Covering a Few Submodular Constraints and Applications

Tanvi Bajpai
Siebel School for Computing and Data Science, University of Illinois, Urbana, IL, USA

Chandra Chekuri
Siebel School for Computing and Data Science, University of Illinois, Urbana, IL, USA

Pooja Kulkarni
Siebel School for Computing and Data Science, University of Illinois, Urbana, IL, USA

Abstract

We consider the problem of covering multiple submodular constraints. Given a finite ground set N, a cost function $c : N \to \mathbb{R}_+$, r monotone submodular functions $f_1, f_2, \ldots, f_r$ over N and requirements $b_1, b_2, \ldots, b_r$ the goal is to find a minimum cost subset $S \subseteq N$ such that $f_i(S) \geq b_i$ for $1 \leq i \leq r$. When $r = 1$ this is the well-known Submodular Set Cover problem. Previous work [8] considered the setting when r is large and developed bi-criteria approximation algorithms, and approximation algorithms for the important special case when each f_i is a weighted coverage function. These are fairly general models and capture several concrete and interesting problems as special cases. The approximation ratios for these problem are at least $\Omega(\log r)$ which is unavoidable when r is part of the input. In this paper, motivated by some recent applications, we consider the problem when r is a *fixed constant* and obtain two main results. When the f_i are weighted coverage functions from a deletion-closed set system we obtain a $(1+\epsilon)(\frac{e}{e-1})(1+\beta)$-approximation where β is the approximation ratio for the underlying set cover instances via the natural LP. Second, for covering multiple submodular constraints we obtain a randomized bi-criteria approximation algorithm that for any given integer $\alpha \geq 1$ outputs a set S such that $f_i(S) \geq (1 - 1/e^{\alpha} - \epsilon)b_i$ for each $i \in [r]$ and $\mathbb{E}[c(S)] \leq (1+\epsilon)\alpha \cdot \mathsf{OPT}$. These results show that one can obtain nearly as good an approximation for any fixed r as what one would achieve for $r = 1$. We also demonstrate applications of our results to implicit covering problems such as fair facility location.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Facility location and clustering; Theory of computation → Rounding techniques

Keywords and phrases covering, linear programming, rounding, fairness

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.25

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2507.09879

Funding *Tanvi Bajpai*: Supported in part by NSF grant CCF-2402667.
Chandra Chekuri: Supported in part by NSF grant CCF-2402667.
Pooja Kulkarni: Supported in part by NSF grant CCF-2402667 and CCF-2334461.

Acknowledgements CC thanks Sayan Bandyapadhyay, Tanmay Inamdar and Kasturi Varadarajan for some earlier discussions on colorful set covering problems with fixed number of colors.

1 Introduction

Covering problems are ubiquitous in algorithms and combinatorial optimization, forming the basis for a wide range of applications and connections. Among the most well-known covering problems are Vertex Cover (VC) and Set Cover (SC). In SC the input consists of a universe $\mathcal{U}$ of n elements and a family $\mathcal{S}$ of subsets of $\mathcal{U}$. The objective is to find a minimum

cardinality sub-collection $\mathcal{S}' \subset \mathcal{S}$ such that the union of the subsets in $\mathcal{S}'$ is (i.e. *covers*) $\mathcal{U}$. VC is a special case of SC where $\mathcal{U}$ corresponds to the edges of a given graph $G = (V, E)$, and the sets correspond to vertices of G. In the cost versions of these problems, each set or vertex is assigned a non-negative cost, and the objective is to find a covering sub-collection with the minimum total cost. A significant generalization of SC is the Submodular Set Cover (SubmodSC) problem where the input includes a normalized monotone submodular function $f : 2^N \to \mathbb{Z}_+$, and the goal is to find a min-cost subset S of N such that $f(S) = f(N)$. Recall that a real-valued set function f is submodular iff $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

The aforementioned covering problems are known to be NP-Hard, but classical approximation algorithms offer strong guarantees. For SC, the well-known Greedy algorithm proposed by Johnson [14] achieves a $(1 + \ln n)$-approximation and has also been shown to achieve a $(1 + \ln(\max_i f(i))$-approximation for SubmodSC by Wolsey [21]. These results are essentially tight, assuming $P \neq NP$ [11]. Nevertheless, VC admits a 2-approximation, and many other special cases of SC and SubmodSC also admit constant-factor approximations. Moreover, settling for *bi-criteria* approximations allows for additional flexibility and improvements. For SubmodSC, a slight modification to the standard Greedy algorithm yields the following tradeoff: for any integer $\alpha \geq 1$, we may obtain a set $S \subseteq N$ such that $f(S) \geq (1 - \frac{1}{e^\alpha})f(N)$ and $c(S) \leq \alpha\mathsf{OPT}$.

Another related covering problem of importance is the Partial Set Cover (PartialSC) problem, in which the input includes an additional integer b, and the goal is to find a minimum-cost collection of sets that cover *at least* b elements. A similar formulation extends to the submodular setting where the goal is to find a set S such that $f(S) \geq b$. Partial covering problems are particularly useful as they naturally model outliers for various settings, and approximation algorithms for these problems have been well studied. It is straightforward to show that PartialSC is equivalent to SC in terms of approximation. In the submodular case, this equivalence is even clearer: we can consider the truncated submodular function f_b where $f_b(S) = \min\{f(S), b\}$. In contrast, understanding the approximability of Partial Vertex Cover (PartialVC) is more nuanced. While a 2-approximation for PartialVC is known [4], it is not straight forward to see this.

In recent years, emerging applications and connections, particularly to fairness, have sparked interest in covering and clustering problems involving multiple groups or colors for elements. The goal in these problems is to ensure that some specified number of elements from each group or color are covered. The partial covering problems discussed above are a special case where the number of groups is one. In the Colorful Vertex Cover problem (ColorVC), multiple partial covering constraints are imposed. The input consists of a graph $G = (V, E)$ and r subsets of edges $E_1, E_2, \ldots, E_r$ where $E_i \subseteq E$. Each subset represents the edges of color i, and an edge can have multiple colors. Each color i has requirement $b_i \leq |E_i|$ and the objective is to find a minimum-cost set of vertices that covers at least b_i edges from each E_i. Bera et al. [2] were the first to consider (a generalization of) ColorVC, obtaining an $O(\log r)$-approximation. This result is essentially tight when r is large and part of the input, and one can show that SC reduces to ColorVC (see [2]). More recently, Bandyapadhyay et al. [1] considered ColorVC when r is a *fixed constant* and achieved a $(2 + \epsilon)$-approximation for any fixed $\epsilon > 0$.

Chekuri et al. [8] explored a broader class of problems involving the covering of multiple (submodular) functions and various special cases. Their general framework not only recovers the $O(\log r)$-approximation from [2] but also provides results for a variety of geometric set cover problems, and combinatorial optimization problems such as facility location and clustering. However, while powerful and general, their framework is designed for cases

where r (i.e., the number of constraints) is part of the input. As a result, it is inherently limited to $\Omega(\log r)$-approximation due to above mentioned reduction from SC to ColorVC. In a recent work, Suriyanarayana et al. [16] considered the Joint Replenishment Problem (JRP), a fundamental problem in supply chain optimization. JRP and its variants can be viewed as a covering problems where the objective is to satisfy demands over a time horizon for multiple items using orders that incur joint costs; readers can refer to [16] for the formal details of the JRP problem. While constant-factor approximations for JRP are well-established [3], Suriyanarayana et al. [16] introduced a colorful version of JRP (CJRP), similar in spirit to ColorVC. In CJRP, demands are grouped by color, and the goal is to satisfy at least b_i demands from each color class. Chekuri et al. [8]'s framework can yield an $O(\log r)$-approximation for CJRP; however, [16] focused on the setting where r is a fixed constant. They developed an intricate algorithm which achieves a $(2.86 + \epsilon)$-approximation for any fixed $\epsilon > 0$. Note that the approximation ratio does not depend on r but the running time is exponential in r. There are several settings in approximation for covering and packing problems where one has a fixed number of constraints or objectives and one obtains similar trade offs.

