Stochastic Submodular Maximization: The Case of Coverage Functions

Mohammad Reza Karimi
Department of Computer Science
ETH Zurich
mkarimi@ethz.ch

Mario Lucic
Department of Computer Science
ETH Zurich
lucic@inf.ethz.ch

Hamed Hassani
Department of Electrical and Systems Engineering
University of Pennsylvania
hassani@seas.upenn.edu

Andreas Krause
Department of Computer Science
ETH Zurich
krausea@ethz.ch

Abstract

Stochastic optimization of *continuous* objectives is at the heart of modern machine learning. However, many important problems are of *discrete* nature and often involve *submodular* objectives. We seek to unleash the power of stochastic continuous optimization, namely stochastic gradient descent and its variants, to such discrete problems. We first introduce the problem of *stochastic submodular optimization*, where one needs to optimize a submodular objective which is given as an expectation. Our model captures situations where the discrete objective arises as an empirical risk (e.g., in the case of exemplar-based clustering), or is given as an explicit stochastic model (e.g., in the case of influence maximization in social networks). By exploiting that common extensions *act linearly* on the class of submodular functions, we employ projected stochastic gradient ascent and its variants in the continuous domain, and perform rounding to obtain discrete solutions. We focus on the rich and widely used family of weighted coverage functions. We show that our approach yields solutions that are guaranteed to match the optimal approximation guarantees, while reducing the computational cost by several orders of magnitude, as we demonstrate empirically.

1 Introduction

Submodular functions are discrete analogs of convex functions. They arise naturally in many areas, such as the study of graphs, matroids, covering problems, and facility location problems. These functions are extensively studied in operations research and combinatorial optimization [22]. Recently, submodular functions have proven to be key concepts in other areas such as machine learning, algorithmic game theory, and social sciences. As such, they have been applied to a host of important problems such as modeling valuation functions in combinatorial auctions, feature and variable selection [23], data summarization [27], and influence maximization [20].

Classical results in submodular optimization consider the *oracle model* whereby the access to the optimization objective is provided through a black box — an oracle. However, in many applications, the objective has to be estimated from data and is subject to stochastic fluctuations. In other cases the value of the objective may only be obtained through simulation. As such, the exact computation might not be feasible due to statistical or computational constraints. As a concrete example, consider the problem of *influence maximization* in social networks [20]. The objective function is defined as the expectation of a stochastic process, quantifying the size of the (random) subset of nodes

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

influenced from a selected seed set. This expectation cannot be computed efficiently, and is typically approximated via random sampling, which introduces an error in the estimate of the value of a seed set. Another practical example is the *exemplar-based clustering* problem, which is an instance of the *facility location* problem. Here, the objective is the sum of similarities of all the points inside a (large) collection of data points to a selected set of centers. Given a distribution over point locations, the true objective is defined as the expected value w.r.t. this distribution, and can only be approximated as a sample average. Moreover, evaluating the function on a sample involves computation of many pairwise similarities, which is computationally prohibitive in the context of massive data sets.

In this work, we provide a formalization of such *stochastic submodular maximization tasks*. More precisely, we consider set functions $f : 2^V \to \mathbb{R}_+$, defined as $f(S) = \mathbb{E}_{\gamma\sim\Gamma}[f_\gamma(S)]$ for $S \subseteq V$, where Γ is an arbitrary distribution and for each realization $\gamma \sim \Gamma$, the set function $f_\gamma : 2^V \to \mathbb{R}_+$ is monotone and submodular (hence f is monotone submodular). The goal is to maximize f subject to some constraints (e.g. the k-cardinality constraint) having access only to i.i.d. samples $f_{\gamma\sim\Gamma}(\cdot)$.

Methods for submodular maximization fall into two major categories: (i) The classic approach is to directly optimize the objective using discrete optimization methods (e.g. the GREEDY algorithm and its accelerated variants), which are state-of-the-art algorithms (both in practice and theory), at least in the case of simple constraints, and are most widely considered in the literature; (ii) The alternative is to lift the problem into a continuous domain and exploit continuous optimization techniques available therein [7]. While the continuous approaches may lead to provably good results, even for more complex constraints, their high computational complexity inhibits their practicality.

In this paper we demonstrate how modern stochastic optimization techniques (such as SGD, ADAGRAD [8] and ADAM [21]), can be used to solve an important class of discrete optimization problems which can be modeled using weighted coverage functions. In particular, we show how to efficiently maximize them under matroid constraints by (i) lifting the problem into the continuous domain using the *multilinear extension* [37], (ii) efficiently computing a concave relaxation of the multilinear extension [32], (iii) efficiently computing an unbiased estimate of the gradient for the concave relaxation thus enabling (projected) stochastic gradient ascent-style algorithms to maximize the concave relaxation, and (iv) rounding the resulting fractional solution without loss of approximation quality [7]. In addition to providing convergence and approximation guarantees, we demonstrate that our algorithms enjoy strong empirical performance, often achieving an order of magnitude speedup with less than 1% error with respect to GREEDY. As a result, the presented approach unleashes the powerful toolkit of stochastic gradient based approaches to discrete optimization problems.

Our contributions. In this paper we (i) introduce a framework for *stochastic submodular optimization*, (ii) provide a general methodology for constrained maximization of stochastic submodular objectives, (iii) prove that the proposed approach guarantees a $(1 - 1/e)-$approximation in expectation for the class of weighted coverage functions, which is the best approximation guarantee achievable in polynomial time unless $\mathrm{P} = \mathrm{NP}$, (iv) highlight the practical benefit and efficiency of using continuous-based stochastic optimization techniques for submodular maximization, (v) demonstrate the practical utility of the proposed framework in an extensive experimental evaluation. We show for the first time that continuous optimization is a highly practical, scalable avenue for maximizing submodular set functions.

2 Background and problem formulation

Let V be a ground set of n elements. A set function $f : 2^V \longrightarrow \mathbb{R}_+$ is *submodular* if for every $A, B \subseteq V$, it holds $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$. Function f is said to be monotone if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq V$. We focus on maximizing f subject to some constraints on $S \subseteq V$. The prototypical example is maximization under the cardinality constraint, i.e., for a given integer k, find $S \subseteq V$, $|S| \leq k$, which maximizes f. Finding an exact solution for monotone submodular functions is NP-hard [10], but a $(1 - 1/e)$-approximation can be efficiently determined [30]. Going beyond the $(1 - 1/e)$-approximation is NP-hard for many classes of submodular functions [30, 24]. More generally, one may consider *matroid constraints*, whereby $(V, \mathcal{I})$ is a matroid with the family of independent sets $\mathcal{I}$, and maximize f such that $S \in \mathcal{I}$. The GREEDY algorithm achieves a $1/2$-approximation [13], but CONTINUOUS GREEDY introduced by Vondrák [37], Calinescu et al. [6] can achieve a $(1 - 1/e)$-optimal solution in expectation. Their approach is based on the *multilinear*

extension of f, $F : [0,1]^V \to \mathbb{R}_+$, defined as

$$F(\mathbf{x}) = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j), \tag{1}$$

for all $\mathbf{x} = (x_1, \cdots, x_n) \in [0,1]^V$. In other words, $F(\mathbf{x})$ is the expected value of of f over sets wherein each element i is included with probability x_i independently. Then, instead of optimizing $f(S)$ over $\mathcal{I}$, we can optimize F over the matroid base polytope corresponding to $(V, \mathcal{I})$: $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}_+^n \mid \mathbf{x}(S) \leq r(S), \forall S \subseteq V, \mathbf{x}(V) = r(V)\}$, where $r(\cdot)$ is the matroid's rank function. The CONTINUOUS GREEDY algorithm then finds a solution $\mathbf{x} \in \mathcal{P}$ which provides a $(1 - 1/e)-$approximation. Finally, the continuous solution $\mathbf{x}$ is then efficiently rounded to a feasible discrete solution without loss in objective value, using PIPAGE ROUNDING [1, 6]. The idea of converting a discrete optimization problem into a continuous one was first exploited by Lovász [28] in the context of submodular minimization and this approach was recently applied to a variety of problems [36, 19, 3].

Problem formulation. The aforementioned results are based on the *oracle model*, whereby the exact value of $f(S)$ for any $S \subseteq V$ is given by an oracle. In absence of such an oracle, we face the additional challenges of *evaluating* f, both statistical and computational. In particular, consider set functions that are defined as *expectations*, i.e. for $S \subseteq V$ we have

$$f(S) = \mathbb{E}_{\gamma \sim \Gamma}[f_\gamma(S)], \tag{2}$$

where Γ is an arbitrary distribution and for each realization $\gamma \sim \Gamma$, the set function $f_\gamma : 2^V \to \mathbb{R}$ is submodular. The goal is to efficiently maximize f subject to constraints such as the k-cardinality constraint, or more generally, a matroid constraint.

