Balancing Utility and Fairness in Submodular Maximization

Yanhao Wang
East China Normal University
Shanghai, China
yhwang@dase.ecnu.edu.cn

Yuchen Li
Singapore Management University
Singapore, Singapore
yuchenli@smu.edu.sg

Francesco Bonchi
CENTAI Institute, Turin, Italy
Eurecat, Barcelona, Spain
bonchi@centai.eu

Ying Wang
East China Normal University
Shanghai, China
yingwang007@stu.ecnu.edu.cn

ABSTRACT

Submodular function maximization is a fundamental combinatorial optimization problem with plenty of applications – including data summarization, influence maximization, and recommendation. In many of these problems, the goal is to find a solution that maximizes the average utility over all users, for each of whom the utility is defined by a monotone submodular function. However, when the population of users is composed of several demographic groups, another critical problem is whether the utility is fairly distributed across different groups. Although the *utility* and *fairness* objectives are both desirable, they might contradict each other, and, to the best of our knowledge, little attention has been paid to optimizing them jointly.

To fill this gap, we propose a new problem called *Bicriteria Submodular Maximization* (BSM) to balance utility and fairness. Specifically, it requires finding a fixed-size solution to maximize the utility function, subject to the value of the fairness function not being below a threshold. Since BSM is inapproximable within any constant factor, we focus on designing efficient instance-dependent approximation schemes. Our algorithmic proposal comprises two methods, with different approximation factors, obtained by converting a BSM instance into other submodular optimization problem instances. Using real-world and synthetic datasets, we showcase applications of our proposed methods in three submodular maximization problems: maximum coverage, influence maximization, and facility location.

1 INTRODUCTION

The awareness that decisions informed by data analytics could have inadvertent discriminatory effects against particular demographic groups has grown substantially over the last few years [29]. Researchers have been studying how to inject *group fairness*, along the lines of *demographic parity* [14] or *equal opportunity* [25], into algorithmic decision-making processes. The bulk of this algorithmic fairness literature has mainly focused on avoiding discrimination against a sensitive attribute (i.e., a protected social group) in supervised machine learning [32], while less attention has been devoted to combinatorial optimization problems such as ranking [2], selection [16], allocation [41], and matching [18, 21].

In this paper, we study how to incorporate group fairness into *submodular maximization* [35] (SM), one of the most fundamental combinatorial optimization problems. Specifically, a set function is submodular if it satisfies the "*diminishing returns*" property, whereby the marginal gain of adding any new item decreases as the set of existing items increases. More formally, a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ defined on a ground set V of items is submodular if $f(S\cup\{v\})-f(S) \geq f(T\cup\{v\})-f(T)$ for any two sets $S \subseteq T \subseteq V$ and item $v \in V \setminus T$. This property occurs in numerous data science applications, such as data summarization [3], influence maximization [33], and recommendation [53].

Submodular maximization with a cardinality constraint is the most widely studied optimization problem on submodular functions. For a monotone submodular function f and a cardinality constraint $k \in \mathbb{Z}^+$, it aims to find a size-k subset $S \subseteq V$ of items such that $f(S)$ is maximized. Although this problem is generally NP-hard [19], the classic greedy algorithm [49] yields a best possible $(1 - 1/e)$-approximation factor.

In many scenarios, the goal of a submodular maximization problem is to find a good solution to serve a large number of users, for each of whom the utility is measured by a monotone submodular function. For such problems, the classic greedy algorithm is effective in maximizing the *average utility* over all users (called the *utility objective*) because it is a non-negative linear combination of monotone submodular functions, which is still monotone and submodular [35]. However, when users are composed of different demographic groups, the utilities should be *fairly distributed* across groups to avoid biased outcomes [34]. A typical choice for the *fairness objective* is to improve the welfare of the least well-off group, as per Rawls's theory of justice [54], which advocates arranging social and financial inequalities so that they are to the greatest benefit of the worst-off. The problem of maximizing the minimum utility among all groups coincides with *robust submodular maximization* [36, 61, 62] (RSM), which is inapproximable within any constant factor in general but admits approximate solutions by violating the cardinality constraints (i.e., providing solutions of sizes larger than k).

Although the two objectives of *utility* and *fairness*, when taken in isolation, can be treated as SM and RSM instances, respectively, optimizing them jointly results to be a much more challenging task, as maximizing one can come at the expense of significantly reducing the other. To our best knowledge, there has been no prior investigation into optimizing both objectives jointly.

To fill this gap, we propose a new problem called *Bicriteria Submodular Maximization* (BSM) to balance utility and fairness. As is often the case when facing two opposite objectives [23, 52, 66], the BSM problem requires finding a fixed-size solution to maximize the utility function subject to that the value of the fairness function is at least a τ-fraction of its optimum for a given factor $\tau \in [0, 1]$.

Our proposal can find application in many real-world submodular maximization problems, such as:

- **Maximum Coverage (MC)** is a well-known NP-hard problem with broad applications in summarization and recommendation [55, 56]. Given a collection of sets over the same ground set, it selects a set of k sets such that their union contains as many elements as possible. When the ground set is a set of individuals divided into different population groups, we also need the solution to cover individuals from every group proportionally [1, 5] for equitable representation. By combining both objectives, we formulate a new problem of maximizing the total coverage of all users and the minimum average coverage for all groups simultaneously.
- **Influence Maximization (IM)** arises naturally in the design of viral marketing campaigns in social networks [33]. The IM problem [33, 38] asks to find a set of k "seed" users in a social network so that, if they are targeted by the campaign, the expected influence spread in the network is maximized. Recent studies [6, 61] introduced group fairness into IM, aimed at balancing the spread of influence among different population groups to reduce information inequality. In such context, BSM corresponds to a new problem of achieving a better trade-off between the influence spread among all users and the influence spread within the least well-off group.
- **Facility Location (FL)** aims at finding a set of items to maximize the total benefits for all users: it finds application in data clustering and location-based services (LBSs) [3, 39, 59]. In submodular facility location problems, the benefit of an item set to a user is defined as the maximum benefit among all items in the set to the user. As an illustrative example, a set of service points deployed in a district can lead to higher benefits for citizens whose residences are closer to their nearest service points. To achieve group-level fairness, it should ensure that each group of users receives approximately the same benefits on average. As such, BSM finds a set of items that maximizes the benefits for all users and the least well-off group to balance utility and fairness in facility location.

Contributions & Roadmap. We first formally define the bicriteria submodular maximization (BSM) problem: Given two functions f, g to measure the *average utility* for all users and the *minimum utility* among all groups, respectively, and a balance factor $\tau \in [0, 1]$, it asks for a size-k set $S \subseteq V$ to maximize $f(S)$, subject to that $g(S) \geq \tau \cdot \mathtt{OPT}_g$, where $\mathtt{OPT}_g = \max_{S' \subseteq V, |S'|=k} g(S')$. We then prove that BSM cannot be approximated within any constant factor for all $\tau > 0$ and $k \geq 1$ (Section 3).

Due to the inapproximability of BSM, we focus on designing efficient instance-dependent approximation schemes for BSM. Specifically, we first propose a two-stage greedy algorithm combining the two greedy algorithms for *submodular maximization* [49] and *submodular cover* [68], for maximizing f and g in turn to obtain a BSM solution (Section 4.1). Then, we devise an improved algorithm with a better approximation ratio at the expense of violating the cardinality constraint by converting a BSM instance into several *submodular cover* [68] instances and running the Saturate algorithm [36] for submodular cover to find a BSM solution (Section 4.2). We also analyze the approximation factors and computational complexities of both algorithms. Moreover, we formulate specific classes of BSM problems, i.e., *maximum coverage* and *facility location*, as integer linear programs (ILPs), thus acquiring their optimal solutions on small instances using any ILP solver (Appendix A). This can help quantify the gap between the optimal and approximate solutions to BSM in practice.

Finally, we conduct extensive experiments to evaluate the proposed algorithms on three submodular maximization problems, namely *maximum coverage*, *influence maximization*, and *facility location*, using real-world and synthetic datasets in comparison to non-trivial baselines, including approximation algorithms for SM [49], RSM [36], and submodular maximization under submodular cover [52] (SMSC). Our experimental results demonstrate that our algorithms offer better trade-offs between the utility and fairness objectives than the competing algorithms and are scalable against large datasets (Section 5).

Our main technical contributions are summarized as follows:

(1) The definition of the bicriteria submodular maximization (BSM) problem to balance utility and fairness in submodular maximization.
(2) The proof of the inapproximability of BSM.
(3) Two instance-dependent approximation algorithms for BSM with theoretical analyses of their approximation factors and complexities as well as problem-specific ILP formulations for BSM.
(4) Comprehensive experimental results to verify the effectiveness and efficiency of our proposed BSM framework and algorithms in real-world applications.

2 RELATED WORK

Before presenting our technical results, we first collocate our contribution to the literature by discussing the related work.

Submodular Maximization (SM). Maximizing a monotone submodular function with a cardinality constraint k is one of the most widely studied problems in combinatorial optimization. A simple greedy algorithm [49] runs in $O(nk)$ time and provides the best possible $(1 - 1/e)$-approximate solution for this problem unless $P = NP$ [19]. Several optimization techniques, e.g., *lazy forward* [37] and *subsampling* [44], were proposed to improve the efficiency of the greedy algorithm. Any of the above algorithms can be used as a subroutine in our algorithmic frameworks for BSM. Furthermore, there have been approximation algorithms for submodular maximization problems with more general constraints beyond cardinality, such as *knapsack* [57], *matroid* [9], *k-system* [12], and more [43]. In addition, submodular maximization problems were also studied in various settings, including the *streaming* [3], *distributed* [46], *dynamic* [48], and *sliding-window* [17, 64, 65] models. However, unlike our BSM problem, all the above problems only consider maximizing a single submodular objective function.

Previous work [15, 63] has also introduced fairness into submodular maximization problems. Specifically, they considered that the items to select represent persons and are partitioned into several demographic groups based on sensitive attributes and proposed efficient approximation algorithms to find a set of items for maximizing a submodular function subject to the constraint that the number of items chosen from each group must fall within the pre-defined lower and upper bounds to achieve an equal (or proportional) group representation in the solution. Nevertheless, such a notion of fairness is distinct from ours. In our BSM problem, the items are assumed to be non-sensitive; thus, no specific restriction is imposed on the attributes of items to pick in the solution. We alternatively focus on picking a fixed-size set of items to distribute utilities fairly across sensitive user groups. Due to the differences in data models and fairness notions, the

algorithms in [15, 63] are not comparable to those in this paper and thus ignored in the experimental evaluation.

Robust Submodular Maximization (RSM). The problem of *robust (or multi-objective) submodular maximization* aims to maximize the minimum of $c > 1$ submodular functions. The RSM problem is inapproximable within any constant factor in polynomial time unless $P = NP$ [36]. The algorithms that provide approximate solutions for RSM by violating the cardinality constraints were proposed in [36, 60]. The *multiplicative weight updates* (MWU) algorithms for RSM, which achieves constant approximation factors when $c = o(k \log^{-3} k)$, were developed in [20, 62]. In this paper, our second algorithm for BSM adopts a similar scheme to that for RSM in [36]. But the analyses are different because an additional objective of maximizing the average utility is considered.

The deletion-robust submodular maximization problem [31, 45], which aims to maintain a "robust" solution when a part of the data may be adversarially deleted, is also identified as RSM in the literature. However, this is a different problem, and the algorithms in [31, 45] do not apply to BSM.

Submodular Optimization with More Than One Objective. There exist several other variants of submodular optimization problems to deal with more than one objective. Iyer and Bilmes [27] studied the problem of minimizing the difference between two submodular functions. Approximation algorithms proposed in [30, 51] aim to maximize the difference between a submodular utility function and a modular cost function. Bai *et al.* [4] considered minimizing the ratio of two submodular functions. Iyer and Bilmes [28] investigated the problem of maximizing a submodular utility function while minimizing a submodular cost function.

