{\mathcal{P}}$ of $\mathcal{C}$ is the convex hull of $\widehat{\mathcal{C}}$: $\hat{\mathcal{P}}_\mathcal{C} = \mathcal{P}_{\widehat{\mathcal{C}}} = conv(1_X, X \in \widehat{\mathcal{C}})$. For any up-monotone constraint $\mathcal{C}$, the relaxation is tight: $\hat{\mathcal{P}}_\mathcal{C} = \mathcal{P}_\mathcal{C}$.*

Although this Proposition seems intuitive, we prove it here for completeness.

Proof. Let $\mathcal{P}_{\hat{\mathcal{C}}} = \text{conv-hull}(1_X, X \in \hat{\mathcal{C}})$. We need to show that $\mathcal{P}_{\hat{\mathcal{C}}} = \hat{\mathcal{P}}_\mathcal{C}$.

First, we observe that the characteristic vector 1_X for every set $X \in \hat{\mathcal{C}}$ lies in $\hat{\mathcal{P}}_\mathcal{C}$. This follows because, by definition, for every set $X \in \widehat{\mathcal{C}}$, there exists a set $Z \subseteq V$ such that $X \backslash Z \in \mathcal{C}$. Since $1_Z \in \mathcal{P}_\mathcal{C}$ and $1_{X\backslash Z} \in \mathbb{R}^n_+$, we conclude that $1_X = 1_Z + 1_{X\backslash Z} \in \hat{\mathcal{P}}_\mathcal{C}$, and therefore, $\mathcal{P}_{\hat{\mathcal{C}}} \subseteq \hat{\mathcal{P}}_\mathcal{C}$.

We next show $\mathcal{P}_{\hat{\mathcal{C}}} \supseteq \hat{\mathcal{P}}_\mathcal{C}$ by investigating the polytope $\hat{\mathcal{P}}_\mathcal{C}$. Since it is an intersection of $\mathcal{P}_\mathcal{C} + \mathbb{R}^n_+$ (which is an integral polyhedron) and $[0,1]^n$, it follows from [51] (Theorem 5.19), that $\hat{\mathcal{P}}_\mathcal{C}$ is also integral. Let 1_Z be any extreme point of $\hat{\mathcal{P}}_\mathcal{C}$. We will show that $Z \in \hat{\mathcal{C}}$, and this implies $\mathcal{P}_{\hat{\mathcal{C}}} \supseteq \hat{\mathcal{P}}_\mathcal{C}$. Since $1_Z \in \hat{\mathcal{P}}_\mathcal{C}$, there exists a vector $x \in \mathcal{P}_\mathcal{C}$ and $y \geq 0$ such that $x = 1_Z - y$. This implies that $x = \sum_{i=1}^K \lambda_i 1_{X_i}, X_i \in \mathcal{C}$. Since $y \geq 0$, it must hold that $X_i \subseteq Z$ for all $1 \leq i \leq K$. As Z contains at least one feasible set X_i, $Z \in \hat{\mathcal{C}}$, proving the result.

The second statement of the result follows from the first, because an up-monotone constraint satisfies $\mathcal{C} = \hat{\mathcal{C}}$. □

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 first assume that the submodular function f is monotone nondecreasing (which often holds in applications), and later relax this assumption for a large class of constraints.

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 methods [16] apply. The projection onto matroid polyhedra can also be cast as a form of unconstrained submodular function minimization and is hence polynomial time solvable [18].

To apply splitting methods such as the alternating directions method of multipliers (ADMM) [4], we write the problem as $\min_{x,y:x=y} \breve{f}(x) + I(y \in \hat{\mathcal{P}}_\mathcal{C})$. One iteration of ADMM requires (1) computing the proximal operator of f, (2) projecting onto $\mathcal{P}_\mathcal{C}$, and (3) doing a simple dual update step. 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 [45, 53, 34]. An approximate proximal operator for generalized graph cuts [33] can be obtained via parametric max-flows in $O(n^2)$ [45, 8].

Algorithm 1 The constrained θ-rounding scheme

```
Input: Continuous vector $\hat{x}$, Constraints $\mathcal{C}$
Output: Discrete set $\hat{X}$
1: Obtain the chain of sets $\emptyset \subset X_1 \subset X_2 \subset X_k$ corresponding to $\hat{x}$.
2: for $j = 1, 2, \cdots, k$ do
3:   if $\exists \hat{X} \subseteq X_j : \hat{X} \in \mathcal{C}$ then
4:     Return $\hat{X}$
5:   end if
6: end for
```

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 $\hat{x}$ or n checks in the unsorted vector $\hat{x}$. 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 $\hat{X}_\theta = \{i \mid \hat{x}_i \geq \theta\} \in \hat{\mathcal{C}}$,*

$$f(\hat{X}_\theta) \leq \frac{1}{\theta} \breve{f}(\hat{x}). \tag{7}$$

If, moreover, $\breve{f}(\hat{x}) \leq \beta \min_{x \in \hat{\mathcal{P}}_\mathcal{C}} \breve{f}(x)$, then it holds that $f(\hat{X}_\theta) \leq \frac{\beta}{\theta} \min_{X \in \mathcal{C}} f(X)$.

Proof. The proof follows from the positive homogeneity of $\breve{f}$ and the monotonicity of f and $\breve{f}$:

$$\begin{aligned} \theta f(\hat{X}_\theta) &= \theta \breve{f}(1_{X_\theta}) && (8)\\ &= \breve{f}(\theta 1_{X_\theta}) && (9)\\ &\leq \breve{f}(\hat{x}) && (10)\\ &\leq \beta \min_{x \in \hat{\mathcal{P}}_\mathcal{C}} \breve{f}(x) && (11)\\ &\leq \beta \min_{X \in \hat{\mathcal{C}}} f(X) && (12)\\ &\leq \beta \min_{X \in \mathcal{C}} f(X) && (13) \end{aligned}$$

The second equality follows from the positive homogeneity of $\breve{f}$ and the third one follows from the monotonicity of f. Inequality (11) follows from the approximation bound of $\hat{x}$ with respect to $\min_{x\in\hat{\mathcal{P}}_{\mathcal{C}}} \breve{f}(x)$, and (12) uses the observation that the optimizer of the continuous problem is smaller than the discrete one. Finally (13) follows from (12) since it is optimizing over a smaller set. $\square$

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 $\hat{\mathcal{P}}_{\mathcal{C}}$ can be described as

$$\hat{\mathcal{P}}_{\mathcal{C}} = \Big\{x \in [0,1]^n \;\Big|\; \sum_{i\in W} x_i \geq b_W \text{ for all } W \in \mathcal{W}\Big\}. \tag{14}$$

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 with a bounded approximation factor. This is the case for s-t paths and cuts, covering problems, and spanning trees. This means we are addressing the following class of optimization problems:

$$\begin{aligned} \min_x \quad & \breve{f}(x) \\ \text{subject to} \quad & x \in [0,1]^n, \; \sum_{i\in W} x_i \geq b_W, \; \forall W \in \mathcal{W} \end{aligned} \tag{15}$$

The following main result states approximation bounds and integrality gaps for the class of problems described by Equation (14).

