Approximation Algorithm for Submodular Maximization under Submodular Cover

Naoto Ohsaka[1] **Tatsuya Matsuoka**[1]

[1]NEC Corporation

Abstract

We study a new optimization problem called *submodular maximization under submodular cover* (SMSC), which requires to find a fixed-size set such that one monotone submodular function f is maximized subject to that another monotone submodular function g is maximized approximately. SMSC is preferable to submodular function maximization when one wants to maximize two objective functions simultaneously. We propose an optimization framework for SMSC, which guarantees a constant-factor approximation. Our algorithm's key idea is to construct a new instance of submodular function maximization from a given instance of SMSC, which can be approximated efficiently. Besides, if we are given an approximation oracle for submodular function maximization, our algorithm provably produces nearly optimal solutions. We experimentally evaluate the proposed algorithm in terms of sensor placement and movie recommendation using real-world data.

1 INTRODUCTION

Background. Submodularity is a property of set functions arising in artificial intelligence applications, including influence maximization [Kempe et al., 2003], document summarization [Lin and Bilmes, 2011], and sensor placement [Krause et al., 2008b]. A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is said to be *submodular* if $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$ for all $S \subseteq T \subseteq V$ and $e \in V \setminus T$, which is also known as the "diminishing returns" property. One of the most common optimization problems on submodular functions is *submodular function maximization* (SFM); i.e., solving $\max_{S:|S|=k} f(S)$ for a monotone submodular function f. What makes SFM attractive is that a simple greedy heuristic guarantees a $(1 - 1/\mathrm{e})$-factor approximation to the optimum [Nemhauser et al., 1978].

In this paper, we study the following optimization problem derived from *two* submodular functions:

> *Given two monotone submodular functions $f : 2^V \to \mathbb{R}_{\geq 0}$ and $g : 2^V \to \mathbb{R}_{\geq 0}$ and an approximation threshold β, we are asked to find a size-k set $S \subseteq V$ with $|S| = k$ such that $f(S)$ is maximized subject to that $g(S) \geq \beta \cdot \max_{S':|S'|=k} g(S')$.*

We call this problem *submodular maximization under submodular cover* (SMSC), which can be thought of as an extension of SFM constrained by g. SMSC is preferable to SFM whenever one wants to maximize two objective functions *simultaneously*. We shall present three examples below to motivate SMSC. See Section 2 for the difference of SMSC from existing problems.

- **Sensor Placement:** The effective allocation of a fixed number of sensors can be determined by maximizing the Shannon entropy f of the selected sensors, which is an instance of SFM. Besides, we can express a more complex demand with the second function g. Suppose one wants to improve the sensor placement in operation but does not want to modify the current placement significantly as it would be time-consuming. Then, we can define g as the similarity between a new placement and the current placement. (See Section 5.1.)
- **Movie Recommendation:** A movie recommendation problem involves extracting movies to be displayed in recommender systems. The utility (e.g., diversity and rating) of a set of movies is often expressed as a monotone submodular function f. Suppose one wants to include as many movies as possible from a fixed set T of movies that an agency wants to advertise, for example. We can use the second function g to describe the requirement that many movies from T should be included. (See Section 5.2.)

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

- **Information gain**: Consider selecting the most informative subset of variables for a graphical model. Instead of the Shannon entropy, we can adopt the *information gain* f, which is a more direct measure for the value of information and defined as the expected reduction in uncertainty over the variables in the ground set given the selected observations. The information gain as a set function of the variable set is shown to be monotone and submodular under mild assumptions [Krause and Guestrin, 2005], which applies to attribute selection for naive Bayes classifiers. By using the second objective g, we can further impose a complex constraint introduced so far.

It is easy to see that SMSC is at least as hard as SFM; i.e., we cannot approximate SMSC within a factor better than $(1 - 1/\mathrm{e})$ [Nemhauser and Wolsey, 1978]. On the other hand, unlike the case of SFM, merely applying the greedy heuristic on either f or g does not provide an accurate solution, which will be verified in our experimental evaluation in Section 5. Further, we show that SMSC cannot be solved by simply maximizing the sum of f and g, giving evidence that SFM and SMSC are inherently different problems (Observation 3). The present study aims at developing an algorithm for SMSC that has a provable approximation guarantee.

Our Contributions. In this paper, we develop an optimization framework for SMSC. Our framework's key idea is to construct an instance of SFM from a given instance of SMSC, which can be approximated efficiently. When using the greedy heuristic to solve the SFM instance approximately, the proposed framework guarantees a constant-factor bi-criteria approximation; namely, it returns a set S in polynomial time such that $f(S) \geq 0.16 f(S^*)$ and $g(S) \geq 0.16\beta \cdot \max_{S':|S'|=k} g(S')$, where S^* is an optimal solution for SMSC. Besides, we can enlist the help of *oracles* to solve the SFM instance more accurately. Conceptually, an approximation oracle returns a solution for SFM, which approximates the optimum within a $(1-\epsilon)$-factor. Even though such an oracle requires exponential time in the worst case, the proposed framework with an approximation oracle provably produces a set S such that $f(S) \geq (1-3\epsilon) f(S^*)$ and $g(S) \geq (1-3\epsilon)\beta \cdot \max_{S':|S'|=k} g(S')$.

We evaluate the proposed framework by performing experiments on sensor placement and movie recommendation using real-world data. We confirm that our proposed framework outperforms the greedy heuristic significantly and runs in a reasonable time.

Organization. Section 2 reviews existing studies on submodular function optimization. Section 3 introduces basic notions and presents the definition of submodular function maximization and submodular maximization under submodular coverage. Section 4 develops our framework for SMSC and analyzes its approximation guarantee and time complexity. Section 5 evaluates the proposed algorithm empirically on sensor placement and movie recommendation.

2 RELATED WORK

2.1 SUBMODULAR FUNCTION MAXIMIZATION

Submodularity arises as a property of set functions in numerous combinatorial notions, including coverage functions, matroid rank functions, graph cuts, and entropy; see, e.g., Krause and Golovin [2014]. Optimization of submodular functions has been actively studied, and we here review monotone submodular function maximization under the size constraint (SFM), which is included in SMSC as a special case. The simple greedy algorithm, which runs in quadratic time, achieves an approximation factor of $1 - 1/\mathrm{e} \approx 0.63$ [Nemhauser et al., 1978], and this is the best possible approximation factor because no polynomial-time algorithm can achieve a better approximation factor [Nemhauser and Wolsey, 1978, Feige, 1998]. More recently, nearly linear-time approximation algorithms [Badanidiyuru and Vondrák, 2014, Mirzasoleiman et al., 2015] and approximation algorithms for complex constraints [Călinescu et al., 2011, Sviridenko, 2004] have been developed; see, e.g., the survey of Buchbinder and Feldman [2018] for details.