The most related problem to BSM is *submodular maximization under submodular cover* [52] (SMSC), which finds a fixed-size solution to maximize one submodular function under the constraint that the value of the other submodular function is not below a given threshold. A similar problem to SMSC was also defined by Gershtein *et al.* [23] in the context of influence maximization. BSM differs from SMSC as the minimum of $c > 1$ submodular functions is not a submodular function, and thus the theoretical results for SMSC [23, 52] do not hold for BSM. Nonetheless, SMSC can be treated as a special case of BSM when $c = 2$, i.e., there are only two groups of users. Thus, our experiments empirically compare SMSC with BSM in the case of $c = 2$. The simultaneous optimization of the average and robust submodular objectives was also considered by Wei *et al.* [66]. This work is different from BSM in two aspects: (1) the objective function is a linear combination of average and robust objective functions, to which an approximate solution might be unbounded for each of them; (2) the algorithms are specific for submodular data partitioning and not directly applicable for submodular maximization.

Balancing Utility and Fairness in ML and Optimization. The problem of balancing utility and fairness has also been widely considered for many machine learning and optimization problems other than submodular optimization. In supervised learning, the utility is typically measured by the difference in accuracy [14, 42] or distribution of model output [10] before and after fairness intervention, and the fairness is defined by *demographic parity* [14] or *equal opportunity* [25] for different groups. The overall goal is thus to improve the fairness metrics at little utility expense. In unsupervised learning, e.g., center-based clustering [24, 40], and optimization, e.g., allocation [41] and matching [18], similar to BSM, the utility is the same as for the original (fairness-unaware) problem, and the fairness is measured by the utilities for different protected groups. Various schemes have also been proposed to balance both objectives in those problems. However, although the above works are similar to our work in formulation, their algorithmic techniques do not apply to BSM, and vice versa, since none of them is submodular.

3 PROBLEM STATEMENT

In this section, we first introduce the basic notions, then give the formal definition of *Bicriteria Submodular Maximization* (BSM), and finally analyze the theoretical hardness of BSM.

For a positive integer n, we use $[n]$ to denote the set of integers $\{1, \ldots, n\}$. Let V be a set of n items indexed by $[n]$ and U be a set of m users indexed by $[m]$. For each user $u \in [m]$, we define a non-negative set function $f_u : 2^V \to \mathbb{R}_{\geq 0}$ to measure the utility of any subset $S \subseteq V$ of items for user u. We assume that f_u is normalized, i.e., $f_u(\emptyset) = 0$, monotone, i.e., $f_u(S) \leq f_u(T)$ for any $S \subseteq T \subseteq V$, and submodular, i.e., $f_u(S \cup \{v\}) - f_u(S) \geq f_u(T \cup \{v\}) - f_u(T)$ for any $S \subseteq T \subseteq V$ and $v \in V \setminus T$. We define the following function to measure the utility of a set S for each user in U on *average*:

$$f(S) := \frac{1}{m} \sum_{u \in [m]} f_u(S) \tag{1}$$

Following existing submodular maximization literature [3, 49], we assume that the value of $f_u(S)$ is given by an oracle in $O(1)$ time. Hence, the time complexity of computing $f(S)$ is $O(m)$.

We consider the case in which the set U of users is partitioned into c (disjoint) demographic groups according to a certain sensitive attribute, e.g., *gender* or *race*, and $U_i \subseteq U$ is the set of users from the i-th group for $i \in [c]$. We want to distribute the utilities across groups fairly to achieve group-level fairness. Specifically, the average utility function of the i-th group is $f_i(S) := \frac{1}{m_i} \sum_{u \in U_i} f_u(S)$, where $m_i = |U_i|$. Then, the widely adopted *maximin fairness* [20–22, 61], which improves the conditions for the *least well-off* group by maximizing the minimum average utility for any of the i groups, is used as the notion of fairness. This leads to the following function for measuring how fairly the utilities are distributed among groups:

$$g(S) := \min_{i \in [c]} f_i(S) = \min_{i \in [c]} \frac{1}{m_i} \sum_{u \in U_i} f_u(S) \tag{2}$$

Towards the balance between *utility* and *fairness*, we should consider optimizing both objective functions jointly: on the one hand, we should maximize f for the effectiveness of the solution; on the other hand, we should maximize g for the group-level fairness. To achieve both goals, we generalize a common framework for bicriteria optimization [23, 52]. Specifically, the first objective, i.e., maximizing f, is considered the primary objective of the new problem. Meanwhile, the second objective, i.e., maximizing g, is modeled as the constraint of the new problem, where a factor $\tau \in [0, 1]$ is used to restrict that the function value of g must be at least τ-fraction of its optimum to seek the balance between both objectives. We define the *Bicriteria Submodular Maximization* (BSM) problem as follows.

Problem 1. *[Bicriteria Submodular Maximization] Given an item set V, a user set U partitioned into c groups $U_1, \ldots, U_c$ with $\bigcup_{i=1}^{c} U_i = U$ and $U_i \cap U_j = \emptyset$ for any $i, j \in [c]$ $(i \neq j)$, two functions f and g in Eqs. 1 and 2, the cardinality constraint $k \in \mathbb{Z}^+$, and the balance factor $\tau \in [0, 1]$, return a set $S^* \subseteq V$ such that*

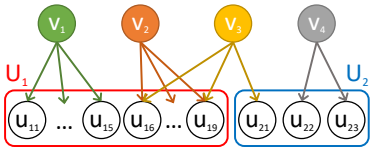


Figure 1: Running example for BSM based on an instance of maximum coverage.

$|S^*| = k$, $f(S^*)$ *is maximized, and* $g(S^*) \geq \tau \cdot \mathtt{OPT}_g$, *where* $\mathtt{OPT}_g = \max_{S' \subseteq V, |S'|=k} g(S')$. *Formally,*

$$\begin{aligned} \max_{S \subseteq V, |S|=k} \quad & f(S) \\ \text{subject to} \quad & g(S) \geq \tau \cdot \mathtt{OPT}_g \end{aligned}$$

Here, S^* is called an α^*-approximate solution if $f(S^*) = \alpha^* \cdot \mathtt{OPT}_f$ where $\mathtt{OPT}_f = \max_{S \subseteq V, |S|=k} f(S)$. Accordingly, α^* is called the best achievable factor for the BSM instance.

Example 3.1. Let us consider the BSM instance for maximum coverage in Figure 1. It consists of four items $V = \{v_1, \ldots, v_4\}$ and a ground set U of twelve users divided into two groups $U_1 = \{u_{11}, \ldots, u_{19}\}$ and $U_2 = \{u_{21}, u_{22}, u_{23}\}$. The sets of users covered by v_1, v_2, v_3, and v_4 are $S(v_1) = \{u_{11}, \ldots, u_{15}\}$, $S(v_2) = \{u_{16}, \ldots, u_{19}\}$, $S(v_3) = \{u_{16}, u_{19}, u_{21}\}$, and $S(v_4) = \{u_{22}, u_{23}\}$.

The maximum coverage problem with a cardinality constraint $k = 2$ aims to select two items to cover the most number of users. Intuitively, it returns $S_{12} = \{v_1, v_2\}$ since $f(S_{12}) = \frac{1}{12} \cdot |S(v_1) \cup S(v_2)| = 0.75$ is the largest among all combinations of size-2 sets in V. The robust maximum coverage problem with the same constraint $k = 2$ aims to maximize the minimum of average coverage between U_1 and U_2. It alternatively returns $S_{14} = \{v_1, v_4\}$ with $\mathtt{OPT}_g = g(S_{14}) = \min_{i \in \{1,2\}} f_i(S_{14}) = \min\{\frac{5}{9}, \frac{2}{3}\} \approx 0.556$. A BSM instance with balance factor $\tau \in [0, 1]$ finds a solution S that maximizes $f(S)$ while ensuring that $g(S) \geq \tau \cdot \mathtt{OPT}_g$. When $\tau = 0$ (i.e., no constraint on g), the same solution S_{12} as the vanilla maximum coverage problem is returned for BSM; when $0 < \tau \leq 0.6$, $S_{13} = \{v_1, v_3\}$ is returned for BSM since $g(S_{13}) = \frac{1}{3} \geq \tau \cdot \mathtt{OPT}_g$ and $f(S_{13}) = 8$ is the maximum among all size-2 sets satisfying the constraint on g; when $0.6 < \tau \leq 1$, the solution for BSM becomes S_{14} since it is the size-2 set that has the largest coverage on U and can satisfy the constraint on g.

We next analyze the hardness of BSM. First, when $\tau = 0$, BSM is equivalent to maximizing f subject to a cardinality constraint k, which is NP-hard and cannot be approximated within a factor better than $1 - 1/e$ (i.e., $\alpha^* \leq 1 - 1/e$) unless $P = NP$ [19]. Furthermore, when $\tau = 1$, we need to compute the optimum $\mathtt{OPT}_g$ for maximizing g with a cardinality constraint k to ensure the satisfaction of the constraint. This is an instance of *robust submodular maximization*, which is generally NP-hard to approximate within any constant factor [36]. Due to the intractability of computing $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$, we consider a bicriteria approximation scheme for BSM. Specifically, a set S is called an (α, β)-approximate solution to a BSM instance for some factors $\alpha, \beta \in (0, 1)$ if $f(S) \geq \alpha \cdot \mathtt{OPT}_f$ and $g(S) \geq \beta\tau \cdot \mathtt{OPT}_g$. Finally, we will show by the following lemma that the objectives of maximizing f and g might contradict each other. A BSM instance with any $\tau > 0$ and $k \geq 1$ can be inapproximable within any constant factors $\alpha, \beta > 0$ even when $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$ are known.

Lemma 3.2. *For any* $\tau > 0$ *and* $k \geq 1$, *there exists a BSM instance without any* (α, β)*-approximate solution for constants* $\alpha, \beta > 0$.

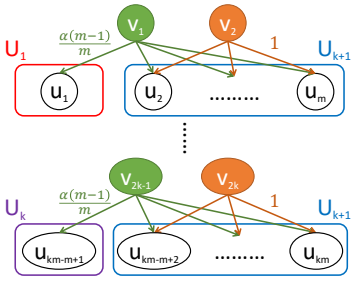


Figure 2: BSM instance for the proof of Lemma 3.2.

Proof. We construct a BSM instance $\mathcal{I}$ that does not have any (α, β)-approximate solution for all constants $\alpha, \beta > 0$ as shown in Figure 2. We first consider the procedure of constructing $\mathcal{I}$ for the case of $k = 1$. We have an item set $V = \{v_1, v_2\}$ and a user set $U = \{u_1, u_2, \ldots, u_m\}$ with two groups $U_1 = \{u_1\}$ and $U_2 = \{u_2, \ldots, u_m\}$. Then, the utility function of u_1 is defined as

$$f_{u_1}(S) = \begin{cases} \frac{\alpha(m-1)}{m}, \text{ if } v_1 \in S \\ 0, \text{ otherwise} \end{cases}$$

and the utility function of u_j for each $j > 1$ is defined as

$$f_{u_j}(S) = \begin{cases} 1, \text{ if } v_2 \in S \\ \frac{\alpha(m-1)}{m}, \text{ if } v_1 \in S \text{ and } v_2 \notin S \\ 0, \text{ otherwise.} \end{cases}$$

Obviously, f_u is monotone and submodular for each $u \in U$. To maximize f on $\mathcal{I}$ with $k = 1$, $S_2 = \{v_2\}$ is the optimal solution and $\mathtt{OPT}_f = \frac{1}{m} \sum_{u \in [m]} f_u(S_2) = \frac{m-1}{m}$. To maximize g on $\mathcal{I}$ with $k = 1$, $S_1 = \{v_1\}$ is the optimal solution and $\mathtt{OPT}_g = f_1(S_1) = \frac{\alpha(m-1)}{m}$. However, as $g(S_2) = f_1(S_2) = 0$, the approximation factor β of S_2 on g for any $\tau > 0$ is 0 and thus S_2 is a $(1, 0)$-approximate solution. In addition, $f(S_1) = \frac{\alpha(m-1)}{m} = \alpha \cdot \mathtt{OPT}_f$. Therefore, S_1 is a $(\alpha, 1)$-approximate solution and the best achievable factor α^* of $\mathcal{I}$ is equal to α. Since the value of α can be arbitrarily small, e.g., $\alpha \leq \frac{1}{m}$, there does not exist any (α, β)-approximate solution to $\mathcal{I}$ for constants $\alpha, \beta > 0$ when $k = 1$.