Theorem 1. *The θ-rounding scheme for constraints $\mathcal{C}$ whose relaxed polytope $\hat{\mathcal{P}}_{\mathcal{C}}$ can be described by Equation (14) achieves a worst case approximation bound of $\max_{W\in\mathcal{W}} |W| - b_W + 1$. If we assume that the sets in $\mathcal{W}$ are disjoint, the integrality gap for these constraints matches the approximation: $\mathcal{I}^S_{\mathcal{C}} = \max_{W\in\mathcal{W}} |W| - b_W + 1$.*

The proof of this result is in Appendix A. Note that the integrality gap matches the approximation factor, thus showing the tightness of the rounding strategy for a large class of constraints. In particular, for the class of constraints we consider below, we can provide instances where the integrality gap matches the approximation factors.

A result similar to Theorem 1 was shown in [37] for a different, greedy algorithmic technique. While their result also holds for a large class of constraints, for the constraints in Equation (14) 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.

Non-monotone submodular functions. A simple trick extends the methods above directly to non-monotone submodular functions over up-monotone constraints. The monotone extension of f is defined as $f^m(X) = \min_{Y:Y\supseteq X} f(Y)$.

Lemma 3. *If f is submodular, then f^m is monotone submodular and computable in polynomial time. If $\mathcal{C}$ is up-monotone, then*

$$\min_{X\in\mathcal{C}} f(X) = \min_{X\in\mathcal{C}} f^m(X). \tag{16}$$

The solution for f can, moreover, be recovered: given an approximate minimizer $\widehat{X}$ of f^m over $\mathcal{C}$, the set $Z \in \operatorname{argmin}_{Y:Y\supseteq\hat{X}} f(X)$ is an approximate minimizer of f with the same approximation bounds for f over $\mathcal{C}$ as $\widehat{X}$ has for f^m.

Proof. It is well known that, for any submodular f, the function f^m is monotone submodular [18]. To show the equivalence (16), let X^{m*} be the exact minimizer of f^m over $\mathcal{C}$. The definition of f^m implies that there exists a set $X \supseteq X^{m*}$ such that $f^m(X^{m*}) = f(X)$ and moreover, since $\mathcal{C}$ is up-monotone, $X \in \mathcal{C}$. Hence $\min_{X\in\mathcal{C}} f(X) \leq f(X) = \min_{X\in\mathcal{C}} f^m(X)$. Conversely, let X^* be the minimizer of f under $\mathcal{C}$. Then $f^m(X^*) = \min_{X\supseteq X^*} f(X) \leq f(X^*)$, and therefore $\min_{Y\in\mathcal{C}} f^m(Y) \leq f^m(X^*) = \min_{X\in\mathcal{C}} f(X)$.

To show the second part, let X^m be an approximate optimizer of f^m with an approximation factor α, i.e.,

$f^m(X^m) \leq \alpha f^m(X^{m*})$. From the first part, it follows that both f and f^m have the same optimal value in $\mathcal{C}$. The definition of f^m implies that there exists a set $Y^m \supseteq X^m$ such that $f(Y^m) = f^m(X^m)$. Hence $Y^m \in \operatorname{argmin}_{Y \supseteq X^m} f(Y)$ satisfies that $f(Y^m) = f^m(X^m) \leq \alpha f^m(X^{m*}) = \alpha \min_{X \in \mathcal{C}} f(X)$. □

While f^m can be obtained directly from f via submodular function minimization, it can be quite costly for general submodular functions. Fortunately however, many useful subclasses of submodular functions admit much faster algorithms. For example, those functions expressible as generalized graph cuts [33] can be minimized via max flows. By using f^m instead of f, any algorithm for constrained minimization of monotone submodular functions straightforwardly generalizes to the non-monotone case. The pruning step above does not apply since it could lead to a higher objective value, so we instead utilize that $\mathcal{C}$ is up-monotone and finish off with a final unconstrained minimization problem. This result holds for the relaxation and rounding discussed above, as well as the algorithms in [30, 22]. Examples of up-monotone constraints are matroid spans (Sec. 3.2.1) and covers (Sec. 3.2.2).

Down-monotone constraints. For down-monotone constraints, the up-monotone closure $\hat{C}$ will be the entire power set 2^V. Here, a different construction is needed. Define a monotone extension $f^d(X) = \min_{Y: Y \subseteq V \setminus X} f(Y)$. Also, define $\mathcal{C}' = \{X : V \setminus X \in \mathcal{C}\}$. It is easy to see that $\mathcal{C}'$ is up-monotone. Notice that if we assume f to be normalized and nonnegative, then the empty set will always be a great solution for down-monotone constraints, and this extension may not make sense. However in general the submodular function may not necessarily be normalized, and all we need for this is that the function f^d is normalized and non-negative. In other words, this implies that the function f itself be non-negative and $\min_{X \subseteq V} f(X) = 0$.

Lemma 4. *Given a submodular function f, the function f^d is monotone non-decreasing submodular, and can be evaluated in polynomial time. It holds that*

$$\min_{X \in \mathcal{C}} f(X) = \min_{X \in \mathcal{C}'} f^d(X). \tag{17}$$

Moreover, minimizing f^d over $\mathcal{C}'$ is quivalent to minimizing f over $\mathcal{C}$ in terms of the approximation factor.

Proof. The proof of this result is very similar to the previous Lemma. To show submodularity, note that the function $g(Z) = \min_{Y: Y \subseteq Z} f(Y)$ as a function of Z is submodular [18]. Then, $f^d(X) = g(V \setminus X)$ is also submodular. We also observe that f^d is monotone.