Since there is still a need for more accurate or even exact solutions for SFM beyond the theoretical limit of a $(1 - 1/\mathrm{e})$-factor, several heuristic algorithms have been developed. Chen, Chen, and Weinberger [2015] developed the Filtered Search algorithm for submodular maximization under the knapsack constraint. Given an approximation threshold α, Filtered Search performs a best-first search to find an α-approximate solution. Sakaue and Ishihata [2018] improved heuristic functions used in Filtered Search for bounding the optimal value accurately. Uematsu, Umetani, and Kawahara [2019] proposed an exact algorithm for SFM based on binary integer programming, which dates back to Nemhauser and Wolsey [1981]. Those algorithms require exponentially increasing steps in the worst case, but they can manage moderately large problems in practice. Our optimization framework can use such practically-efficient algorithms as well as the greedy algorithm as a subroutine.

2.2 VARIANTS OF SUBMODULAR OPTIMIZATION

We review the recent work on variants of submodular function optimization. Narasimhan and Bilmes [2005], Iyer and Bilmes [2012] studied minimizing the *difference* between two submodular functions (*DS function*); i.e., $\min_S \{f(S) - g(S)\}$, where f and g are submodular. Narasimhan and Bilmes [2005] proved that DS functions can express an arbitrary set function. Iyer and Bilmes [2012] proved a multiplicative inapproximability result and designed algorithms with an additive approximation guarantee. Iyer and Bilmes [2013] considered two optimization problems: One is the *submodular cost submodular cover (SCSC)* problem defined as $\min_{S:g(S)\geq c} f(S)$, and the other is the *submodular cost submodular knapsack (SCSK)* problem defined as $\max_{S:f(S)\leq b} g(S)$, for two monotone submodular functions f and g. Both problems can be recast into DS function minimization. Iyer and Bilmes [2013] developed bi-criteria approximation algorithms for SCSC and SCSK, whose approximation factor depends on the size n of the ground set V. Bai, Iyer, Wei, and Bilmes [2016] considered minimizing the *ratio* of two monotone submodular functions (*RS function*), i.e., $\min_{S} f(S)/g(S)$. RS function minimization can be approximated using algorithms for DS minimization, SCSC, or SCSK. Bai et al. [2016] proposed approximation algorithms for RS function minimization, whose approximation factor is $1/\mathcal{O}(\sqrt{n}\log n)$. *Robust submodular maximization (RSM)* is defined as $\max_{S:|S|=k} \min_{i\in[m]} f_i(S)$ for m monotone submodular functions $f_1, \dots, f_m$. Krause, McMahan, Guestrin, and Gupta [2008a] gave an approximation algorithm whose output is logarithmically larger than the optimal solution. Powers, Bilmes, Wisdom, Krout, and Atlas [2016] considered RSM under the matroid constraint and devised an algorithm that returns a feasible set that maximizes not all but a certain fraction of the m functions. Anari, Haghtalab, Naor, Pokutta, Singh, and Torrico [2019] studied RSM under the matroid and knapsack constraint and developed an approximation algorithm that returns a $(1-\epsilon)$-approximate solution of size $\mathcal{O}(k\log\frac{m}{\epsilon})$.

We finally differentiate this study from the previous work.

- Our problem SMSC can be expressed as $\max_{S:|S|=k, g(S)\geq c} f(S)$ for some c, which belongs to neither SCSC, SCSK, DS maximization, nor RS minimization. The decision version of SMSC (see Problem 4) is a special case of RSM; however, existing algorithms for RMS do not satisfy the size constraint [Krause et al., 2008a, Anari et al., 2019] or provide explicit approximation guarantees [Powers et al., 2016]. On the other hand, the proposed framework produces a *specified-size* set that guarantees a *constant-factor* bi-criteria approximation in *polynomial time* (whenever the greedy heuristic is used as an oracle for SFM).
- Furthermore, our framework allows utilizing a well-engineered implementation for SFM as an approximation oracle, which is not necessarily the case for the problems reviewed above. Though those implementations require exponential time in the worst case, our experiments demonstrate that the proposed framework with an approximation oracle provides a more accurate solution than that with the greedy algorithm in a reasonable running time. Our approximation framework in Section 4 is inspired by the SATURATE algorithm [Krause et al., 2008a].

3 PRELIMINARIES

Notations. For a positive integer n, let $[n]$ denote the set $\{1, 2, \dots, n\}$, and let V be a finite ground set of size n, e.g., $V = [n]$. We assume every set function $f : 2^V \to \mathbb{R}$ to be nonnegative; i.e., $f(S) \geq 0$ for all $S \subseteq V$. A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is said to be *monotone* if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq V$ and *submodular* if $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$ for all $S \subseteq T \subseteq V$ and $e \in V \setminus T$. Submodularity is equivalent to the following inequality: $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all $S, T \subseteq V$.

Submodular Function Maximization and Optimization Oracles. We define submodular function maximization as follows.

Problem 1 (SFM). *Given a monotone submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ and a solution size $k \in [n]$, the problem of* submodular function maximization (SFM) *asks to find a size-k set that has the maximum function value; i.e., to compute* $\operatorname{argmax}_{S\subseteq V:|S|=k} f(S)$.

Hereafter, we denote the optimum for SFM on f by OPT_f; i.e., $\mathrm{OPT}_f \triangleq \max_{S\subseteq V:|S|=k} f(S)$. We say that a set $S \subseteq V$ with $|S| = k$ is an *α-approximation* for SFM on f if $f(S) \geq \alpha\,\mathrm{OPT}_f$. Theoretically, the greedy algorithm shown in Algorithm 1 delivers a $(1-1/\mathrm{e})$-approximation by evaluating f for $\mathcal{O}(nk)$ subsets [Nemhauser et al., 1978]. This is the best possible approximation factor because no polynomial-time algorithm can achieve a $(1-1/\mathrm{e}+\epsilon)$-approximation for any $\epsilon > 0$ [Nemhauser and Wolsey, 1978, Feige, 1998].