As a motivating example, consider the problem of propagation of contagions through a network. The objective is to identify the most influential seed set of a given size. A propagation instance (concrete realization of a contagion) is specified by a graph $G = (V, E)$. The influence $f_G(S)$ of a set of nodes S in instance G is the fraction of nodes reachable from S using the edges E. To handle uncertainties in the concrete realization, it is natural to introduce a probabilistic model such as the Independent Cascade [20] model which defines a distribution $\mathcal{G}$ over instances $G \sim \mathcal{G}$ that share a set V of nodes. The influence of a seed set S is then the expectation $f(S) = \mathbb{E}_{G \sim \mathcal{G}}[f_G(S)]$, which is a monotone submodular function. Hence, estimating the expected influence is computationally demanding, as it requires summing over exponentially many functions f_G. Assuming f as in (2), one can easily obtain an unbiased estimate of f for a fixed set S by random sampling according to Γ. The critical question is, given that the underlying function *is* an expectation, can we optimize it more efficiently?

Our approach is based on continuous extensions that are linear operators on the class of set functions, namely, *linear continuous extensions*. As a specific example, considering the multilinear extension, we can write $F(\mathbf{x}) = \mathbb{E}_{\gamma \sim \Gamma}[F_\gamma(\mathbf{x})]$, where F_γ denotes the extension of f_γ. As a consequence, the value of $F_\gamma(\mathbf{x})$, when $\gamma \sim \Gamma$, is an *unbiased estimator* for $F(\mathbf{x})$ and unbiased estimates of the (sub)gradients may be obtained analogously. We explore this avenue to develop efficient algorithms for maximizing an important subclass of submodular functions that can be expressed as weighted coverage functions. Our approach harnesses a *concave relaxation* detailed in Section 3.

Further related work. The emergence of new applications, combined with a massive increase in the amount of data has created a demand for fast algorithms for submodular optimization. A variety of approximation algorithms have been presented, ranging from submodular maximization subject to a cardinality constraint [29, 39, 4], submodular maximization subject to a matroid constraint [6], non-monotone submodular maximization [11], approximately submodular functions [17], and algorithms for submodular maximization subject to a wide variety of constraints [25, 12, 38, 18, 9]. A closely related setting to ours is online submodular maximization [35], where functions come one at a time and the goal is to provide time-dependent solutions (sets) such that a cumulative regret is minimized. In contrast, our goal is to find a single (time-independent) set that maximizes the objective (2). Another relevant setting is noisy submodular maximization, where the evaluations returned by the oracle are noisy [16, 34]. Specifically, [34] assumes a noisy but unbiased oracle (with an independent sub-Gaussian noise) which allows one to sufficiently estimate the marginal gains of items by averaging. In the context of cardinality constraints, some of these ideas can be carried to our setting by introducing additional assumptions on how the values $f_\gamma(S)$ vary w.r.t. to their expectation $f(S)$. However, we provide a different approach that does not rely on uniform convergence and compare sample and running time complexity comparison with variants of GREEDY in Section 3.

3 Stochastic Submodular Optimization

We follow the general framework of [37] whereby the problem is lifted into the continuous domain, a continuous optimization algorithm is designed to maximize the transferred objective, and the resulting solution is rounded. Maximizing f subject to a matroid constraint can then be done by first maximizing its multilinear extension F over the matroid base polytope and then rounding the solution. Methods such as the projected stochastic gradient ascent can be used to maximize F over this polytope.

Critically, we have to assure that the computed local optima are *good* in expectation. Unfortunately, the multilinear extension F lacks concavity and therefore may have bad local optima. Hence, we consider *concave* continuous extensions of F that are *efficiently computable*, and at most a constant factor away from F to ensure solution quality. As a result, such a concave extension $\bar{F}$ could then be efficiently maximized over a polytope using *projected stochastic gradient ascent* which would enable the application of modern continuous optimization techniques. One class of important functions for which such an extension can be efficiently computed is the class of weighted coverage functions.

The class of weighted coverage functions (WCF). Let U be a set and let g be a nonnegative modular function on U, i.e. $g(S) = \sum_{u \in S} w(u)$, $S \subseteq U$. Let $V = \{B_1, \ldots, B_n\}$ be a collection of subsets of U. The *weighted coverage function* $f : 2^V \longrightarrow \mathbb{R}^+$ defined as

$$\forall S \subseteq V : \; f(S) = g\left(\textstyle\bigcup_{B_i \in S} B_i\right)$$

is monotone submodular. For all $u \in U$, let us denote by $P_u := \{B_i \in V \mid u \in B_i\}$ and by $\mathbb{I}(\cdot)$ the indicator function. The multilinear extension of f can be expressed in a more compact way:

$$\begin{aligned} F(\mathbf{x}) &= \mathbb{E}_S[f(S)] = \mathbb{E}_S \sum_{u \in U} \mathbb{I}(u \in B_i \text{ for some } B_i \in S) \cdot w(u) \\ &= \sum_{u \in U} w(u) \cdot \mathbb{P}(u \in B_i \text{ for some } B_i \in S) = \sum_{u \in U} w(u) \left(1 - \textstyle\prod_{B_i \in P_u} (1 - x_i)\right) \end{aligned} \tag{3}$$

where we used the fact that each element $B_i \in V$ was chosen with probability x_i.

Concave upper bound for weighted coverage functions. To efficiently compute a concave upper bound on the multilinear extension we use the framework of Seeman and Singer [32]. Given that all the weights $w(u)$, $u \in U$ in (3) are non-negative, we can construct a concave upper bound for the multilinear extension $F(\mathbf{x})$ using the following Lemma. Proofs can be found in the Appendix A.

Lemma 1. *For $\mathbf{x} \in [0,1]^\ell$ define $\alpha(\mathbf{x}) := 1 - \prod_{i=1}^{\ell}(1 - x_i)$. Then the Fenchel concave biconjugate of $\alpha(\cdot)$ is $\beta(\mathbf{x}) := \min\left\{1, \sum_{i=1}^{\ell} x_i\right\}$. Also*

$$(1 - 1/e)\, \beta(\mathbf{x}) \leq \alpha(\mathbf{x}) \leq \beta(\mathbf{x}) \quad \forall \mathbf{x} \in [0,1]^\ell.$$

Furthermore, β is an extension of α, i.e. $\forall \mathbf{x} \in \{0,1\}^\ell$: $\alpha(\mathbf{x}) = \beta(\mathbf{x})$.

Consequently, given a weighted coverage function f with $F(\mathbf{x})$ represented as in (3), we can define

$$\bar{F}(\mathbf{x}) := \sum_{u \in U} w(u) \min\left\{1, \sum_{B_v \in P_u} x_v\right\} \tag{4}$$

and conclude using Lemma 1 that $(1 - 1/e)\bar{F}(\mathbf{x}) \leq F(\mathbf{x}) \leq \bar{F}(\mathbf{x})$, as desired. Furthermore, $\bar{F}$ has three interesting properties: (1) It is a concave function over $[0,1]^V$, (2) it is equal to f on vertices of the hypercube, i.e. for $\mathbf{x} \in \{0,1\}^n$ one has $\bar{F}(\mathbf{x}) = f(\{i : x_i = 1\})$, and (3) it can be computed efficiently and deterministically given access to the sets P_u, $u \in U$. In other words, we can compute the value of $\bar{F}(\mathbf{x})$ using at most $\mathcal{O}(|U| \times |V|)$ operations. Note that $\bar{F}$ is not the *tightest* concave upper bound of F, even though we use the tightest concave upper bounds for each term of F.

Optimizing the concave upper bound by stochastic gradient ascent. Instead of maximizing F over a polytope $\mathcal{P}$, one can now attempt to maximize $\bar{F}$ over $\mathcal{P}$. Critically, this task can be done efficiently, as $\bar{F}$ is concave, by using projected stochastic gradient ascent. In particular, one can

Algorithm 1 Stochastic Submodular Maximization via concave relaxation

Require: matroid $\mathcal{M}$ with base polytope $\mathcal{P}$, η_t (step size), T (maximum # of iterations)

$\mathbf{x}^{(0)} \leftarrow$ starting point in $\mathcal{P}$
for $t \leftarrow 0$ **to** $T-1$ **do**
  Choose $\mathbf{g}_t$ at random from a distribution such that $\mathbb{E}[\mathbf{g}_t | \mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(t)}] \in \partial \bar{F}(\mathbf{x}^{(t)})$
  $\mathbf{x}^{(t+1/2)} \leftarrow \mathbf{x}^{(t)} + \eta_t \, \mathbf{g}_t$
  $\mathbf{x}^{(t+1)} \leftarrow \mathrm{Project}_{\mathcal{P}}(\mathbf{x}^{(t+1/2)})$
end for
$\bar{\mathbf{x}}_T \leftarrow \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}^{(t)}$
$S \leftarrow$ RANDOMIZED-PIPAGE-ROUND$(\bar{\mathbf{x}}_T)$
return S such that $S \in \mathcal{M}$, $\mathbb{E}[f(S)] \geq (1-1/e) f(OPT) - \varepsilon(T)$.

control the convergence speed by choosing from the toolbox of modern continuous optimization algorithms, such as SGD, ADAGRAD and ADAM. Let us denote a maximizer of $\bar{F}$ over $\mathcal{P}$ by $\bar{\mathbf{x}}^*$, and also a maximizer of F over $\mathcal{P}$ by $\mathbf{x}^*$. We can thus write

$$F(\bar{\mathbf{x}}^*) \geq (1-1/e)\bar{F}(\bar{\mathbf{x}}^*) \geq (1-1/e)\bar{F}(\mathbf{x}^*) \geq (1-1/e)F(\mathbf{x}^*),$$

which is the exact guarantee that previous methods give, and in general is the best near-optimality ratio that one can give in poly-time. Finally, to round the continuous solution we may apply RANDOMIZED-PIPAGE-ROUNDING [7] as the quality of the approximation is preserved in expectation.