If the constraints $\mathcal{C}$ are down-monotone, then $\mathcal{C}'$ is up-monotone. Let X^{d*} be the exact minimizer of f^d over $\mathcal{C}'$. This implies that $V \setminus X^{d*} \in \mathcal{C}$. By the definition of f^d, it implies that $\exists X \subseteq V \setminus X^{d*}$ such that $f^d(X^{d*}) = f(X)$. Moreover, since $\mathcal{C}$ is down monotone, $X \in \mathcal{C}$ (since $V \setminus X^{d*} \in \mathcal{C}$). Hence $\min_{X \in \mathcal{C}} f(X) \leq f(X) = \min_{X \in \mathcal{C}'} f^d(X)$.

Conversely, let X^* be the minimizer of f under $\mathcal{C}$. Then $f^d(V \setminus X^*) = \min_{X \subseteq X^*} f(X) \leq f(X^*)$. Hence $\min_{X \in \mathcal{C}'} f^d(X) \leq f^m(X^*) = \min_{X \in \mathcal{C}} f(X)$. The approximation factor follows similarly. □

To demonstrate the utility of Theorem 1, we apply it to a variety of problems. 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 (both are equivalent since f is monotone), 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 [51] 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 $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^n, x(S) \geq r_\mathcal{M}(V) - r_\mathcal{M}(V \setminus S), \forall S \subseteq V\}$ [51]. This is clearly in the form of Eqn. 14. Although this polytope is described via an exponential number of inequalities, a linear program can still be solved efficiently over it [13]. Furthermore, projecting onto this polytope is also easy, since it corresponds to submodular function minimization [18].

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 $n - k + 1$.*

Proof. This result follows directly as a Corollary from Theorem 1, by observing that the approximation factor in this case is $\max_{S \subseteq V} |S| - r_\mathcal{M}(V) + r_\mathcal{M}(V \setminus S) - 1$. Then notice that $r_{\mathcal{M}^*}(S) = |S| - r_\mathcal{M}(V) + r_\mathcal{M}(V \setminus S)$, where $\mathcal{M}^*$ is the dual matroid of $\mathcal{M}$ [51]. Correspondingly the approximation factor can be expressed as $\max_{S \subseteq V} r_{\mathcal{M}^*}(S) - 1 = r_{\mathcal{M}^*}(V) - 1 = n - k + 1$. To show the integrality gap, we use Lemma 13. Consider the simple uniform matroid, with $\mathcal{W} = \{V\}$. A straightforward application of Lemma 13 then reveals the integrality gap. □

[2]These results were shown in [21, 25, 55]

	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	$\lvert VC \rvert \leq n$	$\lvert EC \rvert \leq n$	$C_{max} \leq m$	$P_{max} \leq n$	$\lvert M \rvert \leq n$
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 [30] (SG), the Ellipsoidal Approximation (EA) algorithm of [22], hardness [21, 25, 55], and the integrality gaps of the corresponding constrained submodular minimization problems. Note the complementarity between CR and SG. See text for further details.

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 [30, 22]. The semi-gradient framework of [30] guarantees a bound of k, while more complex (and less practical) sampling based methods [55] and approximations [22] yield factors of $O(\sqrt{n})$. The factor k of [30] is the best for small k, while our continuous relaxation works well when k is large. Moreover, for a monotone function under matroid span constraints, the pruning step will always find a set in the base, and hence a matroid span constraint is, for all intents and purposes, identical to a base constraint.

These results also extend to non-monotone cost functions, with approximation bounds of $n-k+1$ (Cor. 1) and k (using [30]) for matroid span constraints. We can also handle matroid independence constraints. Note that

$$\min_{X \in \text{Indep}(\mathcal{M})} f(X) = \min_{X \in \text{Span}(\mathcal{M}^*)} f'(X), \qquad (18)$$

where $f'(X) = f(V \backslash X)$ is a submodular function and Span($\mathcal{M}$) and Indep($\mathcal{M}$) refer to the Span and Independence sets of a Matroid $\mathcal{M}$, and $\mathcal{M}^*$ is the dual matroid of $\mathcal{M}$ [51]. Recall that the rank function of the dual Matroid satisfies $r_{\mathcal{M}^*}(V) = n - r_{\mathcal{M}}(V)$. Hence, the approximation factors for the matroid independence constraints are $k+1$ for our relaxation based framework and $n-k$ for the discrete framework of [30]. Again, we see how our results complement those of [30]. Moreover, the algorithm of [22] achieves an approximation factor of $O(\sqrt{n})$, in both cases.

Cardinality Constraints. This is a special class of 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: $\mathcal{P}_\mathcal{C} = \{x \in [0,1]^n : \sum_i x_i = k\}$. The projection onto this polyhedron can be easily performed using bisection [1]. Furthermore, the rounding step in this context is very intuitive. It corresponds to choosing the elements with the k largest entries in the continuous solution $\hat{x}$.

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. The bound of Corollary 1 becomes $|\mathcal{E}|-|\mathcal{V}|+1 = m-n+1$. The discrete algorithms of [30, 21] 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. (14) 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* $\gamma = \max_{u \in \mathcal{U}} |\{i : u \in S_i\}|$.

Proof. The proof of this corollary follows from the expression of the set cover polytope and Theorem 1. To show the integrality gap, notice that the sets $\mathcal{W}$ here are $W_u = \{i : u \in S_i\}$. Consider an instance of the set cover problem when these sets are disjoint. A direct application of Lemma 13 then provides the integrality gap (since the sets in $\mathcal{W}$ are disjoint). □

The approximation factor in Corollary 2 (without the integrality gap) was first shown in [25]. 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 [50], 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.

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 immediately provides a 2-approximation algorithm for the minimum submodular vertex cover. The 2-approximation for the special case of vertex was also shown in [21, 25].

The corrsponding integrality gap is two as well. To show this gap, consider a graph such that each vertex is incident to exactly one edge. The sets $\mathcal{W}$ are the sets of vertices for each edge and, in this case, are disjoint. As a result, Theorem 1 implies that the integrality gap is exactly 2. We may even take a complete graph and use the modular function $f(X) = |X|$. This shows that the integrality gap is 2 even when the function is modular (linear). In fact, no polynomial-time algorithm can guarantee an approximation factor better than $2 - \epsilon$, for any $\epsilon > 0$ [21].

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 $|\mathcal{V}|$ (for simple graphs), but is often much smaller. The algorithm in [30] 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 [21].