In this paper, we assume access to "oracles" for SFM. An *α-approximation oracle* for $\alpha \in [0, 1]$ is an abstract machine that returns an α-approximation for SFM, i.e., $S \subseteq V$ with $|S| = k$ such that $f(S) \geq \alpha\,\mathrm{OPT}_f$, in a single step. While we cannot obtain $(1-1/\mathrm{e}+\epsilon)$-approximation in polynomial time, the recent development of practically-efficient algorithms [Sakaue and Ishihata, 2018, Chen et al., 2015, Uematsu et al., 2019] encourages to adopt such oracles. Our framework's efficiency will be measured by the number of oracle calls because the empirical running time of a "single step" depends on its implementation. Note that the greedy algorithm is a $(1-1/\mathrm{e})$-approximation oracle that runs in *polynomial time*.

Submodular Maximization under Submodular Cover. We now define the central problem referred to as submodular maximization under submodular cover as follows.

Algorithm 1 Greedy algorithm for SFM.

```
Input: monotone submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$;
    solution size $k$.
1: $S_0 \leftarrow \emptyset$.
2: for all $i = 1$ to $k$ do
3:     $S_i \leftarrow S_{i-1} \cup \left\{ \operatorname*{argmax}_{e \in V \setminus S_{i-1}} f(S_{i-1} \cup \{e\}) \right\}$.
4: return set $S_k$.
```

Problem 2 (SMSC)**.** *Given two monotone submodular functions $f : 2^V \to \mathbb{R}_{\geq 0}$ and $g : 2^V \to \mathbb{R}_{\geq 0}$, an approximation threshold β, and a solution size $k \in [n]$, the problem of* submodular maximization under submodular cover (SMSC) *asks to find a size-k solution $S \subseteq V$ with $|S| = k$ such that S is a β-approximation to* SFM *on g and $f(S)$ is maximized, i.e., to compute the following:*

$$\operatorname*{argmax}_{S \subseteq V : |S| = k} f(S) \text{ subject to } g(S) \geq \beta \operatorname{OPT}_g . \quad (1)$$

Since SMSC with $\beta = 0$ includes SFM as a special case, we cannot approximate SMSC within a $(1 - 1/\mathrm{e} + \epsilon)$-factor in polynomial time. We develop in Section 4 an optimization framework for SMSC *given* access to oracles for SFM, which successfully circumvents the theoretical barrier in practice.

Failed Attempt based on Lagrangian Formulation. We here describe a failed attempt based on the Lagrangian formulation. Given two monotone submodular functions $f : 2^V \to \mathbb{R}_{\geq 0}$ and $g : 2^V \to \mathbb{R}_{\geq 0}$, let us consider a monotone submodular function $F_\lambda : 2^V \to \mathbb{R}_{\geq 0}$ defined as $F_\lambda(S) \triangleq f(S) + \lambda g(S)$ for all $S \subseteq V$, where $\lambda \geq 0$ is a parameter. Note that the conversion from SMSC to the Lagrangian formulation using F_λ can be regarded as transforming the Ivanov formulation to the Tikhonov formulation [Oneto et al., 2016]. We denote the size-k set that maximizes $F_\lambda(S)$ by S_λ. It is easy to see that $f(S_\lambda)$ is a nonincreasing function of λ and $g(S_\lambda)$ is a nondecreasing function of λ. One might thus think of performing a bisection search on λ to find the minimum value of λ such that $g(S_\lambda) \geq \beta \operatorname{OPT}_g$, in the hope that S_λ is optimal to SMSC. However, we show that this strategy does not work due to the following observation; in particular, SMSC cannot be approximated by merely maximizing F_λ for any $\lambda \geq 0$, which means that SMSC is inherently different from SFM.

Observation 3. *For any approximation threshold β in $(0, \frac{3}{4})$ and a solution size $k = 1$, there exists two monotone submodular functions f and g and such that the following conditions are satisfied:*

- *an optimal solution S^* for* SMSC *defined by f, g, β, k satisfies that $f(S^*) > 0$ and $g(S^*) > 0$;*
- *for any $\lambda \geq 0$ and $\epsilon \in [0, \frac{1}{5})$, invoking a $(1 - \epsilon)$-approximation oracle for* SFM *on F_λ yields a set S that satisfies at least either of $f(S_\lambda) = 0$ or $g(S_\lambda) = 0$; i.e., S_λ has no positive approximation guarantee for* SMSC.

Proof. Let $V \triangleq [3]$, $\beta \in (0, \frac{3}{4})$, $(a_1, a_2, a_3) \triangleq (1, 0, 32)$, and $(b_1, b_2, b_3) \triangleq (15, \frac{15}{\beta}, 0)$. We define a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ such that $f(S) \triangleq \sum_{i \in S} a_i$ for each $S \subseteq V$, a set function $g : 2^V \to \mathbb{R}_{\geq 0}$ such that $g(S) \triangleq \sum_{i \in S} b_i$ for each $S \subseteq V$, and a set function $F_\lambda : 2^V \to \mathbb{R}_{\geq 0}$ such that $F(S) \triangleq f(S) + \lambda g(S)$ for each $S \subseteq V$. Consider an instance of SMSC defined by $f, g, \beta, k = 1$. An optimal solution S^* for the SMSC instance is $\{1\}$, which satisfies that $f(S^*) = 1$ and $g(S^*) = 15$. On one hand, if $\lambda \geq 1$, then we have that $\frac{F_\lambda(\{1\})}{F_\lambda(\{2\})} \leq \frac{4}{5}$. On the other hand, if $\lambda < 1$, then we have that $\frac{F_\lambda(\{1\})}{F_\lambda(\{3\})} \leq \frac{1}{2}$. Therefore, for any $\lambda \geq 0$, a $(1 - \epsilon)$-approximation oracle for $\epsilon \in [0, \frac{1}{5})$ does not choose $\{1\}$ but either $\{2\}$ or $\{3\}$, which meet the second property in the statement. □

4 PROPOSED FRAMEWORK

In this section, we develop an optimization framework for SMSC, which guarantees a "bi-criteria approximation." Here, a solution $S \subseteq V$ with $|S| = k$ is said to be a $[\sigma, \rho]$-*approximation to* SMSC for some $\sigma, \rho > 0$ if it holds that $f(S) \geq \sigma f(S^*)$ and $g(S) \geq \rho \beta \operatorname{OPT}_g$, where S^* is an optimal solution to SMSC.

Our framework produces a $[1 - 3\epsilon, 1 - 3\epsilon]$-approximation for SMSC if we are given a $(1 - \epsilon)$-approximation oracle for SFM (Theorem 6). The number of oracle calls is bounded by $\mathcal{O}(\log \epsilon^{-1})$. Furthermore, we prove that if we adopt the greedy algorithm (Algorithm 1) as a $(1 - 1/\mathrm{e})$-approximation oracle, the proposed framework returns a $[0.16, 0.16]$-approximation for SMSC in *polynomial time* (Theorem 7).