Matroid constraints. Constrained optimization can be efficiently performed by projected gradient ascent whereby after each step of the stochastic ascent, we need to project the solution back onto the feasible set. For the case of matroid constraints, it is sufficient to consider projection onto the matroid base polytope. This problem of projecting on the base polytope has been widely studied and fast algorithms exist in many cases [2, 5, 31]. While these projection algorithms were used as a key subprocedure in constrained submodular minimization, here we consider them for submodular maximization. Details of a fast projection algorithm for the problems considered in this work are presented the Appendix D. Algorithm 1 summarizes all steps required to maximize f subject to matroid constraints.

Convergence rate. Since we are maximizing a concave function $\bar{F}(\cdot)$ over a matroid base polytope $\mathcal{P}$, convergence rate (and hence running time) depends on $B := \max_{\mathbf{x} \in \mathcal{P}} ||\mathbf{x}||$, as well as maximum gradient norm ρ (i.e. $||\mathbf{g}_t|| \leq \rho$ with probability 1). [1] In the case of the base polytope for a matroid of rank r, B is $\sqrt{r}$, since each vertex of the polytope has exactly r ones. Also, from (4), one can build a rough upper bound for the norm of the gradient:

$$||\mathbf{g}|| \leq ||\textstyle\sum_{u \in U} w(u) \mathbf{1}_{P_u}|| \leq \big(\max_{u \in U} |P_u|\big)^{1/2} \sum_{u \in U} w(u),$$

which depends on the weights $w(u)$ as well as $|P_u|$ and is hence problem-dependent. We will provide tighter upper bounds for gradient norm in our specific examples in the later sections. With $\eta_t = B/\rho\sqrt{t}$, and classic results for SGD [33], we have that

$$\bar{F}(\mathbf{x}^*) - \mathbb{E}[\bar{F}(\bar{\mathbf{x}}_T)] \leq B\rho/\sqrt{T},$$

where T is the total number of SGD iterations and $\bar{\mathbf{x}}_T$ is the final outcome of SGD (see Algorithm 1). Therefore, for a given $\varepsilon > 0$, after $T \geq B^2\rho^2/\varepsilon^2$ iterations, we have

$$\bar{F}(\mathbf{x}^*) - \mathbb{E}[\bar{F}(\bar{\mathbf{x}}_T)] \leq \varepsilon.$$

Summing up, we will have the following theorem:

Theorem 2. *Let f be a weighted coverage function, $\mathcal{P}$ be the base polytope of a matroid $\mathcal{M}$, and ρ and B be as above. Then for each $\epsilon > 0$, Algorithm 1 after $T = B^2\rho^2/\varepsilon^2$ iterations, produces a set $S^* \in \mathcal{M}$ such that $\mathbb{E}[f(S^*)] \geq (1-1/e) \max_{S \in \mathcal{M}} f(S) - \varepsilon$.*

[1] Note that the function $\bar{F}$ is neither smooth nor strongly concave as functions such as $\min\{1, x\}$ are not smooth or strongly concave.

Remark. Indeed this approximation ratio is the best ratio one can achieve, unless P=NP [10]. A key point to make here is that our approach also works for more general constraints (in particular is efficient for *simple* matroids such as partition matroids). In the latter case, GREEDY only gives $\frac{1}{2}$-approximation and fast discrete methods like STOCHASTIC-GREEDY [29] do not apply, whereas our method still yields an $(1 - 1/e)$-optimal solution.

Time Complexity. One can compute an upper bound for the running time of Algorithm 1 by estimating the time required to perform gradient computations, projection on $\mathcal{P}$, and rounding. For the case of uniform matroids, projection and rounding take $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$ time, respectively (see Appendix D). Furthermore, for the applications considered in this work, namely expected influence maximization and exemplar-based clustering, we provide linear time algorithms to compute the gradients. Also when our matroid is the k-uniform matroid (i.e. k-cardinality constraint), we have $B = \sqrt{k}$. By Theorem 2, the total computational complexity of our algorithm is $\mathcal{O}(\rho^2 kn(\log n)/\varepsilon^2)$.

Comparison to GREEDY. Let us relate our results to the classical approach. When running the GREEDY algorithm in the stochastic setting, one estimates $\hat{f}(S) := \frac{1}{s}\sum_{i=1}^{s} f_{\gamma_i}(S)$ where $\gamma_1, \ldots, \gamma_s$ are i.i.d. samples from Γ. The following proposition bounds the sample and computational complexity of GREEDY. The proof is detailed in the Appendix B.

Proposition 3. *Let f be a submodular function defined as* (2)*. Suppose $0 \leq f_\gamma(S) \leq H$ for all $S \subseteq V$ and all $\gamma \sim \Gamma$. Assume S^* denotes the optimal solution for f subject to k-cardinality constraint and S_k denotes the solution computed by the greedy algorithm on $\hat{f}$ after k steps. Then, in order to guarantee*

$$\mathbb{P}[f(S_k) \geq (1 - 1/e)f(S^*) - \varepsilon] \geq 1 - \delta,$$

it is enough to have

$$s \in \Omega\bigg(H^2(k \log n + \log(1/\delta))/\varepsilon^2\bigg),$$

i.i.d. samples from Γ. The running time of GREEDY *is then bounded by*

$$\mathcal{O}\bigg(\tau H^2 nk(k \log n + \log(1/\delta))/\varepsilon^2\bigg),$$

where τ is an upper bound on the computation time for a single evaluation of $f_\gamma(S)$.

As an example, let us compare the worst-case complexity bound obtained for SGD (i.e. $\mathcal{O}(\rho^2 kn(\log n)/\varepsilon^2)$) with that of GREEDY for the influence maximization problem. Each single function evaluation for GREEDY amounts to computing the total influence of a set in a sample graph, which makes $\tau = O(n)$ (here we assume our sample graphs satisfy $|E| = O(|V|)$). Also, a crude upper bound for the size of the gradient for each sample function is $H\sqrt{n}$ (see Appendix E.1). Hence, we can deduce that SGD can have a factor k speedup w.r.t. to GREEDY.

4 Applications

We will now show how to instantiate the *stochastic submodular maximization framework* using several prototypical discrete optimization problems.

Influence maximization. As discussed in Section 2, the Independent Cascade [20] model defines a distribution $\mathcal{G}$ over instances $G \sim \mathcal{G}$ that share a set V of nodes. The influence $f_G(S)$ of a set of nodes S in instance G is the fraction of nodes reachable from S using the edges $E(G)$. The following Lemma shows that the influence belongs to the class of WCF.

Lemma 4. *The influence function $f_G(\cdot)$ is a WCF. Moreover,*

$$F_G(\mathbf{x}) = \mathbb{E}_S[f_G(S)] = \frac{1}{|V|}\sum_{v \in V}(1 - \textstyle\prod_{u \in P_v}(1 - x_u)) \tag{5}$$

$$\bar{F}_G(\mathbf{x}) = \frac{1}{|V|}\sum_{v \in V}\min\{1, \textstyle\sum_{u \in P_v} x_u\}, \tag{6}$$

where P_v is the set of all nodes having a (directed) path to v.

We return to the problem of maximizing $f_{\mathcal{G}}(S) = \mathbb{E}_{G\sim\mathcal{G}}[f_G(S)]$ given a distribution over graphs $\mathcal{G}$ sharing nodes V. Since $f_{\mathcal{G}}$ is a weighted sum of submodular functions, it is submodular. Moreover,

$$F(\mathbf{x}) = \mathbb{E}_S[f_{\mathcal{G}}(S)] = \mathbb{E}_S[\mathbb{E}_G[f_G(S)]] = \mathbb{E}_G[\mathbb{E}_S[f_G(S)]] = \mathbb{E}_G[F_G(\mathbf{x})]$$
$$= \mathbb{E}_G\left[\frac{1}{|V|}\sum_{v\in V}(1-\textstyle\prod_{u\in P_v}(1-x_u))\right].$$

Let $\mathcal{U}$ be the uniform distribution over vertices. Then,

$$F(\mathbf{x}) = \mathbb{E}_G\left[\tfrac{1}{|V|}\textstyle\sum_{v\in V}(1-\prod_{u\in P_v}(1-x_u))\right] = \mathbb{E}_G\left[\mathbb{E}_{v\sim\mathcal{U}}\left[1-\textstyle\prod_{u\in P_v}(1-x_u)\right]\right], \quad (7)$$

and the corresponding upper bound would be

$$\bar{F}(\mathbf{x}) = \mathbb{E}_G\left[\mathbb{E}_{v\sim\mathcal{U}}\left[\min\{1, \textstyle\sum_{u\in P_v} x_u\}\right]\right]. \quad (8)$$

This formulation proves to be helpful in *efficient* calculation of subgradients, as one can obtain a random subgradient in linear time. For more details see Appendix E.1. We also provide a more efficient, *biased* estimator of the expectation in the Appendix.