These recent developments inspire us to explore whether the general framework in [8], which applies to a wide variety of covering problems, can be adapted to the setting of a fixed number of submodular covering constraints. The goal is to remove the dependence of the approximation ratio on r. Specifically, we consider two settings: (1) the Multiple Submodular Covering problem, and (2) Colorful Set Cover and a generalization.

Multiple Submodular Covering (MSC). The MSC problem generalizes the above ColorSC problem. Formally, the input to the MSC problem consists of r monotone submodular functions $f_i : 2^N \to \mathbb{Z}_+$ over a finite ground set N (provided as value oracles), a non-negative cost function $c : N \to \mathbb{R}_+$, and non-negative integer requirements $b_1, b_2, \ldots, b_r$. The objective is to find a minimum-cost set $S \subseteq N$ such that $f_i(S) \geq b_i$ for each $i \in [r]$. Note that when $r = 1$, this corresponds to SubmodSC. The Greedy algorithm (modified slightly) yields the following trade off for SubmodSC: for any integer $\alpha \geq 1$ the algorithm outputs a set S such that $f(S) \geq (1 - 1/e^{\alpha})b$ and $c(S) \leq \alpha\mathsf{OPT}$. We note that this trade off is essentially tight for any fixed α via the hardness result for SC [11]. It is easy to reduce multiple submodular covering constraints to a single submodular covering constraint [12, 8], but in doing so one is limited to $\Omega(\log f(N))$ approximation to the cost to satisfy all constraints exactly – it is not feasible to distinguish the individual constraints via this approach. In contrast, Chekuri et al.[8] obtained the following bi-criteria approximation result for MSC: there is an efficient randomized algorithm that outputs a set $S \subseteq N$ such that $f(S) \geq (1 - 1/e - \epsilon)b_i$ for each $i \in [r]$ and $E[c(S)] \leq O(\frac{1}{\epsilon} \log r)$. In this paper we ask whether it is possible to achieve bi-criteria bounds for MSC instances with a fixed number of constraints that match those obtainable for single submodular constraint. We prove the following theorem and its corollary which together affirmatively answers this (modulo a small dependence on ϵ).

▶ **Theorem 1.** *There is a randomized polynomial-time algorithm that given an instance of the* MSC *with fixed r and $\epsilon > 0$ outputs a set $S \subseteq N$ such that (i) $f_i(S) \geq (1 - 1/e - \epsilon)b_i$ for all $i \in [r]$ and (ii) $E[c(S)] \leq (1 + \epsilon)\mathsf{OPT}$ (where* OPT *is the cost of the optimal solution to the instance).*

▶ **Corollary 2.** *For any fixed integer $\alpha > 1$ and ϵ, Algorithm 1 (from Theorem 1) can be used to construct a solution $S \subseteq N$ such that $f(S) \geq (1 - 1/e^{\alpha} - \epsilon)b_i$ for all $i \in [r]$ and $E[c(S)] \leq \alpha(1 + \epsilon)\mathsf{OPT}$.*

We remark that the underlying algorithms give *additive* guarantees. In terms of running time, the most expensive part is to convert this additive guarantee into a multiplicative guarantee by guessing sufficiently many elements from some fixed optimum solution. The additive guarantees provide useful insights and are often sufficient if one assumes that the optimum value is large.

Colorful Set Cover (ColorSC) and generalizations. Prior work of Inamdar and Varadarajan [13] showed a general setting when positive results for SC can be extended to PartialSC ($r = 1$). We describe the setting first. A class of SC instances (i.e. set systems), $\mathcal{F}$, is called *deletion-closed* if for any given instance in the class, removing a set or an element of the universe results in an instance that also belongs to $\mathcal{F}$. Many special cases of SC are naturally deletion-closed. For example, VC instances are deletion-closed as are many geometric SC instance (say covering points by disks in the plane). Suppose one can obtain a β-approximation to SC on instances from $\mathcal{F}$ via the natural LP relaxation. For example, VC has $\beta = 2$ and there are many geometric settings where $\beta = O(1)$ or $o(\log n)$ [6, 13]. Inamdar and Varadarajan [13] additionally formalized the concept of deletion-closed set systems and obtained a $2(\beta+1)$-approximation for PartialSC on a set system from deletion-closed $\mathcal{F}$. Their framework gave a straightforward way to approximate geometric partial covering problems by building on existing results for SC. Chekuri et al. [8] extended this framework to address deletion-closed instances of ColorSC problem (which is analogous to the previously discussed ColorVC). Formally, the input for ColorSC consists of a set system from $\mathcal{F}$, costs on the sets, and r sets $\mathcal{U}_1, \ldots, \mathcal{U}_r$ where each $\mathcal{U}_i$ is a subset of the universe $\mathcal{U}$, and non-negative integer requirements $b_1, \ldots, b_r$. The objective is to find a minimum-cost collection of sets such that for each $i \in [r]$, at least b_i elements from $\mathcal{U}_i$ are covered. In this setting, Chekuri et al. [8] achieved two key results. First, when $r = 1$, they improved the approximation bound from [13] to $\frac{e}{e-1}(\beta + 1)$. Second, for general r they obtained an $O(\beta + \log r)$-approximation which generalized [2] for ColorVC. Here we consider this same setting but when r is a fixed constant and obtain the following result.

▶ **Theorem 3.** *For any instance of* ColorSC *with fixed r from a β-approximable deletion-closed set system, and any fixed $\epsilon > 0$, there is a randomized polynomial-time algorithm that yields a $\left(\frac{e}{e-1}\right)(1+\beta)(1+\epsilon)$-approximate feasible solution.*

Chekuri et al. [8] prove a more general result that applies to covering integer programs induced by deletion-closed set systems (this is similar to the context considered by Bera et al. [2] for ColorVC). This generality is significant in applications such as facility location and clustering. We obtain the preceding result also in this more general setting (see Theorem 15); we defer the formal description of the problem setting to Section 2, and prove the theorem in Section 5.

Implications and Applications. At a high level, our two main theorems show that approximation results for covering problems with a single submodular constraint, i.e. where $r = 1$, can be extended to the case with a fixed number of constraints. We believe that these results, in addition to being independently interesting, are broadly applicable and valuable due to the modeling power of submodularity and set covering. However, one should expect additional technical details in applying these results to certain applications such as facility location, clustering, JRP, and others. In this paper, we prove our main technical results and lay down the general framework. We also demonstrate how the framework can be applied and adapted. We discuss these applications next.

Colorful covering problems: Recall that ColorVC admits a $(2+\epsilon)$-approximation for fixed r [1]. Now consider the problem of covering points in plane by disks with costs. This well-studied geometric set cover problem admits an $O(1)$ approximation via the natural LP [6, 18]. [13, 8] obtained a $\left(\frac{e}{e-1}\right)(1+\beta)$ approximation for PartialSC version of this problem ($r = 1$) and [8] obtained an $O(\beta + \log r)$-approximation for the colorful version of this problem (arbitrary r). In this paper we obtain a $\left(\frac{e}{e-1}\right)(1+\beta)(1+\epsilon)$ approximation for any fixed r. In effect all the PartialSC problems considered in [13, 8] for various geometric settings admit essentially the same approximation for the colorful versions with fixed r. The colorful problems capture outliers and fairness constraints, and thus, as long as the number of such constraints is small, the approximation ratio is independent of r. Our bi-criteria approximation for MSC, in a similar vein, allows tight trade offs in cost vs coverage for any fixed number of constraints. See [12, 8] for an application to splitting point sets by lines where a bi-criteria approximation is adequate.