Since an implementation of $(1 - \epsilon)$-approximation oracles becomes faster empirically as the value of ϵ increases [Sakaue and Ishihata, 2018], we can trade between the accuracy and efficiency by adjusting the value of ϵ.

4.1 DECISION VERSION AND ITS BI-CRITERIA APPROXIMATION

Before designing our framework, we consider the following decision problem:

Problem 4. *Given two monotone submodular functions $f : 2^V \to \mathbb{R}_{\geq 0}$ and $g : 2^V \to \mathbb{R}_{\geq 0}$, approximation thresholds α and β, and a solution size $k \in [n]$, find $S \subseteq V$ with $|S| = k$ that is an α-approximation to* SFM *on f (i.e., $f(S) \geq \alpha \operatorname{OPT}_f$) and a β-approximation to* SFM *on g (i.e., $g(S) \geq \beta \operatorname{OPT}_g$) simultaneously, or declare that there cannot be such a solution.*

Algorithm 2 Bi-criteria approximation algorithm for Problem 4 with an approximation SFM oracle.

```
Input: $(1-\epsilon)$-approximate optimum $\mathrm{OPT}'_f$ and $\mathrm{OPT}'_g$;
    approximation thresholds $\alpha$ and $\beta$.
1: invoke approximation oracle to find $(1-\epsilon)$-
   approximation $\hat{S}$ to Eq. (2).
2: if (objective value of $\hat{S}$ for Eq. (2) ) $\geq 2(1-\epsilon)$ then
3:     return $\hat{S}$.
4: else
5:     declare "not found."
```

Observe that Problem 4 is a special case of robust submodular maximization in which two monotone submodular functions f_1 and f_2 are given by $f_1(S) = \min\Big\{1, \dfrac{f(S)}{\alpha \,\mathrm{OPT}_f}\Big\}$ and $f_2(S) = \min\Big\{1, \dfrac{g(S)}{\beta \,\mathrm{OPT}_g}\Big\}$, and the objective is $\max\limits_{S:|S|=k} \min\{f_1(S), f_2(S)\}$. Once an approximate solver for Problem 4 is designed, we can repeatedly use it to perform a bisection search on an approximation threshold α to solve SMSC.

Algorithm 2 describes an approximation algorithm for Problem 4 given access to a $(1-\epsilon)$-approximation oracle for SFM. Our algorithm approximately solves the following problem:

$$\max_{S \subseteq V : |S|=k} \min\left\{1, \frac{f(S)}{\alpha \,\mathrm{OPT}'_f}\right\} + \min\left\{1, \frac{g(S)}{\beta \,\mathrm{OPT}'_g}\right\}, \tag{2}$$

where OPT'_f (resp. OPT'_g) is a $(1-\epsilon)$-approximate optimum for SFM on f (resp. g); i.e., $\mathrm{OPT}'_f \geq (1-\epsilon)\,\mathrm{OPT}_f$ and $\mathrm{OPT}'_g \geq (1-\epsilon)\,\mathrm{OPT}_g$. Since submodularity is preserved under taking nonnegative linear combinations[1] and truncation,[2] Eq. (2) is an instance of SFM. Solving Eq. (2) approximately by invoking an approximation oracle, Algorithm 2 returns the resulting solution $\hat{S}$ if $\hat{S}$ has an objective value in Eq. (2) at least $2(1-\epsilon)$, and otherwise, we declare that there exists no feasible solution. Algorithm 2 has the following bi-criteria approximation guarantee.

Lemma 5. *Given a $(1-\epsilon)$-approximation oracle for* SFM*, Algorithm 2 returns a bi-criteria approximation $S \subseteq V$ with $|S| = k$ to Problem 4 such that $f(S) \geq (1 - 3\epsilon + 2\epsilon^2)\alpha \,\mathrm{OPT}_f$ and $g(S) \geq (1 - 3\epsilon + 2\epsilon^2)\beta \,\mathrm{OPT}_g$, or it correctly declares that there cannot be a feasible solution, by invoking a single call of the oracle.*

Proof. Suppose there exists a feasible solution $S^* \subseteq V$ with $|S^*| = k$ such that $f(S^*) \geq \alpha \,\mathrm{OPT}_f$ and $g(S^*) \geq \beta \,\mathrm{OPT}_g$ (i.e., the answer is "yes"). The optimum of Eq. (2) is then equal to 2; a $(1-\epsilon)$-approximation $\hat{S}$ to Eq. (2) must satisfy that

$$\min\left\{1, \frac{f(\hat{S})}{\alpha \,\mathrm{OPT}'_f}\right\} + \min\left\{1, \frac{g(\hat{S})}{\beta \,\mathrm{OPT}'_g}\right\} \geq 2(1-\epsilon).$$

In this case, $f(\hat{S})$ is bounded from below as follows.

$$\begin{aligned} f(\hat{S}) &\geq \alpha \,\mathrm{OPT}'_f \left[2(1-\epsilon) - \min\left\{1, \frac{g(\hat{S})}{\beta \,\mathrm{OPT}'_g}\right\}\right] \\ &\geq \alpha \,\mathrm{OPT}'_f [2(1-\epsilon) - 1] \\ &\geq \alpha \,\mathrm{OPT}_f (1-\epsilon)(1-2\epsilon) \\ &\geq (1 - 3\epsilon + 2\epsilon^2)\alpha \,\mathrm{OPT}_f \,. \end{aligned}$$

$g(\hat{S})$ is similarly bounded from below by $(1 - 3\epsilon + 2\epsilon^2)\beta \,\mathrm{OPT}_g$. Consequently, $\hat{S}$ is a $[1-3\epsilon+2\epsilon^2, 1-3\epsilon+2\epsilon^2]$-approximation to Problem 4.

On the other hand, when $\hat{S}$ has an objective value strictly less than $2(1-\epsilon)$, we can safely ensure that $f(S) < \alpha \,\mathrm{OPT}_f$ or $g(S) < \beta \,\mathrm{OPT}_g$ for all $S \subseteq V$ with $|S| = k$. □

4.2 ALGORITHM DESCRIPTION

Algorithm 3 describes the proposed framework for SMSC with an approximation SFM oracle. Our algorithm first computes $(1-\epsilon)$-approximate optimum OPT'_f and OPT'_g by invoking a $(1-\epsilon)$-approximation SFM oracle. It then performs a bisection search on an approximation threshold α starting with a range $[\alpha_{\max}, \alpha_{\min}] \triangleq [0, 1]$, which requires to solve Problem 4 by Algorithm 2. It terminates the bisection search if we have that $(1-\epsilon^4)\alpha_{\max} > \alpha_{\min}$, and returns a solution obtained by running Algorithm 2 on $(\mathrm{OPT}'_f, \mathrm{OPT}'_g, \alpha_{\min}, \beta)$, where we use the value of $\alpha_{\min}$ obtained during the last iteration of the bisection search. The following theorem presents the approximation guarantee and the number of oracle calls of Algorithm 3.