We further generalize the above construction procedure to the case of $k > 1$. Specifically, we replicate the above constructed instance k times, denoted as $\mathcal{I}_1, \ldots, \mathcal{I}_k$. For each $\mathcal{I}_i$ where $i \in [k]$, we have an item set $\{v_{2i-1}, v_{2i}\}$ and a user set $\{u_{im-m+1}, \ldots, u_{im}\}$. Then, we assign the first user u_{im-m+1} to group U_i and all remaining users to group U_{k+1}. The utility functions of all users are defined in the same way as in $\mathcal{I}$. For the new instance $\mathcal{I}'$, the sets of items and users are the unions of the sets of items and users for $\mathcal{I}_1, \ldots, \mathcal{I}_k$. Then, the optimal solution to maximizing f with a cardinality constraint k on $\mathcal{I}'$ is $S_{2i} = \{v_{2i} : i \in [k]\}$, and the optimal solution to maximizing g with a cardinality constraint k on $\mathcal{I}'$ is $S_{2i-1} = \{v_{2i-1} : i \in [k]\}$. We can see that, for every subset $S \subseteq V'$ with $|S| \leq k$ except S_{2i-1}, there must exist some $i \in [k]$ such that $g(S) = f_i(S) = 0$. Hence, their approximation ratios on g all equal 0. Moreover, the approximation ratio of S_{2i-1} on f is $\frac{f(S_{2i-1})}{f(S_{2i})} = \alpha$. Therefore, S_{2i-1} is a $(\alpha, 1)$-approximate solution and the best achievable factor α^* of $\mathcal{I}'$ is also equal to α. Accordingly, the inapproximability result holds for arbitrary $k \geq 1$. □

Given the above inapproximability result, we will focus on instance-dependent bicriteria approximation schemes for BSM in the subsequent section.

Algorithm 1 BSM-TSGreedy

Input: Two functions $f, g : 2^V \to \mathbb{R}_{\geq 0}$, balance factor $\tau \in (0, 1)$, solution size $k \in \mathbb{Z}^+$
Output: Solution S' with $|S'| = k$ for BSM

1: Run Greedy [49] on f to compute S_f and $\mathtt{OPT}'_f$
2: Run Saturate [36] on g to compute S_g and $\mathtt{OPT}'_g$
3: Initialize $S' \leftarrow \emptyset$
4: Define $g'_\tau(S) := \frac{1}{c}\sum_{i\in[c]} \min\left\{1, \frac{f_i(S)}{\tau\cdot \mathtt{OPT}'_g}\right\}$
5: **while** $g'_\tau(S') < 1$ and $|S'| < k$ **do**
6: $\quad v^* \leftarrow \arg\max_{v\in V} g'_\tau(S' \cup \{v\}) - g'_\tau(S')$
7: $\quad S' \leftarrow S' \cup \{v^*\}$
8: **if** $|S'| = k$ and $g'_\tau(S') < 1$ **then**
9: $\quad S' \leftarrow S_g$
10: **else if** $|S'| < k$ **then**
11: $\quad$ **for** $l \leftarrow 1, \ldots, k$ **do**
12: $\quad\quad$ Let $v_{f,l}$ be the l-th item added to S_f by Greedy
13: $\quad\quad S' \leftarrow S' \cup \{v_{f,l}\}$
14: $\quad\quad$ **if** $|S'| = k$ **then**
15: $\quad\quad\quad$ **break**
16: **return** S'

4 ALGORITHMS

In this section, we propose our algorithms for BSM. Our first algorithm is a two-stage greedy algorithm that combines the two greedy algorithms for *submodular maximization* [49] and *submodular cover* [68] (Section 4.1). Our second algorithm converts BSM into the *submodular cover* [68] problem and adapts the Saturate algorithm [36] to obtain a solution to BSM (Section 4.2). We also analyze the approximation factors and complexities of both algorithms.

4.1 The Two-Stage Greedy Algorithm for BSM

By definition, BSM can be divided into two sub-problems, i.e., computing a solution S that (1) maximizes $f(S)$ and (2) satisfies the constraint $g(S) \geq \tau \cdot \mathtt{OPT}_g$. Therefore, an intuitive scheme to solve BSM is first to find a solution for each sub-problem independently and then to combine both solutions as the BSM solution. The above scheme is feasible in practice because the prior and latter problems are instances of *submodular maximization* and *submodular cover*, respectively, on which Greedy algorithms [49, 68] can provide approximate solutions, despite of their NP-hardness [19]. Accordingly, we propose a two-stage greedy algorithm called BSM-TSGreedy, which first runs the greedy submodular cover algorithm [68] to obtain an initial solution based on the constraint on g and then uses the greedy submodular maximization algorithm [49] to add items to the initial solution for maximizing f. In this way, the final solution after two stages can not only approximately satisfy the constraint on g due to the items in the first stage but also maximizes f with an approximation factor depending on the number of items added in the second stage, thus achieving an instance-dependent bicriteria approximation as described in Section 3.

In particular, the procedure of BSM-TSGreedy is presented in Algorithm 1. First of all, it calls Greedy [49] for SM and Saturate [36] for RSM separately to obtain the approximations $\mathtt{OPT}'_f$ and $\mathtt{OPT}'_g$ for $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$, respectively. Then, in the first stage, a function $g'_\tau(S) := \frac{1}{c}\sum_{i\in[c]} \min\left\{1, \frac{f_i(S)}{\tau\cdot \mathtt{OPT}'_g}\right\}$ is defined based on τ and $\mathtt{OPT}'_g$. It holds that g'_τ is monotone and submodular because (1) $\overline{f}(t, S) := \min\{t, f(S)\}$ is monotone and submodular for any constant $t \geq 0$ and monotone submodular function f [35] and (2) any nonnegative linear combination of monotone submodular functions is still monotone and submodular [35]. Starting from $S' = \emptyset$, the item to maximize g'_τ is greedily added to S' until $g'_\tau(S') = 1$, i.e., $g(S') \geq \tau \cdot \mathtt{OPT}'_g$ and the constraint on g is (approximately) satisfied, or $|S'| = k$. After that, if $g'_\tau(S') < 1$, the partial solution S' will be replaced with S_g. The purpose of this step is to guarantee that a size-k solution satisfying $g'_\tau(S') = 1$ is provided, as $g'_\tau(S_g) = 1$ always holds from the definition of g'_τ, in the case that the Greedy algorithm fails to find a size-k solution S' with $g(S') \geq \tau \cdot \mathtt{OPT}'_g$. Then, it executes the second stage by further adding the items in the greedy solution S_f for maximizing f to S' until $|S'| = k$ to increase the value of $f(S')$ as largely as possible. Finally, the solution S' after two stages is returned for BSM.

Example 4.1. We consider how BSM-TSGreedy works on the BSM instance with $k = 2$ in Figure 1. As shown in Example 3.1, it first runs Greedy and Saturate to compute $S_f = \{v_1, v_2\}$ with $\mathtt{OPT}'_f = 0.75$ and $S_g = \{v_1, v_4\}$ with $\mathtt{OPT}'_g = 5/9 \approx 0.556$. Accordingly, $g'_\tau(S) := \frac{1}{2}\big(\min\{1, \frac{9f_1(S)}{5\tau}\} + \min\{1, \frac{9f_2(S)}{5\tau}\}\big)$. When $\tau = 0.2$, it adds v_3 to S' in the first stage since $g'_{0.2}(\{v_3\}) = 1$, then includes v_1 into S' in the second stage because v_1 is the first item in S_f, and finally returns $S' = \{v_1, v_3\}$ for BSM. When $\tau = 0.5$, it also first adds v_3 to S' since $g'_{0.5}(\{v_3\}) = 0.9$ is the largest among all four items. Then, either v_1 or v_2 can be added to S' because $g'_{0.5}(\{v_1, v_3\}) = g'_{0.5}(\{v_2, v_3\}) = 1$. Assuming that v_2 is added, it will return $S' = \{v_2, v_3\}$ after the first stage, which is inferior to S_{13} since $f(S') = 5 < f(S_{13}) = 8$, though they both satisfy the constraint on g. When $\tau = 0.8$, it still first adds v_3 to S' with $g'_{0.8}(\{v_3\}) = 0.625$. Then, $g'_{0.8}(S') < 1$ and $|S'| = 2$ after any remaining item is added to S'. Thus, it returns $S' = S_g = \{v_1, v_4\}$ according to Line 8 of Algorithm 1.

Subsequently, we analyze the approximation factor and time complexity of BSM-TSGreedy in the following theorem.

Theorem 4.2. *Algorithm 1 returns a $\big(1 - \exp(-\frac{k'}{k}), 1 - \varepsilon_g\big)$-approximate solution S' with $|S'| = k$ for an instance of BSM in $O\big(nmk\log(cm)\big)$ time, where k' is the number of iterations in the second stage, $\varepsilon_g \leq 1 - \frac{1}{m} - \frac{\widehat{\mathtt{OPT}_g}}{\mathtt{OPT}_g}$, and $\widehat{\mathtt{OPT}_g}$ is the optimum of maximizing g with a cardinality constraint $O\big(k\log^{-1}(cm)\big)$.*

Proof. First of all, Algorithm 1 either finds a solution S' such that $g'_\tau(S') = 1$ and $g(S') \geq \tau \cdot \mathtt{OPT}'_g$ after the first stage, or replaces S' with S_g, which always satisfies $g(S_g) = \mathtt{OPT}'_g \geq \tau \cdot \mathtt{OPT}'_g$. According to the analysis for Saturate in [50, Thm. 8], we have $\mathtt{OPT}'_g \in [(1-\theta)\cdot\widehat{\mathtt{OPT}_g}, \mathtt{OPT}_g]$ where $\widehat{\mathtt{OPT}_g}$ is the optimum of maximizing g with a cardinality constraint $O\big(k\log^{-1}(\frac{c}{\theta})\big)$. By setting $\theta = \frac{1}{m}$, we obtain that $\varepsilon_g \leq 1 - \frac{1}{m} - \frac{\widehat{\mathtt{OPT}_g}}{\mathtt{OPT}_g}$ and the approximation factor holds for a cardinality constraint $O\big(k\log^{-1}(cm)\big)$. In the second stage, since the first k' items of S_f are added to S', by applying the approximation analysis in [49] for k' instead of k, it holds that $f(S') \geq (1 - \exp(-\frac{k'}{k}))\cdot \mathtt{OPT}_f$. Therefore, S' is a $\big(1 - \exp(-\frac{k'}{k}), 1 - \varepsilon_g\big)$-approximate solution of size k for BSM.

Moreover, Greedy and Saturate run in $O(nmk)$ and $O\big(nmk\log(cm)\big)$ time, respectively. The time complexity of Greedy for maximizing g'_τ is $O(nmk)$ since it always terminates after k iterations no matter whether $g'_\tau(S') = 1$. Then, adding the first k' items of S_f to S' takes $O(k)$ time. Thus, the time complexity of Algorithm 1 is $O\big(nmk\log(cm)\big)$. □

4.2 The Saturate Algorithm for BSM

The BSM-TSGreedy algorithm has two limitations. First, its approximation factor might be arbitrarily bad (e.g., dropping to 0 when $k' = 0$). Also, how close its approximation factor is to the best achievable factor α^* is unavailable. Second, as will be shown in Section 5, it suffers from significant losses in solution quality when the number of items added to S' in the first stage for ensuring $g'_\tau(S') = 1$ is equal or close to k. To achieve better performance for BSM, we alternatively consider applying a Lagrangian-like formulation similar to that for SMSC [52], which combines the maximization of f and the satisfaction of the constraint on g into a single problem, to convert BSM into *submodular cover* [68] instances. Then, we run the Saturate algorithm [36] for submodular cover to provide a BSM solution that not only achieves a bicriteria approximation guarantee theoretically but also strikes a good balance between f and g empirically.

Next, we present the detailed conversion procedure from BSM to submodular cover. Let us first consider the decision version of BSM defined as follows.

Definition 4.3 (BSM Decision). For any approximation factor $\alpha \in (0, 1)$, determine if a set $S \subseteq V$ with $|S| = k$ such that $f(S) \geq \alpha \cdot \mathtt{OPT}_f$ and $g(S) \geq \tau \cdot \mathtt{OPT}_g$ exists.

