Logarithmic Approximations for Fair k-Set Selection

Shi Li[1] , **Chenyang Xu**[2] and **Ruilong Zhang**[3]

[1]School of Computer Science, Nanjing University, Nanjing, China
[2] Software Engineering Institute, East China Normal University, Shanghai, China
[3]Department of Mathematics, Technical University of Munich, Munich, Germany
shili@nju.edu.cn, cyxu@sei.ecnu.edu.cn, ruilong.zhang@tum.de

Abstract

We study the fair k-set selection problem where we aim to select k sets from a given set system such that the (weighted) occurrence times that each element appears in these k selected sets are balanced, i.e., the maximum (weighted) occurrence times are minimized. By observing that a set system can be formulated into a bipartite graph $G := (L \cup R, E)$, our problem is equivalent to selecting k vertices from R such that the maximum (weighted) number selected neighbors of vertices in L is minimized. The problem arises in a wide range of applications in various fields, such as machine learning, artificial intelligence, and operations research.

We first prove that the problem is NP-hard even if the maximum degree Δ of the input bipartite graph is 3, and the problem is in P when $\Delta = 2$. We then show that the problem is also in P when the input set system forms a laminar family. Based on intuitive linear programming, we show that two rounding algorithms achieve $O(\frac{\log n}{\log \log n})$-approximation on general bipartite graphs, and an independent rounding algorithm achieves $O(\log \Delta)$-approximation on bipartite graphs with a maximum degree Δ. We demonstrate that our analysis is almost tight by providing a hard instance for this linear programming.

1 Introduction

The problem of fair k-set selection is to select k sets from a given set system such that the (weighted) occurrence times that each element appears in these k selected sets are balanced, i.e., the maximum (weighted) occurrence times among all elements are minimized. Observe that a set system can be formulated into a bipartite graph $G := (L \cup R, E)$, i.e., each element and set corresponds to a vertex in L and R respectively; there is an edge between an element vertex and a set vertex if the element is included in the set. Thus, the fair k-set selection problem is equivalent to finding k vertices from R such that the maximum (weighted) number of selected neighbors of vertices in L are minimized.

The above problem aims to balance the frequency of element occurrence within a selected subset of sets. Thus, it falls under the umbrella of the subset selection problem, which arises in a wide range of applications in various fields, such as artificial intelligent [De and Chakrabarti, 2022; Mehrotra and Vishnoi, 2023; Tschiatschek *et al.*, 2017], machine learning [Boehmer *et al.*, 2023; Lang *et al.*, 2022; Mirzasoleiman *et al.*, 2020; Tukan *et al.*, 2023], data mining [Bao *et al.*, 2022; Yi *et al.*, 2023], operations research [Hazimeh and Mazumder, 2020; Mazumder *et al.*, 2023], to name just a few. In the following, we present three concrete applications of the problem in the recommended system, feature selection, and facility location. As a typical application of subset selection algorithms, our problem also captures applications in feature selection on unbalanced datasets, which is described in the full version of the paper [Li *et al.*, 2025].

Fair Ad Recommendation. The first application of the above problem is about ad recommendation. Fairness plays a vital role in recommendation systems and an unfair system may harm the benefits of multiple stakeholders [Abdollahpouri and Burke, 2019; Beutel *et al.*, 2019; Li *et al.*, 2023]; see [Wang *et al.*, 2023] for an excellent survey. The fair k-set selection problem can model a fair ad recommendation scenario as follows: a recommendation system has n users, and the company needs to select k types of ads from m types to push to these users. A user may dislike some ad types, and this can be learned from her block history. If the user receives too many ads that she does not like to see, this will cause her to resent the recommendation system. We use *disagreement* to denote the case where a user sees an ad that she dislikes. The goal is to fairly pick k ads from m ads, i.e., minimize the maximum disagreement among all users.

Fair Facility Location. The last impetus for studying the above problem is an application in facility location. Fairness in facility location has been extensively studied in the field of both game theory and computational social choices [Aziz *et al.*, 2022; Wang *et al.*, 2021; Zhou *et al.*, 2022]. Consider the following scenario where the government aims to select k positions from m positions to build facilities (e.g., garbage recycle stations). There are n agents, and all agents do not want many garbage recycling stations built near them. Each agent has multiple adjacent positions, which can be viewed as a subset of the given m positions. We also use the disagreement to denote that case where some recycling station is built near an agent. The goal is to find a fair way to build k

facilities, i.e., minimize the maximum disagreement.

Our problem is related to the anonymous refugee housing problem, which was recently proposed by [Knop and Schierreich, 2023; Schierreich, 2023]. In this problem, the input is an undirected graph, and each vertex corresponds to a house that is either an empty house or occupied by an inhabitant. There are k anonymous refugees, each of which is required to be assigned an empty house. Each inhabitant has an upper bound, and an assignment is called inhabitant respected if the number of refugees in the neighborhood of every inhabitant is at most its upper bound. The goal is to determine whether an inhabitant-respected assignment exists. By considering (i) each inhabitant as an element; (ii) each empty house as a set of inhabitants that are adjacent to this empty house, the minimax fairness version of the anonymous refugee housing problem is equivalent to our problem. So, the fair anonymous refugee housing problem can be viewed as another application in Fair Facility Location. The previous works only focus on the computation complexity while we aim to design approximate algorithms, and this leads to a completely different technique.

For convenience of description, we mainly use the bipartite-graph-based formulation of our problem. By following the convention of the computational social choice community, we consider each vertex in L as an agent and the integer k as the *demand*, where k is the number of vertices that we need to select. For each vertex in L, we use *disagreement* to denote the number of selected neighbors of this vertex. The formal definition of the problem is shown in Section 2.

1.1 Our Contributions

We consider the Fair k-Set Selection problem (FKSS) in which we mainly use the bipartite graph $G := (L \cup R, E)$ to describe the input. We distinguish several cases according to the type of the input bipartite graph. For each case, we give either an exact or approximation algorithm running in polynomial time. In the following, we summarize the main results of this work.

Main Result 1. The problem FKSS is NP-hard even on bipartite graphs with $\Delta = 3$. For bipartite graphs with $\Delta = 2$, the problem FKSS is in P. The problem is also in P if the input set system forms a laminar family. Moreover, the maximin criteria do not admit any polynomial time α-approximate algorithm unless P = NP, where α is an arbitrary function of the input.