Theorem 6. *Given a $(1-\epsilon)$-approximation* SFM *oracle, Algorithm 3 provides a $[1-3\epsilon, 1-3\epsilon]$-approximation to* SMSC*, by invoking the oracle $\mathcal{O}(\log \epsilon^{-1})$ times.*

Proof. When the algorithm terminates, the following holds by Lemma 5: $f(S) \leq \alpha_{\max} \,\mathrm{OPT}_f$ for any $S \subseteq V$ with $|S| = k$ such that $g(S) \geq \beta \,\mathrm{OPT}_g$, and $g(\hat{S}) \geq (1 - 3\epsilon + 2\epsilon^2)\beta \,\mathrm{OPT}_g \geq (1-3\epsilon)\beta \,\mathrm{OPT}_g$, where $\hat{S}$ is the algorithm's return. We then have that

$$\begin{aligned} f(\hat{S}) &\geq (1 - 3\epsilon + 2\epsilon^2)\alpha_{\min} \,\mathrm{OPT}_f & (3) \\ &\geq (1 - 3\epsilon + 2\epsilon^2)(1-\epsilon^4)\alpha_{\max} \,\mathrm{OPT}_f & (4) \\ &\geq (1-3\epsilon) \max_{S \subseteq V : |S|=k, g(S) \geq \beta \,\mathrm{OPT}_g} f(S). & (5) \end{aligned}$$

[1]If $g_1, \ldots, g_m : 2^V \to \mathbb{R}$ are monotone submodular, and $\alpha_1, \ldots, \alpha_m \geq 0$, then so is $f(S) = \sum_{i \in [m]} \alpha_i g(S)$.

[2]If $g : 2^V \to \mathbb{R}$ is monotone submodular, then so is $f(S) = \min\{g(S), c\}$ for any constant c.

Algorithm 3 Optimization framework for SMSC with an approximation SFM oracle.

```
Input: two monotone submodular functions $f : 2^V \to \mathbb{R}_{\geq 0}$ and $g : 2^V \to \mathbb{R}_{\geq 0}$; approximation threshold $\beta$; solution size $k$.
 1: invoke $(1-\epsilon)$-approximation SFM oracle on $f, k$ and $g, k$ to compute a $(1-\epsilon)$-approximate optimum $\mathrm{OPT}'_f$ and $\mathrm{OPT}'_g$, respectively; i.e., $\mathrm{OPT}'_f \geq (1-\epsilon) \cdot \mathrm{OPT}_f$ and $\mathrm{OPT}'_g \geq (1-\epsilon) \cdot \mathrm{OPT}_g$.
 2: $\alpha_{\min} \leftarrow 0$ and $\alpha_{\max} \leftarrow 1$.
 3: while $(1-\epsilon^4)\alpha_{\max} > \alpha_{\min}$ do
 4:     $\bar{\alpha} = \frac{\alpha_{\min}+\alpha_{\max}}{2}$.
 5:     call Algorithm 2 on $(\mathrm{OPT}'_f, \mathrm{OPT}'_g, \bar{\alpha}, \beta)$.
 6:     if $[1-3\epsilon+2\epsilon^2, 1-3\epsilon+2\epsilon^2]$-approximation to Problem 4 was found then
 7:         $\alpha_{\min} \leftarrow \bar{\alpha}$.
 8:     else
 9:         $\alpha_{\max} \leftarrow \bar{\alpha}$.
10: return a solution $\hat{S}$ obtained by calling Algorithm 2 on $(\mathrm{OPT}'_f, \mathrm{OPT}'_g, \alpha_{\min}, \beta)$.
```

Hence, $\hat{S}$ is a $[1-3\epsilon, 1-3\epsilon]$-approximation to SMSC. The number of oracle calls is equal to the number of iterations plus a constant, which is bounded by $\mathcal{O}(\log \epsilon^{-1})$. □

4.3 POLYNOMIAL-TIME $[0.16, 0.16]$-APPROXIMATION

Obviously, the greedy algorithm (Algorithm 1) is a polynomial-time oracle for SFM with the best approximation guarantee. We show that Algorithm 3 with the greedy algorithm returns a $[0.16, 0.16]$-approximation for SMSC.

Theorem 7. *Given the greedy algorithm (Algorithm 1) as an approximation oracle for* SFM*, Algorithm 3 provides a* $[0.16, 0.16]$*-approximation to* SMSC *in polynomial time, by evaluating two input monotone submodular functions for at most* $\mathcal{O}(nk)$ *subsets.*

Proof. According to the proof of Theorem 6, Algorithm 3 with a $(1-\epsilon)$-approximation oracle returns a set $\hat{S}$ such that

$$g(\hat{S}) \geq (1-3\epsilon+2\epsilon^2)\beta\,\mathrm{OPT}_g,$$
$$f(\hat{S}) \geq (1-3\epsilon+2\epsilon^2)(1-\epsilon^4)f(S^*),$$

where S^* is an optimal solution for SMSC. Since a greedy algorithm is a $(1-1/\mathrm{e})$-approximation oracle for SFM, we have that $(1-3/\mathrm{e}+2/\mathrm{e}^2)(1-1/\mathrm{e}^4) \geq 0.16$, which proves the approximation factor. The number of function evaluations is bounded by $\mathcal{O}(nk)$ because a single call of the greedy algorithm requires $\mathcal{O}(nk)$ time and the number of iterations in the bisection search is $\mathcal{O}(\log \mathrm{e})$, which is a constant. □

5 EXPERIMENTS

We evaluate the proposed framework by performing experiments on sensor placement and movie recommendation using real-world data. In particular, the purpose of this section is to answer the following questions:

- How does the value of β affect the approximation quality in terms of f?
- How efficient is Algorithm 3?
- Does Algorithm 3's solution S satisfy the requirement that $g(S) \geq \beta\,\mathrm{OPT}_g$?

We compare the following algorithms.