Facility location. Let $G = (X \dot\cup Y, E)$ be a complete weighted bipartite graph with parts X and Y and nonnegative weights $w_{x,y}$. The weights can be considered as utilities or some similarity metric. We select a subset $S \subseteq X$ and each $y \in Y$ selects $s \in S$ with the highest weight $w_{s,y}$. Our goal is to maximize the average weight of these selected edges, i.e. to maximize

$$f(S) = \frac{1}{|Y|}\sum_{y\in Y}\max_{s\in S} w_{s,y} \quad (9)$$

given some constraints on S. This problem is indeed the *Facility Location* problem, if one takes X to be the set of facilities and Y to be the set of customers and $w_{x,y}$ to be the utility of facility x for customer y. Another interesting instance is the *Exemplar-based Clustering* problem, in which $X = Y$ is a set of objects and $w_{x,y}$ is the similarity (or inverted distance) between objects x and y, and one tries to find a subset S of exemplars (i.e. *centroids*) for these objects.

The stochastic nature of this problem is revealed when one writes (9) as the expectation $f(S) = \mathbb{E}_{y\sim\Gamma}[f_y(S)]$, where Γ is the uniform distribution over Y and $f_y(S) := \max_{s\in S} w_{s,y}$. One can also consider this more general case, where y's are drawn from an unknown distribution, and one tries to maximize the aforementioned expectation.

First, we claim that $f_y(\cdot)$ for each $y \in Y$ is again a weighted coverage function. For simplicity, let $X = \{1, \dots, n\}$ and set $m_i \doteq w_{i,y}$, with $m_1 \geq \cdots \geq m_n$ and $m_{n+1} \doteq 0$.

Lemma 5. *The utility function $f_y(\cdot)$ is a WCF. Moreover,*

$$F_y(\mathbf{x}) = \textstyle\sum_{i=1}^n (m_i - m_{i+1})(1 - \prod_{j=1}^i (1-x_j)), \quad (10)$$
$$\bar{F}_y(\mathbf{x}) = \textstyle\sum_{i=1}^n (m_i - m_{i+1}) \min\{1, \sum_{j=1}^i x_j\}. \quad (11)$$

We remark that the gradient of both F_y and $\bar{F}_y$ can be computed in linear time using a recursive procedure. We refer to Appendix E.2 for more details.

5 Experimental Results

We demonstrate the practical utility of the proposed framework and compare it to standard baselines. We compare the performance of the algorithms in terms of their wall-clock running time and the obtained utility. We consider the following problems:

- **Influence Maximization for the Epinions network[2].** The network consists of 75 879 nodes and 508 837 directed edges. We consider the subgraph induced by the top 10 000 nodes with the largest out-degree and use the independent cascade model [20]. The diffusion model is specified by a fixed probability for each node to influence its neighbors in the underlying graph. We set this probability p to be 0.02, and chose the number of seeds $k = 50$.

[2] `http://snap.stanford.edu/`

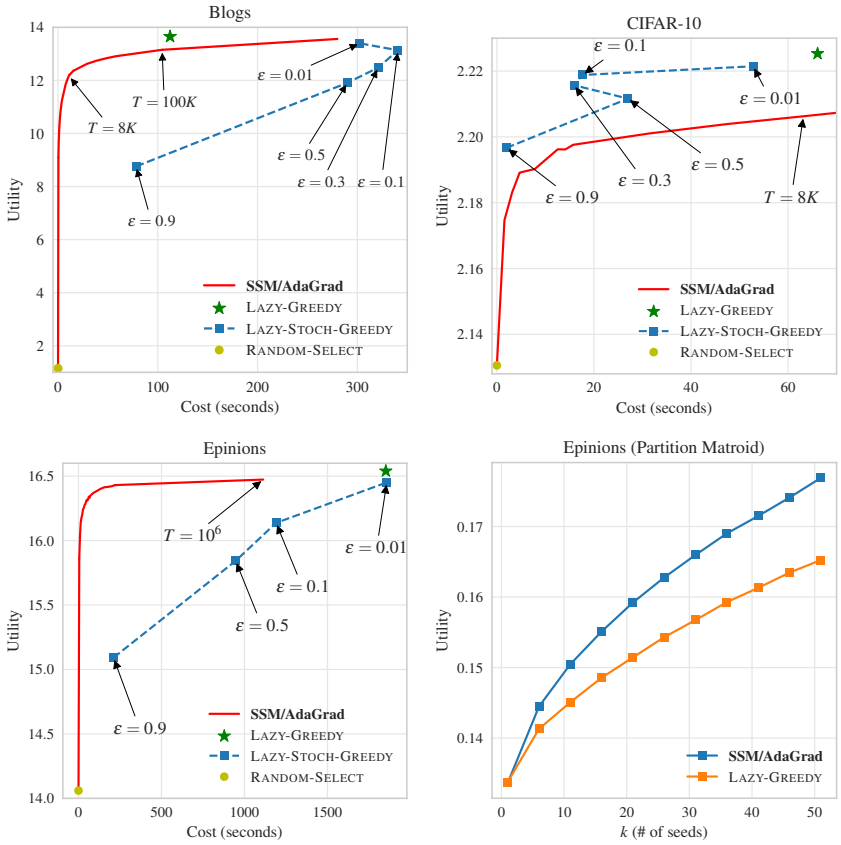

Figure 1: In the case of Facility location for Blog selection as well as on influence maximization on Epinions, the proposed approach reaches the same utility *significantly* faster. On the exemplar-based clustering of CIFAR, the proposed approach is outperformed by STOCHASTIC-GREEDY, but nevertheless reaches 98.4% of the GREEDY utility in a few seconds (after less than 1000 iterations). On Influence Maximization over *partition matroids*, the proposed approach significantly outperforms GREEDY.

- **Facility Location for Blog Selection.** We use the data set used in [14], consisting of 45 193 blogs, and 16 551 cascades. The goal is to detect information cascades/stories spreading over the blogosphere. This dataset is *heavy-tailed*, hence a small random sample of the events has high variance in terms of the cascade sizes. We set $k = 100$.

- **Exemplar-based Clustering on CIFAR-10.** The data set contains 60 000 color images with resolution 32×32. We use a single batch of 10 000 images and compare our algorithms to variants of GREEDY over the full data set. We use the Euclidean norm as the distance function and set $k = 50$. Further details about preprocessing of the data as well as formulation of the submodular function can be found in Appendix E.3.

Baselines. In the case of cardinality constraints, we compare our stochastic continuous optimization approach against the most efficient discrete approaches (LAZY-)GREEDY and (LAZY-)STOCHASTIC-GREEDY, which both provide optimal approximation guarantees. For STOCHASTIC-GREEDY, we vary the parameter ε in order to explore the running time/utility tradeoff. We also report the performance of randomly selected sets. For the two facility location problems, when applying the greedy variants we can evaluate the exact objective (true expectation). In the Influence Maximization application, computing the exact expectation is intractable. Hence, we use an empirical average of s samples (cascades) from the model. We note that the number of samples suggested by Proposition 3 is overly conservative, and instead we make a practical choice of $s = 10^3$ samples.

Results. The results are summarized in Figure 1. On the blog selection and influence maximization applications, the proposed continuous optimization approach outperforms STOCHASTIC-GREEDY in terms of the running time/utility tradeoff. In particular, for blog selection we can compute a solution with the same utility $26\times$ faster than STOCHASTIC-GREEDY with $\varepsilon = 0.5$. Similarly, for influence maximization on Epinions we the solution $88\times$ faster than STOCHASTIC-GREEDY with $\varepsilon = 0.1$. On the exemplar-based clustering application STOCHASTIC-GREEDY outperforms the proposed approach. We note that the proposed approach is still competitive as it recovers 98.4% of the value after less than thousand iterations.

We also include an experiment on Influence Maximization over *partition matroids* for the Epinions network. In this case, GREEDY only provides a $1/2$ approximation guarantee and STOCHASTIC-GREEDY does not apply. To create the partition, we first sorted all the vertices by their out-degree. Using this order on the vertices, we divided the vertices into two partitions, one containing vertices with even positions, other containing the rest. Figure 1 clearly demonstrates that the proposed approach outperforms GREEDY in terms of utility (as well as running time).

Acknowledgments The research was partially supported by ERC StG 307036. We would like to thank Yaron Singer for helpful comments and suggestions.

References

[1] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3): 307–328, 2004.

[2] Francis Bach et al. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373, 2013.

[3] Francis R. Bach. Convex analysis and optimization with submodular functions: a tutorial. *CoRR*, abs/1010.4207, 2010.

[4] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514. SIAM, 2014.

[5] P. Brucker. An o(n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3 (3):163–166, 1984.

[6] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 182–196. Springer, 2007.

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

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

[9] Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond 1/e. pages 248–257, 2016.

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

[11] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[12] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 570–579. IEEE, 2011.

[13] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978.

[14] Natalie Glance, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. Deriving marketing intelligence from online discussion. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 419–428, 2005.

[15] Ryan Gomez and Andreas Krause. Budgeted nonparametric learning from data streams. *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[16] Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. *CoRR*, abs/1601.03095, 2016.

[17] Thibaut Horel and Yaron Singer. Maximizing approximately submodular functions. *NIPS*, 2016.

[18] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

[19] Rishabh K. Iyer and Jeff A. Bilmes. Polyhedral aspects of submodularity, convexity and concavity. *Arxiv, CoRR*, abs/1506.07329, 2015.

[20] David Kempe, Jon Kleinberg, and Eva. Tardos. Maximizing the spread of influence through a social network. *9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[22] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3(19):8, 2012.