If the answer to the BSM Decision problem in Definition 4.3 is *yes*, then there must exist an α-approximate solution to the BSM instance and vice versa. Assuming that $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$ are known, the BSM decision problem for a given $\alpha \in (0, 1)$ can be divided into two sub-problems: *(i)* is there a size-k set $S \subseteq V$ such that $f_\alpha(S) := \frac{f(S)}{\alpha \cdot \mathtt{OPT}_f} \geq 1$? and *(ii)* is there a size-k set $S \subseteq V$ such that $g_\tau(S) := \frac{g(S)}{\tau \cdot \mathtt{OPT}_g} \geq 1$? The BSM decision problem is thus transformed to decide whether the objective value of the following problem equals 2:

$$\max_{S \subseteq V, |S|=k} \min\{1, f_\alpha(S)\} + \min\{1, g_\tau(S)\}. \tag{3}$$

Then, the problem of Eq. 3 is reduced to *submodular maximization* according to the truncation method as used in [36]:

$$\max_{S \subseteq V, |S|=k} F_\alpha(S) := \min\left\{1, \frac{f(S)}{\alpha \mathtt{OPT}_f}\right\} + \frac{1}{c} \sum_{i \in [c]} \min\left\{1, \frac{f_i(S)}{\tau \mathtt{OPT}_g}\right\}. \tag{4}$$

Intuitively, F_α in Eq. 4 is monotone and submodular because it is a nonnegative linear combination of monotone submodular functions. Since computing the optimums $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$ is NP-hard, we further consider replacing them with approximate values $\mathtt{OPT}'_f$ and $\mathtt{OPT}'_g$, i.e., $\mathtt{OPT}'_f \in [(1-\varepsilon_f) \cdot \mathtt{OPT}_f, \mathtt{OPT}_f]$ and $\mathtt{OPT}'_g \in [(1-\varepsilon_g) \cdot \mathtt{OPT}_g, \mathtt{OPT}_g]$, where ε_f and ε_g are the relative errors for approximating $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$, respectively. The following lemma indicates that the BSM decision problem will still be answered with a theoretical guarantee by solving the approximate version of the problem in Eq. 4.

Lemma 4.4. *Let* $F'_\alpha(S) := \min\left\{1, \frac{f(S)}{\alpha \cdot \mathtt{OPT}'_f}\right\} + \frac{1}{c} \sum_{i \in [c]} \min\left\{1, \frac{f_i(S)}{\tau \cdot \mathtt{OPT}'_g}\right\}$. *On the one hand, any set* $\widehat{S}$ *with* $F'_\alpha(\widehat{S}) \geq 2(1 - \frac{\varepsilon}{c})$ *is an* $(\hat{\alpha}, \hat{\beta})$*-approximate solution to BSM, where* $\hat{\alpha} = (1 - 2\varepsilon - \varepsilon_f)\alpha$ *and* $\hat{\beta} = 1 - 2\varepsilon - \varepsilon_g$. *On the other hand, there is not any* α*-approximate solution to BSM if* $F'_\alpha(S) < 2$ *for any size-*k *set* S.

Proof. On the one hand, if $F'_\alpha(\widehat{S}) \geq 2(1 - \frac{\varepsilon}{c})$, then it will hold that $\frac{f(\widehat{S})}{\alpha \cdot \mathtt{OPT}'_f} \geq 1 - \frac{2\varepsilon}{c}$ and $\frac{1}{c} \sum_{i \in [c]} \frac{f_i(\widehat{S})}{\tau \cdot \mathtt{OPT}'_g} \geq 1 - \frac{2\varepsilon}{c}$. According to the prior inequality, we have

$$\begin{aligned} f(\widehat{S}) &\geq \alpha(1 - \frac{2\varepsilon}{c}) \cdot \mathtt{OPT}'_f \\ &\geq \alpha(1 - 2\varepsilon)(1 - \varepsilon_f) \cdot \mathtt{OPT}_f \\ &> (1 - 2\varepsilon - \varepsilon_f)\alpha \cdot \mathtt{OPT}_f \end{aligned}$$

According to the latter inequality, for each $i \in [c]$,

$$\begin{aligned} f_i(\widehat{S}) &\geq \tau\big((1 - \frac{2\varepsilon}{c})c - (c-1)\big) \cdot \mathtt{OPT}'_g \\ &> \tau(1 - 2\varepsilon)(1 - \varepsilon_g) \cdot \mathtt{OPT}_g \\ &> (1 - 2\varepsilon - \varepsilon_g)\tau \cdot \mathtt{OPT}_g \end{aligned}$$

Thus, $g(\widehat{S}) = \min_{i \in [c]} f_i(\widehat{S}) > (1-2\varepsilon-\varepsilon_g)\tau \cdot \mathtt{OPT}_g$. We prove that $\widehat{S}$ is an $(\hat{\alpha}, \hat{\beta})$-approximate solution to BSM, where $\hat{\alpha} = (1-2\varepsilon-\varepsilon_f)\alpha$ and $\hat{\beta} = 1 - 2\varepsilon - \varepsilon_g$. On the other hand, if $F'_\alpha(S) < 2$ for any size-k set $S \subseteq V$, then either $f(S) < \alpha \cdot \mathtt{OPT}'_f$ or there exists some $i \in [c]$ such that $f_i(S) < \tau \cdot \mathtt{OPT}'_g$ and thus $g(S) < \tau \cdot \mathtt{OPT}'_g$. In either case, it must hold that $f(S) < \alpha \cdot \mathtt{OPT}'_f < \alpha \cdot \mathtt{OPT}_f$ or $g(S) < \tau \cdot \mathtt{OPT}'_g < \tau \cdot \mathtt{OPT}_g$. Therefore, S must not be an α-approximate solution to BSM. □

Algorithm 2 BSM-Saturate

Input: Two functions $f, g : 2^V \to \mathbb{R}_{\geq 0}$, balance factor $\tau \in (0, 1)$, solution size $k \in \mathbb{Z}^+$, error parameter $\varepsilon \in (0, 1)$
Output: Solution $\widehat{S}$ with $|\widehat{S}| \leq k \ln \frac{c}{\varepsilon}$ for BSM

```
Run Greedy [49] on f to compute OPT'_f
Run Saturate [36] on g to compute OPT'_g
α_max ← 1 and α_min ← 0
while (1 − ε)α_max > α_min do
    Set α ← (α_max + α_min)/2
    Define F'_α := min{1, f(S)/(α·OPT'_f)} + (1/c) Σ_{i∈[c]} min{1, f_i(S)/(τ·OPT'_g)}
    Initialize S ← ∅
    for i ← 1, ..., k ln(c/ε) do
        v* ← arg max_{v∈V} F'_α(S ∪ {v}) − F'_α(S)
        S ← S ∪ {v*}
    if F'_α(S) ≥ 2(1 − ε/c) then
        α_min ← α and Ŝ ← S
    else
        α_max ← α
return Ŝ
```

Based on Lemma 4.4, we complete the conversion by connecting a BSM decision problem with a submodular cover problem on F'_α. Accordingly, we propose BSM-Saturate in Algorithm 2 by solving submodular cover instances on F'_α with different α's to find an appropriate α value and thus provide a good BSM solution. First, it also utilizes Greedy [49] for SM and Saturate [36] for RSM to compute the approximate values $\mathtt{OPT}'_f$ and $\mathtt{OPT}'_g$ for $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$, respectively. Next, it performs a bisection search on α within $[0, 1]$. For each value of α, it runs Greedy [49] to maximize the submodular function F'_α defined in Lemma 4.4 with size constraint $k \ln \frac{c}{\varepsilon}$. If the function value $F'_\alpha(S)$ of the greedy solution S reaches $2(1 - \frac{\varepsilon}{c})$, it will set S as the current solution $\widehat{S}$ and search on the upper half to find a better solution. Otherwise, the search will be performed on the lower half to find a feasible solution. Finally, it terminates the bisection search when the ratio between the upper and lower bounds of α is within $1 - \varepsilon$ and returns the solution $\widehat{S}$ obtained for the lower bound of α at the last iteration for BSM.

Subsequently, we analyze the approximation factor and time complexity of BSM-Saturate in the following theorem.

Theorem 4.5. *Algorithm 2 returns a* $\big((1-3\varepsilon-\varepsilon_f)\alpha^*, 1-2\varepsilon-\varepsilon_g\big)$*-approximate solution* $\widehat{S}$ *with* $|\widehat{S}| \leq k \ln \frac{c}{\varepsilon}$ *for an instance of BSM in* $O\big(nmk \log^2(\frac{cm}{\alpha^*\varepsilon})\big)$ *time, where* α^* *is the best possible approximation factor of the BSM instance,* $\varepsilon_f \leq \frac{1}{e}$, $\varepsilon_g \leq 1 - \frac{1}{m} - \frac{\widehat{\mathsf{OPT}_g}}{\mathsf{OPT}_g}$*, and* $\widehat{\mathsf{OPT}_g}$ *is the optimum of maximizing* g *with a cardinality constraint* $O\big(k \log^{-1}(cm)\big)$.

Proof. First, since the approximation factor of Greedy [49] for SM is $1-1/e$, we have $\mathsf{OPT}'_f \in [(1-1/e)\cdot\mathsf{OPT}_f, \mathsf{OPT}_f]$ and thus $\varepsilon_f \leq \frac{1}{e}$. Second, according to the analysis of Saturate in [50], we have $\mathsf{OPT}'_g \in [(1-\frac{1}{m})\cdot\widehat{\mathsf{OPT}_g}, \mathsf{OPT}_g]$ where $\widehat{\mathsf{OPT}_g}$ is the optimum of maximizing g with a cardinality constraint $O\big(k \log^{-1}(cm)\big)$. And thus, $\varepsilon_g \leq 1 - \frac{1}{m} - \frac{\widehat{\mathsf{OPT}_g}}{\mathsf{OPT}_g}$. For the lower bound α_{min} of α when the bisection search in Algorithm 2 is terminated, as it holds that $F'_{\alpha_{min}}(\widehat{S}) \geq 2(1-\frac{\varepsilon}{c})$, we have $\widehat{S}$ is an $\big((1-2\varepsilon-\varepsilon_f)\alpha_{min}, 1-2\varepsilon-\varepsilon_g\big)$-approximate solution to BSM according to Lemma 4.4. Furthermore, for the upper bound α_{max} of α when the bisection search in Algorithm 2 is terminated, it holds that $F'_{\alpha_{max}}(S_{\alpha_{max}}) < 2(1-\frac{\varepsilon}{c}) < 2$, where $S_{\alpha_{max}}$ is the greedy solution of size $k \ln \frac{c}{\varepsilon}$ for maximizing $F'_{\alpha_{max}}$. By generalizing the analysis of Greedy in [49], for any monotone submodular function F, the greedy solution S_l after l iterations satisfies that $F(S_l) \geq \big(1-\exp(-\frac{l}{k})\big)\cdot\mathsf{OPT}_F$, where OPT_F is the optimum of maximizing F with a size constraint k. Taking the above inequality into $F'_{\alpha_{max}}$ and $S_{\alpha_{max}}$, we have

$$F'_{\alpha_{max}}(S_{\alpha_{max}}) \geq \Big(1-\exp(-\frac{k\ln\frac{c}{\varepsilon}}{k})\Big)\cdot\mathsf{OPT}_{F'} \geq (1-\frac{\varepsilon}{c})\cdot\mathsf{OPT}_{F'}.$$

Therefore, we have $\mathsf{OPT}_{F'} < 2$, i.e., $F'_{\alpha_{max}}(S) < 2$ for any size-k set $S \subseteq V$. Then, we can safely say there is no α_{max}-approximate solution to the BSM instance and $\alpha^* \leq \alpha_{max}$. Considering all the above results, we prove that $\widehat{S}$ returned by Algorithm 2 is an $\big((1-3\varepsilon-\varepsilon_f)\alpha^*, 1-2\varepsilon-\varepsilon_g\big)$-approximate solution of size at most $k \ln \frac{c}{\varepsilon}$ to any BSM instance.

In terms of complexity, Greedy and Saturate take $O(nmk)$ and $O\big(nmk \log(cm)\big)$ time, respectively. Moreover, the bisection search attempts $O\big(\log(\frac{1}{\alpha^*\varepsilon})\big)$ different α's before termination. For each value of α, it takes $O(nmk \log \frac{c}{\varepsilon})$ time to compute a solution S. Therefore, the time complexity of Algorithm 2 is $O\big(nmk \log^2(\frac{cm}{\alpha^*\varepsilon})\big)$. □

Theorem 4.5 indicates that BSM-Saturate improves upon BSM-TSGreedy with better theoretical guarantees: its approximation factor is close to the best achievable α^* when the error terms are ignored. However, such an improvement comes at the expense of providing solutions of sizes greater than k. In practice, to ensure that the solution size is at most k, we substitute $k \ln \frac{c}{\varepsilon}$ in Line 8 of Algorithm 2 with k, while keeping the remaining steps unchanged.