Our first result focuses on the computation complexity of the problem. This part is completely omitted in this version due to the space limit and can be found in [Li *et al.*, 2025]. We first show that the problem is NP-hard even in the case $\Delta = 3$. The hardness result is built on the maximum independent set on planar graphs, which is shown to be NP-hard in [Garey and Johnson, 1977]. We then show that the complement case ($\Delta = 2$) is polynomially solvable by giving a simple and efficient combinatorial algorithm. For the laminar set family, we give a dynamic programming algorithm that computes the optimal solution in polynomial time. One may expect that maximin criteria are also candidates to investigate the fairness of the k-set selection problem. Namely, find k sets such that the minimum number of selected neighbors of vertices in L is maximized, where we call these neighbors *agreement*. This is not the case because it is NP-hard to determine whether the optimal solution has a zero agreement; this case is equal to determining whether there exist k sets that cover all elements.

Main Result 2 (Theorem 1, 2). Given any instance of FKSS on general bipartite graphs, there is a randomized algorithm that achieves $O(\frac{\log n}{\log \log n})$-approximation with high probability running in $\mathrm{poly}(n)$ times, where n is the number of vertices in the graph. Moreover, our analysis is optimal up to a constant factor.

Our second result focuses on the general bipartite graph (Section 3). To have a better understanding of our algorithmic ideas, we focus on the unweighted case in Section 3, and then we extend our algorithms to the weighted case in [Li *et al.*, 2025]. We give two algorithms and both algorithms are LP-based rounding algorithms. The first algorithm is a simple independent rounding algorithm that is $O(\frac{\log n}{\log \log n})$-approximate and it only can satisfy the demand requirement with high probability. The second algorithm is a dependent rounding algorithm which can surely satisfy the demand requirement while achieving the same ratio. The random variables used in the second algorithm are negatively correlated, which enables us to use the strong concentration bounds (e.g., Chernoff bound). The algorithm is a special case of the classical pipage rounding technique [Chekuri *et al.*, 2010], which is used to handle the more general matroid constraints. We also demonstrate that our analysis is tight up to some constant factor by showing that the LP has a $\Omega(\frac{\log n}{\log \log n})$ integrality gap. This part is omitted in this version.

Main Result 3 (Theorem 3). Given any instance of FKSS on bipartite graphs with a maximum degree Δ, there is a randomized algorithm that achieves $O(\log \Delta)$-approximation with running time $\mathrm{poly}(n)$ in expectation. Moreover, our analysis is nearly tight.

By observing that the maximum degree of many practiced in real-life scenarios is often small compared to the number of vertices in the whole graph, our third result focuses on graphs with a maximum degree Δ (Section 4). To have a better understanding of our algorithm, we first focus on the unweighted case in Section 4 and then extend our algorithms to the weighted case in [Li *et al.*, 2025]. We prove that there is a randomized algorithm achieving $O(\log \Delta)$-approximation ratio, which significantly improves the Δ approximation achieved by a trivial algorithm. Our algorithm is based on the same linear programming formulation as the general graph. To get rid of the dependency on the number of vertices, we employ the powerful *Lovász Local Lemma*. Our analysis is also almost tight since the same integrality gap instance also implies a $\Omega(\frac{\log \Delta}{\log \log \Delta})$ gap of the natural linear programming.

1.2 Other Related Works

Subset selection has numerous variants in literature. To the best of our knowledge, FKSS is a novel problem and has never been addressed before. One of the most representative subset selection problems is to select a subset of elements such that a monotone submodular function is opti-

mized. If the selected subset is required to satisfy a cardinality constraint, the submodular maximization admits a $(1-\frac{1}{e})$-approximate algorithm [Nemhauser *et al.*, 1978]. This result can be further extended to a more general matroid constraint, and the approximation ratio remains the same [Călinescu *et al.*, 2011; Filmus and Ward, 2014]. In contrast, the submodular minimization subject to some constraint has a strong lower bound [Svitkina and Fleischer, 2011]. When the selected subset is required to satisfy a knapsack constraint, the submodular maximization also admits a $(1-\frac{1}{e})$-approximate algorithm [Sviridenko, 2004].

1.3 Roadmap

In Section 2, we give a formal definition of our problem. In the full version [Li *et al.*, 2025], we show that our problem is NP-hard even if $\Delta = 3$, and we give a simple polynomial-time algorithm that solves the $\Delta = 2$ case. We also show that the problem is in P when the input set system forms a laminar family. In Section 3, we consider the problem on general bipartite graphs and give $O(\frac{\log n}{\log\log n})$-approximate algorithms. In Section 4, we show that the problem admits a $O(\log \Delta)$-approximate algorithm on graphs with a maximum degree Δ. In the full version, we extend our logarithmic-approximation algorithms in Section 3 and Section 4 to the weighted case. For a better understanding of our algorithm, we will focus on the unweighted version of our problem when presenting our logarithmic approximation algorithms.

2 Preliminaries

We consider the Fair k-Set Selection problem (FKSS) and mainly use the bipartite-graph-based formulation. An instance of FKSS consists of a bipartite graph $G := (L \cup R, E)$ with $|L| = n$ and $|R| = m$ and a positive integer k with $k \leq |R|$, where k is called the *demand*. Each vertex v in R has a non-negative weight w_v. When all w_v are equal, we refer it to *unweighted case*. A feasible solution $S \subseteq R$ is a subset of vertices in R with $|S| \geq k$. In the corresponding set system, each vertex in L represents an element and each vertex in R corresponds to a set. So, we aim to choose at least k sets if we use the set system description. For each vertex i in G, let $N_G(i)$ be the set of neighbors of i in G. The disagreement of a vertex $i \in L$ is the total weight of its neighbors that are also in S, i.e., $\mathsf{dis}_i(S) := \sum_{v \in N_G(i) \cap S} w_v$. The goal is to select a subset of vertices $S \subseteq R$ with $|S| \geq k$ such that $\max_{i \in L} \mathsf{dis}_i(S)$ is minimized. In the remainder of this paper, we will use Δ to denote the maximum degree of the input bipartite graph $G = (L \cup R, E)$, i.e., $\Delta := \max_{i \in L \cup R} |N_G(i)|$.