[23] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2005.

[24] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 324–331. AUAI Press, 2005.

[25] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 545–554. Society for Industrial and Applied Mathematics, 2009.

[26] K. S. Sesh Kumar and Francis Bach. Active-set methods for submodular minimization problems. *hal-01161759v3*, 2016.

[27] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[28] László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[29] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. *Association for the Advancement of Artificial Intelligence*, 2015.

[30] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - i. *Mathematical Programming*, 14(1):265–294, 1978.

[31] P. M. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1):321–328, 1990.

[32] Lior Seeman and Yaron Singer. Adaptive seeding in social networks. pages 459–468, 2013.

[33] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning : From Theory to Algorithms*. Cambridge University Press, 2014.

[34] Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *Proc. Conference on Artificial Intelligence (AAAI)*, February 2016.

[35] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. *NIPS*, 2008.

[36] Jan Vondrák. Submodularity in combinatorial optimization. *Charles University, Prague*, 2007.

[37] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008.

[38] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304, 2013.

[39] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *International Conference on Machine Learning (ICML)*, Beijing, China, 2014.

A Proof of Lemma 1

Here we prove the inequality mentioned in the lemma. Proof of the fact of being Fenchel biconjugate is in Appendix F.

We prove the left-hand-side inequality, since the right-hand-side inequality is a consequence of Fenchel biconjugate-ness.

Let $\theta := \sum_{i=1}^{\ell} x_i$. We note from the inequality $1 - x \leq \exp(-x)$ that $\prod_{i=1}^{\ell}(1 - x_i) \leq \exp(-\theta)$. We thus obtain

$$1 - \prod_{i=1}^{\ell}(1 - x_i) \geq 1 - \exp(-\theta).$$

Now, if $\theta \geq 1$ then the result is clear. Also, if $\theta < 1$, then we note that the function $(1 - \exp(-\theta))/\theta$ is decreasing for $\theta \in (0, 1)$, and hence, $1 - \exp(-\theta) \geq \theta(1 - 1/e)$. The left-hand-side inequality thus follows immediately.

B Proof of Proposition 3

Note that the total number of subsets of cardinality less than k is bounded from above by $k\binom{n}{k}$. For each such set S we want the estimate $\hat{f}(S) := \frac{1}{s}\sum_{i=1}^{s} f_{\gamma_i}(S)$ to be at most ϵ away from $f(S)$. Also, note that the function $\hat{f}$ is itself a submodular function and maximizing it would give a $(1 - 1/e)$-approximation to its optimum. Hence, it is enough to have enough samples such that for all subsets S of cardinality at most k the two values $f(S)$ and $\hat{f}(S)$ differ by at most epsilon. By using Hoeffding's inequality and a union bound over all the subsets of cardinality at most k (note that $\log(n\binom{n}{k}) = \mathcal{O}(k\log(n))$ we get the result.

C Proof of Lemmas 4 and 5

C.1 Lemma 4

Proof. Let $A := \{C_v \mid v \in V\}$, where C_v is the set of vertices reachable from v. By construction, there is a one-to-one correspondence between elements of A and V, namely $C_v \leftrightarrow v$. For $T \subseteq A$, let $S \subseteq V$ be its corresponding subset in V, i.e. $S = \{v \in V \mid v \leftrightarrow C_v, C_v \in T\}$. It's obvious that $\bigcup_{v\in S} C_v = \bigcup_{C_v \in T} C_v$. Setting $g(T) = \frac{|T|}{|V|}$, makes $f'_G(T) := g(\bigcup_{C_v\in T} C_v)$ a WCF. But $f'_G(T) = f_G(S)$, so $f_G(\cdot)$ is also a WCF.

Moreover, for each $v \in V$, the set P_v is the set of all elements of A that contain v, which are precisely those vertices from which there is a (directed) path to v. We also relax our notation, and replace any element of A by its correspondent in V. Hence,

$$F_G(\mathbf{x}) = \mathbb{E}_S[f_G(S)] = \frac{1}{|V|}\sum_{v\in V}(1 - \textstyle\prod_{u\in P_v}(1 - x_u))$$

$$\bar{F}_G(\mathbf{x}) = \frac{1}{|V|}\sum_{v\in V}\min\{1, \textstyle\sum_{u\in P_v} x_u\},$$

which are poly-time computable since one can find P_v with a simple BFS algorithm in $\mathcal{O}(|V| + |E|)$ for each $v \in V$.

□

C.2 Lemma 5

Proof. Write $f(\cdot)$ instead of $f_y(\cdot)$.

Let $V = \{C_i \mid 1 \leq i \leq n\}$, where $C_i = \{i, \ldots, n\}$, and let $w(i) = m_i - m_{i+1}$ (set $m_{n+1} = 0$). Note that there is a natural bijection between V and U, namely $C_i \leftrightarrow i$. Let g be the modular function with weights $w(i)$, defined on 2^U, and define the WCF $f' : 2^V \to \mathbb{R}_+$ as

$$f'(S) := g(\textstyle\bigcup_{i\in S} C_i) = \sum_{j\in\bigcup_{i\in S} C_i} w(j). \tag{12}$$

Since C_i's are forming a decreasing chain, $\bigcup_{i\in S} C_i = C_{\min S}$ and (12) becomes

$$f'(S) = \sum_{j\in C_{\min S}} w(j) = \sum_{j=\min S}^{n} w(j) = m_{\min S} - m_{n+1} = \max_{i\in S} m_i,$$

which is exactly $f(S)$.

Furthermore, P_i is simply the set $\{1,\dots,i\}$. Hence, we can write the multilinear extension and the corresponding upper bound as

$$F_y(\mathbf{x}) = \textstyle\sum_{i=1}^{n}(m_i - m_{i+1})(1 - \prod_{j=1}^{i}(1-x_j)),$$
$$\bar{F}_y(\mathbf{x}) = \textstyle\sum_{i=1}^{n}(m_i - m_{i+1})\min\{1, \sum_{j=1}^{i} x_j\}.$$

□

D Fast Algorithms for Projection and Rounding

In this section, we show how projection (w.r.t. Mahalanobis norm) can be done in time $O(n\log n)$ and rounding in time $O(n)$ for the uniform matroid. This projection algorithm also proves to be useful in case of partition matroid polytope. We also discuss a projection method on general matroid base polytopes, based on the method of Kumar and Bach [26], which needs to solve a total number of n submodular function minimization (SFM) tasks (details below).

D.1 Efficient projection on the uniform matroid

Let G be a diagonal matrix with positive entries, $G = \operatorname{diag}(g_1,\dots,g_n)$. Our aim is to project a vector $\mathbf{y}\in\mathbb{R}^n_+$ on the uniform matroid base polytope defined as

$$P_k = \{\mathbf{x}\in\mathbb{R}^n_+ \mid \textstyle\sum x_i = k,\ 0\le x_i\le 1\}.$$

The polytope P_k is the convex hull of all the vectors that have precisely k ones and $n-k$ zeros. Projecting $\mathbf{y}$ onto P_k entails finding a point $\mathbf{x}$ in P_k, such that

$$\mathbf{x} = \operatorname*{argmin}_{\mathbf{x}\in P_k} \|\mathbf{x}-\mathbf{y}\|_G^2 := \operatorname*{argmin}_{\mathbf{x}\in P_k} (\mathbf{x}-\mathbf{y})^\top G\,(\mathbf{x}-\mathbf{y}),$$

where $\|\cdot\|_G$ is the Mahalanobis norm (i.e. the Mahalanobis distance to $\mathbf{0}$). Note that in the special case of $G = I$, this problem boils down to orthogonal projection of $\mathbf{y}$ onto P_k. We first transform this problem into an orthogonal projection, and solve that projection in $O(n\log n)$.

$$\begin{aligned}
\mathbf{x} &= \operatorname*{argmin}_{\mathbf{x}\in P_k} (\mathbf{x}-\mathbf{y})^\top G\,(\mathbf{x}-\mathbf{y}) \\
&= \operatorname*{argmin}_{\mathbf{x}\in P_k} \|\mathbf{u}-\mathbf{w}\|_2^2, \quad \text{where } \mathbf{u} = G^{1/2}\mathbf{x} \text{ and } \mathbf{w} = G^{1/2}\mathbf{y} \\
&= G^{-1/2} \operatorname*{argmin}_{\mathbf{u}\in G^{1/2}P_k} \|\mathbf{u}-\mathbf{w}\|_2^2, \qquad (13)
\end{aligned}$$

where (13) suggests an orthogonal projection on the polytope $G^{1/2}P_k$. By defining the vector $\mathbf{c} = (g_1^{-1/2},\dots,g_n^{-1/2})$, one has $G^{1/2}P_k = \{\mathbf{x}\in\mathbb{R}^n_+ \mid \mathbf{c}^\top\mathbf{x} = k, 0\le x_i\le \frac{1}{c_i}\}$. Theorem 6 shows that this projection can be done in $O(n\log n)$, and Algorithm 2 depicts the algorithm achieving the solution.