3.2.3 CUTS, PATHS AND MATCHINGS

Even though Eqn. (14) 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 [51]). The extended polytope for s-t paths can be described as a *cut-cover* [51] (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 [51]. Conversely, the extended s-t cut polytope can be described as a *path-cover* [51, 31]: $\hat{\mathcal{P}}_{\mathcal{C}} = \{x \in [0,1]^{|\mathcal{E}|} \mid \sum_{e \in P} x(e) \geq 1$, 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). These match the integrality gaps for both problems.*

Proof. The approximation factors directly follow from Theorem 1. In order to show the integrality gaps, we need to construct a set $\mathcal{W}$ of s-t paths and cuts that are disjoint. It is easy to construct such graphs. For example, in the case of s-t cuts, consider a graph with m/P parallel paths between s and t, each of length P. The integrality gap in this setting is exactly P, which matches the approximation factor. Similarly, for the s-t path case, consider a graph of m/C cuts in series. In other words, construct a chain of m/C vertices. Connect each adjacent vertex with C edges. We have m/C disjoint cuts each of size C. The integrality gap and approximation factor both are C in this setting. □

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. For example, the convex relaxation corresponding to the extended cut polytope can be described as a convex optimization problem subject to $m+1$ linear inequality constraints [49, 31]. In particular, the relaxed optimization problem can be expressed as:

$$\begin{aligned} &\min_x \breve{f}(x) \\ \text{subject to } & x \in [0,1]^{|\mathcal{E}|}, \pi \in [0,1]^{|\mathcal{V}|} \\ & \pi(v) - \pi(u) + x(e) \geq 0, \forall e = (u,v) \in \mathcal{E} \\ & \pi(s) - \pi(t) \geq 1 \end{aligned} \tag{19}$$

In the above, the variables π are additional variables that intuitively represent which vertices are reachable from s. Similarly, the extended path polytope is equivalent to the set of s-t flows with $x \leq 1$ [51, §13.2a]. This polytope can be described via $n+1$ linear inequalities.

$$\begin{aligned} &\min_x \breve{f}(x) \\ \text{subject to } & x \in [0,1]^{|\mathcal{E}|} \\ & \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = 0, \forall v \in \mathcal{V}, v \neq s,t \\ & \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e = 1, \\ & \sum_{e \in \delta^+(t)} x_e - \sum_{e \in \delta^-(t)} x_e = -1 \end{aligned} \tag{20}$$

where $\delta^+(v)$ represents the set of edges entering the vertex v and $\delta^-(v)$ represents the set of edges leaving the vertex v.

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 [30].

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 $\mathcal{P}_{\mathcal{C}} = \{x \in [0,1]^{|\mathcal{E}|} \mid \sum_{e \in \delta(v)} x(e) = 1 \text{ for all } v \in \mathcal{V}\}$, 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 [21, 29].

4 SUBMODULAR MAXIMIZATION

To relax submodular maximization, we use the multilinear extension and the concave extension. We first show that this extension can be efficiently computed for a large subclass of submodular functions. 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, 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. 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:

$$f(X) = \sum_i \max\{w_i(A) | A \subseteq X, A \in \mathcal{I}_i\}, \tag{21}$$

for linear weights $w_i(j)$. These functions form a rich class of coverage functions for summarization tasks [40]. Interestingly, the concave extension of this class of functions is efficiently computable [7, 57]. Moreover, for a number of matroids, the multilinear extension can also be efficiently computed. In particular, consider the weighted uniform matroid rank function, $f(X) = \sum_i \max\{w_i(A) | A \subseteq X, |A| \leq k\}$. The multilinear extension takes a nice form:

Lemma 5. *The multilinear extension corresponding to the weighted uniform matroid rank function, $f(X) = \max\{w(A)|A \subseteq X, |A| \leq k\}$, for a weight vector w can be expressed as (without loss of generality, assume that the vector w is ordered as $w_1 \geq w_2 \cdots \geq w_n$),*

$$\tilde{f}(x) = \sum_{i \in V} w_i x_i \sum_{l=1}^{\min(i,k)} P(x_1, \cdots, x_{i-1}, l-1) \tag{22}$$

where

$$P(x_1, \cdots, x_i, l) = \sum_{Z \subseteq S^i, |Z| = l} \prod_{s \in Z} x_s \prod_{t \in S^i \backslash Z} (1 - x_t).$$

and $S^i = \{1, 2, \cdots, i\}$.

Proof. Recall that the multilinear extension of f is

$$\tilde{f}(x) = \sum_{X \subseteq V} f(X) \prod_{s \in X} x_s \prod_{t \in V \backslash X} (1 - x_t), \tag{23}$$

where $f(X) = \max\{w(A)|A \subseteq X, |A| \leq k\}$. We can rewrite this sum in terms of the weights. Any $i \in V$ is only counted if it is among the k elements in X that have largest weight. Formally, let $\mathcal{L}_l^i = \{X : i \text{ occurs as the } l^{\text{th}} \text{ largest element in } X\}$. Then

$$\tilde{f}(x) = \sum_{i \in V} w_i \sum_{l=1}^{\min(i,k)} \sum_{X \in \mathcal{L}_l^i} \prod_{s \in X} x_s \prod_{t \in V \backslash X} (1 - x_t). \tag{24}$$

This sum has a nice form. We can break the sets $X \in \mathcal{L}_l^i$ into $Y \cup Z$, where Y is a subset of $\{1, 2, \cdots, i-1\}$ and Z is a subset of $V \backslash \{1, 2, \cdots, i-1\}$. With this, we may rewrite

$$\begin{aligned} &\sum_{X \in \mathcal{L}_l^i} \prod_{s \in X} x_s \prod_{t \in V \backslash X} (1 - x_t) \\ &= \sum_{\substack{Y \subseteq S^{i-1}, \\ |Y| = l-1, \\ Z \subseteq V \backslash S^{i-1}}} \prod_{s \in Y} x_s \prod_{t \in S^{i-1} \backslash Y} (1 - x_t) \prod_{u \in Z} x_u \prod_{v \in S_i' \backslash Z} (1 - x_v), \end{aligned}$$

where we wrote $S_i' = V \backslash S^{i-1}$. Note that $\sum_{Z \subseteq S_i'} \prod_{u \in Z} x_u \prod_{v \in S_i' \backslash Z} (1 - x_v) = 1$ and hence,

$$\sum_{X \in \mathcal{L}_l^i} \prod_{s \in X} x_s \prod_{t \in V \backslash X} (1 - x_t) = \sum_{\substack{Z \subseteq S^i, \\ |Z| = l-1}} \prod_{s \in Z} x_s \prod_{t \in S^i \backslash Z} (1 - x_t).$$

□

Interestingly, $P(x_1, \cdots, x_i, l)$ admits a nice recursive relationship, and can be computed efficiently, and therefore also the multilinear extension of the weighted matroid rank function.