Example 4.6. We consider running BSM-Saturate on the BSM instance with $k = 2$ in Figure 1. Here, we set $\varepsilon = 0.1$ and replace $k \ln \frac{c}{\varepsilon} = 2 \ln 20$ in Line 8 with $k = 2$ to ensure that the size of $\widehat{S}$ is 2. As shown in Example 3.1, it first runs Greedy and Saturate to compute $S_f = \{v_1, v_2\}$ with $\mathsf{OPT}'_f = 0.75$ and $S_g = \{v_1, v_4\}$ with $\mathsf{OPT}'_g \approx 0.556$. For a given $\tau \in [0, 1]$, it maximizes F'_α in Lemma 4.4 for different α's using the greedy algorithm until $\frac{\alpha_{min}}{\alpha_{max}} > 0.9$ and returns the solution w.r.t. α_{min} for BSM. When $\tau = 0.2$ and 0.5, it attempts to maximize F'_α for $\alpha = 0.5$, 0.75, 0.875, and 0.9375 one by one. For each α value, it adds v_3 and v_1 into S and gets $F'_\alpha(\{v_1, v_3\}) > 1.9$. Thus, it terminates with $\alpha_{min} = 0.9375$, $\alpha_{max} = 1$, and $F'_{0.9375}(\{v_1, v_3\}) \approx 0.95 + 1 > 1.9$, and returns $\widehat{S} = \{v_1, v_3\}$ for BSM. When $\tau = 0.8$, it also attempts to maximize F'_α for $\alpha = 0.5$, 0.75, and 0.875. However, it obtains $F'_{0.875}(\{v_1, v_3\}) = 1.875 < 1.9$ for $\alpha = 0.875$. Next, it sets $\alpha = 0.8125$ and gets $F'_{0.8125}(\{v_1, v_4\}) \approx 0.96 + 1 > 1.9$. Thus, it finishes with $\alpha_{min} = 0.8125$, $\alpha_{max} = 0.875$, and returns $\widehat{S} = \{v_1, v_4\}$ for BSM.

5 EXPERIMENTAL EVALUATION

In this section, we evaluate our optimization framework, i.e., BSM, and algorithms, i.e., BSM-TSGreedy and BSM-Saturate, by extensive experiments on three problems, namely *maximum coverage*, *influence maximization*, and *facility location*, using real-world and synthetic datasets. The goal of the experiments is to answer the following questions:

- **Q1:** How does the factor τ affect the balance between the values of two objective functions f and g?
- **Q2:** How far are the solutions produced by the proposed approximation algorithms from optimal in practice?
- **Q3:** How effective and efficient are the proposed algorithms?
- **Q4:** Are the proposed algorithms scalable to large data?

Algorithms and Baselines. The following algorithms and baselines are compared in the experiments.

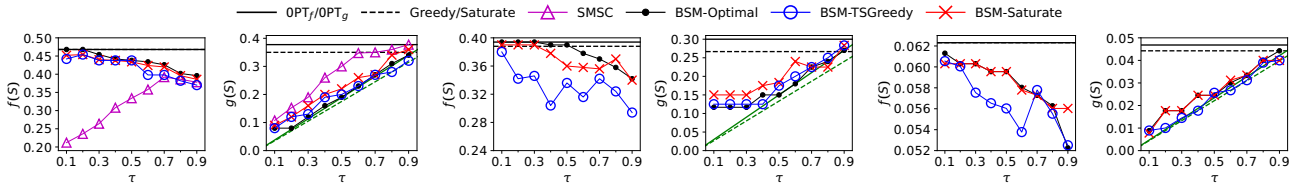
- Greedy [49]: The classic $(1-1/e)$-approximation greedy algorithm for SM, which is also used as a subroutine to maximize f in our algorithms.
- Saturate [36]: The bicriteria approximation algorithm for RSM, which is also used as a subroutine to maximize g in our algorithms.
- SMSC [52]: The $(0.16, 0.16)$-approximation algorithm for submodular maximization under submodular cover, which can be used for BSM only when $c = 2$ by maximizing two submodular functions f_1 and f_2 simultaneously.
- BSM-Optimal: The exponential-time exact algorithm for BSM. Since the *maximum coverage* and *facility location* problems can be formulated as integer linear programs (ILPs) [67], we find the optimal solutions of small BSM instances using an ILP solver (see Appendix A). We aim to measure the gap between the optimal and approximate solutions to BSM.
- BSM-TSGreedy: Our first instance-dependent bicriteria approximation algorithm for BSM (Algorithm 1).
- BSM-Saturate: Our second instance-dependent bicriteria approximation algorithm for BSM (Algorithm 2), which is adapted to provide solutions of size at most k for a fair comparison. In preliminary experiments, we observe that the value of ε hardly affects the performance of BSM-Saturate unless $\varepsilon \geq 0.5$ (see Appendix B). To guarantee the good performance of BSM-Saturate in all cases, we set $\varepsilon = 0.05$ throughout the experiments in this section.

We implemented all the above algorithms in Python 3. We used the *lazy-forward* strategy [37] to accelerate all algorithms except BSM-Optimal (for which the *lazy-forward* strategy is not applicable). The Gurobi optimizer[1] was applied to solve ILPs in BSM-Optimal. All experiments were run on a server with an Intel Xeon E5-2650v4 2.2GHz processor and 96GB memory running Ubuntu 18.04 LTS. Our code and data are published on https://github.com/yhwang1990/code-bsm-release.

[1] https://www.gurobi.com/products/gurobi-optimizer/

Table 1: Statistics of datasets in the MC and IM experiments.

Dataset	n (and m)	$\|E\|$	Percentage of users from each group
RAND ($c = 2$)	500 or 100	8,946 or 360	[‘U_0’: 20%, ‘U_1’: 80%]
RAND ($c = 4$)	500 or 100	6,655 or 257	[‘U_0’: 8%, ‘U_1’: 12%, ‘U_2’: 20%, ‘U_3’: 60%]
Facebook (Age, $c = 2$)	1,216	42,443	[‘< 20’: 8%, ‘≥ 20’: 92%]
Facebook (Age, $c = 4$)	1,216	42,443	[‘19’: 8%, ‘20’: 28%, ‘21’: 31%, ‘22’: 33%]
DBLP (Continent, $c = 5$)	3,980	6,966	[‘Asia’: 21%, ‘Europe’: 23%, ‘North America’: 52%, ‘Oceania’: 3%, ‘South America’: 1%]
Pokec (Gender, $c = 2$)	1,632,803	30,622,564	[‘Female’: 51%, ‘Male’: 49%]
Pokec (Age, $c = 6$)	1,632,803	30,622,564	[‘0–20’: 17%, ‘21–30’: 45%, ‘31–40’: 29%, ‘41–50’: 6%, ‘51–60’: 2%, ‘60+’: 1%]



(a) RAND ($c = 2, k = 5$) **(b) RAND ($c = 4, k = 5$)** **(c) DBLP ($c = 5, k = 10$)**

Figure 3: Results of different algorithms for maximum coverage by varying the factor τ on two random graphs and DBLP. The green straight lines in the plots for $g(S)$ denote $\tau \cdot \mathtt{OPT}_g$ (solid) and $\tau \cdot \mathtt{OPT}'_g$ (dashed), where $\mathtt{OPT}_g$ and $\mathtt{OPT}'_g$ are computed by ILP and Saturate, to show whether the constraints $g(S) \geq \tau \cdot \mathtt{OPT}_g$ and $g(S) \geq \tau \cdot \mathtt{OPT}'_g$ are satisfied.

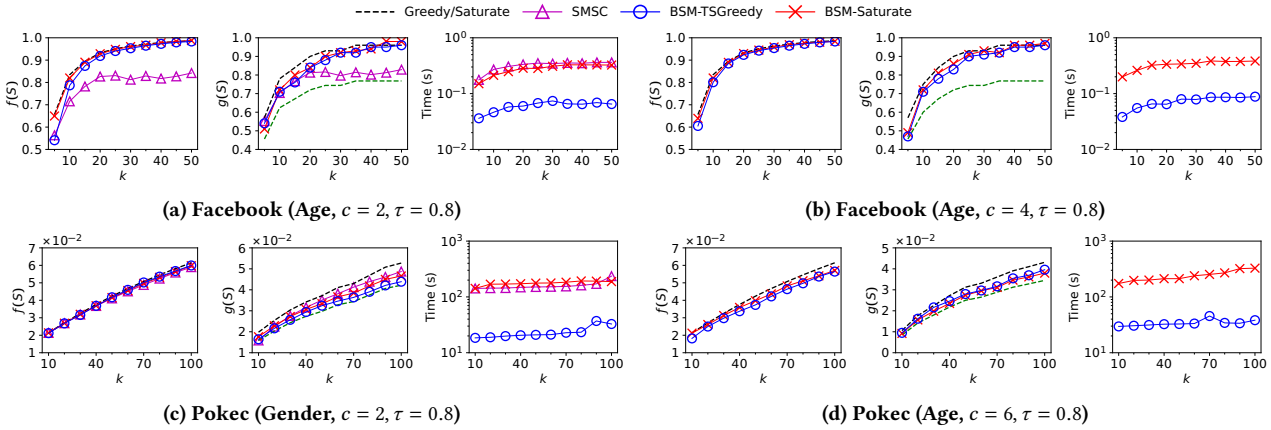
5.1 Maximum Coverage

Setup. We first apply the BSM framework to the maximum coverage (MC) problem to maximize the overall coverage while ensuring group-level fairness. Specifically, for a set U of m users and a collection V of n sets defined on U, we suppose that the utility $f_u(S)$ of a set $S \subseteq V$ to user u is equal to 1 if u is contained by the union of the sets in S and 0 otherwise. Given the definition of f_u, the function $f(S)$ in Eq. 1 captures the average coverage of S over U and the function $g(S)$ in Eq. 2 denotes the minimum of average coverages of S among different groups on U.

We use two synthetic and three real-world datasets in the MC experiments. The two synthetic datasets are undirected random graphs generated by the stochastic block model [26] (SBM) with 500 nodes consisting of two and four groups, respectively. We set the intra-group and inter-group connection probabilities in the generation procedure to 0.1 and 0.02. We also use three publicly available real-world datasets, namely *Facebook* [47], *DBLP* [13], and *Pokec*[2], in the experiments. The *Facebook* dataset is an undirected graph representing the friendships between students at Rice University on Facebook. Profile data contains the *age* attribute to divide students into two (i.e., age < 20 and age ≥ 20) or four (i.e., age $= 19, 20, 21, 22$) groups. The *DBLP* dataset is an undirected graph denoting the co-authorships between researchers. We divide them into five groups based on which continent (i.e., ‘Asia’, ‘Europe’, ‘North America’, ‘Oceania’, ‘South America’) their affiliations are located in. The *Pokec* dataset is a directed graph representing the follower-followee relationships of users in a social network in Slovakia. We use the *gender* and *age* information in profile data to divide users into two and six groups, respectively. Table 1 shows the statistics of all the above datasets, where n (and m) is the number of nodes (and users) and $|E|$ is the number of edges. We also present the percentage of users from each group in the population. Following a common dominating set [8] formulation, we construct a set system on each graph as follows. First, the ground set U is equal to the set of nodes. Then, for each node v, we create a set $S(v)$ containing all its out-neighbors $N_{out}(v)$ plus itself. Finally, the set collection V consists of the sets created for all the nodes in the graph. Based on the construction procedure, the task of BSM is to select a set of k nodes that not only contain the largest number of users in their neighborhoods but also ensure at least a minimum average coverage for every group.