Theorem 6. *Let $P = \{\mathbf{x}\in\mathbb{R}^n_+ \mid \mathbf{c}^\top\mathbf{x} = k, 0\le x_i\le \frac{1}{c_i}\}$, where $\mathbf{c}\in\mathbb{R}^n_+$ is given. Then for any given point $\mathbf{y}\in\mathbb{R}^n_+$ one can find the solution to $\operatorname{argmin}_{\mathbf{x}\in P}\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2$ in $O(n\log n)$ time. Moreover this solution is unique.*

Proof. Let us begin by writing the KKT optimality conditions for the projected vector $\mathbf{x}$. The Lagrangian is defined by

$$\mathcal{L}(\mathbf{x},\alpha,\boldsymbol{\beta},\boldsymbol{\gamma}) = \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \alpha(\mathbf{c}^\top\mathbf{x}-k) - \boldsymbol{\beta}^\top\mathbf{x} + \boldsymbol{\gamma}^\top(\mathbf{x}-1/\mathbf{c}),$$

Algorithm 2 Projection on the Scaled Uniform Matroid Polytope

Input: vectors $\mathbf{y}, \mathbf{c} \in \mathbb{R}^n_+$ and $k \in \mathbb{N}$, s.t. $k \leq n$.
$\underline{\alpha}_i \leftarrow \frac{y_i c_i - 1}{c_i^2}, \bar{\alpha}_i \leftarrow \frac{y_i}{c_i}, \forall i \in [n]$
$S \leftarrow \{\underline{\alpha}_i\} \cup \{\bar{\alpha}_i\}$
Sort elements in S, so that $S = \{\alpha_1 < \ldots < \alpha_s\}$
$h \leftarrow n, \quad \alpha \leftarrow \min S - 1, \quad m \leftarrow 0$
for $i \in [s]$ **do**
  $h' \leftarrow h + (\alpha_i - \alpha)m$ {calculate function value at the new point using current slope m}
  {check if α^* is between α_i and α_{i-1}}
  if $h' < k \leq h$ **then**
    $\alpha^* \leftarrow (\alpha_i - \alpha)\frac{h-k}{h-h'} + \alpha$
    return the projected vector $\mathbf{x}$ as follows:

$$x_j = \begin{cases} 1/c_j & \alpha^* < \underline{\alpha}_j \\ y_j - \alpha^* c_j & \underline{\alpha}_j \leq \alpha^* \leq \bar{\alpha}_j \\ 0 & \bar{\alpha}_j < \alpha^* \end{cases}$$

  end if
  $m \leftarrow m - \sum_{j:\underline{\alpha}_j = \alpha} c_j^2$ {for these j, $x(\alpha)_j$'s slope is changing from 0 to $-c_j$}
  $m \leftarrow m + \sum_{j:\bar{\alpha}_j = \alpha} c_j^2$ {for these j, $x(\alpha)_j$'s slope is changing from $-c_j$ to 0}
  $h \leftarrow h', \quad \alpha \leftarrow \alpha_i$
end for

where $\alpha \in \mathbb{R}$ and $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^n_+$. Minimizing the Lagrangian w.r.t. $\mathbf{x}$ gives for each $i \in [n]$:

$$x_i = y_i - \alpha c_i + \beta_i - \gamma_i, \tag{14}$$

and also considering complementary slackness, we should have $\beta_i x_i = 0$ and $\gamma_i(x_i - 1/c_i) = 0$. If one provides suitable $\mathbf{x}$ and $\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}$ that satisfy the equations above, then $\mathbf{x}$ would be the optimal solution. In what follows, we construct $\mathbf{x}$ and provide suitable $\alpha, \boldsymbol{\beta}, \boldsymbol{\gamma}$.

For each $\alpha \in \mathbb{R}$, define $\mathbf{x}(\alpha) := \min\{\frac{1}{\mathbf{c}}, \max\{0, \mathbf{y} - \alpha\mathbf{c}\}\}$, where $\min$ and $\max$ are applied element-wise. By definition, one has $\mathbf{0} \leq \mathbf{x}(\alpha) \leq \frac{1}{\mathbf{c}}$. Let $h(\alpha) := \mathbf{c}^\top \mathbf{x}(\alpha)$. We claim that if for a value of α, $h(\alpha) = k$, we are done, since $\mathbf{x}(\alpha) \in P$, and it satisfies the KKT conditions: If $x(\alpha)_i = 0$, by definition of $\mathbf{x}(\alpha)$ it means that $y_i - \alpha c_i \leq 0$, so we can set $\beta_i = -(y_i - \alpha c_i) \geq 0$ and $\gamma_i = 0$. If $x(\alpha)_i = \frac{1}{c_i}$, it means $y_i - \alpha c_i \geq \frac{1}{c_i}$, so we can set $\beta_i = 0$ and $\gamma_i = y_i - \alpha c_i - \frac{1}{c_i} \geq 0$. Otherwise, $0 < x(\alpha)_i < \frac{1}{c_i}$, which in that case we set $\beta_i = \gamma_i = 0$.

So it suffices to provide an α such that $h(\alpha) = k$. For each $i \in [n]$, define $\underline{\alpha}_i := \frac{y_i c_i - 1}{c_i^2}$ and $\bar{\alpha}_i := \frac{y_i}{c_i}$. It's obvious that if $\alpha \leq \underline{\alpha}_i$ then $x(\alpha)_i = \frac{1}{c_i}$, if $\alpha \geq \bar{\alpha}_i$ then $x(\alpha)_i = 0$, and otherwise $x(\alpha)_i = y_i - \alpha c_i$. So $x(\alpha)_i$ is a continuous decreasing function, and so will be $h(\alpha)$. Note that if $\alpha \leq \min\{\underline{\alpha}_i\}$, then $h(\alpha) = n$ and if $\alpha \geq \max\{\underline{\alpha}_i\}$, then $h(\alpha) = 0$. So by continuity, there is some α^* such that $h(\alpha^*) = k$. Now let $\alpha_1 < \ldots < \alpha_s$ be the set of all distinct values among $\underline{\alpha}_i$ and $\bar{\alpha}_i$. It's clear that for all $\alpha \in [\alpha_i, \alpha_{i+1}]$, $h(\cdot)$ is a linear function. By exploiting this fact, we can find α^* by searching through these endpoints. Detailed procedure is explained in Algorithm 2. □

D.2 Efficient projection on Partition matroid base polytope

Let V be a ground set and $A_1, \ldots, A_m$ be a partition of V. A *partition matroid*, includes all sets $S \subseteq V$ such that for all $i \in [m]$ we have $|A_i \cap S| \leq k$. It's easy to see that the base polytope would be

$$\mathcal{P} = \left\{ \mathbf{x} \in [0,1]^V \mid \forall i \in [m] : \sum_{j \in A_i} x_j = k \right\}.$$

In order to project onto $\mathcal{P}$, we first note that it becomes a separable objective, partitioned over A_i. This means that it is sufficient to project $\mathbf{y}|_{A_i}$ onto the uniform matroid of A_i, for all $i \in [m]$. Since each projection takes $\mathcal{O}(|A_i| \log |A_i|)$ time, the total process would be $\mathcal{O}(n \log n)$.

D.3 Projection on general matroid base polytopes

Let us now ask whether there is an efficient projection algorithm for general matroid polytopes. Here, we argue that the method proposed by Kumar and Bach [26] would be a reasonable candidate in the case of general matroid polytopes.

Let $g : 2^V \to \mathbb{R}_+$ be a submodular function, such that $g(\emptyset) = 0$, and let $g_L : \mathbb{R}^n \to \mathbb{R}_+$ be its Lovasz extension. We define the *base polytope* of g as the set

$$\mathcal{B} = \{\mathbf{s} \in \mathbb{R}^n \mid \mathbf{s}(V) = g(V), \forall A \subset V : \mathbf{s}(A) \leq g(A)\}.$$

It can be shown [2] that the Lovasz extension is the *support function* of this polytope, i.e.

$$g_L(\mathbf{x}) = \sup_{\mathbf{s} \in \mathcal{B}} \mathbf{s}^\top \mathbf{x}. \tag{15}$$

For any $\mathbf{y} \in \mathbb{R}^n$ consider the task of minimizing the following objective with respect to $\mathbf{x} \in \mathbb{R}^n$:

$$g_L(\mathbf{x}) - \mathbf{y}^\top \mathbf{x} + \tfrac{1}{2}\|\mathbf{x}\|_2^2. \tag{16}$$

By using (15), we can rewrite (16) in the following dual form

$$\min_{\mathbf{x} \in \mathbb{R}^n} g_L(\mathbf{x}) - \mathbf{y}^\top \mathbf{x} + \tfrac{1}{2}\|\mathbf{x}\|_2^2 = \max_{\mathbf{s} \in \mathcal{B}} -\tfrac{1}{2}\|\mathbf{s} - \mathbf{y}\|_2^2, \tag{17}$$

in which the latter expression is precisely the projection of $\mathbf{y}$ on $\mathcal{B}$. In Kumar and Bach [26], the authors have exploited the structural properties of the Lovasz extension and the faces of the base polytope to create the so-called "Active-set" algorithm. The Active-set algorithm iteratively solves instances of isotonic regression as well as submodular function minimization tasks, whose overall complexity is less than a single submodular function minimization call (recall that by submodular function minimization, we mean the task of solving $\min_{\mathbf{x} \in [0,1]^n}(g_L(\mathbf{x}) - \mathbf{y}^\top \mathbf{x})$). By knowing (17), the algorithm can be viewed as a sequence of iterative projections on outer-approximations of the base polytope.