Lemma 6. *$P(x_1, \cdots, x_i, l)$ admits the following relationship,*

$$\begin{aligned} P(x_1, \cdots, x_i, l) = &\, x_i P(x_1, \cdots, x_{i-1}, l-1) \\ &+ (1 - x_i) P(x_1, \cdots, x_i, l-1). \end{aligned}$$

Moreover, for every $i \in V$ and $l \in [1,n]$, the matrix of values of $P(x_1, \cdots, x_i, l)$, and hence the multilinear extension of $\tilde{f}$ can be computed in $O(n^2)$ time. Moreover, the gradient $\nabla^a \tilde{f}(x)$ can be computed in $O(n^3)$ time.

Proof. The proof of this follows directly from the fact that we divide the possible sets into those containing i and those not containing i. Given this recursion, it is easy to see that the entire matrix of values of $P(x_1, \cdots, x_i, l)$ can be obtained for all values of i and l in $O(n^2)$ iterations. Moreover, given this matrix, we can obtain the expression of $\tilde{f}$ also in $O(n^2)$. □

The above results immediately provide an expression of the multilinear extension of the Facility Location function.

Corollary 5. *The multilinear extension corresponding of the Facility Location function $f(X) = \sum_{i \in V} \max_{j \in X} s_{ij}$, for a similarity matrix s_{ij}, can be expressed as*

$$\tilde{f}(x) = \sum_{i \in V} \sum_{l=1}^{n} s_{ij_i^l} x_{j_i^l} \prod_{m=1}^{l-1} (1 - x_{j_i^m}), \qquad (25)$$

where $j_i^1, j_i^2, \cdots, j_i^n$ denote the elements closest to $i \in V$, sorted in decreasing order by their similarity s_{ij}. The fnction $\tilde{f}(x)$ can be computed in $O(n^2 \log n)$ time, and likewise the alternate gradient.

Proof. The expression for the multilinear extension follows immediately from Lemma 5 and 6. The extension can be computed by, for each $i \in V$, sorting the elements by s_{ij}, computing the products in one linear pass for each l, and then the sum over l in another pass. Repeating this for each $i \in V$ results in a running time of $O(n(n + n \log n))$.

We next consider the gradient $\nabla^a \tilde{f}(x)$. For simplicity, we focus on the max function, i.e $f(X) = \max_{i \in X} s_i$. Assume w.l.o.g. that $s_1 \geq s_2 \geq \cdots \geq s_n$. The gradient of this function is

$$\nabla_j^a \tilde{f}(x) = s_j \prod_{l=1}^{j-1} (1 - x_l) - \sum_{i=j+1}^{n} s_i x_i \prod_{l=1}^{i-1} (1 - x_l).$$

This follows since $\nabla_j^a \tilde{f}(x) = \tilde{f}(x | x_j = 1) - \tilde{f}(x)$, where $\tilde{f}(x | x_j = 1)$ is the expression of $\tilde{f}(x)$ with x_j set to be 1. The multilinear extension for the max-function is $\tilde{f}(x) = \sum_{i=1}^{n} s_i x_i \prod_{l=1}^{i-1} (1 - x_l)$, and hence,

$$\tilde{f}(x | x_j = 1) = \sum_{i=1}^{j-1} s_i x_i \prod_{l=1}^{i-1} (1 - x_l) + s_j \prod_{l=1}^{j-1} (1 - x_l).$$

Plugging both into the expression of the gradient, we get the above.

Then for every j, we precompute $M(x, j) = \prod_{l=1}^{j-1} (1 - x_l)$ and store it (this can be done in $O(n)$, and $O(n \log n)$ for sorting). Then

$$\nabla_j^a \tilde{f}(x) = s_j M(x, j) - \sum_{i=j+1}^{n} s_i x_i M(x, i). \qquad (26)$$

Hence the entire alternate gradient can be computed in $O(n \log n)$ time for the max function, and correspondingly in $O(n^2 \log n)$ for facility location. □

Lemma 5 provides an expression for the partial alternate gradient (since it is useful for maximizing monotone functions). For the complete gradient $\nabla^a \tilde{f}(x)$, there is a similar expression. The complete gradient is required for non-monotone submodular maximization, when the facility location function is used together with a non-monotone submodular function.

We can rewrite the gradients for some other matroids too, using similar techniques as above. For example, consider partition matroids. Let $B_1, B_2, \cdots, B_m$ be m blocks of a partition of the ground set such that $B_i \cap B_j = \emptyset, \forall i, j$ and $\cup_i B_i = V$. Also let $d_1, \cdots, d_m$ be numbers such that $d_i \leq |B_i|, \forall i$. A set X is independent in a partition matroid if and only if $|X \cap B_i| \leq d_i, \forall i$. The weighted rank function corresponding to the partition matroid is,

$$\begin{aligned} f(X) &= \max\{w(Y) | \forall i, Y \cap B_i \subseteq X \cap B_i, |Y \cap B_i| \leq d_i\} \\ &= \max\{\sum_{i=1}^{m} w(Y_i), Y_i \subseteq X \cap B_i, |Y_i| \leq d_i\} \\ &= \sum_{i=1}^{m} \max\{w(Y_i), Y_i \subseteq X \cap B_i, |Y_i| \leq d_i\}. \end{aligned}$$

The last equality holds since the B_i's are disjoint. Hence, the weighted matroid rank function for a partition matroid is a sum of rank functions corresponding to uniform matroids defined over the subsets B_i. Hence Lemma 23 directly provides an expression for the multilinear extension.

Set Cover function. This function is widely used in applications, capturing notions of coverage [40]. Given a collection of sets $\{\mathcal{S}_1, \cdots, \mathcal{S}_n\}$ 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:

Lemma 7. *The multilinear extension corresponding to the set cover function $f(X) = w(\Gamma(X))$ is*

$$\tilde{f}(x) = \sum_{j \in \mathcal{U}} w_j [1 - R(x, j)], \qquad (27)$$

where $R(x, j) = \prod_{i \in \Gamma^{-1}(j)} (1 - x_i)$. The multilinear extension can be computed in $O(n^2)$ time. Moreover, the gradient $\nabla_k^a \tilde{f}(x)$ is

$$\nabla_k^a \tilde{f}(x) = \sum_{j \notin \Gamma(k)} w_j R(x, j), \qquad (28)$$

and the entire vector $\nabla^a \tilde{f}(x)$ can be computed in $O(n^2)$ time.