Facility location with multiple outliers: Facility location and clustering problems have long been central in both theory and practice, with a wide variety of objective functions studied in the literature. At a high level, the goal in facility-location problems is to select a subset of facilities and assign clients to them in a way that balances assignment cost with facility opening cost. Several objectives and variants are studied in the literature. Motivated by robustness considerations there is also extensive work on these problems when there are outliers. The goal in such problems is to solve the underlying clustering or facility location problem where certain number of points/clients can be omitted; equivalently the goal is to connect/cluster at least a given number of points. These problems do not fall neatly in the framework of (partial) covering problems. Nevertheless it is possible to view them as *implicit* covering problems; in some settings one needs to consider an exponential sized set system. Thus, although the problems cannot be directly cast in as MSC or CCF, some of the ideas can be transferred with additional technical work. [8] used this perspective and additional ideas to obtain $O(\log r)$-approximation for two facility location problems with multiple outlier classes. In this paper, we consider these two problems when r is a fixed constant and obtain constant factor approximations. The first one involves facility location to minimize the objective of sum of radii; it is easier to adapt the ideas for CCF to this problem and we discuss this in the full version of our paper.

Our main new technical contribution is the second problem related to the outlier version of the well-known uncapacitated facility location problem (UCFL). In UCFL the input consists of a facility set F and a client set C that are in a metric space $(F \cup C, d)$. Each facility $i \in F$ has an opening cost $f_i \geq 0$ and connecting a client j to a facility i costs $d(i, j)$. The goal is to open a subset of the facilities $S \subseteq F$ to minimize $\sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$ where $d(j, S)$ is the distance of j to the closest open facility. This classical problem has seen extensive work in approximation algorithms; the current best approximation ratio is 1.488 [15]. Charikar et al. [7] studied the outlier version of this problem (under the name robust facility location) and obtained a 3-approximation; in this version the goal is to connect at least some specified number $b \leq |C|$ of clients, and this corresponds to having a single color class to cover. In [8], the generalization of this to r outlier classes was studied under the name Facility Location with Multiple Outliers (FLMO). In this paper we prove the following result.

▶ **Theorem 4.** *Let β be the approximation ratio for* UCFL *via the natural LP relaxation. Then there is a randomized polynomial-time algorithm that given an instance of* FLMO *with fixed r and $\epsilon > 0$ yields a $\frac{e}{e-1}(\beta + 1 + \epsilon)$-approximate solution.*

We remark that we use LP-based algorithms for UCFL as a black-box and have not attempted to optimize the precise ratio. Using $\beta = 1.488$ from [15] we get an approximation factor of $3.936(1+\epsilon)$. We also note that the case of r being fixed requires new technical ideas from those in [8]. We recall the non-trivial work of [16] who consider CJRP for fixed r by adapting the ideas from [8] for an implicit problem. One of our motivations for this paper is to obtain a general framework for CCF for fixed r, and to showcase its applicability to concrete problems.

Summary of Technical Contributions. Our algorithms follow a streamlined framework (outlined in Section 3.2) that builds upon the ideas in [8], while introducing additional new ingredients to obtain improved approximations when r is fixed. The framework consists of four stages; at a high level, these stages can are: guess high-cost elements, solve a continuous relaxation, randomly round the fractional relaxation, and greedy fix in a post-processing step to ensure that constraints are appropriately satisfied. The main new ingredient is the rounding procedure employed in the third stage, which is centered around a rounding lemma (Lemma 10). The lemma is based on the concentration bound for independent rounding of a fractional solution to the multilinear relaxation of a submodular function. The improved approximation for fixed r is based on obtaining a strong concentration bound. In order to obtain this we need to ensure that the underlying submodular function has a Lipschitz property. Our new insight is that one can use a greedy procedure to select a small number of elements to ensure the desired Lipschitz property. However this means that the algorithms will incur an additive approximation for selecting a constant number (that depends only on r and $1/\epsilon$) of elements. We are able to convert this additive approximation into a multiplicative approximation by guessing a sufficiently large number of elements in the first stage. We note that several packing and covering problems with a fixed number of constraints follow this high-level template. Typically one uses properties of basic feasible solutions to underlying LP relaxations for packing and covering problems. We cannot use standard methods because our constraints are submodular and hence non-linear. Our fractional solutions come from solving non-linear programs or auxiliary LPs with many constraints. This is the reason for relying on our rounding approach. The high-level overview suppresses several non-trivial technical details in the algorithms, some of which are inherited from [8]. There are important differences in the details for the two problems, and are explained in the relevant sections.

Organization. Section 2 provides the formal definitions for MSC and CCF, as well as some relevant background on submodularity. In Section 3 we describe the Rounding Lemma (Lemma 10), which is our key ingredient. In the same section, we provide an overview of our algorithmic framework. Section 4 describes our algorithm for Theorem 1, and Section 5 describes our algorithm for CCF which generalizes Theorem 3. Finally, in Section 6, we discuss the application of CCF to Facility Location. We sketch the analysis of the algorithms in the respective sections and defer the reader to the full version for complete details.

2 Notation and Preliminaries

2.1 Problem Definitions

Below we provide the formal problem statements for Multiple Submodular Covering Constraints (MSC) and Covering Coverage Functions (CCF) problems [8].

▶ **Definition 5** (MSC). *Let N be a finite ground set, and we are given r monotone submodular functions $f_i : 2^N \to \mathbb{R}_+$ for $i \in [r]$ each with corresponding demands $b_i \in \mathbb{R}_+$. Additionally, we are given a non-negative cost function $c : N \to \mathbb{R}_+$. The goal is to find a subset $S \subseteq N$ that solves the following optimization problem:*

$$\min_{S \subseteq N} \quad c(S) \quad s.t. \quad f_i(S) \geq b_i \quad \forall i \in [r].$$

▶ **Definition 6** (CCF). *The* CCF *problem consists of a set system $(\mathcal{U}, \mathcal{S})$ where $\mathcal{U}$ is the universe of n elements, $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a collection of m subsets of $\mathcal{U}$. Each set $S \in \mathcal{S}$ has a cost associated with it and we are given a set of inequalities $A\mathbf{z} \geq \mathbf{b}$ where $A \in \mathbb{R}^{r \times n}_{\geq 0}$. The goal is to optimize the integer program given in* IP-CCF.

$$\begin{aligned} & \min \textstyle\sum_{i \in [m]} c_i x_i \\ \text{s.t.} \quad & \textstyle\sum_{i : j \in S_i} x_i \geq z_j \\ & A\mathbf{z} \geq b \\ & z_j \leq 1 \qquad \text{for all } j \in [n] \\ & x_i \in \{0,1\} \qquad \text{for all } i \in [m] \end{aligned} \tag{IP-CCF}$$

$$\begin{aligned} & \min \textstyle\sum_{i \in [m]} c_i x_i \\ \text{s.t.} \quad & \textstyle\sum_{i : j \in S_i} x_i \geq z_j \\ & A\mathbf{z} \geq b \\ & z_j \leq 1 \qquad \text{for all } j \in [n] \\ & x_i \in [0,1] \qquad \text{for all } i \in [m] \end{aligned} \tag{LP-CCF}$$

In this program, the variables x_i for $i \in [m]$ represent the indicator variable for whether the set i is selected. The variables z_j for $j \in [n]$ are indicators for whether the element j is covered. If entries of A are in $\{0,1\}$ then we obtain ColorSC as a special case.

CCF as a special case of MSC. We view the constraints as submodular functions as follows: for the i^{th} constraint, given by the i^{th} row of matrix, we define a submodular function $f_i(\cdot) : 2^{\mathcal{S}} \to \mathbb{R}^+_{\geq 0}$ as follows: $f_i(\mathcal{S}) = \sum_{j \in [n]} A_{i,j} z_j$ where z_j is a variable indicating whether $j \in \cup_{S \in \mathcal{S}} S$ i.e., whether j is covered by a set in the collection $\mathcal{S}$ of sets, and $A_{i,j}$ is the entry in i^{th} row and j^{th} column of matrix A. This is a submodular function since it is a weighted coverage function with weights coming from the matrix A. In Section 5, we will use either the matrix form or the submodular function form as convenient.

In this paper we assume that input instances of each problem will have a fixed number of constraints, i.e., r is taken to be some positive fixed constant.