For any matroid, its associated rank function is a monotone submodular function. Also, the base polytope for a matroid's rank function is exactly the matroid base polytope. As a result of (17), we can use the Active-set algorithm to perform projections on the matroid base polytope. Interestingly, in the case of uniform matroids, the main parts of our projection scheme has similar counterparts as in the Active-set scheme. However, runtime complexity is significantly different due to several differences such as optimality checks: In our approach, this check is done in $\mathcal{O}(1)$, but in Active-set scheme, in each iteration, one should solve approximately $\mathcal{O}(n)$ submodular minimization tasks. However, the Active-set approach is more general, as explained above.

D.4 The RANDOMIZED-PIPAGE-ROUNDING procedure

The randomized pipage rounding procedure was first proposed in [7] for any matroid $\mathcal{M}$. Here, we show how this procedure can be efficiently done (in linear time) for the uniform matroid. Suppose we have a matroid $\mathcal{M}$ and a point $\mathbf{y} := (y_1, \cdots, y_n)$ in its corresponding base polytope. We want to round $\mathbf{y}$ to a vertex of the base polytope. In each step of the algorithm, one has a fractional solution $\mathbf{y}$ and a tight set T containing at least two fractional variables (recall that if the matroid rank function is $r(\cdot)$, a set T is tight if $\mathbf{y}(T) = r(T)$; Tight sets are exactly those constraints in the base polytope who are tight at y). It modifies two fractional variables in such a way that their sum remains constant, until some variable becomes integral or a new constraint becomes tight. Note that since the sum of all of elements of $\mathbf{y}$ is an integer (rank of the matroid), there exist at least two fractional variables in the case that the point is fractional.

For our purpose, we are faced with uniform matroid, which we argue that finding tight constraints is easy, i.e., we can compute the HITCONSTRAINT subroutine in a very fast way. This subroutine is given a fractional point $\mathbf{y}$ and two variables i and j, and tries to increase y_i and decrease y_j simultaneously, and find a new tight constraint A. For sure, one should search for this new tight set through the sets having i inside them but not j. So let $\mathcal{A}$ denote the family of all subsets containing i and not containing j. So we are interested in $\delta = \min_{A \in \mathcal{A}}(r(A) - \mathbf{y}(A))$, the maximum increase in y_i (and decrease in y_j) that does not violate any polytope condition, but produces a new tight constraint. We claim that δ is trivial in case of the uniform matroid: $\delta = \min\{1 - y_i, y_j\}$. Also the new tight set A is either $\{i\}$ or $V - j$.

This simple form of the HITCONSTRAINT gives an efficient algorithm for RANDOMIZED-PIPAGE-ROUNDING, which we describe in Algorithm 3. Moreover, one has the following Theorem:

Algorithm 3 RANDOMIZED-PIPAGE-ROUNDING for the Uniform Matroid

Input: fractional $\mathbf{y}$; $k \in \mathbb{N}$ defining the matroid rank
while $\mathbf{y}$ fractional **do**
  Select i and j among fractional variables
  if $y_i + y_j < 1$ **then**
    Let $p = y_j/(y_i + y_j)$
    With probability p, set $y_i \leftarrow 0$ and $y_j \leftarrow y_i + y_j$, and with probability $1-p$, set $y_i \leftarrow y_i + y_j$ and $y_j \leftarrow 0$.
  else
    Let $p = (1 - y_i)/(2 - y_i - y_j)$
    With probability p, set $y_i \leftarrow y_i + y_j - 1$ and $y_j \leftarrow 1$, and with probability $1 - p$, set $y_i \leftarrow 1$ and $y_j \leftarrow y_i + y_j - 1$.
  end if
end while
return $\mathbf{y}$

Theorem 7. *Let $\mathcal{M}$ be the uniform matroid and $\mathbf{y}$ be a fractional point inside $\mathcal{P}(\mathcal{M})$. Then* RANDOMIZED-PIPAGE-ROUNDING *returns an integral point $\mathbf{y}_{rnd}$ in $\mathcal{O}(n)$ time, such that*

$$\mathbb{E}[F(\mathbf{y}_{rnd})] \geq F(\mathbf{y}).$$

The proof of this algorithm's correctness is similar to the original one given in [7]. It is also noteworthy that our algorithm runs in $\mathcal{O}(n)$ time compared to $\mathcal{O}(n^2)$, as described in [7].

E Details on experiments

E.1 Influence Maximization

Our approach is to obtain samples from the product distribution, $(G, v) \sim \mathcal{G} \times \mathcal{U}$, and compute the set P_v (see the definitions for the class of weighted coverage functions in Section 3). Note that the vertex v is chosen uniformly at random. Since P_v is smaller compared to G, it is less efficient to sample G completely. Instead, while doing the BFS starting from v, we select edges with probability p and proceed. Note that in case of maximizing the upper-bound (8), whenever the sum of x_i visited so far exceeds 1, one can stop and return $\mathbf{0}$, otherwise return $\mathbf{1}_{P_v}$ in the end. This approach is quite fast, but may need too many iterations to converge, because of its locality (i.e. we only take one vertex in each iteration). Note that the size of the gradient in this case is at most $\sqrt{|P_v|} \leq \sqrt{n}$.

E.2 Facility Location

Computing the (stochastic) gradient for the concave upper bound can be done in linear time. Let h be the first index that $\sum_{j=1}^{h} x_j \geq 1$, then a vector in sub-gradient of $\bar{F}(\cdot)$ is simply

$$\mathbf{g} = (m_1 - m_h, \ldots, m_{h-1} - m_h, 0, 0, \ldots, 0). \tag{18}$$

In case of the multilinear extension, we give a linear time algorithm for computing the gradient. Let h be the first index that $x_h = 1$ (if no such index exists, then set $h = n + 1$). It's clear from (10) that $\frac{\partial F}{\partial x_i}(\mathbf{x}) = 0$ for $i = h + 1, \ldots, n$. For $i = h, h - 1, \ldots, 1$ one has the following recursion:

$$\frac{\partial F_e}{\partial x_i}(\mathbf{x}) = \frac{1 - x_{i+1}}{1 - x_i} \frac{\partial F_e}{\partial x_{i+1}}(\mathbf{x}) + (m_i - m_{i+1}) \prod_{j=1}^{i-1} (1 - x_j),$$

which can be done completely in linear time.

E.3 Exemplar-based Clustering

Let V be a set of points. One can quantify the representativeness a set of exemplars $S \subseteq V$ by the loss function $L(S) = \frac{1}{|S|} \sum_{v \in V} \min_{s \in S} \|v - s\|_2$. Finding the best k exemplars is equivalent to

solving $\min_{|S|=k} L(S)$. By introducing an appropriate phantom element e_0 we can turn $L(\cdot)$ into a monotone submodular function [15]: $f(S) = L(\{e_0\}) - L(S \cup \{e_0\})$. Thus maximizing f is equivalent to minimizing L. In our experiments, to ensure non-negativity of the function values, we transform our dataset V by the transformation $T : \mathbb{R}^m \to \mathbb{R}^m$,

$$T(\mathbf{x}) = \frac{3}{\sqrt{m}}\mathbf{1} + \frac{\mathbf{x} - \bar{\mathbf{x}}}{\|\mathbf{x} - \bar{\mathbf{x}}\|_2}, \quad \text{where } \bar{\mathbf{x}} = \frac{1}{|V|}\sum_{\mathbf{x} \in V} \mathbf{x},$$

and set $e_0 = \mathbf{0}$.

F Concave Envelope Evaluation

Here we prove Lemma 1. Define $f(\mathbf{x})$ as follows:

$$f(\mathbf{x}) = \begin{cases} 1 - \prod_{i=1}^n (1 - x_i) & \mathbf{x} \in [0,1]^n \\ -\infty & \text{Otherwise} \end{cases}.$$

We first compute the Fenchel concave dual of f, which is defined as

$$f_*(\mathbf{y}) = \inf\{\mathbf{y}^\top \mathbf{x} - f(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n\}. \tag{19}$$

For brevity, let us define $h(\mathbf{x}, \mathbf{y}) := \mathbf{y}^\top \mathbf{x} - f(\mathbf{x})$. We partition $\mathbb{R}^n$ into several subsets (cases below) and compute the infimum (19) for each subset and take the minimum over all partitions.

Case I, $\mathbf{x} \in (0,1)^n$: Here we can compute the infimum by setting the gradient equal to zero. For a fixed $\mathbf{y} \in \mathbb{R}^n$ we have

$$\nabla_{\mathbf{x}} h(\mathbf{x}, \mathbf{y}) = \mathbf{0} \iff y_i = \frac{\partial f}{\partial x_i} = \frac{\prod_{j=1}^n (1 - x_j)}{(1 - x_i)}.$$

Clearly $y_i > 0$. Let us define $P = \prod_i (1 - x_i)$. We then have $y_1 \cdots y_n = P^{n-1}$, and then $x_i = 1 - P/y_i$. Since $x_i > 0$, we should have $y_i > P$. The following lemma gives the necessary condition on $\mathbf{y}$ for this to happen.