3 General Graphs

In this section, we give two logarithmic approximate algorithms. To have a better understanding of our algorithmic idea, we shall focus on the case where all vertices in R have the same weight. We will extend our algorithms to the general FKSS instance in [Li *et al.*, 2025]. Both algorithms are LP-based randomized rounding algorithms. The first algorithm is a simple independent rounding algorithm that (i) achieves $O(\frac{\log n}{\log\log n})$-approximation with high probability; (ii) satisfies the demand requirement with high probability; see Section 3.2 for details. The second algorithm is a dependent rounding algorithm which strictly improves the first algorithm, i.e., it (i) achieves $O(\frac{\log n}{\log\log n})$-approximation with high probability; (ii) satisfies the demand requirement with probability 1. Due to the space limits, this part is omitted in this version. The dependent rounding algorithm is a special pipage rounding algorithm proposed by [Chekuri *et al.*, 2010], which preserves the negative correlation property. Compared to dependent rounding, independent rounding is simple and easy to analyze. The analysis of special pipage rounding requires proof of negative correlation property. In the full version [Li *et al.*, 2025], we also show that our analysis is optimal up to the constant factor by giving a hard instance, i.e., the linear programming formulation that we used has an integrality gap of $\Omega(\frac{\log n}{\log\log n})$.

3.1 LP Formulation

The intuitive linear programming formulation is to add the objective "min T" to (Feas-LP); if so, the integrality gap shall be $\Omega(n)$. The detailed discussion is deferred to the full version. In the following, we shall use the feasibility-checking LP (Feas-LP), which is a standard technique to cut the integrality gap.

(Feas-LP)

$$\begin{aligned} \sum_{v \in N_G(u)} x_v &\leq T, && \forall u \in L \\ \sum_{v \in R} x_v &\geq k, \\ 0 \leq x_v &\leq 1, && \forall v \in R \end{aligned}$$

We shall guess the value of the optimal solution whose range is from 1 to k. Hence, (Feas-LP) eliminates the bad instance of the intuitive LP because the guessed value $T \geq 1$. Let T^* be the minimum integer in $[1, k]$ to make (Feas-LP) admit a feasible solution; T^* can be obtained by solving $O(\log k)$ times (Feas-LP). Observe that T^* is a lower bound of the optimal solution, i.e., any optimal integral solution has a maximum disagreement of at least T^*. Let $\mathbf{x}^* := (x_v^*)_{v \in R}$ be the solution to (Feas-LP) that achieves T^*.

3.2 Independent Rounding

In this section, we show that there is a simple independent rounding algorithm that (i) achieves $O(\log n)$-approximation with high probability; (ii) satisfies the demand requirement with high probability. Formally, we aim to prove the following theorem (Theorem 1).

Theorem 1. *There is a randomized independent rounding algorithm that (i) achieves $O(\frac{\log n}{\log\log n})$-approximation with probability at least $1 - \frac{1}{n}$; (ii) satisfies the demand requirement with probability at least $1 - O(1/\frac{\log n}{\log\log n})$, where n is the number of vertices in the input bipartite graph.*

Algorithmic Intuition. The algorithmic idea of Algorithm 1 is simple. We first solve (Feas-LP) to obtain a fractional solution $(x_v^*)_{v\in R}$, and then, we interpret x_v^* as the probability that we choose v. To ensure that we can sample a sufficient number of vertices from R, we raise the probability that we choose v to $O(\frac{\ln n}{\ln\ln n})\cdot x_v^*$. Then, by using the Chebyshev bound, we can show that the sampled solution satisfies the demand requirement with high probability. Meanwhile, the disagreement of each vertex in L will also be raised to $O(\frac{\ln n}{\ln\ln n})\cdot T^*$, where T^* is a lower bound of the optimal solution. By using the right tail Chernoff bound and union bound, we can show that the disagreement of all vertices in L is at most $O(\frac{\ln n}{\ln\ln n})\cdot \mathrm{OPT}$ with high probability.

Algorithm 1 Independent Rounding Algorithm.

Input: The fractional solution $\mathbf{x}^*$.
Output: A set of vertices $S \subseteq R$ with $|S| \geq k$ with high probability.

$A \leftarrow \{ v \in R : x_v^* \geq \frac{\ln\ln n}{10\ln n} \}$;
$B \leftarrow \{ v \in R : x_v^* < \frac{\ln\ln n}{10\ln n} \}$; $B' \leftarrow \emptyset$.
if $\sum_{v\in B} x_v^* \leq 1$ **then**
    $B' \leftarrow \{ u \}$; u is an arbitrary vertex in B.
end if
if $\sum_{v\in B} x_v^* > 1$ **then**
    for each $v \in B$ **do**
        $p_v \leftarrow \frac{10\ln n}{\ln\ln n}\cdot x_v^*$. ▷ Note that $p_v \leq 1$.
        Independently add v to B' with probability p_v.
    end for
end if
return $S \leftarrow A \cup B'$.

In the following, we show that the returned solution of Algorithm 1 satisfies the demand requirement (Lemma 1) and ratio requirement (Lemma 2) with high probability. In the following proof, we will use the same notations stated in Algorithm 1.

Lemma 1. *Algorithm 1 returns a vertex set that satisfies the demand requirement with probability at least* $1-O(1/\frac{\ln n}{\ln\ln n})$*, where* n *is the number of vertices.*

Lemma 2. $\Pr[|N_G(u)\cap B'| \geq \frac{10\ln n}{\ln\ln n}\cdot T^*] \leq \frac{1}{n^2}$ *for all vertices* $v \in L$.

Theorem 1 can be proven by Lemma 1 and Lemma 2 with the union bound applied. The formal proofs of these lemmas are omitted in this version.

As we have seen above, independent rounding allows us to use the strong concentration bounds but it may not be able to guarantee that the total number of selected vertices satisfies the demand. Fortunately, several dependent rounding techniques can ensure the *negative correlation* property with which the strong concentration bounds are still applicable. One of the most elegant methods is called *pipage rounding* proposed by [Chekuri *et al.*, 2010]. The initial pipage rounding technique in [Chekuri *et al.*, 2010] is used to handle the matroid structure. In our problem, the constraint can be viewed as a simple uniform matroid. Thus, we can obtain the following theorem via the pipage rounding algorithm. More details can be found in [Li *et al.*, 2025].