2.2 Submodularity

For a submodular function, $f(\cdot)$ defined on ground set N, we will use $f_{|A}(\cdot)$ to denote the submodular function that gives marginals of f on set $A \subseteq N$ i.e., $f_{|A}(S) = f(S \cup A) - f(A)$. Further, for any $A \subseteq N$ and $e \in N$, we will use $f(e \mid A)$ to denote the marginal value of e when added to A.

Multilinear Extensions of Set Functions. Our main rounding algorithm (discussed in Section 3) makes use of the multilinear extension of set functions. Here are some relevant preliminaries.

▶ **Definition 7** (Multilinear extension). *The multilinear extension of a real-valued set function $f : 2^N \to R$, denoted by F, is defined as follows: For $x \in [0,1]^N$*

$$F(x) = \sum_{S \subseteq N} f(S) \prod_{i \in S} x_i \prod_{j \in N \setminus S} (1 - x_j).$$

Equivalently, $F(x) = \mathbb{E}[f(R)]$ where R is a random set obtained by picking each $i \in N$ independently with probability x_i.

Lipschitzness and Concentration Bounds. One benefit of utilizing the multilinear extension that is relevant to our algorithm is a concentration bound. Before we state that bound, we define *Lipschitzness* of a submodular function.

▶ **Definition 8** (ℓ-Lipschitz). *A submodular function $f : 2^N \to \mathbb{R}_+$ is ℓ-Lipschitz if $\forall i \in N$ and $A \subset N$: $|f(A \cup \{i\}) - f(A)| \leq \ell$. When f is monotone, this amounts to $f(\{i\}) \leq \ell$.*

▶ **Lemma 9** (Vondrák [20]). *Let $f : 2^N \to \mathbb{R}_+$ be an ℓ-Lipschitz monotone submodular function. For $\mathbf{x} \in [0,1]^N$, let R be a random set where each element i is drawn independently using the marginals induced by $\mathbf{x}$. Then for $\delta \geq 0$,*

$$Pr[f(R) \leq (1-\delta)F(\mathbf{x})] \leq e^{-\frac{\delta^2 F(\mathbf{x})}{2\ell}}.$$

Discretizing costs and guessing. Our problems involve minimizing the cost of objects. Via standard guessing and scaling tricks, if we are willing to lose a $(1 + o(1))$ factor in the approximation, we can assume that all costs are integers and are polynomially bounded in n. In particular this implies that there are only $\text{poly}(n)$ choices for the optimum value. We will thus assume, in some algorithmic steps, knowledge of the optimum value with the implicit assumption that we are guessing it from the range of values. This will incur a polynomial-overhead in the run-time, the details of which we ignore in this version.

3 High-Level Algorithmic Framework and Rounding Lemma

Our algorithms for approximating MSC and CCF share a unified framework. The framework relies on a Rounding Lemma (Lemma 10), which provides a method to convert a fractional solution to an integral one in a way that preserves key properties of the original fractional solution while incurring only a minimal additive cost. In this section, we begin by presenting and proving our Rounding Lemma (Section 3.1). We then provide a high-level overview of the algorithmic framework (Section 3.2) built upon this Rounding Lemma.

3.1 Rounding Lemma

A key step in our framework is the ability to round fractional solutions to integral ones while maintaining important guarantees. Our Rounding Lemma (stated below) provides a formal method for achieving this. It shows that for r submodular functions ($r \in O(1)$), a fractional solution satisfying certain constraints with respect to their multilinear extensions can be (randomly) rounded in polynomial time to an integral solution. This rounding incurs only a small (constant sized) additive increase in cost and ensures that the values of the submodular functions are sufficiently preserved with high probability.

▶ **Lemma 10** (Rounding Lemma). *Let N be a finite set, $c : N \to \mathbb{R}_+$ be a non-negative cost function and let $c_{\max} = \max_{j \in N} c_j$. Let $f_1, \ldots, f_r$ be monotone submodular functions over N. For each f_i, let F_i be its corresponding multilinear extension. Suppose we are given a fractional point $\mathbf{x} \in [0,1]^N$ such that $\sum_j c_j x_j \leq B$ and $F_i(\mathbf{x}) \geq b_i$ for each $i \in [r]$. Then, for any fixed $\epsilon > 0$ there is a randomized algorithm that outputs a set $S \subseteq N$ such that (i) $\mathbb{E}[c(S)] \leq B + r\lceil \frac{2\ln(r/\epsilon)\ln(1/\epsilon)}{\epsilon^2} \rceil c_{\max}$, and (ii) $\forall i \in [r]$ $f_i(S) \geq (1-\epsilon)b_i$ with probability at least $1 - \epsilon/r$.*

To prove this lemma, we first establish the following helpful claim, which ensures that after selecting a subset of elements, a monotone function becomes Lipschitz continuous (Definition 8), with a Lipschitz constant ℓ that depends on the number of elements chosen.

▷ Claim 11. Let N be a finite set and f be a non-negative monotonic[1] function over N. For any $\epsilon > 0$, $b > 0$ and $\ell > 0$, there exists a polynomial time algorithm that returns a set $S \subseteq N$ of cardinality at most $\lceil \frac{1}{\ell} \ln(\frac{1}{\epsilon}) \rceil$ such that one of the following conditions hold: (i) $f(S) \geq (1-\epsilon)b$, or (ii) $\forall e \in N \setminus S$, $f(e \mid S) < \ell \cdot (b - f(S))$.

Proof. Consider the following greedy procedure to construct S: Initialize S as the empty set. While there exists an element $e \in N \setminus S$ satisfying $f(e \mid S) \geq \ell \cdot (b - f(S))$ and $f(S) < (1-\epsilon)b$, add e to S. Finally, return S.

We prove that in at most $\lceil \frac{1}{\ell} \ln \frac{1}{\epsilon} \rceil$ iterations of the while loop, the set S must satisfy (i) or (ii). Note that trivially, the returned set S satisfies one of these conditions. In the rest of the proof we will bound the cardinality of S.

Suppose that we never satisfy (ii) in $\lceil \frac{1}{\ell} \ln \frac{1}{\epsilon} \rceil$ iterations. We will show using induction that we must then satisfy (i). In particular, let $S^{(t)}$ denote the set S after t iterations of the while-loop in the given procedure (where $S^{(0)} = \emptyset$), and let s_{t+1} be the element that is to be added to S in the $(t+1)$-th iteration. Now, using induction on t, we will show that for all $t \geq 0$ the following inequality will hold

$$f(S^{(t)}) \geq (1 - (1-\ell)^t)b.$$

For $t = 0$, S is the empty set, hence $f(S^{(0)}) = 0$, and $(1 - (1-\ell)^0)b = 0$. Assume $f(S^{(t)}) \geq (1 - (1-\ell)^t)b$ for some fixed t. We will show that the inequality must hold for $t+1$. To see this, observe that if (ii) does not hold, then there exists an element, s_{t+1} such that $f(s_{t+1} \mid S^{(t)}) \geq \ell(b - f(S^{(t)}))$. Therefore we will have,

$$\begin{aligned} f(S^{(t+1)}) &= f(s_{t+1} \mid S^{(t)}) + f(S^{(t)}) \\ &\geq \ell(b - f(S^{(t)}) + f(S^{(t)}) \\ &= \ell b + (1-\ell) f(S^{(t)}) \\ &\geq \ell b + (1-\ell)(1 - (1-\ell)^t)b \\ &= (1 - (1-\ell)^{t+1})b. \end{aligned}$$

Therefore, for $t = \lceil \frac{\ln \epsilon}{\ln(1-\ell)} \rceil$, $(1 - (1-\ell)^t)b \geq (1-\epsilon)b$ holds. Thus, we are guaranteed that after selecting $\frac{\ln \epsilon}{\ln(1-\ell)} \leq \lceil \frac{1}{\ell} \ln \frac{1}{\epsilon} \rceil$ elements, if (ii) is not met (i) holds. ◁

Using Claim 11, we prove the Rounding Lemma.