Lemma 8. *Let $y_1, \ldots, y_n \in \mathbb{R}^+$, $n \geq 2$, and assume that $\forall i \in [n] : \; y_i > \sqrt[n-1]{y_1 \cdots y_n}$. Then we have $y_i < 1$ for all $i \in [n]$.*

Proof. We prove the argument by induction. The case $n = 2$ is obvious since

$$y_1 > y_1 y_2 \Rightarrow y_2 < 1, \quad y_2 > y_1 y_2 \Rightarrow y_1 < 1.$$

Suppose the claim is true for $n - 1$. We now prove it for n. W.l.o.g. assume $y_1 \leq \cdots \leq y_n$. We have

$$y_1^{n-1} > y_1 y_2 \cdots y_n \Rightarrow \forall i \geq 2 : \; y_i \geq y_1 > \sqrt[n-2]{y_2 \cdots y_n}.$$

So $y_2, \ldots, y_n$ satisfy the lemma's conditions, and by the induction hypothesis, we have $y_2, \ldots, y_n < 1$. Since $y_1 \leq y_2$, we also have $y_1 < 1$. □

So far, we know that there is a minimum in this case if $\mathbf{y} \in (0,1)^n$. The minimum value would be

$$\begin{aligned} h(\mathbf{x}, \mathbf{y}) &= \sum x_i y_i - f(\mathbf{x}) = \sum (1 - P/y_i) y_i - (1 - P) \\ &= \sum y_i - (n-1)P - 1 \end{aligned} \tag{20}$$

This minimum value, as will be clear shortly, is not the best (lower values are available on other partitions), because of the following lemma:

Lemma 9. *Let $\mathbf{y} \in (0,1)^n$. Then the minimum value of $h(\mathbf{x}, \mathbf{y})$ over $\mathbf{x} \in (0,1)^n$ is strictly greater than -1.*

Proof. Because of (20), we have

$$\begin{aligned} \min_{\mathbf{x} \in (0,1)^n} h(\mathbf{x}, \mathbf{y}) &= \sum_{i=1}^n y_i - (n-1)\sqrt[n-1]{y_1 \cdots y_n} - 1 \\ &> \sum_{i=2}^n y_i - (n-1)\sqrt[n-1]{y_2 \cdots y_n} - 1 && \text{since } 1 > y_1 > 0 \\ &\geq -1 && \text{by AM-GM inequality} \end{aligned}$$

□

Case II, at least for one index $i \in [n]$ we have $x_i = 1$: In this case, we have $f(\mathbf{x}) = 1$, and $h(\mathbf{x}, \mathbf{y}) = \mathbf{y}^\top \mathbf{x} - 1$. W.l.o.g. assume $y_1 \leq \cdots \leq y_n$. It's clear that the minimum value for h over these values of $\mathbf{x}$ would be

$$\min_{\mathbf{x} \in [0,1]^n, \exists_i x_i = 1} h(\mathbf{x}, \mathbf{y}) = \begin{cases} y_1 - 1 & \mathbf{y} \geq \mathbf{0} \\ \sum_{y_i < 0} y_i - 1 & \text{Otherwise} \end{cases}.$$

Case III, for some $i \in [n]$ we have $x_i = 0$: In this case, f would have the same form (with x_i deleted) and also y_i is deleted from h, and hence the problem is reduced to $n - 1$ dimensional case. So the same type of solutions as the previous cases would occur.

In total, for $\mathbf{y}$, such that $y_1 \leq \cdots y_n$ we can write

$$f_*(\mathbf{y}) = \begin{cases} \sum_{y_i < 0} y_i - 1 & y_1 < 0 \\ y_1 - 1 & 0 \leq y_1 < 1 \\ 0 & 1 \leq y_1 \end{cases}$$

Now that we have computed the Fenchel dual of f, we can compute the Fenchel dual of f_*, which would be the Fenchel bi-conjugate of f. By definition we have

$$f_{**}(\mathbf{z}) = \inf\{\mathbf{z}^\top \mathbf{y} - f_*(\mathbf{y}) \mid \mathbf{y} \in \mathbb{R}^n\}.$$

Let's define $g(\mathbf{y}, \mathbf{z}) = \mathbf{z}^\top \mathbf{y} - f_*(\mathbf{y})$. We will find the minimum on the orthant $y_1 \leq \cdots \leq y_n$.

Case IV, $y_1 \geq 1$: We have $g(\mathbf{y}, \mathbf{z}) = \mathbf{z}^\top \mathbf{y}$. If $\mathbf{z}$ has negative components, then infimum would become $-\infty$. So from now on, we assume $\mathbf{z} \geq 0$. In this case, the minimum of g over this case is $\sum z_i$.

Case V, $y_i \in (0, 1)$: We have

$$g(\mathbf{y}, \mathbf{z}) = \mathbf{z}^\top \mathbf{y} - y_1 + 1 \geq y_1(\textstyle\sum z_i - 1) + 1.$$

Now if $\sum z_i \geq 1$, the righthand side's minimum would be 1 and this minimum is achieved by setting $\mathbf{y} = 0$. But if $\sum z_i < 1$, then

$$y_1(\textstyle\sum z_i - 1) + 1 \geq \sum z_i - 1 + 1 = \sum z_i,$$

and this minimum is achieved by setting $\mathbf{y} = \mathbf{1}$.

Case VI, $y_1 < 0$: We have

$$\begin{aligned} g(\mathbf{y}, \mathbf{z}) &= \mathbf{z}^\top \mathbf{y} - \sum_{y_i < 0} y_i + 1 \\ &= \sum_{i: y_i < 0} y_i(z_i - 1) + \sum_{j: y_j \geq 0} y_j z_j + 1 \end{aligned}$$

If for some i, $z_i > 1$, then infimum of g over this case would become $-\infty$, so we have also $\mathbf{z} \leq \mathbf{1}$. In this case, the minimum would become 1.

Summing all up, we have:

$$f_{**}(\mathbf{z}) = \begin{cases} \min\{1, \sum z_i\} & \mathbf{z} \in [0, 1]^n \\ -\infty & \text{Otherwise} \end{cases},$$

and this is what we wanted to prove.

G Pathological Examples

Here we present a special case, where GREEDY fails to give a proper solution, but our method works well. Our example would be about influence maximization with partition matroid condition. It is well known that for general matroids (matroids other than the uniform matroid), GREEDY is guaranteed to give $1/2$-optimal solution.

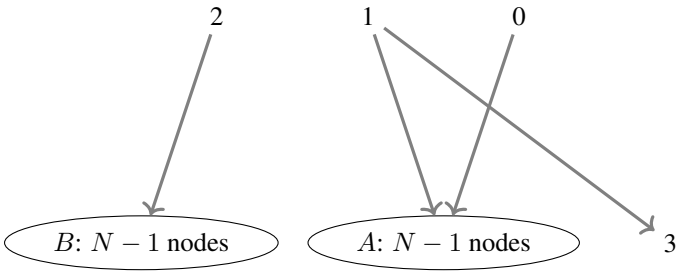

Figure 2: Pathological graph for Influence Maximization

Construct a graph $G = (V, E)$ as follows: $V = \{0, 1, 2, \ldots, 2N + 1\}$, and connect vertices like in the figure. Also take the partitions to be $\{0\}$ and $\{1, 2, \ldots, 2N + 1\}$, meaning that one should take from each partition at most one vertex.

The GREEDY algorithm, first chooses the vertex with the highest out-degree, which is 1, and then is forced to choose 0 as the second vertex because of matroid condition, leading to the $(1/2+\epsilon)$-optimal answer $\{0, 1\}$.

On the other hand, our algorithm successfully gives the optimal answer $\{0, 2\}$. It's a good practice to show why. Let us choose $\mathbf{x}^{(0)}$ be the projection of $\mathbf{1}$ on the partition matroid polytope, namely

$$\mathbf{x}^{(0)} = (1, \tfrac{1}{2N+1}, \ldots, \tfrac{1}{2N+1}).$$

The (sub-)gradient of $\bar{F}$ at $\mathbf{x}^{(0)}$ is calculated as follows:

$$\begin{aligned}\nabla \bar{F}(\mathbf{x}^{(0)}) &= \mathbf{0}_{\{0\}} + \mathbf{1}_{\{1\}} + \mathbf{1}_{\{2\}} + \mathbf{1}_{\{3,1\}} + \sum_{a\in A} \mathbf{0}_{\{0,1,a\}} + \sum_{b\in B} \mathbf{1}_{\{2,b\}} \\ &= (0, 2, N, 1, \underbrace{0, \ldots, 0}_{a\in A}, \underbrace{1, \ldots, 1}_{b\in B})\end{aligned}$$

where the reason of $\mathbf{0}_{\{0\}}$ and $\mathbf{0}_{\{0,1,a\}}$ for $a \in A$ is $\mathbf{x}_0 = 1$ and $\mathbf{x}_0 + \mathbf{x}_1 + \mathbf{x}_a > 1$ respectively. It's obvious that moving along this gradient, and projecting, makes $\mathbf{x}_0 = \mathbf{x}_2 = 1$ and all others to zero, selecting $\{0, 2\}$ as the solution.

The difference here with GREEDY is apparently in two places: (i) being on the matroid base polytope forces the algorithm tochoose a vertex from the first partition, and simultaneously (ii) selecting 0 means all vertices $a \in A$ are influenced, so there is no need to select 1. GREEDY does not take into account that in the future it should select a node in some other partition, that may lose his achievement in the first step.