Theorem 2. *Given any instance of* FKSS *on general bipartite graphs, there is a randomized algorithm with running time* $\mathrm{poly}(n)$ *that (i) satisfies the demand requirement with probability 1; (ii) returns a* $O(\frac{\log n}{\log\log n})$*-approximate solution with probability at least* $1-\frac{1}{n}$*, where* n *is the number of vertices in the input bipartite graph.*

4 Graphs with Bounded Degree

In this section, we consider FKSS on the graphs with a maximum degree Δ and mainly show that there is a $O(\log\Delta)$-approximate algorithm (Theorem 3). Similar to Section 3, to have a better understanding of our algorithmic ideas, we shall focus on the unweighted case. We will extend our algorithms to the general FKSS instances in [Li *et al.*, 2025]. We remark that our analysis is almost tight since (Feas-LP) also has an integrality gap $\Omega(\frac{\log\Delta}{\log\log\Delta})$.

Theorem 3. *Given any instance of* FKSS *on graphs with a maximum degree* Δ*, there is a randomized algorithm with running time* $\mathrm{poly}(n)$ *in the expectation that returns a* $O(\log\Delta)$*-approximate solution, where* n *is the number of vertices in the input bipartite graph.*

In practice, the maximum degree of many graphs is often small compared to the number of vertices in the whole graph. The graph with a maximum degree Δ admits a trivial Δ-approximate algorithm. Namely, arbitrarily picking k-vertices from R is a Δ-approximate algorithm by observing the following two simple facts: (i) the optimal solution to the given instance is at least 1; (ii) the maximum disagreement of any algorithm is at most Δ since the maximum degree is Δ. However, if we use the algorithm for general graphs, we can only obtain a $O(\frac{\log n}{\log\log n})$-approximate solution; this is not even a constant factor when Δ is a constant. In this section, we give a $O(\log\Delta)$-approximate algorithm, which significantly improves the ratio of the trivial Δ-approximate algorithm.

Main Obstacles and Our Ideas. Our algorithm is still based on (Feas-LP). To get rid of the dependency of n on the approximation ratio, there are three main obstacles that we need to overcome. The first obstacle comes from the union bound we used in the proof of Theorem 2. For each vertex $u \in L$, we define the following event as a "bad event": the disagreement of vertex u is larger than $\alpha\cdot\mathrm{OPT}$ for some parameter α. To ensure that these bad events do not occur at the same time, we need to use the union bound over all vertices in L; thus, we have to lose the factor n on the approximation ratio. However, if we do the analysis more carefully, we may realize that these bad events are not so independent. For example, assuming that two vertices u, v in L have the same set of neighbors in R. If the bad event of u does not occur, this shall imply that the bad event of v does not occur either. Based on this observation, we may not need to do union bound over all vertices, so bypassing the dependency of n. To this end, we shall use the powerful *Lovász Local Lemma*; see Section 4.1 for the definition of the lemma.

The next two obstacles come from the use of the Lovász Local Lemma. The second obstacle is that the Lovász Local

Lemma requires us to do independent rounding, but the independent rounding cannot ensure the feasibility of the solution, i.e., it cannot ensure the algorithm selects at least k vertices. Our idea to overcome this obstacle is that, aside from defining a bad event for the approximation ratio, we also define a bad event for the feasibility. In this way, as long as the Lovász Local Lemma ensures all bad events do not occur, we obtain a solution that achieves the desired approximation ratio and satisfies the demand requirement simultaneously. The third obstacle comes from the implementation of the above feasibility idea. Namely, we need to find an appropriate definition for the feasibility bad events so that (i) all feasibility bad events do not occur, ensuring a solution that satisfies the demand requirement; (ii) the defined bad events should not dependent on many other bad events to get rid of the dependency on n. Our idea for this obstacle is a grouping technique; see Definition 1 for details.

4.1 Lovász Local Lemma

There are several versions of Lovász Local Lemma. In our work, we shall use the *variable version* proposed by [Moser and Tardos, 2010]. In this version, there is an underlying family of mutually independent random variables on a common probability space, denoted by $\mathcal{X} := \{ X_1, \ldots, X_m \}$. Let $\mathcal{A} := \{ A_1, \ldots, A_n \}$ be a set of bad events. Each bad event A_i is determined by a subset $\mathcal{A}_i \subseteq \mathcal{X}$ of variables in $\mathcal{X}$. The *dependency graph* $G := (V, E)$ for $\mathcal{A}$ is an undirected graph such that (i) each vertex $v \in V$ corresponds a bad event A_v in $\mathcal{A}$; (ii) there is an edge between $u \in V$ and $v \in V$ if and only if $\mathcal{A}_u \cap \mathcal{A}_v \neq \emptyset$. For each bad event A_i, let $\mathcal{N}(A_i) \subseteq \mathcal{A}$ be a set of bad events that are adjacent to A_i in the dependency graph G.

Lemma 3 (Lovász Local Lemma [Moser and Tardos, 2010])**.** *If there exists a real number $x_i \in (0, 1)$ for each bad event $A_i \in \mathcal{A}$ such that $\Pr[A_i] \leq x_i \prod_{A_j \in \mathcal{N}(A_i)} (1 - x_j)$ for all $A_i \in \mathcal{A}$, then (i) there exists an assignment of the random variables in $\mathcal{X}$ such that $\Pr[\wedge_{i \in [n]} \neg A_i] > 0$; (ii) there exists an algorithm that finds such an assignment for $\mathcal{X}$ in expected time $\sum_{i \in [n]} \frac{x_i}{1 - x_i}$.*

The algorithm proposed by [Moser and Tardos, 2010] is a simple local search algorithm; let $\texttt{LocalSearch}(\cdot, \cdot)$ represent this algorithm. It takes two parameters as the input: the independent random variables $\mathcal{X}$ and the bad events $\mathcal{A}$, and it outputs an assignment $\mathfrak{X}$ of $\mathcal{X}$ that makes all bad events in $\mathcal{A}$ not occur. Let d be the maximum degree of the dependency graph G. By setting $x_i = \frac{1}{d+1}$ for each $i \in [n]$, the Lovász Local Lemma can be simplified, which produces the *Symmetric Lovász Local Lemma*. The symmetric Lovász Local Lemma was first proposed by [Erdos and Lovász, 1975].