Proof of Rounding Lemma (Lemma 10). Consider the following algorithm. For a value of ℓ that we will determine later, for each $i \in [r]$, create a set S_i according to the greedy algorithm in Claim 11. We are therefore guaranteed that for each constraint $i \in [r]$ $f_i(S_i) \geq (1-\epsilon)b_i$ or $f_i(e \mid S_i) \leq \ell(b_i - f(S_i))$ for all $e \in N \setminus S_i$. Note that S_i's may not be disjoint. Next, perform independent randomized rounding with marginal probabilities given by $\mathbf{x}$ to find a set S'. Let $S = \cup_{i \in [r]} S_i \cup S'$.

We first prove that each constraint is satisfied to a $(1-\epsilon)$ approximation with probability $1 - \epsilon/r$. Fix any constraint $i \in [r]$. Suppose the set S_i returned from Claim 11's algorithm satisfies $f_i(S_i) \geq (1-\epsilon)b_i$, then we are done. Otherwise, we know that for all $e \in N \setminus S_i$, $f(e \mid S_i) < \ell(b - f_i(S_i))$. Denote by $f_{i|S_i}$ the submodular function that is defined by $f_{i|S_i}(X) \coloneqq f_i(X \mid S_i)$. Let $b'_i = b_i - f_i(S_i)$. We have that for all $e \in N \setminus S_i$, $f_i(e \mid S_i) \leq \ell b'_i$ and therefore, $f_{i|S_i}(e) \leq \ell b'_i$ for all $e \in N \setminus S_i$. Then, when we (independently) randomly round using marginal probabilities given by $\mathbf{x}$, we get using Lemma 9 for the function $f_{i|S_i}$[2],

[1] Note that we do not need submodularity in this claim.

[2] Note that the Lemma 9 applies when we round elements in $N \setminus S_i$. Here we round all elements in N and, since f_i is monotone, this is guaranteed to have a higher value than rounding some of them, therefore we can safely apply the concentration bound.

$$\Pr[f_{i|S_i}(S') \leq (1-\epsilon)F_{i|S_i}(x)] \leq e^{-\frac{\epsilon^2 F_{i|S_i}(x)}{2\ell b'_i}} \tag{1}$$

Now, $b'_i = b_i - f_i(S_i) \leq F_i(\mathbf{x}) - f_i(S_i) = F_i(\mathbf{x}) - F_i(S_i)$. Furthermore,

$$\begin{aligned}
F_i(\mathbf{x}) - f_i(S_i) &= \sum_{S\sim\mathbf{x}} f_i(S)\Pr[S] - f_i(S_i) && \text{(By Definition 7)}\\
&= \sum_{S\sim\mathbf{x}} f_i(S)\Pr[S] - \sum_{S\sim\mathbf{x}} f_i(S_i)\Pr[S] && \text{(Since the probabilities sum to one)}\\
&= \sum_{S\sim\mathbf{x}} (f_i(S) - f_i(S_i))\Pr[S]\\
&\leq \sum_{S\sim\mathbf{x}} (f_i(S\cup S_i) - f_i(S_i))\Pr[S]\\
&= \sum_{S\sim\mathbf{x}} f_{i|S_i}(S)\Pr[S] = F_{i|S_i}(\mathbf{x}) && \text{(Using monotonicity of } f_i \text{ and Definition 7)}
\end{aligned}$$

Therefore, $b'_i \leq F_{i|S_i}(\mathbf{x})$. Substituting this in Equation 1, $\Pr[f_{i|S_i}(S') \leq (1-\epsilon)F_{i|S_i}(x)] \leq e^{-\frac{\epsilon^2 b'_i}{2\ell b'_i}}$ Now, choosing $\ell = \frac{\epsilon^2}{2\ln(r/\epsilon)}$, we have $\Pr[f_{i|S_i}(S') \leq (1-\epsilon)F_{i|S_i}(x)] \leq e^{-\frac{\epsilon^2}{2\epsilon^2}\cdot 2\ln(r/\epsilon)} = \frac{\epsilon}{r}$. Therefore, with probability at least $1 - \epsilon/r$,

$$\begin{aligned}
f_i(S) &= f_i(S_i) + f_i(S' \mid S_i) \geq f_i(S_i) + (1-\epsilon)F_{i|S_i}(x)\\
&\geq (1-\epsilon)F_i(x) = (1-\epsilon)b_i.
\end{aligned}$$

Now, we will look at the cost of the sets chosen. To get the required probability bound, we need to choose $\ell = \frac{\epsilon^2}{2\ln(r/\epsilon)}$. Therefore, following claim 11, the size of set S_i for each $i \in [r]$ is at most $\lceil \frac{1}{\ell}\ln\frac{1}{\epsilon}\rceil = \lceil \frac{2\ln(r/\epsilon)\ln(1/\epsilon)}{\epsilon^2}\rceil$. Therefore, together the total cardinality of $\cup_{i\in[r]} S_i$ is bounded by $r\lceil \frac{2\ln(r/\epsilon)\ln(1/\epsilon)}{\epsilon^2}\rceil$. Finally, the expected cost of S' is B. Together, we get $\mathbb{E}[c(S)] \leq B + r\lceil \frac{2\ln(r/\epsilon)\ln(1/\epsilon)}{\epsilon^2}\rceil c_{\max}$.

It is easy to see that the algorithm runs in polynomial time. ◀

3.2 Overview of Algorithmic Framework

Our algorithmic framework consists of four stages. Let N denote the initial set of objects, and let r denote the number of covering constraints.

- **Stage 1: Guessing Highest-Cost Objects from the Optimal Solution.** First, we "guess" S_{pre} as the L highest-cost objects from the optimal solution, where L is chosen to be some sufficiently large constant number that depends on r and ϵ. By guessing we mean enumerating all possible L-sized subsets of objects from N. We use this subset to construct a *residual instance* in which all objects with costs higher than any object in S_{pre} are discarded.
- **Stage 2: Constructing a Fractional Solution.** Let N' denote the set of objects that remain after Stage 1. We solve a continuous relaxation to obtain a a fractional solution $\mathbf{x} \in [0,1]^{N'}$ to the residual instance.
- **Stage 3: Rounding the Fractional Solution.** $\mathbf{x}$ is rounded into an integral solution $S \subseteq N'$ using Rounding Lemma (Lemma 10). S satisfies each covering constraint with adequate probability while incurring a small additive cost.
- **Stage 4: Fixing S.** Finally, since S only satisfies the constraints probabilistically, additional adjustments are needed to ensure all constraints are satisfied. This is done in a simple way by fixing each constraint individually by greedy methods.

While our algorithms for MSC (Section 4) and CCF (Section 5) follow the high-level framework described above, the execution of stages have several different aspects that are covered in the next two sections.

4 Multiple Submodular Cover

We prove Theorem 1 (restated below). In the full version, we show how this algorithm can be extended to prove Corollary 2.

We will use $\mathcal{J}$ to denote the input instance of MSC, which consists of a finite ground set N, a non-negative cost function $c : N \to \mathbb{R}_+$, and $\forall i \in [r]$ monotone submodular functions f_i with associated requirement b_i. Recall that we assume r is fixed. We will assume (without loss of generality) that $f_i(N) = b_i$ for all $i \in [r]$.

▶ **Theorem 1.** *There is a randomized polynomial-time algorithm that given an instance of the* MSC *with fixed r and $\epsilon > 0$ outputs a set $S \subseteq N$ such that (i) $f_i(S) \geq (1 - 1/e - \epsilon)b_i$ for all $i \in [r]$ and (ii) $E[c(S)] \leq (1 + \epsilon)$OPT (where* OPT *is the cost of the optimal solution to the instance).*

4.1 Algorithm Preliminaries

4.1.1 Constructing residual MSC instances

Recall that Stage 1 of our algorithmic framework (Section 3.2) constructs a residual instance after guessing a subset of elements from the optimal solution. Below is a definition which describes how to construct a residual and cost-truncated residual MSC instances.