Results on Effect of τ. Figure 3 shows the values of $f(S), g(S)$, where S is the solution returned by each algorithm, for each value of factor $\tau = \{0.1, 0.2, \ldots, 0.9\}$ on two random graphs when $k = 5$ and on *DBLP* when $k = 10$. The results for SMSC are ignored on the datasets with more than two groups because it does not provide any valid solution when $c > 2$. The optimums $\mathtt{OPT}_f$ and $\mathtt{OPT}_g$ for solely maximizing f and g as well as their approximations $\mathtt{OPT}'_f$ and $\mathtt{OPT}'_g$ returned by Greedy and Saturate are plotted as black horizontal lines to illustrate the trade-offs between *utility* (f) and *fairness* (g) of different algorithms w.r.t. τ. In general, our BSM framework achieves a good utility-fairness trade-off. On the one hand, when the value of τ is close to 0, $f(S)$ approaches or even reaches $\mathtt{OPT}_f$; on the other hand, when the value of τ increases, $f(S)$ decreases but $g(S)$ increases accordingly. In contrast, the SMSC framework fails to balance two objectives well. Compared with the optimal solutions returned by BSM-Optimal, the approximate solutions returned by BSM-Saturate and BSM-TSGreedy have up to 9% and 26% losses in $f(S)$. BSM-TSGreedy provides lower-quality solutions than BSM-Saturate in terms of $f(S)$ for almost all τ values on all the three graphs. This is mostly because the solution of BSM-TSGreedy has contained nearly k items after the first stage, and the number k' of items added in the second stage is thus close to 0. Nevertheless, BSM-Saturate achieves better trade-offs between $f(S)$ and $g(S)$ by combining them into a single objective function. We find that BSM-Saturate and BSM-TSGreedy provide solutions higher in $f(S)$ than BSM-Optimal in very few cases. Such results do not challenge the optimality of BSM-Optimal because BSM-Saturate and BSM-TSGreedy do not provide valid BSM solutions satisfying the constraint of $g(S) \geq \tau \cdot \mathtt{OPT}_g$ (i.e.,

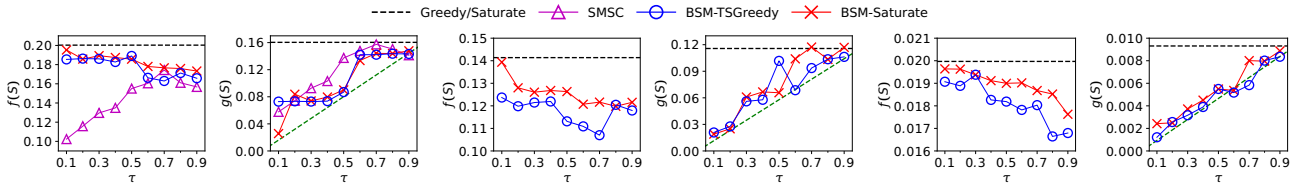
[2]https://snap.stanford.edu/data/soc-Pokec.html



(a) **Facebook (Age,** $c = 2, \tau = 0.8$**)** (b) **Facebook (Age,** $c = 4, \tau = 0.8$**)**

(c) **Pokec (Gender,** $c = 2, \tau = 0.8$**)** (d) **Pokec (Age,** $c = 6, \tau = 0.8$**)**

Figure 4: Results of different algorithms for maximum coverage by varying solution size k on Facebook and Pokec. The green dashed lines in the plots for $g(S)$ denote $\tau \cdot \mathtt{OPT}'_g$, where $\mathtt{OPT}'_g$ is computed by Saturate, to indicate whether $g(S) \geq \tau \cdot \mathtt{OPT}'_g$.



(a) **RAND** ($c = 2, k = 5$) (b) **RAND** ($c = 4, k = 5$) (c) **DBLP** ($c = 5, k = 10$)

Figure 5: Results of different algorithms for influence maximization by varying the factor τ on two random graphs and DBLP. The green, dashed straight lines denote $\tau \cdot \mathtt{OPT}'_g$, where $\mathtt{OPT}'_g$ is computed by Saturate, to show whether $g(S) \geq \tau \cdot \mathtt{OPT}'_g$.

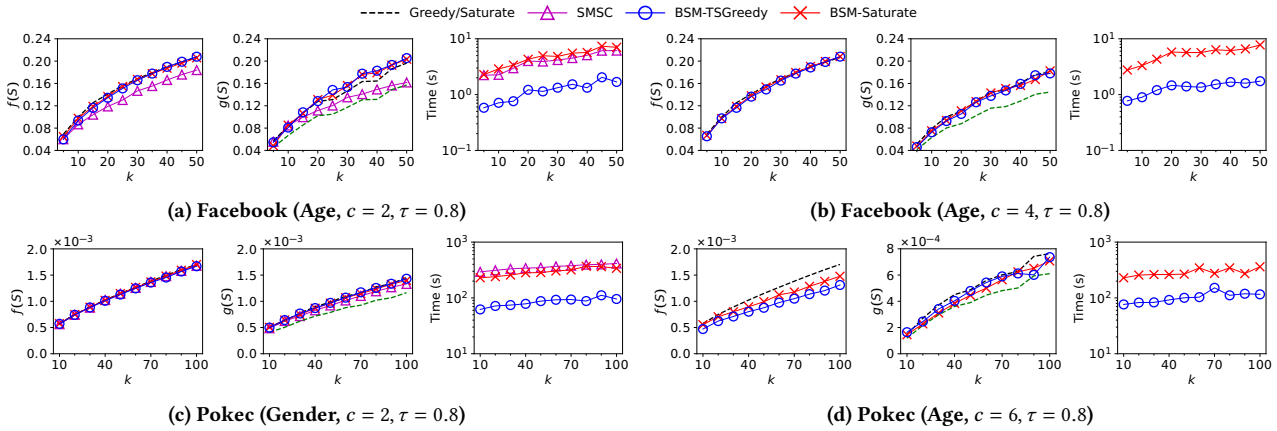
above the solid green line) in these cases. In fact, the solutions of both BSM-Saturate and BSM-TSGreedy violate the constraint $g(S) \geq \tau \cdot \mathtt{OPT}_g$ of BSM (i.e., below the solid green line) in some cases, because $\mathtt{OPT}_g$ is unknown to them, but always satisfy a "weaker" constraint $g(S) \geq \tau \cdot \mathtt{OPT}'_g$ (i.e., above the dashed green line), since $\mathtt{OPT}'_g$ is used as their input.

Results on Effect of k. Figure 4 present the values of $f(S), g(S)$, where S is the solution returned by each algorithm, and the runtime of each algorithm for the cardinality constraint $k = \{5, 10, \ldots, 50\}$ on *Facebook* and $k = \{10, 20, \ldots, 100\}$ on *Pokec* with different group partitions. According to Figure 3, we observe that the solutions of each algorithm are close to those for solely maximizing f (i.e., SM) when $\tau \leq 0.5$ and to those for solely maximizing g (i.e., RSM) when $\tau \geq 0.9$. By following the 80% rule, a common practice in algorithmic fairness, and distinguishing BSM from SM and RSM, we set the value of τ to 0.8 in the experiments to better evaluate the effect of k. Since computing the optimal solutions becomes infeasible on large datasets, $\mathtt{OPT}_f$, $\mathtt{OPT}_g$, and BSM-Optimal are omitted. In addition, the runtime of Greedy and Saturate are not presented separately in Figure 4 since they are used as subroutines in BSM-TSGreedy and BSM-Saturate. Generally, the values of $f(S)$ and $g(S)$ increase with k for all algorithms. Meanwhile, the runtime of each algorithm only slightly grows with k. This is because a great number of function evaluations are reduced by applying the *lazy forward* strategy, and thus the total number of function evaluations increases marginally with k. BSM-Saturate provides higher-quality solutions than BSM-TSGreedy at the expense of lower efficiency for different k's. The solutions of both algorithms satisfy $g(S) \geq \tau \cdot \mathtt{OPT}'_g$ in all cases. Finally, their time efficiencies on *Pokec* confirm that they are scalable to large graphs with over one million nodes and thirty million edges. We note that the values of $f(S)$ and $g(S)$ are much smaller on *Pokec* than on *Facebook* because *Pokec* is larger and sparser, on which only a very small portion of users ($< 7\%$) are covered by at most 100 nodes.

5.2 Influence Maximization

Setup. In the BSM framework, we propose a new problem that integrates the classic IM [33] aimed to maximize the overall influence spread over all users and the group fairness-aware IM [6, 61] aimed to guarantee a balanced influence distribution among groups. In particular, we are given a graph $G = (V, E, p)$, where V is a set of nodes, E is a set of edges, and $p : E \rightarrow \mathbb{R}^+$ is a function that assigns a propagation probability to each edge. We follow the *independent cascade* (IC) model [33] to describe the diffusion process[3] and set $p(e) = 0.1$ or 0.01 for each $e \in E$. We define $f_u(S) = \mathbb{P}_u(S)$, where $\mathbb{P}_u(S)$ is the probability that user u is influenced by a set $S \subseteq V$ of seeds under the IC model. Accordingly, the two functions f and g in Eqs. 1 and 2 are equivalent to the influence spread function in [33] divided by the number of users and the maximin welfare function in [61], respectively. We use a *reverse influence set* [7] (RIS) based algorithm called IMM [58] for influence spread estimation. And for any solution S, we run Monte-Carlo simulations 10,000 times to estimate $f(S)$ and $g(S)$. The datasets we use in the IM experiments are almost identical to those in the MC experiments, except that the number of nodes in random graphs is reduced to 100.

[3] Note that all the algorithms we compare can be trivially extended to any diffusion model, e.g., *linear threshold* and *triggering* models [33], and parameter settings where the influence spread function is monotone and submodular.



(a) Facebook (Age, $c = 2, \tau = 0.8$)

(b) Facebook (Age, $c = 4, \tau = 0.8$)

(c) Pokec (Gender, $c = 2, \tau = 0.8$)

(d) Pokec (Age, $c = 6, \tau = 0.8$)

Figure 6: Results of different algorithms for influence maximization by varying the solution size k on Facebook and Pokec. The green dashed lines denote $\tau \cdot \mathtt{OPT}'_g$, where $\mathtt{OPT}'_g$ is computed by SATURATE, to show whether $g(S) \geq \tau \cdot \mathtt{OPT}'_g$.

Table 2: Statistics of datasets in the FL experiments.

Dataset	n	m	d	Percentage of users from each group
RAND ($c = 2$)	100	100	5	['U_0': 15%, 'U_1': 85%]
RAND ($c = 3$)	100	100	5	['U_0': 5%, 'U_1': 20%, 'U_2': 75%]
Adult-Small (Race, $c = 5$)	100	100	6	['Amer-Indian-Eskimo': 1%, 'Asian-Pac-Islander': 2%, 'Black': 14%, 'White': 82%, 'Others': 1%]
Adult (Gender, $c = 2$)	1,000	1,000	6	['Female': 34%, 'Male': 66%]
Adult (Race, $c = 5$)	1,000	1,000	6	['Amer-Indian-Eskimo': 1%, 'Asian-Pac-Islander': 3%, 'Black': 10%, 'White': 85%, 'Others': 1%]
FourSquare-NYC ($c = 1,000$)	882	1,000	2	['u_0': 0.1%, ..., 'u_{999}': 0.1%]
FourSquare-TKY ($c = 1,000$)	1,132	1,000	2	['u_0': 0.1%, ..., 'u_{999}': 0.1%]

Results. Figures 5–6 show the results by varying the factor $\tau = \{0.1, 0.2, \ldots, 0.9\}$ on random graphs when $k = 5$ and *DBLP* when $k = 10$, as well as the results by varying the solution size $k = \{5, 10, \ldots, 50\}$ on *Facebook* and $k = \{10, 20, \ldots, 100\}$ on *Pokec* when $\tau = 0.8$. For the IM problem, the optimal solutions are infeasible even on very small graphs because computing the influence spread under the IC model is #P-hard [11]. Thus, the results of $\mathtt{OPT}_f$, $\mathtt{OPT}_g$, and BSM-OPTIMAL are all ignored. Generally, the results for IM exhibit similar trends to those for MC. Specifically, BSM-SATURATE and BSM-TSGREEDY strike better balances between f and g than SMSC. But the solution quality of BSM-TSGREEDY is mostly close to and sometimes higher than that of BSM-SATURATE for IM. Meanwhile, it still runs 1.5–4× faster than BSM-SATURATE. Nevertheless, BSM-TSGREEDY breaks the "weak" constraint of $g(S) \geq \tau \cdot \mathtt{OPT}'_g$ in a few cases due to the errors in influence estimations by IMM. Finally, although our algorithms take much longer for IM than MC, they are still scalable to large graphs such as *Pokec*, where the computation is always finished within 1,000 seconds.