Lemma 4 (Symmetric Lovász Local Lemma [Erdos and Lovász, 1975])**.** *Let $\mathcal{A} = \{ A_1, \ldots, A_m \}$ be a set of bad events with $\Pr[A_i] \leq p$ for all $i \in [n]$. If $e \cdot p \cdot (d + 1) \leq 1$, then $\Pr[\wedge_{i \in [n]} \neg A_i] > 0$.*

Lemma 3 implies Lemma 4. More discussions about Lemma 3 and Lemma 4 can be found in the full version.

4.2 Rounding Algorithm

Algorithm 2 is an independent rounding algorithm that is built based on the local search algorithm for Lovász Local Lemma ($\texttt{LocalSearch}(\cdot, \cdot)$). As is typical, we interpret each variable in the optimal fractional solution $\mathbf{x}^* = (x_v^*)_{v \in R}$ as the probability. For each $v \in R$, we raise the probability of the event that selects v to $O(\ln \Delta) \cdot x_v^*$ (see line 2 of Algorithm 2). Algorithm 2 consists of three phases: (Phase 1) Fixed Choice Phase (lines 4-7 of Algorithm 2); (Phase 2) Local Search Phase (lines 8-18 of Algorithm 2); (Phase 3) One Vertex Phase (lines 19-22 of Algorithm 2). In the first phase, we select all vertices whose probability is 1. If the number of selected vertices already satisfies the demand requirement (line 5 of Algorithm 2), then we stop; otherwise, we enter the second phase to select more vertices. Due to some technical reason, Algorithm 2 may not be able to select sufficient vertices at the end of the second phase. Fortunately, such a demand gap is at most 1. Thus, Algorithm 2 enters the third phase and arbitrarily selects one vertex.

The choice of the scaling factor $O(\ln \Delta)$ is crucial to the Lovász Local Lemma, i.e., it ensures that the probability of the bad events (which will be defined in Definition 1) satisfies the conditions stated in Lemma 4. So, we can employ $\texttt{LocalSearch}(\cdot, \cdot)$ to find the right selection of vertices in R that ensures all bad events do not occur.

Algorithm 2 Algorithm for Degree Bounded Graphs

Input: The bipartite graph $G := (L \cup R, E)$; The maximum degree Δ of G; The demand k; The optimal fractional solution $\mathbf{x}^* = (x_v^*)_{v \in R}$.
Output: A vertex set $S \subseteq R$ with $|S| \geq k$.

1: **for** each vertex $v \in R$ **do**
2: $\quad p_v \leftarrow \min\{1, (x_v^* + \frac{1}{\Delta}) \cdot 4 \ln(2e\Delta^2)\}$.
3: **end for**
4: $S_1 \leftarrow \{ v \in R \mid p_v = 1 \}$; ▷ Phase 1
5: **if** $|S_1| \geq k$ **then**
6: $\quad$ **return** $S \leftarrow S_1$.
7: **end if**
8: **if** $|S_1| < k$ **then** ▷ Phase 2
9: $\quad$ **for** each vertex $v \in R \setminus S_1$ **do**
10: $\quad\quad$ Define $X_v \in \{0, 1\}$ s.t. $\Pr[X_v = 1] = p_v$.
11: $\quad$ **end for**
12: $\quad \mathcal{X} \leftarrow \{ X_v \}_{v \in R \setminus S_1}$.
13: $\quad \mathfrak{X} \leftarrow \texttt{LocalSearch}(\mathcal{X}, \mathcal{A} \cup \mathcal{B})$. ▷ Definition 1
14: $\quad S_2 \leftarrow \{ v \in R \setminus S_1 \mid X_v = 1 \text{ in } \mathfrak{X} \}$.
15: **end if**
16: **if** $|S_1| + |S_2| \geq k$ **then**
17: $\quad$ **return** $S \leftarrow S_1 \cup S_2$.
18: **end if**
19: **if** $|S_1| + |S_2| < k$ **then** ▷ Phase 3
20: $\quad$ Pick an arbitrary vertex v^* from $R \setminus (S_1 \cup S_2)$.
21: $\quad$ **return** $S \leftarrow S_1 \cup S_2 \cup \{ v^* \}$.
22: **end if**

4.3 Ratio Analysis

We start by introducing some notations for the purpose of analysis. Let $S_1 \subseteq R$ be the set of vertices that are selected

in the first phase by Algorithm 2. Let $|S_1| := k_1$. Recall that T^* is the optimal fractional solution that is achieved by the solution $\mathbf{x}^* = (x_v^*)_{v \in R}$. Let OPT be the maximum disagreement of the optimal solution. Let ALG be the maximum disagreement of Algorithm 2's solution.

Analysis Framework. To analyze the ratio, we distinguish three cases according to the existence of the second and third phases, i.e., Case (i): Algorithm 2 ends at the first phase (Lemma 5); Case (ii): Algorithm 2 ends at the second phase (Lemma 8); Case (iii): Algorithm 2 ends at the third phase (Lemma 9). The first phase of Algorithm 2 is equivalent to a simple deterministic rounding algorithm, and the third phase just picks at most one vertex; thus, their analysis is easy. The analysis of the second phase depends on the Lovász Local Lemma. To this end, we shall first define the bad events and show that the defined bad events achieve the desired approximation ratio as well as (almost) satisfy the demand requirement (Lemma 7). Then, we will upper bound the probability that a bad event occurs (Lemma 10, 11) and show that Algorithm 2 finds an appropriate solution in desired running time; this is the place where we use the Lovász Local Lemma.

We start with the easy case where Algorithm 2 does not enter the second phase; see Lemma 5.

Lemma 5. *If Algorithm 2 does not enter the second phase, then the following two claims are true: (i) S_1 is a feasible solution; (ii)* $\mathrm{ALG} \leq 4 \cdot \ln(2e\Delta^2) \cdot (\mathrm{OPT} + 1)$.

Now, we focus on the second case where Algorithm 2 ends at the second phase, which implies that $k_1 < k$. Let S_2 be the set of vertices that are selected by Algorithm 2 in the second phase. Let $R' := R \setminus S_1$, and if a vertex in L has no neighbors in R', we delete such a vertex. Let L' be the set of remaining vertices in L. We shall focus on the subgraph $G' := (L' \cup R', E')$. Note that for each $v \in R'$, we have $(x_v^* + \frac{1}{\Delta}) \cdot 4\ln(2e\Delta^2) < 1$. For each vertex $v \in R'$, define a random variable $X_v \in \{0, 1\}$ to indicate whether the vertex v is added to S_2. Let $\mathcal{X} := \{X_v\}_{v \in R'}$. We now give the definition of the *bad events* based on the random variable set $\mathcal{X}$, which is crucial to the Lovász Local Lemma.