▶ **Definition 12** (Residual and Cost-Truncated Residual MSC Instances)**.** *Given an* MSC *instance $\mathcal{J} = (N, \{f_i, b_i\}_{i\in[r]}, c)$ and a set $A \subseteq N$, a **residual instance of $\mathcal{J}$ with respect to A** is $\mathcal{J}' = (N', \{f'_i, b'_i\}_{i\in[r]}, c)$ where $N' := N \setminus A$, for all $i \in [r]$, $f'_i(\cdot) := f_{i|A}(\cdot)$ and $b'_i := b_i - f_i(A)$. We do not change c since the cost values remain unchanged, but assume its domain is restricted to N'. A **cost-truncated residual instance** of $\mathcal{J}$ with respect to A is a residual instance $\mathcal{J}' = (N', \{f'_i, b'_i\}_{i\in[r]}, c)$ in which elements in N' that have a higher cost than any element in A are also removed. More precisely, $N' = \{e \mid e \in N \setminus A$ and s.t. $\forall e' \in A\ c(e) \leq c(e')\}$.*

4.1.2 Continuous relaxation of MSC and approximation results

To construct the desired fractional $\mathbf{x}$ in Stage 2, we use the continuous relaxation of MSC, stated below as a feasibility program in MSC-Relax. For an instance $\mathcal{J} = (N, \{f_i, b_i\}_{i\in[r]}, c)$, the objective of MSC-Relax is to find a fractional point $\mathbf{x} \in [0, 1]^N$ whose cost is at most C and where the value of the multilinear extension of each constraint f_i, denoted by F_i, at $\mathbf{x}$ satisfies the demand b_i.

Note that for $C \geq$ OPT, where OPT is the cost of the optimal solution to the MSC instance, MSC-Relax is guaranteed to be feasible.

$$\begin{aligned} & \textstyle\sum_j c_j x_j \leq C \\ \text{s.t.} \quad & F_i(\mathbf{x}) \geq b_i \quad \forall i \in [r] \\ & \mathbf{x} \geq 0 \end{aligned} \tag{MSC-Relax}$$

Finding a feasible solution to the above continuous optimization problem is NP-Hard. The following result can be used to get an approximate solution.

▶ **Theorem 13** (Chekuri, Vondrák, and Zenklusen [10] and Chekuri, Jayram, and Vondrák [9])**.** *For any fixed $\epsilon > 0$, there is a randomized polynomial-time algorithm that, given an instance of* MSC*-Relax and a value oracle access to the submodular functions $f_1, ..., f_r$, with high probability, either correctly outputs that the instance is not feasible or outputs an $\mathbf{x}$ such that (i) $\sum_j c_j x_j \leq C$ and (ii) $F_i(\mathbf{x}) \geq (1 - 1/e - \epsilon)b_i \ \forall i \in [r]$.*

4.1.3 Algorithm to fix MSC constraints

In Stage 4 of our MSC algorithm, we need to fix unsatisfied constraints, and for this we can focus on a single submodular covering constraint. The relevant optimization problem is the following:

$$\min c(T) \text{ s.t. } f(T) \geq b$$

Suppose the optimum cost for this is C^*. Our goal is to show that we can find a set S such that $f(S) \geq (1 - 1/e)b$ and $c(S) \leq C^*$. For this purpose we guess C^* and recast the problem as a submodular function *maximization* problem subject to a knapsack constraint with budget C^*.

$$\max f(T) \text{ s.t. } c(T) \leq C^*$$

Note that there is a feasible solution to this maximization problem of value at least b. We can now use the following result. It is based on slight modification of the standard greedy algorithm.

▶ **Lemma 14** (Sviridenko [17])**.** *There is a $(1-1/e)$-approximation for monotone submodular function maximization subject to a knapsack constraint.*

4.2 Algorithm for MSC

We provide pseudo-code for our algorithm in Algorithm 1 and demarcate the various steps involved in each of the framework stages outlined in Section 3.2. We additionally provide more detailed descriptions of how each of the four stages in the context of our MSC approximation.

Stage 1: Choosing L and constructing $\mathcal{J}'$. In the first stage we guess sufficiently many (i.e. L) high-cost elements from the optimal solution, denoted by S^*. To properly set L, we define ϵ' and ϵ'' such that $0 < \epsilon', \epsilon'' \leq \epsilon$ and $(1 - \epsilon'')(1 - 1/e - \epsilon') \leq (1 - 1/e - \epsilon)$. Next, we define $\ell := \frac{(\epsilon'')^2}{2 \ln r/\epsilon''}$. With this, we finally set $L := r \cdot \lceil \frac{1}{\epsilon''} (\frac{1}{\ell} \ln \frac{1}{\epsilon''}) \rceil$. All these choices become clear in the analysis. We additionally assume that $|S^*| > L$, otherwise we can simply return S_{pre} as our final solution.

Now, upon guessing (i.e. enumerating) S_{pre}, we construct a cost-truncated residual instance of $\mathcal{J}$ with respect to S_{pre} called $\mathcal{J}'$ per Definition 12, in which N' will not contain any element whose cost is higher than elements in S_{pre}.

Stage 2: Constructing a fractional solution $\mathbf{x}$ using the continuous relaxation of $\mathcal{J}'$. To construct our fractional solution $\mathbf{x}$, we use the continuous relaxation of $\mathcal{J}'$, as defined in Section 4.1.2. Let OPT' denote the cost of an optimal solution to $\mathcal{J}'$. Using Theorem 13 we obtain a vector $\mathbf{x}$ s.t. $\sum_j c_j x_j \leq \mathsf{OPT}'$ and $\forall i \in [r]$ $F'_i(\mathbf{x}) \geq (1 - 1/e - \epsilon')b'_i$.

Algorithm 1 Pseudocode for Bi-criteria Approximation for MSC.

Input : An instance of MSC problem denoted by $\mathcal{J} = (N, \{f_i, b_i\}_{i=1}^r, c)$, ϵ
Output : Solution $S \subseteq N$ that satisfies Theorem 1

$S_{pre} \leftarrow$ guess the L highest cost elements from an optimal solution to $\mathcal{J}$
Eliminate all elements with higher costs than those of S_{pre}, and create a *cost-truncated* residual instance of $\mathcal{J}$ with respect to S_{pre} called $\mathcal{J}' := (N', \{f_i', b_i'\}_{i=1}^r, c)$. // Stage 1
Using the algorithm from Theorem 13, obtain vector $\mathbf{x} \in \{0,1\}^{N'}$ as an approximate solution of MSC-Relax for instance J'. // Stage 2
Using Lemma 10, round $\mathbf{x}$ to obtain a set $R \subseteq N'$. // Stage 3
for $i \in [r]$ **do**
  $T_i \leftarrow \emptyset$ // Stage 4
  if $f_i'(R) < (1 - 1/e - 2\epsilon)b_i'$ **then**
    $T_i \leftarrow$ Elements using Greedy Algorithm by [17] s.t. $f_i'(T_i) \geq (1 - 1/e)b_i'$.
Let $T := \bigcup_{i \in [r]} T_i$
return $S_{pre} \cup R \cup T$

Stage 3: Rounding x to construct set R. Now we use our Rounding Lemma (Lemma 10) to round $\mathbf{x}$ to an integral solution R. The precise usage of the Rounding Lemma for this algorithm is as follows: Given N', c, $\{f_i'\}_{i=1}^r$ and their corresponding multi-linear extensions $\{F_i'\}_{i=1}^r$, our fractional point $\mathbf{x} \in [0,1]^{N'}$ satisfies $\sum_j c_j x_j \leq \mathsf{OPT}'$ and for all $i \in [r]$, $F_i'(\mathbf{x}) \geq b_i''$ (where $b_i'' := (1 - 1/e - \epsilon')b_i'$). Rounding Lemma thus outputs an $R \subseteq N'$ s.t. (i) $\mathbb{E}[c(R)] \leq \mathsf{OPT}' + r\lceil \frac{2\ln(r/\epsilon'')\ln(1/\epsilon'')}{(\epsilon'')^2} \rceil c_{\max}$ (where $c_{\max}$ is the cost of the most expensive element in N') and (ii) $\forall i \in [r]$ $f_i'(R) \geq (1 - \epsilon'')b_i''$ with probability at least $1 - \epsilon''/r$. Note that this is at least $(1 - 1/e - \epsilon)b_i'$ per our choice of ϵ' and ϵ''.