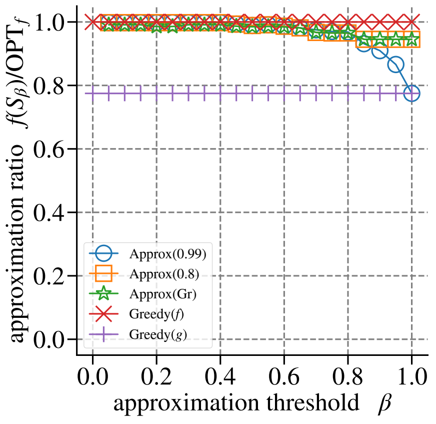
- *Approx(0.99)*: Algorithm 3 with a 0.99-approximation oracle for SFM, which guarantees a $[0.97, 0.97]$-approximation while requiring exponential time in the worst case.
- *Approx(0.8)*: Algorithm 3 with a 0.8-approximation oracle for SFM, which guarantees a $[0.4, 0.4]$-approximation while requiring exponential time in the worst case.
- *Approx(Gr)*: Algorithm 3 with the greedy algorithm for SFM, which guarantees a $[0.16, 0.16]$-approximation and runs in polynomial time.
- *Greedy(f)* and *Greedy(g)*: We run the greedy algorithm on f and g, which does not have any approximation guarantee.

We implemented the above-described algorithms in Python 3.6. For an approximation oracle for SFM, we implemented Uematsu et al.'s algorithm *ICG* using the Gurobi Optimizer ver. 9.0.1 [Gurobi Optimization, LLC, 2020]. We conducted all experiments on a Linux server with an Intel Xeon E5-2699 2.30GHz CPU and 792GB memory.

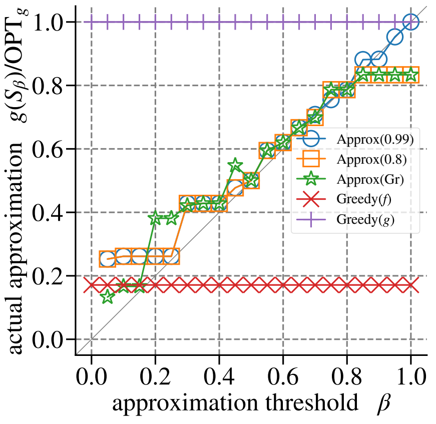
5.1 SENSOR PLACEMENT

We applied the proposed framework to the sensor placement problem, which requires allocating a small number of sensors so that the expected uncertainty is most effectively reduced.

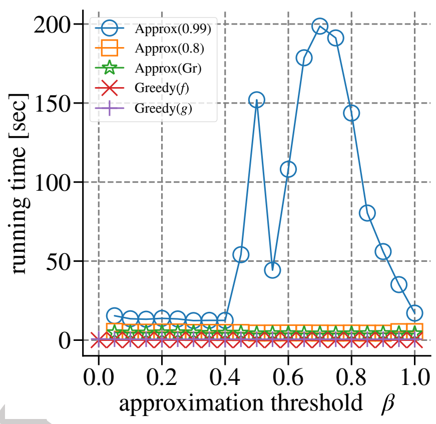
Setup. We used the publicly-available Intel Lab data [Madden, 2014], which contains a log of approximately 2.3 million readings regarding temperature, humidity, and light for 65,536 epochs collected from $n = 54$ sensors in the Intel Berkeley Research Laboratory between February 28th and April 5th. We replaced missing readings for each sensor using linear interpolation and used temperature measurements every 512 epochs. This data also includes the x and y coordinates of each installation location.



(a) β vs. $f(S_\beta)/\,\mathrm{OPT}_f$



(b) β vs. $g(S_\beta)/\,\mathrm{OPT}_g$



(c) β vs. running time

Figure 1: Sensor placement results, where S_β denotes the solution of SMSC with β.

We designed two monotone submodular functions f and g as follows. Consider the following scenario: One is going to improve the sensor placement in operation but does not want to modify the current placement significantly as it would require time and effort. We can apply SMSC to such scenarios by setting so that f captures the expected uncertainty of a new sensor placement S and g captures the similarity between S and the current sensor placement, say, T. We first design $f : 2^V \to \mathbb{R}_{\geq 0}$ based on entropy, which has been adopted in the literature [Ohsaka and Yoshida, 2015, Krause et al., 2008b, Sharma et al., 2015]. Let $V = [n]$, and for each $i \in V$, let X_i be the random variable representing the temperature measurement collected from a sensor located at the i-th location, and let $\Omega = \{X_i\}_{i \in V}$. The *Shannon entropy* of $X \subseteq \Omega$ is defined as

$$H(X) \triangleq -\sum_{x \in \mathrm{dom}(X)} \Pr[x] \cdot \log_2 \Pr[x],$$

where $\mathrm{dom}(X)$ denotes the set of possible outcomes of X, and $\Pr[x]$ is the probability of observing an outcome x. The Shannon entropy $H(\cdot)$ is known to be monotone and submodular [Fujishige, 1978]. Since the conditional entropy of Ω having observed S is $H(\Omega \mid X) = H(\Omega) - H(X)$, to reduce the uncertainty of Ω, it suffices to find a set having the largest entropy $H(X)$. Hence, we define f such that $f(S) \triangleq H(\{X_i\}_{i \in S})$.

We then designed a similarity function $g : 2^V \to \mathbb{R}_{\geq 0}$ based on maximum-weight bipartite matching. Here, a *matching* between two sets S and T is defined as a pair set $M \subseteq S \times T$ in which no two pairs share the first or second element. The *weight* of a matching M is defined as $\sum_{(i,j) \in M} w_{i,j}$, where $w_{i,j}$ is a similarity score between $i \in S$ and $j \in T$. Then, the *maximum-weight matching* is defined as a matching with the maximum weight over all possible matchings between S and T. The maximum-weight matching as a set function in S is known to be monotone submodular [Ito et al., 2002, Sakashita et al., 2008]. Let x_i and y_i be the x and y coordinates of installation location $i \in [n]$, respectively. We define the similarity score between two installation locations i and j as

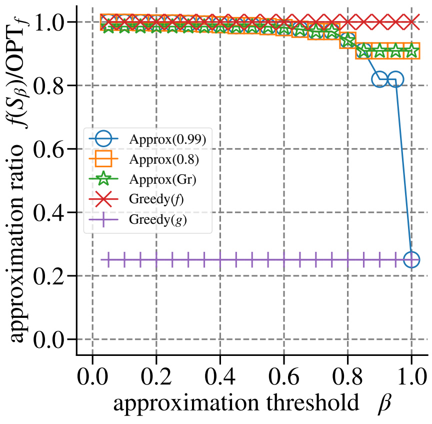
$$w_{i,j} \triangleq \exp\Big(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{\sigma^2}\Big),$$

where $\sigma = 16$ is a parameter. Then $g : 2^V \to \mathbb{R}_{\geq 0}$ is defined as the weight of the maximum-weight matching between S and T; i.e.,