Bad Events. Intuitively, the definition of bad events needs to ensure that when all of them do not occur, the following two statements are true: (i) the selected vertices in the second phase satisfy the remaining demands; (ii) the maximum disagreement of the selected vertices in the second phase is $O(\log \Delta) \cdot \mathrm{OPT}$. This motivates us to define two groups of bad events: *performance bad events* $\mathcal{A}$ and *feasibility bad events* $\mathcal{B}$. The bad event in $\mathcal{A}$ is used to ensure that the selected vertices are able to achieve a good ratio. To this end, for each vertex in $u \in L'$, we define a bad event A_u stands for $|N_{G'}(u) \cap S_2| > O(\log \Delta) \cdot \mathrm{OPT}$, where $N_{G'}(u)$ is a set of neighbors of vertex u in graph G'. The bad event in $\mathcal{B}$ is used to ensure that the number of vertices in S_2 satisfies the remaining demand requirement. To this end, we group vertices in R' into several subgroups according to the index of vertices such that each subgraph consists of Δ vertices from R'. Let $D_1, \ldots, D_\ell \subseteq R'$ be these subgroups. Note that the size of the last subgroup D_ℓ may be smaller than Δ. For each subgroup $D_i, i \in [\ell - 1]$, we define a bad event B_i to represent $|D_i \cap S_2| < \sum_{u \in D_i} x_u^*$. Intuitively, this inequality requires that the number of selected vertices in D_i is at least $\sum_{u \in D_i} x_u^*$. For the last subgroup D_ℓ, we shall not define a bad event if $\sum_{u \in D_\ell} (x_u^* + \frac{1}{\Delta}) < 1$; otherwise, D_ℓ also has a bad event B_ℓ. Intuitively, if D_ℓ has no bad event, then $\sum_{u \in D_\ell} x_u^*$ is small; thus, we will not create a big demand gap in the case where no vertex is selected from D_ℓ. The reason why we need to distinguish the last subgroup is that we have to upper bound the probability that a bad event happens; its meaning will be clear in the proof of Lemma 10. See Definition 1 for the formal definition of the bad events. An example can be found in Figure 1.

Definition 1 (Bad Events)**.** *The bad event set $\mathcal{A} \cup \mathcal{B}$ consists of two types of bad events, where bad events $\mathcal{A}$ and $\mathcal{B}$ are called performance bad events (P-BE) and feasibility bad events (F-BE), respectively.*

(P-BE). *For each vertex $u \in L'$, we have a bad event A_u in $\mathcal{A}$. Define $\mathcal{A}_u \subseteq \mathcal{X}$ as a subset of random variables, each of which corresponds to a vertex in $N_{G'}(u)$. We say A_u occurs if: $\sum_{i \in \mathcal{A}_u} X_i \geq 8 \cdot \ln(2e\Delta^2) \cdot (T^* + 1)$.*

(F-BE). *For each subgroup $D_j, j \in [\ell - 1]$, we have a bad event in $\mathcal{B}$. The last subgraph D_ℓ has a bad event if $\sum_{u \in D_\ell} (x_u^* + \frac{1}{\Delta}) < 1$; otherwise, D_ℓ does not have a bad event. For each bad event $B_j \in \mathcal{B}$, define $\mathcal{B}_j \subseteq \mathcal{X}$ as a subset of random variables, each of which corresponds to a vertex in D_j. Say B_j occurs if: $\sum_{i \in \mathcal{B}_j} X_i < \sum_{u \in D_j} x_u^*$.*

Let $\mathrm{DG}(\mathcal{A} \cup \mathcal{B})$ be the dependency graph of bad events in $\mathcal{A}$ and $\mathcal{B}$. Lemma 6 gives a lower and upper bound of the maximum degree of $\mathrm{DG}(\mathcal{A} \cup \mathcal{B})$, which will be used later to connect our problem and Lovász Local Lemma.

Lemma 6. *Let d be the maximum degree of the dependency graph $\mathrm{DG}(\mathcal{A} \cup \mathcal{B})$. Then, the following two inequalities hold: (i) $d \geq 1$; (ii) $d \leq \Delta^2$, where Δ is the maximum degree of the input bipartite graph.*

In the following, we split the proof into two parts. In the first part (Part I), we focus on an assignment $\mathfrak{X}$ of random variables in $\mathcal{X}$ that makes all bad events in $\mathcal{A} \cup \mathcal{B}$ not occur. We define $S_2 := \{v \in R' \mid X_v = 1 \text{ in } \mathfrak{X}\}$. We show that S_2 is a set of vertices that satisfies the claimed property. In the second part (Part II), we prove that Algorithm 2 is able to return such an assignment in the desired running time. The proofs in this part shall built on the Lovász Local Lemma stated in Lemma 3 and Lemma 4. The formal proofs of all these lemmas are shown in the full version [Li *et al.*, 2025].

Part I: Properties of $\mathfrak{X}$ and S_2. We start with the correctness of the feasibility bad events' definition. Formally, we shall show that when all bad events in $\mathcal{B}$ do not occur, the demand gap is at most 1 at the end of the second phase.

Lemma 7. *If Algorithm 2 enters the second phase and all bad events in $\mathcal{B}$ do not occur, then $|S_1| + |S_2| \geq k - 1$.*

Lemma 8 is mainly due to Lemma 5 and the definition of performance bad events.

Lemma 8. *If Algorithm 2 enters the second phase but does not enter the third phase, and S_2 is a solution that makes all bad events in $\mathcal{A} \cup \mathcal{B}$ do not occur, then the following two claims are true: (i) $|S_1 \cup S_2| \geq k$; (ii)* $\mathrm{ALG} \leq 12 \cdot \ln(2e\Delta^2) \cdot (\mathrm{OPT} + 1)$.