Stage 4: Greedily fixing unsatisfied constraints. For each constraint f_i' that has yet to be sufficiently satisfied by R, i.e., $f_i'(R) < (1 - 1/e - 2\epsilon)b_i'$, use the procedure outlined in Section 4.1.3 to reformulate the problem of finding a corresponding T_i to "fix" (i.e. satisfy) it as a submodular maximization problem subject to a single knapsack constraint. The budget for the knapsack constraint will be OPT'. Using the greedy procedure from Lemma 14 we obtain, for each unsatisfied f_i', a set T_i s.t. $c(T_i) \leq \mathsf{OPT}'$ and $f_i'(T_i) \geq (1 - 1/e)b_i'$.

Letting $T := \bigcup_{i \in [r]} T_i$, our algorithm returns the elements in $S_{pre} \cup R \cup T$. In the following subsection, we show how this output will satisfy the coverage and cost bounds in Theorem 1

Analysis overview. We defer the formal analysis to the full version and give a brief overview here. The analysis is not difficult modulo the powerful technical tools that we used along the way. Let OPT be the optimum value of $\mathcal{J}$; the residual instance after picking S_{pre} has a feasible solution of cost $\mathsf{OPT}' = \mathsf{OPT} - c(S_{pre})$. Stage 2 solves a continuous relaxation for the problem which outputs a fractional solution $\mathbf{x}$ that satisfies the residual constraints, f_i', to a factor of $(1 - 1/e - \epsilon)b_i'$ with $c(\mathbf{x}) \leq \mathsf{OPT}'$. Then we apply our Rounding Lemma to round $\mathbf{x}$ to a solution that satisfies each constraint with probability at least $1 - \epsilon/r$ while incurring an additive cost that we can bound as $\epsilon c(S_{pre})$. This is because we guessed sufficiently many high-cost elements for S_{pre}. Since the probability of a constraint remaining unsatisfied after rounding is at most ϵ/r, the expected cost of fixing it is at most $\frac{\epsilon}{r}\mathsf{OPT}'$ and hence the total expected cost of fixing the constraints is $\epsilon \cdot \mathsf{OPT}'$. Thus the total expected

cost is $c(S_{pre}) + \epsilon c(S_{pre}) + \mathsf{OPT}' + \epsilon\mathsf{OPT}' \leq (1+\epsilon)\mathsf{OPT}$. Each constraint i is satisfied to $f_i(S_{pre}) + (1 - 1/e - \epsilon)b'_i \geq (1 - 1/e - \epsilon)b_i$. The running time is dominated by the time to guess the elements in S_{pre}, which is polynomial since with r fixed, L is a constant.

5 Covering Coverage Functions

In this section, we prove Theorem 15 (stated below). Recall that the CCF problem (Definition 6) is a useful general problem that can model problems such as ColorVC, ColorSC, various facility location and clustering problems, to name a few. As such, Theorem 15 subsumes Theorem 3, which we discussed in the introduction.

We begin by discussing some technical components that are needed for our algorithm, before proceeding with its description. We conclude the section with a brief overview of the algorithm analysis. Again, complete details of the analysis and proof of Theorem 15 can be found in the full version.

For the remainder of this section, we use $\mathcal{I} = (\mathcal{U}, \mathcal{S}, A, b)$ to denote an instance of the CCF problem with set system $(\mathcal{U}, \mathcal{S})$, constraint matrix A, and requirements b_i for $i \in [r]$, where r is assumed to be fixed.

▶ **Theorem 15.** *Given an instance of the* CCF *problem such that the underlying set cover problem has a β approximation via the natural* LP*, and for a deletion closed set system, for any $\epsilon > 0$, there exists a randomized polynomial time (in $|\mathcal{U}|$, $|\mathcal{S}|$) algorithm that returns a collection $\mathcal{A} \subseteq \mathcal{S}$ such that (i) $\mathcal{A}$ is a feasible solution that satisfies all the covering constraints and (ii) $\mathbb{E}[c(\mathcal{A})] \leq \left(\frac{e}{e-1}\right)(1+\beta)(1+\epsilon)\mathsf{OPT}$.*

5.1 Algorithm Preliminaries

5.1.1 Operations on constraints

At times, we will restrict our instance to a subset of elements of the universe. We define this as follows.

▶ **Definition 16** (Restricting the Universe)**.** *Given an instance $\mathcal{I} = (\mathcal{U}, \mathcal{S}, A, b)$ of the* CCF *problem, and a subset of elements, $\mathcal{U}_{sub} \subseteq \mathcal{U}$, we define* ***restricting the Universe to elements of $\mathcal{U}_{sub}$*** *as follows: For each set $S \in \mathcal{S}$, $S \leftarrow S \cap \mathcal{U}_{sub}$ and update the Universe $\mathcal{U} \leftarrow \mathcal{U}_{sub}$. For each $j \in [n]$ such that $j \notin \mathcal{U}_{sub}$, delete j^{th} column of matrix A.*

Since we assume that the underlying set system of our initial CCF instance is deletion closed, the restricted instance is a valid CCF instance. Next, our algorithm chooses sets to be a part of the solution in multiple stages. Accordingly, we update the instance to reflect this selection. This is defined as follows.

▶ **Definition 17** (Residual Instance-CCF)**.** *Given an initial instance $\mathcal{I} = (\mathcal{U}, \mathcal{S}, A, b)$ and a collection $\mathcal{F} \subseteq \mathcal{S}$ of sets we define* ***a residual instance of $\mathcal{I}$ with respect to $\mathcal{F}$*** *as follows: For all $j \in \cup_{S \in \mathcal{F}} S$ and $i \in [r]$, set $A_{ij} = 0$. For all $i \in [r]$, set $b_i \leftarrow b_i - f_i(\mathcal{F})$. In addition, for all $i \in [r]$, let $f_{i|\mathcal{F}}(\cdot)$ denote the submodular function that corresponds to the i^{th} constraint.*

5.1.2 Greedy algorithm to fix CCF constraints

Recall that our framework requires us to have the ability to satisfy a single constraint. For the CCF problem, we use the following from [8]. Consider a universe of size n with a collection of m subsets of the universe. Each element $j \in [n]$ has a weight w_j and each set $i \in [m]$ has a cost c_i. Suppose we want to pick a sub-collection of sets with maximum weighted coverage such that a budget constraint is satisfied. We can write the following LP for this problem.

$$\begin{aligned} \max & \textstyle\sum_{j\in[n]} w_j z_j \\ \text{s.t.} & \textstyle\sum_{i:j\in S_i} x_i \geq z_j \\ & \textstyle\sum_{i\in[m]} c_i x_i \leq B \\ & z_j \leq 1 \qquad \text{for all } j \in [n] \\ & 0 \leq x_i \leq 1 \qquad \text{for all } i \in [m] \end{aligned} \tag{LP-MBC}$$

This is called the Max-Budgeted-Cover problem. One can run the standard greedy algorithm for this problem and [8] shows the following guarantee.

▶ **Lemma 18** (Chekuri et al. [8]). *Let Z be the optimum value of (LP-MBC) for a given instance of Max-Budgeted-Cover with budget B. Suppose Greedy Algorithm is run till the total cost of sets is equal to or exceeds B for the first time. Then the weight of elements covered by greedy is at least $(1 - \frac{1}{e})Z$.*

▶ Remark 19. The proof of the above lemma yields a stronger result: the Greedy algorithm achieves the same guarantee even when restricted to the support of the fractional solution. This will be useful in implicit settings.