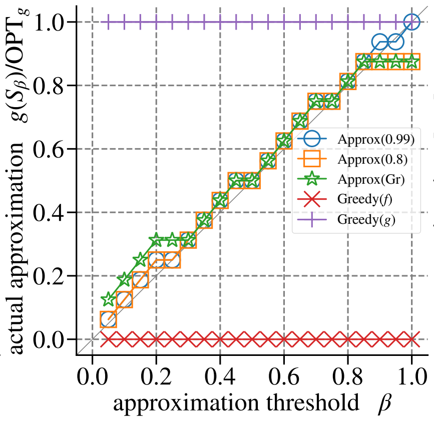
$$g(S) \triangleq \max_{M \subseteq S \times T:\text{matching}} \sum_{(i,j) \in M} w_{i,j}.$$

The computation of g takes $\mathcal{O}(n^3)$ time [Galil, 1986]. We set the solution size k to 6 and construct a (low-entropy) set T of size 6 by running the greedy algorithm on $-f(\cdot)$.

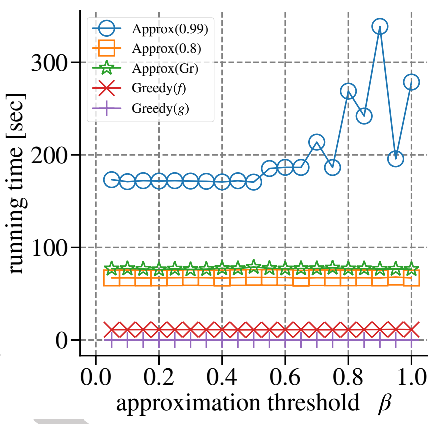
Results. We ran Algorithm 3 and *Greedy* with each approximation threshold $\beta = 0.05, 0.1, \ldots, 0.95, 1$. Figures 1a and 1b plot $f(S_\beta)$ and $g(S_\beta)$ for each value of β, respectively, where S_β is a solution that *Approx(0.99)*, *Approx(0.8)*, *Approx(Gr)*, *Greedy(f)*, or *Greedy(g)* returned with β. Whereas *Approx(0.99)* satisfies that $g(S_\beta) \geq \beta\,\mathrm{OPT}_g$ excepting only a few points of β (e.g., $\beta = 0.9$), *Approx(0.8)* and *Approx(Gr)* violate the constraint for $\beta > 0.8$. Regarding the approximation accuracy $f(S_\beta)/\,\mathrm{OPT}_f$, *Approx(Gr)* is slightly worse than *Approx(0.8)* and *Approx(0.99)* for small β. *Greedy(f)* gives a solution S_f such that $f(S_f) = \mathrm{OPT}_f$ and $g(S_f) \approx 0.17\,\mathrm{OPT}_g$, which significantly violates the constraint on g, *Greedy(g)* on g gives a solution S_g such that $f(S_g) \approx 0.77\,\mathrm{OPT}_f$ and $g(S_g) = \mathrm{OPT}_g$, which is drastically worse than *Approx(0.99)*, *Approx(0.8)*, and *Approx(Gr)* for the case of small β. Figure 1c plots running time for computing S_β for each β. *Greedy(f)* and *Greedy(g)* required only 1 second. *Approx(0.8)* and *Approx(Gr)* required 6 seconds. Though *Approx(0.99)* was



(a) β vs. $f(S_\beta)/\mathrm{OPT}_f$



(b) β vs. $g(S_\beta)/\mathrm{OPT}_g$



(c) β vs. running time

Figure 2: Movie recommendation results, where S_β denotes the solution of SMSC with β.

the slowest because a $(1-\epsilon)$-approximation oracle for SFM requires more time as the value of ϵ decreases, it still required 200 seconds. Note that *Approx(0.99)*, *Approx(0.8)*, and *Approx(Gr)* called approximation oracles 29, 12, and 8 times, respectively, regardless of the value of β. To sum up, the proposed framework even with the greedy algorithm outperformed the simple application of the greedy algorithm, and *Approx(0.99)* provided more accurate solutions than *Approx(0.8)* and *Approx(Gr)* if β is close to 1, at the expense of efficiency.

5.2 MOVIE RECOMMENDATION

We applied the proposed framework to the movie recommendation task, whose objective is to extract a small number of diverse, highly-rated movies given a log of movie ratings.

Setup. We used the publicly-available MovieLens data `ml-latest-small` [Harper and Konstan, 2015], which contains 100,000 ratings from $u = 600$ users on $m = 9{,}000$ movies, where each rating takes a value in $0.5, 1, 1.5, \dots, 4.5, 5$.

We designed two monotone submodular functions f and g in the following way. Consider the following scenario: One is going to extract a few movies to be displayed in recommender systems but wants to include as many movies as possible from a fixed set of movies, say, T. Such a set T consists of movies an agency wants to advertise, for example. We can apply SMSC to such scenarios where f measures how diverse and highly-rated a set of movies S is and g measures how many movies of T are included in a solution S. We first designed f based on the facility-location objective. Similar submodular objectives have been adopted in the literature, e.g., [Lindgren et al., 2016, Mirzasoleiman et al., 2016, Feldman et al., 2017, Mitrovic et al., 2017, Balkanski and Singer, 2018]. Obtaining the user-movie rating matrix $\mathbf{R} \in \mathbb{R}^{u \times m}$ from the data, we constructed feature vectors for each movie. For this purpose, we performed nonnegative matrix factorization to obtain a factorization $\mathbf{R} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{u \times d}$ consists of user feature vectors, $\mathbf{V} \in \mathbb{R}^{m \times d}$ consists of movie feature vectors, and d is the dimension of feature vectors, where we set $d = 32$. Specifically, we applied a nonnegative matrix factorization algorithm from the Python package `scikit-learn`[3] [Cichocki and Phan, 2009, Févotte and Idier, 2011]. The feature vector of movie i is denoted by $\mathbf{v}_i \in \mathbb{R}^d$. Intuitively, the inner product $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ measures the similarity between movies i and j. Let V be a set of $n = 403$ movies with at least 54 reviews. The facility location objective $f : 2^V \to \mathbb{R}_{\geq 0}$, which quantifies how well a movie set S covers V, is then defined as follows:

$$f(S) \triangleq \sum_{t \in V} \max_{s \in S} \langle \mathbf{v}_s, \mathbf{v}_t \rangle,$$

which is known to be monotone and submodular [Frieze, 1974]. A similarity function $g : 2^V \to \mathbb{R}_{\geq 0}$ is simply defined to be the intersection size; i.e., $g(S) \triangleq |S \cap T|$ for a target set T.