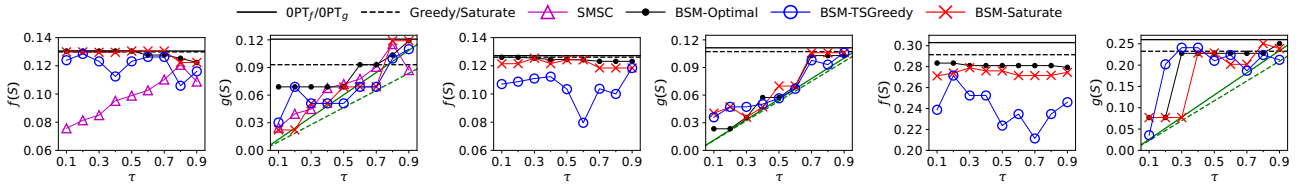
5.3 Facility Location

Setup. Facility location (FL) is a general model for different real-world problems such as *exemplar clustering* [3] and *data summarization* [39]. For a set U of m users, a set V of n items (facilities), and a nonnegative *benefit matrix* $B \in \mathbb{R}^{m \times n}$, where $b_{uv} \in B$ denotes the benefit of item v on user u, we define a function $f_u(S) = \max_{v \in S} b_{uv}$ to measure the benefit of set S on user u. In this way, the two functions f and g in Eqs. 1 and 2 measures the average utility for all users and the minimum of the average utilities for all groups, respectively. And the goal of our BSM framework is to balance both objectives. Suppose that each user u or item v is represented as a vector $\mathbf{p}_u$ or $\mathbf{p}_v$ in $\mathbb{R}^d$. We use two standard methods to compute the benefits in the literature: one is based on the *k-median clustering* [3], i.e., $b_{uv} = \max\{0, \overline{d} - dist(\mathbf{p}_u, \mathbf{p}_v)\}$, where $dist(\mathbf{p}_u, \mathbf{p}_v)$ is the Euclidean distance between $\mathbf{p}_u$ and $\mathbf{p}_v$ and $\overline{d}$ is the distance for normalization; the other is based on the *RBF kernel* [39], i.e., $b_{uv} = e^{-dist(\mathbf{p}_u, \mathbf{p}_v)}$.

In the experiments, we use two public real-world datasets, namely *Adult*[4] and *FourSquare* [69]. The *Adult* dataset contains socioeconomic records of individuals, where the *gender* and *age* attributes are used for group partitioning. We randomly sample 100 or 1,000 records as both facilities and users and adopt RBF for benefit computation. The *FourSquare* dataset includes check-in data in New York City and Tokyo. We extract all locations of *medical centers* as facilities, randomly sample 1,000 distinct check-in locations as representatives of users, and adopt the k-median function for benefit computation. Since user profiles are not available in *FourSquare*, we treat each user as a single group with $c = 1,000$ groups in total. Moreover, we generate two random $5d$ datasets of size 100 with 2 and 3 groups, where each group corresponds to an isotropic Gaussian blob, and RBF is used for benefit computation. Table 2 presents the statistics of all the above datasets, where n is the number of facilities, m is the number of users, and d is the dimension of feature vectors. We also report the percentage of users from each group in the population.

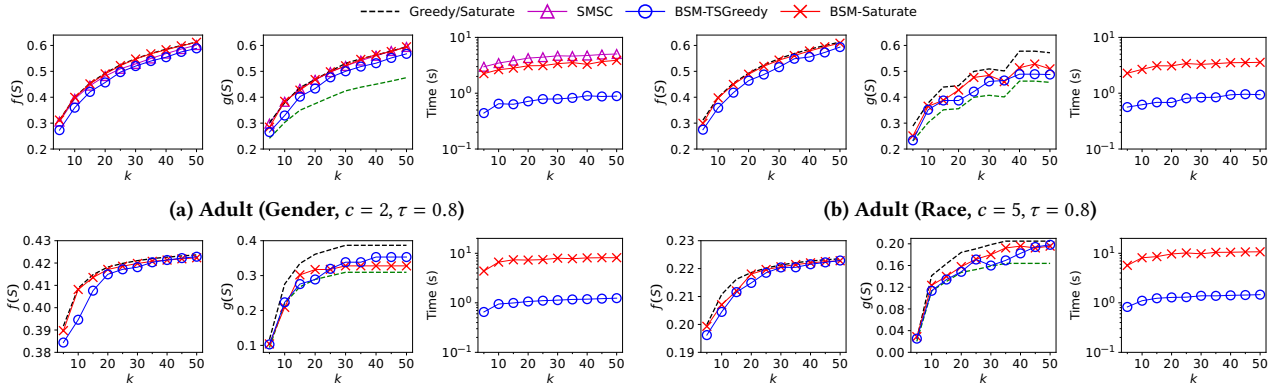
Results. Figures 7–8 show the results by varying the factor $\tau = \{0.1, 0.2, \ldots, 0.9\}$ on two random datasets and *Adult-Small* when $k = 5$ and the solution size $k = \{5, 10, \ldots, 50\}$ on *Adult* and *FourSquare* when $\tau = 0.8$. The results for FL are also generally

[4] https://archive.ics.uci.edu/ml/datasets/adult



(a) RAND ($c = 2, k = 5$) **(b) RAND ($c = 3, k = 5$)** **(c) Adult-Small ($c = 5, k = 5$)**

Figure 7: Results of different algorithms for facility location by varying the factor τ on two random datasets and Adult-Small. The green straight lines are drawn in the same manner as those in Figure 3.



(a) Adult (Gender, $c = 2, \tau = 0.8$) **(b) Adult (Race, $c = 5, \tau = 0.8$)**

(c) FourSquare-NYC ($c = 1000, \tau = 0.8$) **(d) FourSquare-TKY ($c = 1000, \tau = 0.8$)**

Figure 8: Results of different algorithms for facility location by varying the solution size k on Adult and FourSquare. The green dashed lines are drawn in the same manner as those in Figure 4.

similar to those for MC and IM. BSM-Saturate has considerable advantages over BSM-TSGreedy in terms of solution quality, whereas BSM-TSGreedy runs much faster than BSM-Saturate. And they both strike better balances between f and g than SMSC. Finally, their performance on *FourSquare* demonstrates that they are capable of providing high-quality solutions to BSM efficiently when the number c of groups is large (i.e., up to $c = 1,000$).

6 CONCLUSIONS AND FUTURE WORK

In this paper, we studied the problem of balancing utility and fairness in submodular maximization. We formulated the problem as a bicriteria optimization problem called BSM. Since BSM generally could not be approximated within any constant factor, we proposed two instance-dependent approximation algorithms called BSM-TSGreedy and BSM-Saturate for BSM. We showed the effectiveness, efficiency, and scalability of our proposed algorithms by performing extensive experiments on real-world and synthetic data in three problems: maximum coverage, influence maximization, and facility location.

In future work, despite the inapproximability of BSM, we will explore how to further improve the approximation factors for BSM by performing problem-specific analyses and exploiting the correlations between group-specific utility functions. It would also be interesting to generalize BSM to non-monotone or weakly submodular functions.

A ILP FORMULATION

Although BSM is generally inapproximable, for specific classes of BSM problems, it is possible to find the optimal solution of a small instance by solving it as an integer linear programming [67] (ILP) problem using any ILP solver. This approach is referred to as the BSM-Optimal algorithm in the experiments (Section 5). Here, we define the ILP formulations of *maximum coverage* and *facility location* in the context of BSM. Note that these formulations are problem-specific and cannot be extended to other submodular maximization problems, such as *influence maximization*.

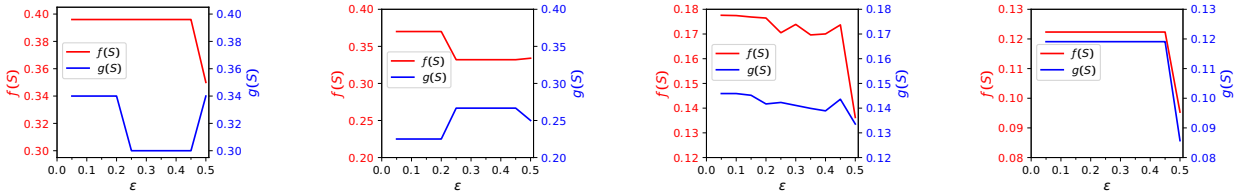
By adapting the standard ILP formulation of maximum coverage[5], we present an ILP to maximize the *average coverage* (i.e., the function f in BSM) on a universe $U = \{u_1, \ldots, u_m\}$ of m elements (users) and a collection $V = \{S_1, \ldots, S_n\}$ of n sets (items) with a cardinality constraint k as follows:

$$\begin{aligned} \max \quad & \sum_{j \in [m]} \frac{y_j}{m} \\ \text{subject to} \quad & \sum_{l \in [n]} x_l \leq k \\ & \sum_{u_j \in S_l} x_l \geq y_j \\ & y_j \in \{0,1\}, \forall j \in [m] \\ & x_l \in \{0,1\}, \forall l \in [n] \end{aligned} \tag{5}$$

where x_l is an indicator of whether set $S_l \in V$ is included in S and y_j is an indicator of whether user $u_j \in U$ is covered by S.

Then, we generalize the ILP in Eq. 5 to robust maximum coverage that aims to maximize the minimum of the average coverage among c groups $U_1, \ldots, U_c$ (i.e., the function g in BSM). We introduce a new variable w to denote the value of $g(S)$. As such, the objective of the generalized ILP is to maximize w. Meanwhile, we should incorporate new constraints on the average coverage of

[5]https://en.wikipedia.org/wiki/Maximum_coverage_problem



(a) RAND (MC, $c = 2, k = 5$) **(b) RAND (MC, $c = 4, k = 5$)** **(c) RAND (IM, $c = 2, k = 5$)** **(d) RAND (FL, $c = 2, k = 5$)**

Figure 9: Results of BSM-Saturate by varying the error parameter ε on random datasets ($\tau = 0.8$).

every group so that it does not exceed w (thus, w is the minimum). In particular, the ILP formulation of *robust maximum coverage* is as follows:

$$\begin{aligned} \max \quad & w \qquad (6)\\ \text{subject to} \quad & \sum_{l\in[n]} x_l \le k \\ & \sum_{u_j\in S_l} x_l \ge y_j \\ & \sum_{u_j\in U_i} \frac{y_j}{m_i} \ge w, \forall i \in [c] \\ & y_j \in \{0,1\}, \forall j \in [m] \\ & x_l \in \{0,1\}, \forall l \in [n] \\ & w \ge 0 \end{aligned}$$

Finally, given the optimal objective value of robust maximum coverage in Eq. 6 as input OPT_g, we can define the BSM version of maximum coverage for a balance parameter $\tau \in [0,1]$ by adding new constraints $\sum_{u_j\in U_i} \frac{y_j}{m_i} \ge \tau \cdot \mathsf{OPT}_g, \forall i \in [c]$ to Eq. 5 to ensure that $g(S) \ge \tau \cdot \mathsf{OPT}_g$.

Then, for our facility location problem with a benefit matrix $B = \{b_{jl} : j \in [m], l \in [n]\} \in \mathbb{R}^{m\times n}$, we extend the ILP formulation for capacitated facility location[6] as follows:

$$\begin{aligned} \max \quad & \sum_{j\in[m]} \sum_{l\in[n]} \frac{b_{jl} y_{jl}}{m} \qquad (7)\\ \text{subject to} \quad & \sum_{l\in[n]} x_l \le k \\ & \sum_{l\in[n]} y_{jl} \le 1, \forall j \in [m] \\ & y_{jl} \le x_l, \forall j \in [m], l \in [n] \\ & y_{jl} \in \{0,1\}, \forall j \in [m], l \in [n] \\ & x_l \in \{0,1\}, \forall l \in [n] \end{aligned}$$

Similarly, we can also generalize Eq. 7 to the robust and BSM versions of facility location by (*i*) changing the objective to max w and adding the new constraints $\sum_{u_j\in U_i} \sum_{l\in[n]} \frac{b_{jl} y_{jl}}{m_i} \ge w, \forall i \in [c]$ and (*ii*) adding the new constraints $\sum_{u_j\in U_i} \sum_{l\in[n]} \frac{b_{jl} y_{jl}}{m_i} \ge \tau \cdot \mathsf{OPT}_g, \forall i \in [c]$, respectively.