Proof. Again, we express this sum in terms of the weights w. In particular,

$$\begin{aligned}\tilde{f}(x) &= \sum_{j\in\mathcal{U}} w_j \sum_{X:j\in\Gamma(X)} \prod_{s\in X} x_s \prod_{t\notin X}(1-x_t) \\ &= \sum_{j\in\mathcal{U}} w_j \sum_{X:X\cap\Gamma^{-1}(j)\neq\emptyset} \prod_{s\in X} x_s \prod_{t\notin X}(1-x_t) \\ &= \sum_{j\in\mathcal{U}} w_j [1 - \sum_{X:X\subseteq V\setminus\Gamma^{-1}(j)} \prod_{s\in X} x_s \prod_{t\notin X}(1-x_t)] \\ &= \sum_{j\in\mathcal{U}} w_j [1 - \prod_{t\in\Gamma^{-1}(j)} (1-x_t)]\end{aligned}$$

where the last inequality follows from the fact that

$$\sum_{X:X\subseteq V\setminus\Gamma^{-1}(j)} \prod_{s\in X} x_s \prod_{t\in V\setminus X} (1-x_t) =$$
$$\prod_{u\in\Gamma^{-1}(j)} (1-x_u) \sum_{X:X\subseteq V\setminus\Gamma^{-1}(j)} \prod_{s\in X} x_s \prod_{t\in\{V\setminus\Gamma^{-1}(j)\}\setminus X} (1-x_t)$$
$$= \prod_{t\in\Gamma^{-1}(j)} (1-x_t)$$

The second part also directly follows from the first, as

$$\begin{aligned}\nabla^a_k \tilde{f}(x) &= \tilde{f}(x|x_k=1) - \tilde{f}(x) \\ &= \sum_{j:k\in\Gamma^{-1}(j)} w_j[1-R(x,j)] + \sum_{j:k\notin\Gamma^{-1}(j)} w_j - \tilde{f}(x) \\ &= \sum_{j:k\notin\Gamma^{-1}(j)} w_j R(x,j) \\ &= \sum_{j\notin\Gamma(k)} w_j R(x,j)\end{aligned}$$

In order to compute the vector $\nabla^a \tilde{f}(x)$, first we precompute $R(x,j)$ for all j. This can be done in $O(n^2)$ time. Then using the values of $R(x,j)$, we can compute $\nabla^a_k \tilde{f}(x), \forall k$ also in $O(n^2)$. □

The gradient $\nabla \tilde{f}(x)$ can be computed analogously.

Probabilistic Coverage Functions. A probabilistic generalization of covering functions of the form $f(X) = \sum_{i\in\mathcal{U}} w_i[1-\prod_{j\in X}(1-p_{ij})]$ has been used for summarization problems [14]. If p_{ij} is binary (either i covers j or not), we obtain a standard set cover function. The multilinear extension of probabilistic coverage functions is also efficiently computable

$$\tilde{f}(x) = \sum_{i\in\mathcal{U}} w_i[1-\prod_{j\in V}(1-p_{ij}x_j)]. \tag{29}$$

This form follows from Proposition 2 (below), since this is a pseudo-Boolean function.

Graph Cut functions. Graph cuts are a widely used class of functions. Their multilinear extension also a admits closed form representation. Graph cut functions are of the form

$$f(X) = \sum_{i\in X, j\notin X} s_{ij}. \tag{30}$$

Its multilinear extension is also easily expressed:

Lemma 8. *The multilinear extension of the graph cut function and its gradient have closed form expressions*

$$\tilde{f}(x) = \sum_{i,j\in V} s_{ij}x_i(1-x_j), \quad \nabla_i \tilde{f}(x) = \sum_{j\in V} s_{ij}(1-2x_j)$$

For asymmetric cut functions $f(X) = \sum_{i\in V, j\in X} s_{ij}$, the expressions are

$$\tilde{f}(x) = \sum_{i,j\in V} s_{ij}x_j, \quad \nabla_i \tilde{f}(x) = \sum_{j\in V} s_{ij} \tag{31}$$

In both cases above, the multilinear extension and the corresponding gradient can be computed in $O(n^2)$ time.

Proof. We rewrite the multilinear extension in terms of the s_{ij} to obtain th equadratic polynomial

$$\begin{aligned}\tilde{f}(x) &= \sum_{i,j\in V} s_{ij} \sum_{X:i\in X, j\notin X} \prod_{s\in X} x_s \prod_{t\notin X}(1-x_t) && (32)\\ &= \sum_{i,j\in V} s_{ij}x_s(1-x_t) && (33)\end{aligned}$$

We can similarly derive the expression for the asymmetric graph cut function and the gradients for both expressions. □

These quadratic functions have been widely used in computer vision. 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 [41, 40].

Lemma 9. *The multilinear extension and the gradient for the function $f(X) = -\sum_{i,j\in X} s_{ij}$ are*

$$\tilde{f}(x) = -\sum_{i,j\in V} s_{ij}x_ix_j, \quad \nabla_i \tilde{f}(x) = -\sum_{j\in V} s_{ij}x_j \tag{34}$$

Proof. The lemma follows analogously to the case of graph cuts:

$$\begin{aligned}\tilde{f}(x) &= -\sum_{i,j\in V} s_{ij} \sum_{X:i\in X, j\in X} \prod_{s\in X} x_s \prod_{t\notin X}(1-x_t) && (35)\\ &= -\sum_{i,j\in V} s_{ij}x_ix_j, && (36)\end{aligned}$$

and similarly for the gradient. □

	Fac. Location [3]	Set Cover	Graph Cuts	Diversity I/ II	Concave over card.	Soft-Max [4]
Multilinear Closed form	$O(n^2 \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$
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	$O(n^2 \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$
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.

This function is often used with other coverage type functions (for example, graph cut or set cover) [40, 41], and since the multilinear extension is commutative, the above forms provide closed form expressions for a number of objective functions that promote diversity and coverage.

Sparse Pseudo-Boolean functions. For graphical models, in particular in computer vision, set functions are often written as polynomials [24]. 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. This is the case for graph cut like functions above and for the functions considered in [54, 24]. This analogy is implicitly known, but we formalize it for completeness.