We set the solution size k to 16 and constructed a (less-diverse) set T of size 16 by running the greedy on $-f(\cdot)$.

Results. Figures 2a and 2b plot $f(S_\beta)$ and $g(S_\beta)$ for each value of $\beta = 0.05, 0.1, \dots, 0.95, 1$, where S_β is a solution of SMSC that Algorithm 3 and *Greedy* returned with β. Examining the results of approximation frameworks, *Approx(0.99)* almost always satisfies that $g(S_\beta) \geq \beta\, \mathrm{OPT}_g$ for every β. This is because the function value of g only takes a number in $\{0, 1, 2, \dots, 16\}$. When $\beta \geq 0.9$, *Approx(0.99)* must choose $S_\beta = T$ uniquely as a solution (regardless of the design of f), and thus the function value f

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html`

drops sharply. On the other hand, *Approx(0.8)* and *Approx(Gr)* violate the constraint if $\beta \geq 0.9$. Note that *Greedy(f)* gave a solution S_f such that $f(S_f) = \mathrm{OPT}_f$ and $g(S_f) = 0$ while *Greedy(g)* gave a solution S_g such that $f(S_g) \approx 0.25\,\mathrm{OPT}_f$ and $g(S_g) = \mathrm{OPT}_g$. Hence, *Greedy* produced poor-quality solutions in practice. Figure 2c plots the running time of each algorithm. Similarly to sensor placement, *Greedy* required at most 20 seconds, which was the fastest. *Approx(0.8)* and *Approx(Gr)* required 80 seconds while *Approx(0.99)* required 6 minutes. Note that the number of oracle calls of *Approx(0.99)*, *Approx(0.8)*, and *Approx(Gr)* were 31, 12, and 8, respectively, regardless of the value of β.

6 CONCLUSION

In this paper, we formulated a new optimization problem called submodular maximization under submodular cover (SMSC). We proposed an approximation framework for SMSC with a provable guarantee. We demonstrated the effectiveness of the proposed framework by performing experiments using real-world data. Future work includes better approximation guarantees and hardness-of-approximation results.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

Nima Anari, Nika Haghtalab, Joseph Seffi Naor, Sebastian Pokutta, Mohit Singh, and Alfredo Torrico. Structured robust submodular maximization: Offline and online algorithms. In *AISTATS*, pages 3128–3137, 2019.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

Wenruo Bai, Rishabh Iyer, Kai Wei, and Jeff A. Bilmes. Algorithms for optimizing the ratio of submodular functions. In *ICML*, pages 2751–2759, 2016.

Eric Balkanski and Yaron Singer. Approximation guarantees for adaptive sampling. In *ICML*, pages 384–393, 2018.

Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 771–806. Chapman and Hall/CRC, 2018.

Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6): 1740–1766, 2011.

Wenlin Chen, Yixin Chen, and Kilian Q. Weinberger. Filtered search for submodular maximization with controllable approximation bounds. In *AISTATS*, pages 156–164, 2015.

Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Trans. Fundamentals*, 92(3):708–721, 2009.

Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, pages 758–784, 2017.

Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Comput.*, 23(9):2421–2456, 2011.

Alan M. Frieze. A cost function property for plant location problems. *Math. Program.*, 7(1):245–248, 1974.

Satoru Fujishige. Polymatroidal dependence structure of a set of random variables. *Inf. Control*, 39(1):55–72, 1978.

Zvi Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18(1):23–38, 1986.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2020. URL `http://www.gurobi.com`.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *TiiS*, 5(4):1–19, 2015.

Hiro Ito, Motoyasu Ito, Yuichiro Itatsu, Kazuhiro Nakai, Hideyuki Uehara, and Mitsuo Yokoyama. Source location problems considering vertex-connectivity and edge-connectivity simultaneously. *Networks*, 40(2):63–70, 2002.

Rishabh Iyer and Jeff Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In *UAI*, pages 407–417, 2012.

Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, pages 2436–2444, 2013.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, pages 324–331, 2005.

Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *J. Mach. Learn. Res.*, 9(Dec):2761–2801, 2008a.

Andreas Krause, Ajit Paul Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, 2008b.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *ACL-HLT*, pages 510–520, 2011.

Erik M. Lindgren, Shanshan Wu, and Alexandros G. Dimakis. Leveraging sparsity for efficient submodular data summarization. In *NIPS*, pages 3414–3422, 2016.

S Madden. Intel lab data. `http://db.csail.mit.edu/labdata/labdata.html`, 2014.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, pages 1358–1367, 2016.

Slobodan Mitrovic, Ilija Bogunovic, Ashkan Norouzi-Fard, Jakub Tarnawski, and Volkan Cevher. Streaming robust submodular maximization: A partitioned thresholding approach. In *NIPS*, pages 4557–4566, 2017.

Mukund Narasimhan and Jeff Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *UAI*, pages 404–412, 2005.

George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978.

George L. Nemhauser and Laurence A. Wolsey. Maximizing submodular set functions: Formulations and analysis of algorithms. *North-Holland Mathematics Studies*, 59:279–301, 1981.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Program.*, 14:265–294, 1978.

Naoto Ohsaka and Yuichi Yoshida. Monotone k-submodular function maximization with size constraints. In *NIPS*, pages 694–702, 2015.

Luca Oneto, Sandro Ridella, and Davide Anguita. Tikhonov, ivanov and morozov regularization for support vector machine learning. *Mach. Learn.*, 103(1):103–136, 2016.

Thomas Powers, Jeff Bilmes, Scott Wisdom, David W. Krout, and Les Atlas. Constrained robust submodular optimization. In *NIPS Workshop on Optimization for Machine Learning*, 2016.

Mariko Sakashita, Kazuhisa Makino, and Satoru Fujishige. Minimum cost source location problems with flow requirements. *Algorithmica*, 50(4):555–583, 2008.

Shinsaku Sakaue and Masakazu Ishihata. Accelerated best-first search with upper-bound computation for submodular function maximization. In *AAAI*, pages 1413–1421, 2018.

Dravyansh Sharma, Ashish Kapoor, and Amit Deshpande. On greedy maximization of entropy. In *ICML*, pages 1330–1338, 2015.

M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32:41–43, 2004.

Naoya Uematsu, Shunji Umetani, and Yoshinobu Kawahara. An efficient branch-and-cut algorithm for approximately submodular function maximization. In *SMC*, pages 3160–3167, 2019.