B ADDITIONAL EXPERIMENTS

Effect of Error Parameter ε: We present how the values of $f(S)$ and $g(S)$ of the solution S returned by BSM-Saturate are affected by the error parameter $\varepsilon \in \{0.05, 0.1, \ldots, 0.5\}$ in Figure 9. In opposition to our theoretical analysis, we find that the value of ε hardly affects the performance of BSM-Saturate until $\varepsilon \ge 0.5$ in practice. Since the ratio between α_{max} and α_{min} (see Line 4 of Algorithm 2) is used as the stop condition, BSM-Saturate will never terminate until $\alpha_{min} > 0$. We observe that the absolute differences between α_{min}'s at different iterations are often small since they are close to 0. Thus, their corresponding solutions are close to each other in terms of $f(S)$ and $g(S)$. Therefore, the values of α_{min}, as well as $f(S)$ and $g(S)$, do not change so much for different ε's smaller than 0.5. Nevertheless, to guarantee the good performance of BSM-Saturate in extreme cases, we use $\varepsilon = 0.05$ in all the experiments of Section 5.

[6] https://en.wikipedia.org/wiki/Facility_location_problem

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under grant number 62202169.

REFERENCES

[1] Abolfazl Asudeh, Tanya Berger-Wolf, Bhaskar DasGupta, and Anastasios Sidiropoulos. 2023. Maximizing Coverage While Ensuring Fairness: A Tale of Conflicting Objectives. *Algorithmica* 85 (2023), 1287–1331.

[2] Abolfazl Asudeh, H. V. Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing Fair Ranking Schemes. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1259–1276.

[3] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. Association for Computing Machinery, New York, NY, USA, 671–680.

[4] Wenruo Bai, Rishabh Iyer, Kai Wei, and Jeff Bilmes. 2016. Algorithms for Optimizing the Ratio of Submodular Functions. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, 2751–2759.

[5] Sayan Bandyapadhyay, Aritra Banik, and Sujoy Bhore. 2021. On Fair Covering and Hitting Problems. In *Graph-Theoretic Concepts in Computer Science – 47th International Workshop, WG 2021, Warsaw, Poland, June 23–25, 2021, Revised Selected Papers*. Springer, Cham, 39–51.

[6] Ruben Becker, Federico Corò, Gianlorenzo D'Angelo, and Hugo Gilbert. 2020. Balancing Spreads of Influence in a Social Network. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (2020), 3–10.

[7] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*. Society for Industrial and Applied Mathematics, 946–957.

[8] Nicolas Bray and Eric W. Weisstein. 2023. Dominating Set. https://mathworld.wolfram.com/DominatingSet.html

[9] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.

[10] Flávio P. Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. 2017. Optimized Pre-Processing for Discrimination Prevention. *Advances in Neural Information Processing Systems* 30 (2017), 3992–4001.

[11] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. Association for Computing Machinery, New York, NY, USA, 1029–1038.

[12] Shuang Cui, Kai Han, Tianshuai Zhu, Jing Tang, Benwei Wu, and He Huang. 2021. Randomized Algorithms for Submodular Function Maximization with a k-System Constraint. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2222–2232.

[13] Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. 2023. Fairness in Graph Mining: A Survey. *IEEE Trans. Knowl. Data Eng.* PrePrints (2023), 1–22. https://doi.org/10.1109/TKDE.2023.3265598

[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12)*. Association for Computing Machinery, New York, NY, USA, 214–226.

[15] Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub M. Tarnawski. 2020. Fairness in Streaming Submodular Maximization: Algorithms and Hardness. *Advances in Neural Information Processing Systems* 33 (2020), 13609–13622.

[16] Vitalii Emelianov, Nicolas Gast, Krishna P. Gummadi, and Patrick Loiseau. 2022. On fair selection in the presence of implicit and differential variance. *Artif. Intell.* 302 (2022), 103609.

[17] Alessandro Epasto, Silvio Lattanzi, Sergei Vassilvitskii, and Morteza Zadimoghaddam. 2017. Submodular Optimization Over Sliding Windows. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. Association for Computing Machinery, New York, NY, USA, 421–430.

[18] Seyed A. Esmaeili, Sharmila Duppala, Vedant Nanda, Aravind Srinivasan, and John P. Dickerson. 2022. Rawlsian Fairness in Online Bipartite Matching: Two-Sided, Group, and Individual. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS '22)*. Association for Computing Machinery, New York, NY, USA, 1583–1585.

[19] Uriel Feige. 1998. A Threshold of ln *n* for Approximating Set Cover. *J. ACM* 45, 4 (1998), 634–652.

[20] Xiaoyun Fu, Rishabh Rajendra Bhatt, Samik Basu, and Aduri Pavan. 2021. Multi-Objective Submodular Optimization with Approximate Oracles and Influence Maximization. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 328–334.

[21] David García-Soriano and Francesco Bonchi. 2020. Fair-by-design matching. *Data Min. Knowl. Discov.* 34, 5 (2020), 1291–1335.

[22] David García-Soriano and Francesco Bonchi. 2021. Maxmin-Fair Ranking: Individual Fairness under Group-Fairness Constraints. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, New York, NY, USA, 436–446.

[23] Shay Gershtein, Tova Milo, and Brit Youngmann. 2021. Multi-Objective Influence Maximization. In *Proceedings of the 24th International Conference on Extending Database Technology (EDBT '21)*. OpenProceedings.org, 145–156.

[24] Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. 2021. Socially Fair k-Means Clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*. Association for Computing Machinery, New York, NY, USA, 438–448.

[25] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. *Advances in Neural Information Processing Systems* 29 (2016), 3315–3323.

[26] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Soc. Netw.* 5, 2 (1983), 109–137.

[27] Rishabh K. Iyer and Jeff A. Bilmes. 2012. Algorithms for Approximate Minimization of the Difference Between Submodular Functions, with Applications. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 407–417.

[28] Rishabh K. Iyer and Jeff A. Bilmes. 2013. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. *Advances in Neural Information Processing Systems* 26 (2013), 2436–2444.

[29] H. V. Jagadish, Francesco Bonchi, Tina Eliassi-Rad, Lise Getoor, Krishna Gummadi, and Julia Stoyanovich. 2019. The Responsibility Challenge for Data. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 412–414.

[30] Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. 2021. Unconstrained Submodular Maximization with Modular Costs: Tight Approximation and Application to Profit Maximization. *Proc. VLDB Endow.* 14, 10 (2021), 1756–1768.

[31] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. 2018. Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2544–2553.

[32] Michael J. Kearns, Aaron Roth, and Zhiwei Steven Wu. 2017. Meritocratic Fairness for Cross-Population Selection. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 1828–1836.

[33] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. Association for Computing Machinery, New York, NY, USA, 137–146.

[34] Jon Kleinberg, Jens Ludwig, Sendhil Mullainathan, and Cass R. Sunstein. 2019. Discrimination in the Age of Algorithms. *J. Leg. Anal.* 10 (2019), 113–174.

[35] Andreas Krause and Daniel Golovin. 2014. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, Cambridge, UK, 71–104.

[36] Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. 2008. Robust Submodular Observation Selection. *J. Mach. Learn. Res.* 9, 12 (2008), 2761–2801.

[37] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne Van Briesen, and Natalie Glance. 2007. Cost-Effective Outbreak Detection in Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. Association for Computing Machinery, New York, NY, USA, 420–429.

[38] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence Maximization on Social Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* 30, 10 (2018), 1852–1872.

[39] Erik M. Lindgren, Shanshan Wu, and Alexandros G. Dimakis. 2016. Leveraging Sparsity for Efficient Submodular Data Summarization. *Advances in Neural Information Processing Systems* 29 (2016), 3414–3422.

[40] Yury Makarychev and Ali Vakilian. 2021. Approximation Algorithms for Socially Fair Clustering. In *Proceedings of Thirty-Fourth Conference on Learning Theory*. PMLR, 3246–3264.

[41] Tasfia Mashiat, Xavier Gitiaux, Huzefa Rangwala, Patrick Fowler, and Sanmay Das. 2022. Trade-Offs between Group Fairness Metrics in Societal Resource Allocation. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 1095–1105.

[42] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2022. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (2022), 35 pages.

[43] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. 2016. Fast Constrained Submodular Maximization: Personalized Data Summarization. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, 1358–1367.

[44] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. 2015. Lazier Than Lazy Greedy. *Proceedings of the AAAI Conference on Artificial Intelligence* 29 (2015), 1812–1818.

[45] Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2017. Deletion-Robust Submodular Maximization: Data Summarization with "the Right to be Forgotten". In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2449–2458.

[46] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. 2016. Distributed Submodular Maximization. *J. Mach. Learn. Res.* 17, 235 (2016), 1–44.

[47] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. 2010. You Are Who You Know: Inferring User Profiles in Online Social Networks. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. Association for Computing Machinery, New York, NY, USA, 251–260.

[48] Morteza Monemizadeh. 2020. Dynamic Submodular Maximization. *Advances in Neural Information Processing Systems* 33 (2020), 9806–9817.

[49] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.* 14, 1 (1978), 265–294.

[50] Lan N. Nguyen and My T. Thai. 2021. Minimum Robust Multi-Submodular Cover for Fairness. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 10 (2021), 9109–9116.

[51] Sofia Maria Nikolakaki, Alina Ene, and Evimaria Terzi. 2021. An Efficient Framework for Balancing Submodularity and Cost. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1256–1266.

[52] Naoto Ohsaka and Tatsuya Matsuoka. 2021. Approximation algorithm for submodular maximization under submodular cover. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR, 792–801.

[53] Shameem Puthiya Parambath, Nishant Vijayakumar, and Sanjay Chawla. 2018. SAGA: A Submodular Greedy Algorithm for Group Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (2018), 3900–3908.

[54] John Rawls. 1971. *A Theory of Justice*. Harvard University Press, Cambridge, MA, USA.

[55] Barna Saha and Lise Getoor. 2009. On Maximum Coverage in the Streaming Model & Application to Multi-topic Blog-Watch. In *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*. Society for Industrial and Applied Mathematics, 697–708.

[56] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in Package-to-Group Recommendations. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. Association for Computing Machinery, New York, NY, USA, 371–379.

[57] Jing Tang, Xueyan Tang, Andrew Lim, Kai Han, Chongshou Li, and Junsong Yuan. 2021. Revisiting Modified Greedy Algorithm for Monotone Submodular Maximization with a Knapsack Constraint. *Proc. ACM Meas. Anal. Comput. Syst.* 5, 1, Article 08 (2021), 22 pages.

[58] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 1539–1554.

[59] Suhas Thejaswi, Bruno Ordozgoiti, and Aristides Gionis. 2021. Diversity-Aware k-median: Clustering with Fair Center Representation. In *Machine Learning and Knowledge Discovery in Databases. Research Track – European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II*. Springer, Cham, 765–780.

[60] Alfredo Torrico, Mohit Singh, Sebastian Pokutta, Nika Haghtalab, Joseph (Seffi) Naor, and Nima Anari. 2021. Structured Robust Submodular Maximization: Offline and Online Algorithms. *INFORMS J. Comput.* 33, 4 (2021), 1590–1607.

[61] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. 2019. Group-Fairness in Influence Maximization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 5997–6005.

[62] Rajan Udwani. 2018. Multi-objective Maximization of Monotone Submodular Functions with Cardinality Constraint. *Advances in Neural Information Processing Systems* 31 (2018), 9513–9524.

[63] Yanhao Wang, Francesco Fabbri, and Michael Mathioudakis. 2021. Fair and Representative Subset Selection from Data Streams. In *Proceedings of the Web Conference 2021 (WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1340–1350.

[64] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. 2017. Real-Time Influence Maximization on Dynamic Social Streams. *Proc. VLDB Endow.* 10, 7 (2017), 805–816.

[65] Yanhao Wang, Yuchen Li, and Kian-Lee Tan. 2019. Efficient Representative Subset Selection over Sliding Windows. *IEEE Trans. Knowl. Data Eng.* 31, 7 (2019), 1327–1340.

[66] Kai Wei, Rishabh K. Iyer, Shengjie Wang, Wenruo Bai, and Jeff A. Bilmes. 2015. Mixed Robust/Average Submodular Partitioning: Fast Algorithms, Guarantees, and Applications. *Advances in Neural Information Processing Systems* 28 (2015), 2233–2241.

[67] Laurence Wolsey. 2020. *Integer Programming, Second Edition.* John Wiley & Sons, Inc., Hoboken, NJ, USA.

[68] Laurence A. Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2, 4 (1982), 385–393.

[69] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Trans. Syst. Man Cybern. Syst.* 45, 1 (2015), 129–142.