Proposition 2. *The polynomial representation is the multilinear extension: $\tilde{f}(x) = p_f(x)$.*

Proof. Recall that the multilinear extension is equivalent to the expectation for a product distribution of Bernoulli random variables, $P(S) = \prod_{i \in S} x_i \prod_{j \notin S} x_j$. We see that

$$\tilde{f}(x) = \sum_{S \subseteq V} p(S) \sum_{T \subseteq S} \alpha_T \tag{37}$$

$$= \sum_{T \subseteq V} \alpha_T \sum_{S \supseteq T} p(S) \tag{38}$$

$$= \sum_{T \subseteq V} \alpha_T \sum_{S \supseteq T} p(T) p(S \setminus T \mid T) \tag{39}$$

$$= \sum_{T \subseteq V} \alpha_T \prod_{i \in T} x_i \Big(\sum_{A \subseteq V \setminus T} p(A) \Big) \tag{40}$$

$$= \sum_{T \subseteq V} \alpha_T \prod_{i \in T} x_i = p_f(x). \tag{41}$$

In the last step, we used that $p(A \mid T) = p(A)$ and that $\sum_{A \subseteq V'} P(A) = 1$ for all $V' \subseteq V$. □

Spectral functions. Diversity can also be encouraged via spectral regularizers [11]. 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 [17]. The resulting class of submodular functions includes the log determinants occurring in DPP inference [20], 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 [20]. This extension is $\tilde{f}^s(x) = \log(\sum_{X \subseteq V} \exp(f(X)) \prod_{i \in X} x_i \prod_{j \notin X}(1 - x_j)$ and shares several desirable theoretical properties with the multilinear extension[5]. A related function that encourages diversity is $f(X) = -\sum_{i=1}^{|X|}(\lambda_i(S_X) - 1)^2$ [11]. Note that $-\sum_{i=1}^{|X|} \lambda_i^2(S_X) = -\text{trace}(S_X^\top S_X) = -\sum_{i,j \in X} s_{ij}^2$. The multilinear extension of this function also takes a nice form.

Lemma 10. *The spectral function $f(X) = -\sum_{i=1}^{|X|}(\lambda_i(S_X) - 1)^2$ can also be directly expressed as*

$$f(X) = -\sum_{i,j \in X} s_{ij}^2 + 2\sum_{i \in X} s_{ii} - |X|. \tag{42}$$

The multilinear extension and its gradient are

$$\tilde{f}(x) = -\sum_{i,j \in V} s_{ij}^2 x_i x_j + \sum_{i \in V}(2 s_{ii} x_i - 1),$$

$$\nabla_i \tilde{f}(x) = -\sum_{j \in V} s_{ij}^2 x_j + 2 s_{ii}$$

Both expressions can be computed in $O(n^2)$ time.

Proof. The proof of this lemma follows from Lemma 9 and from the fact that the multi-linear extension of a modular function is its linear extension – i.e given a function $f(X) = \sum_{i \in X} s_i$, its multilinear extension is $\tilde{f}(x) = \sum_{i \in V} s_i x_i = \langle s, x \rangle$. □

We note that given the above, it is possible to approximate the multilinear extension of the log determinant function by first representing it in its spectral form $f(X) = \log\det(S_X) = \sum_{i=1}^{|X|} \log(\lambda_i(S_X))$, then taking a truncated

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

[4]This is for soft-max extension [20].

[5]The results in [20] are shown only for $\delta = 0$, i.e log-determinant functions. However, it is easy to see that the soft max extension can be computed for any value of $\delta > 0$ efficiently

Taylor series approximation of the log function to two terms, which allows it to be represented in polynomial form where Lemma 9 applies, allowing an $O(n^2)$-cost approximation, something that might be more useful than certain sampling approximations to the multilinear extension.

Concave over modular functions. Finally, we consider functions of the form $f(A) = g(|A|)$ where g is a concave function. Such functions are submodular, and have simple extensions.

Lemma 11. *The multilinear extension of the concave over cardinality function $f(A) = g(|A|)$ is*

$$\tilde{f}(x) = \sum_{i=1}^{n} g(i) P(x_1, \cdots, x_n, i), \tag{43}$$

where

$$P(x_1, \cdots, x_n, i) = \sum_{Z \subseteq V, |Z|=i} \prod_{s \in Z} x_s \prod_{t \notin Z} (1 - x_t).$$

The term $P(x_1, \cdots, x_n, i)$ can be computed in linear time (excluding the computation of constants), and correspondingly $\tilde{f}(x)$ in $O(n^2)$.

Proof. The proof of this Lemma follows directly from definition of the multilinear extension, and from Lemmas 5 and 6. □

One may generalize this to sums $f(X) = \sum_i g_i(m_i(X))$ of concave over modular functions, where the g_i are concave and the m_i are modular. This class of functions has a natural concave extension: $\tilde{f}(x) = \sum_i g_i(\langle x, m_i \rangle)$.

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 [58], which is a slight modification of the Frank-Wolfe algorithm [16], with a fixed step size. The algorithm proceeds as follows.

- Find $h^t = \text{argmax}_{h' \in \mathcal{P}_\mathcal{C}} \langle h', \nabla^a \tilde{f}(x^t) \rangle$.
- $x^{t+1} = x^t + \delta h^t$, with the step size $\delta = 1/n^2$.

Here ∇^a is 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 $\tilde{f}(x) \geq (1 - 1/e)\tilde{f}(x^*)$ [7, 57]. Moreover, using the pipage rounding technique (in particular, the deterministic variant [58]) ensures that we can round the continuous solution to a set in $O(n^2)$ function calls.

A naïve computation of the generic multilinear extension in Eqn. (2) or its gradient takes exponential time. To compute these in polynomial time, we can use sampling. For obtaining an accuracy better than $1/n^2$, we need $O(n^5)$ samples for the multilinear extension or for each coordinate of its gradient [58, 57]. 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 $O(n^9)$.

The specialized expressions in Section 4 however 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 practically very useful submodular functions. Moreover, we can 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 algorithm [44], 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 [39], 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 [16] 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 [9]. 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 [20].

A generic rounding strategy for submodular maximization problems was given by [9], 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 [6] 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 [9]. For DPPs, [20] 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 be used 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 [7, 9]). The concave extension however, can have integrality gaps arbitrarily close to $e/(e-1)$ even for simple matroids [57]. Hence, even though it is possible to exactly optimize it in certain cases (for example, for weighted matroid rank functions), the rounding only guarantees a $1-1/e$ approximation factor.