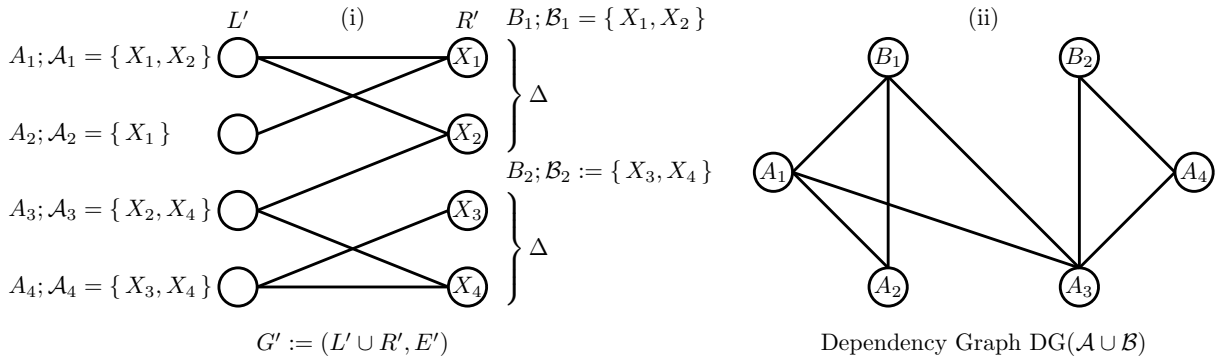

Figure 1: An example for bad events (Definition 1) and their dependency graph. The remaining graph G' has four vertices in L' and four vertices in R', and the maximum degree Δ of the original graph is 2. The bad events are shown in the subfigure (i). For each vertex v in R', there is a random variable $X_v \in \{ 0, 1 \}$ indicating whether v is selected. All bad events in $\mathcal{A} \cup \mathcal{B}$ are defined over $\mathcal{X} := \{ X_v \}_{v \in R'}$. For each vertex $u \in L'$, we have one bad event A_u in $\mathcal{A}$, and A_u is determined by a set $\mathcal{A}_u$ of random variables in $\mathcal{X}$. For example, bad event A_1 is determined by $\{ X_1, X_2 \}$ which are neighbors of vertex 1. We group all vertices in R' into several subgroups, each of which consists of at most Δ vertices in R'. Each subgroup (might) corresponds to a bad event B_v, e.g., B_1 is determined by $\mathcal{B}_1 = \{ X_1, X_2 \}$. The dependency graph of $\mathcal{A} \cup \mathcal{B}$ is shown in the subfigure (ii). The dependency graph describes the relationship between all bad events. For example, the occurrence of bad event B_1 depends on bad events A_1, A_2, A_3.

Lemma 9 can be proved by Lemma 7 and Lemma 8, i.e., adding one more vertex increases the solution's value by at most 1.

Lemma 9. *If Algorithm 2 enters the third phase and selects v^* in the third phase. Assume that S_2 is a solution that makes all bad events in $\mathcal{A} \cup \mathcal{B}$ do not occur, then the following two claims are true: (i) $S_1 \cup S_2 \cup \{ v^* \}$ is a feasible solution; (ii)* $\mathrm{ALG} \leq 12 \cdot \ln(2e\Delta^2) \cdot (\mathrm{OPT} + 2)$.

Part II: Finding $\mathfrak{X}$ and S_2. In this part, we upper bound the probability that a bad event occurs (Lemma 10 and Lemma 11). This shall prove that our bad events satisfy the condition of the Lovász Local Lemma (Lemma 4). And then, we use Lemma 3 to bound the running time of Algorithm 2.

Lemma 10. *For each bad event $B_i \in \mathcal{B}$, $\Pr[B_i \text{ occurs}] \leq \frac{1}{2ed}$, where d is the maximum degree of the dependency graph $\mathrm{DG}(\mathcal{A} \cup \mathcal{B})$.*

Lemma 11. *For each bad event $A_i \in \mathcal{A}$, $\Pr[A_i \text{ occurs}] \leq \frac{1}{2ed}$, where d is the maximum degree of the dependency graph $\mathrm{DG}(\mathcal{A} \cup \mathcal{B})$.*

Wrapping Up. Now, we are ready to prove Theorem 3.

Proof of Theorem 3. By Lemma 5, Lemma 8, and Lemma 9, we know that if Algorithm 2 returns a solution that makes all bad events not occur, then Algorithm 2 is a $12\ln(2e\Delta^2)(\mathrm{OPT} + 2)$-approximate algorithm. In the following, we show that Algorithm 2 finds a desired solution in polynomial time. By Lemma 10 and Lemma 11, we know that the probability that a bad event occurs is at most $\frac{1}{2ed}$, where d is the maximum degree of the dependency graph. By Lemma 6, we know $d \geq 1$. Hence, we have $\frac{1}{2ed} \leq \frac{1}{e(d+1)}$. Thus, we have $\Pr[C \text{ occurs}] \leq \frac{1}{e(d+1)}$ for any bad event $C \in \mathcal{A} \cup \mathcal{B}$. We define $p := \frac{1}{e(d+1)}$. Therefore, we have that $e(d+1) \cdot p = e(d+1) \cdot \frac{1}{e(d+1)} = 1 \leq 1$. Thus, the probability of defined bad events satisfies the condition of Lemma 4. Since Lemma 3 implies Lemma 4 by setting $x_C = \frac{1}{d+1}$ for each bad event $C \in \mathcal{A} \cup \mathcal{B}$, then Algorithm 2 finds a solution that makes all bad events not occur in expected time at most $\sum_{C \in \mathcal{A} \cup \mathcal{B}} \frac{x_C}{1 - x_C} = (1 + \frac{1}{d}) \cdot |\mathcal{A} \cup \mathcal{B}|$ by Lemma 3. Recall that $|L| = n$ and $|R| = m$. By Definition 1, we have $|\mathcal{A}| \leq n$ and $|\mathcal{B}| \leq \lceil \frac{m}{\Delta} \rceil \leq m$. By Lemma 6, we have $1 \leq d \leq \Delta^2$. Thus, $(1 + \frac{1}{d}) \cdot |\mathcal{A} \cup \mathcal{B}|$ is $\mathrm{poly}(n, m)$. Hence, the expected running time of Algorithm 2 is in $\mathrm{poly}(n)$, where n is the number of vertices in the input bipartite graph. □

5 Conclusion

We study the fair k-set selection problem where we aim to select k sets from a given set system such that the maximum (weighted) occurrence times that an element appears in these k selected sets are minimized. Given a bipartite graph $G := (L \cup R, E)$, our problem is equivalent to selecting k vertices from R such that the maximum (weighted) number of selected neighbors of vertices in L is minimized. We demonstrate that the problem is NP-hard when the maximum degree $\Delta = 3$, and then we give a simple optimal algorithm for the $\Delta = 2$ case. For the laminar family case, we show that the problem is also in P. We give three LP-rounding algorithms that are logarithmic approximate for general bipartite graphs and bipartite graphs with a maximum degree Δ.