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 [46, 26]. 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 problem is a DC programming problem, and can be addressed (often very efficiently) using the convex-concave procedure [60]. Moreover, thanks to the special structure of the Lovász extension, there exists a simple rounding scheme for the unconstrained version.

Lemma 12. *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.*

Proof. Recall that given a point $z \in [0,1]^n$, we can find a chain of sets $\emptyset = Z_0 \subset Z_1 \subset Z_2 \subset \cdots \subset Z_k$ corresponding to z, such that $z = \sum_{j=1}^{k} \lambda_j 1_{Z_j}$. Then the Lovász extension can be written as $\breve{f}(z) = \sum_{j=1}^{k} \lambda_j f(Z_j)$. This chain is independent of the function f and hence, given functions f and g, we have that

$$\tilde{h}(z) = \sum_{j=1}^{k} \lambda_j h(Z_j) \tag{44}$$

It is then easy to see that one of $h(Z_j)$ for $j = 1, 2, \cdots, k$ must have a value less than or equal to $\tilde{h}(z)$, thus completing the proof. ☐

The above lemma shows that in at most $O(n)$, we can round the continuous solution without any loss. Unfortunately, these results do not seem to extend straightforwardly to combinatorial constraints. Although the relaxed difference of convex optimization problem can itself be solved via the convex-concave procedure if the polytope $\mathcal{P}_\mathcal{C}$ corresponding to the constraints can be characterized efficiently, the θ-rounding procedure no longer retains any guarantees. However, a procedure like threshold rounding might still provide a feasible solution if the constraints are up-monotone, and taking the best amongst the feasible rounded sets might still work well in practice.

6 DISCUSSION

In this work, we have offered a unifying view on 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 summarize 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 complex. Recently, [27] investigated forms of submodular minimization and maximization with submodular constraints. The proposed algorithms there were all discrete. It is an interesting question whether the relaxations discussed here 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] W. Y. Adams, H. Su, and L. Fei-Fei. Efficient euclidean projections onto the intersection of norm balls. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 433–440, 2012.

[2] I. Averbakh and O. Berman. Categorized bottleneck-minisum path problems on networks. *Operations Research Letters*, 16:291–297, 1994.

[3] F. Bach. Learning with Submodular functions: A convex Optimization Perspective (updated version). *Arxiv*, 2013.

[4] 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.

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

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

[7] 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.

[8] A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *Int. Journal of Computer Vision*, 84(3):288–307, 2009.

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

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

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

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

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

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

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

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

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

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

[19] S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7:3–17, 2011.

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

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

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

[23] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

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

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

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

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

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

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

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

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

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

[33] S. Jegelka, H. Lin, and J. Bilmes. On fast approximate submodular minimization. *In NIPS*, 2011.

[34] S. Jegelka, F. Bach, and S. Sra. Reflections for user-friendly submodular minimization. In *NIPS*, 2013.

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

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

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

[38] 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.

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

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

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

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

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

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

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

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

[47] M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. *NIPS*, 18:979, 2006.

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

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

[50] 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.

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

[52] P. Stobbe. *Convex Analysis for Minimizing and Learning Submodular Set Functions*. PhD thesis, California Institute of Technology, 2013.

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

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

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

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

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

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

[59] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcasting in static ad hoc wireless networks. *Wireless Networks*, 8:607–617, 2002.

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

A Proof of Theorem 1

Proof. To show the first part of the theorem, we invoke Lemma 2. The constraints of Equation (14) demand that for every set $W \in \mathcal{W}$, at least $b_W \leq |W|$ elements need to be chosen, or "covered". Consequently, to round a vector $x \in \hat{\mathcal{P}}_{\mathcal{C}}$, it is sufficient to choose $\theta = \min_{W \in \mathcal{W}} x_{[b_W, W]}$ as the rounding threshold, where $x_{[k,A]}$ denotes the k^{th} largest entry of x in a set A. The worst case scenario is that the $b_W - 1$ entries of x with indices in the set W are all 1, and the remaining mass of 1 is equally distributed over the remaining elements in W. In this case, the value of $x_{[b_W,W]}$ is $1/(|W| - b_W + 1)$. Since the constraint requires $\sum_{i \in W} x_i \geq b_W$, it must hold that $x_{[b_W,W]} \geq 1/(|W| - b_W + 1)$. The approximation factor then follows with Lemma 2.

To analyze the integrality gap, we first show the following Lemma.

Lemma 13. *If $\hat{\mathcal{P}}_{\mathcal{C}}$ is described by Equation 14, then the integrality gap can be as large as $\mathcal{I}^S_{\mathcal{C}} \geq \max_{W \in \mathcal{W}} |W| - b_W + 1$. Moreover, if $\mathcal{C}$ and θ are such that θ-rounding provides a valid set in $\widehat{\mathcal{C}}$, the integrality gap is never larger than that: $\mathcal{I}^S_{\mathcal{C}} \leq \frac{\beta}{\theta}$.*

To show the lower bound on the gap, we construct a simple example. Assume $W' = \operatorname{argmax}_{W \in \mathcal{W}} |W| - b_W + 1$ satisfies $W \cap W' = \emptyset, \forall W \in \mathcal{W}, W \neq W'$. This is true since the $\mathcal{W}$ consists of disjoint sets. Let B' be a subset of W' such that $|B'| = |W'| - b_{W'} + 1$. In other words, every feasible solution must intersect B' Now we define $f(X) = \min\{|X \cap B'|, 1\}$. The Lovász extension of this function is $\breve{f}(x) = \max_{j \in X \cap B'} x_j$. An optimal continuous solution for $\breve{f}$ and $\mathcal{W}$ has entries $x^*_j = 1$ for $j \notin B'$ and $x^*_j = 1/(|W'| - b_{W'} + 1)$ for $j \in B'$. In this case, the integrality gap is $f(X^*)/\breve{f}(x^*) = 1/(W'| - b_{W'} + 1)^{-1} = W'| - b_{W'} + 1$.

The upper bound on the gap follows from the approximation factor:

$$\begin{aligned} I^S_{\mathcal{C}} &= \max_f \frac{f(X^*)}{\min_{x \in \mathcal{P}_{\mathcal{C}}} \breve{f}(x)} \\ &\leq \max_f \frac{f(\hat{X}_\theta)}{\min_{x \in \mathcal{P}_{\mathcal{C}}} \breve{f}(x)} \\ &= \frac{\beta}{\theta} \end{aligned}$$

where the second inequality follows from the fact that $f(X^*) \leq \hat{X}_\theta$ and the last one from Lemma 2. □