This work points out many interesting directions. Firstly, it would be interesting to see whether the approximation can be further improved. This may need a more involved LP formulation. In particular, it would be interesting to see whether the problem admits an $\Omega(\frac{\log n}{\log \log n})$ lower bound. Secondly, considering a more general case is interesting where we additionally require that the selected vertices form a base of some matroid. The pipage rounding still works for this case, but it is not clear whether an $O(\log \Delta)$-approximate exists.

Acknowledgements

Chenyang Xu is supported by the National Natural Science Foundation of China (No. 62302166), the National Key Research Project of China under Grant No. 2023YFA1009402, and the Key Laboratory of Interdisciplinary Research of Computation and Economics (SUFE), Ministry of Education. Shi Li is supported by the State Key Laboratory for Novel Software Technology and the New Cornerstone Science Laboratory. We thank the anonymous reviewers for their valuable comments.

Contribution Statement

All authors in the paper have equal contributions and are corresponding authors, and thus, it is ordered alphabetically.

References

[Abdollahpouri and Burke, 2019] Himan Abdollahpouri and Robin Burke. Multi-stakeholder recommendation and its connection to multi-sided fairness. In *RMSE@RecSys*, volume 2440 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[Aziz *et al.*, 2022] Haris Aziz, Alexander Lam, Mashbat Suzuki, and Toby Walsh. Random rank: The one and only strategyproof and proportionally fair randomized facility location mechanism. In *NeurIPS*, 2022.

[Bao *et al.*, 2022] Wei-Xuan Bao, Jun-Yi Hang, and Min-Ling Zhang. Submodular feature selection for partial label learning. In *KDD*, pages 26–34. ACM, 2022.

[Beutel *et al.*, 2019] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation ranking through pairwise comparisons. In *KDD*, pages 2212–2220. ACM, 2019.

[Boehmer *et al.*, 2023] Niclas Boehmer, L. Elisa Celis, Lingxiao Huang, Anay Mehrotra, and Nisheeth K. Vishnoi. Subset selection based on multiple rankings in the presence of bias: Effectiveness of fairness constraints for multiwinner voting score functions. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 2641–2688. PMLR, 2023.

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

[Chekuri *et al.*, 2010] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, pages 575–584. IEEE Computer Society, 2010.

[De and Chakrabarti, 2022] Abir De and Soumen Chakrabarti. Neural estimation of submodular functions with applications to differentiable subset selection. In *NeurIPS*, 2022.

[Erdos and Lovász, 1975] Paul Erdos and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.

[Filmus and Ward, 2014] Yuval Filmus and Justin Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.*, 43(2):514–542, 2014.

[Garey and Johnson, 1977] M. R. Garey and David S. Johnson. The rectilinear steiner tree problem is NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

[Hazimeh and Mazumder, 2020] Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Oper. Res.*, 68(5):1517–1537, 2020.

[Knop and Schierreich, 2023] Dusan Knop and Simon Schierreich. Host community respecting refugee housing. In *AAMAS*, pages 966–975. ACM, 2023.

[Lang *et al.*, 2022] Hunter Lang, Aravindan Vijayaraghavan, and David A. Sontag. Training subset selection for weak supervision. In *NeurIPS*, 2022.

[Li *et al.*, 2023] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, Juntao Tan, Shuchang Liu, and Yongfeng Zhang. Fairness in recommendation: Foundations, methods, and applications. *ACM Trans. Intell. Syst. Technol.*, 14(5):95:1–95:48, 2023.

[Li *et al.*, 2025] Shi Li, Chenyang Xu, and Ruilong Zhang. Polylogarithmic approximation for robust s-t path. *arXiv*, abs/2505.12123, 2025.

[Mazumder *et al.*, 2023] Rahul Mazumder, Peter Radchenko, and Antoine Dedieu. Subset selection with shrinkage: Sparse linear modeling when the SNR is low. *Oper. Res.*, 71(1):129–147, 2023.

[Mehrotra and Vishnoi, 2023] Anay Mehrotra and Nisheeth K. Vishnoi. Maximizing submodular functions for recommendation in the presence of biases. In *WWW*, pages 3625–3636. ACM, 2023.

[Mirzasoleiman *et al.*, 2020] Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 6950–6960. PMLR, 2020.

[Moser and Tardos, 2010] Robin A. Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57(2):11:1–11:15, 2010.

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

[Schierreich, 2023] Simon Schierreich. Anonymous refugee housing with upper-bounds. *CoRR*, abs/2308.09501, 2023.

[Sviridenko, 2004] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

[Svitkina and Fleischer, 2011] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.*, 40(6):1715–1737, 2011.

[Tschiatschek *et al.*, 2017] Sebastian Tschiatschek, Adish Singla, and Andreas Krause. Selecting sequences of items via submodular maximization. In *AAAI*, pages 2667–2673. AAAI Press, 2017.

[Tukan *et al.*, 2023] Murad Tukan, Samson Zhou, Alaa Maalouf, Daniela Rus, Vladimir Braverman, and Dan Feldman. Provable data subset selection for efficient neural networks training. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 34533–34555. PMLR, 2023.

[Wang *et al.*, 2021] Chenhao Wang, Xiaoying Wu, Minming Li, and Hau Chan. Facility's perspective to fair facility location problems. In *AAAI*, pages 5734–5741. AAAI Press, 2021.

[Wang *et al.*, 2023] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. A survey on the fairness of recommender systems. *ACM Trans. Inf. Syst.*, 41(3):52:1–52:43, 2023.

[Yi *et al.*, 2023] Lu Yi, Hanzhi Wang, and Zhewei Wei. Optimal dynamic subset sampling: Theory and applications. In *KDD*, pages 3116–3127. ACM, 2023.

[Zhou *et al.*, 2022] Houyu Zhou, Minming Li, and Hau Chan. Strategyproof mechanisms for group-fair facility location problems. In *IJCAI*, pages 613–619. ijcai.org, 2022.