5.2 Algorithm for CCF

Stage 1: Guess the high cost elements. Following the framework, we guess $(\frac{r}{\ell} \ln \frac{1}{\epsilon} + r)$ high cost elements from some fixed optimal solution (Line 1). The number ℓ is chosen to be $\frac{\epsilon^2}{2 \ln r/\epsilon}$. After this, we construct a residual instance to reflect this selection and work with the residual instance for remaining stages.

Stage 2: Construct Fractional Solution $\mathbf{x}$ using the LP Relaxation. Recall that the objective in CCF is to solve the integer program IP-CCF. To obtain a fractional solution, in Line 3 we solve the linear relaxation of this program LP-CCF for the residual instance obtained after Stage 1 and obtain a solution $(\mathbf{x}^*, \mathbf{z}^*)$.

Stage 2′: Obtain a slack in covering. We divide the elements into two categories: The *heavy* elements (H_e) are those for which $z_j^* \geq (1 - \frac{1}{e})(1 - \epsilon)$, while the *shallow* elements are the remaining non-heavy elements.

We will first cover the heavy elements completely (in Line 4). This is done by restricting the universe to H_e, using $(\mathbf{x}^*, \mathbf{z}^*)$ restricted to H_e as a solution for canonical LP of the canonical set cover problem and then using the β-approximation promised in the problem to obtain an integral covering. Here we use the fact that the instance is deletion closed. We then scale the shallow elements by $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right)$ to obtain a corresponding slack in covering.

Stage 3: Round fractional solution. We take the scaled fractional covering of shallow elements and round it using a call to Lemma 10 (Line 5).

Algorithm 2 Pseudocode for Covering Coverage Functions.

Input : An instance of CCF problem denoted by $\mathcal{I} = (\mathcal{U}, \mathcal{S}, A, b)$
Output : A subset, $\mathcal{A} \subseteq \mathcal{S}$ of the sets satisfying the claims in Theorem 15

1 With $\ell = \frac{\epsilon^2}{2 \ln r/\epsilon}$ construct $\mathcal{S}_{pre}$ by guessing the $r \cdot \lceil \frac{1}{\ell} \ln \frac{1}{\epsilon} \rceil + r$ highest cost sets from a fixed optimal solution of $\mathcal{I}$. `// Stage 1`
2 Eliminate all sets with cost higher than any set in $\mathcal{S}_{pre}$ and then create $\mathcal{I}'$ as residual instance of $\mathcal{I}$ with respect to $\mathcal{S}_{pre}$
3 Compute $(\mathbf{x}^*, \mathbf{z}^*)$ as the optimal solution to linear program relaxation for the instance $\mathcal{I}'$, given in LP-CCF `// Stage 2`
4 Define $H_e := \{j \mid z^*_j \geq (1 - \frac{1}{e}) \cdot (1 - \epsilon)\}$. Restrict the $\mathcal{U}$ to H_e, use $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right)(\mathbf{x}^*, \mathbf{z}^*)$ as solution for the canonical set covering LP and use the corresponding algorithm to cover all elements in H_e. Call the sets selected here $\mathcal{S}_{he}$ `// Stage 2'`
5 Restrict $\mathcal{U}$ to $\mathcal{U} \setminus H_e$ and use $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right) \cdot (\mathbf{x}^*, \mathbf{z}^*)$ as solution to fairness constrained LP-CCF for the instance $\mathcal{I}'$. Round this using Lemma 10 and call the sets chosen here $\mathcal{S}_{sh}$ `// Stage 3`
6 For $i \in [r]$ if f_i is not satisfied, create $\mathcal{G}_i$ by choosing greedily (with respect to marginal value) sets from $\mathcal{S}$ till it is satisfied. `// Stage 4`
7 **return** $\mathcal{S}_{pre} \cup \mathcal{S}_{he} \cup \mathcal{S}_{sh} \cup \left(\bigcup_{i \in [r]} \mathcal{G}_i\right)$

Stage 4: Greedy fixing of the solution. For any constraint $i \in [r]$ that is not satisfied, we will fix it via a greedy algorithm. Following the greedy fix for Max-Budgeted-Cover in 5.1.2, this works as follows: From the collection of sets not picked, pick the sets greedily in order of decreasing bang-per-buck (ratio of marginal value of the set and its cost).

Analysis overview. We defer the formal analysis to the full version and give a brief overview here. First, we define, $\mathsf{OPT}' = \mathsf{OPT} - c(\mathcal{S}_{pre})$, where OPT and OPT' are the optimal values for $\mathcal{I}$ and $\mathcal{I}'$ (the residual instance of $\mathcal{I}$ with respect to $\mathcal{S}_{pre}$), and $\mathcal{S}_{pre}$ is the set of elements guessed in Stage 1. In Stage 2 we solve an LP relaxation to obtain a fractional solution $\mathbf{x}$ that satisfies all the constraints exactly with $c(\mathbf{x}) \leq \mathsf{OPT}'$.

Following our framework, we want to use our key lemma to round $\mathbf{x}$. However, the lemma is based on the multilinear relaxation F'_i of the residual function f'_i while $\mathbf{x}$ satisfies the constraint based on the LP. Hence, we must address the issue of the correlation gap [5, 19], since working with $F'_i(\mathbf{x})$ loses a factor of $(1 - 1/e)$ in covering the constraint and our goal is to cover it exactly. To address this, we scale up the LP solution by a factor of $\frac{e}{e-1}$. Note that in doing so, there will be some points that are already covered beyond $(1 - 1/e)$ and their coverage should not exceed 1. Following [13, 8], we first separate out the points that are covered already to more than $(1 - 1/e)$. Scaling covers them in the LP to 1, and we can cover all those points via the LP-based Set Cover algorithm promised to us by the deletion-closed set system. The cost of covering them is $\beta \frac{e}{e-1} \mathsf{OPT}'$. For points that are covered less than $(1 - 1/e)$ the scaling increases their coverage by a factor of $\frac{e}{e-1}$, allowing us to work with the multilinear relaxation and avoid the $(1 - 1/e)$ correlation gap. This allows us to use the Rounding Lemma on these points to fully cover them. A minor technicality to note is that the Rounding Lemma loses a $(1 - \epsilon)$-factor in the coverage so we actually must scale up the LP by $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right)$. Thus, the expected cost of the random rounding is $(1 + O(\epsilon))\frac{e}{e-1}\mathsf{OPT}'$ plus the additive cost which we bound as an ϵ-fraction of $c(L)$ as in MSC analysis.

To fix the unsatisfied constraints, and achieve exact coverage, we do the analysis with respect to the scaled up LP solution and take advantage of Lemma 18. The expected cost of fixing can be bound by $\epsilon(1+O(\epsilon))\frac{e}{e-1}\mathsf{OPT}'$ since each constraint is satisfied with probability ϵ/r in the random rounding. Thus the total cost is

$$c(\mathcal{S}_{pre}) + \epsilon c(\mathcal{S}_{pre}) + (1+O(\epsilon))\beta\frac{e}{e-1}\mathsf{OPT}' + (1+O(\epsilon))\frac{e}{e-1}\mathsf{OPT}' + \epsilon(1+O(\epsilon))\frac{e}{e-1}\mathsf{OPT}'$$
$$\leq (1+O(\epsilon))\frac{e}{e-1}(\beta+1)\mathsf{OPT}.$$

Each constraint i is satisfied fully as per the discussion. The running time of our algorithm is dominated by guessing $\mathcal{S}_{pre}$. With r fixed, the size of $\mathcal{S}_{pre}$ is a constant and the running time is polynomial.

6 Applications to Facility Location with Multiple Outliers

In this section, we give an overview for applying the CCF framework to the problem of Facility Location with Multiple Outliers (FLMO). For complete details and an application to facility location to minimize sum of radii, we refer the reader to full version of the paper. FLMO is a generalization of the classical facility location problem, in which clients belong to color classes, and the objective is to cover a required number of clients from each class. Specifically, we are given a set of facilities F, a set of clients C, where each client belongs to at least one of r color classes C_k (i.e. $C = \cup_{k\in[r]} C_k$), and a metric space $(F \cup C, d)$. Each facility $i \in F$ has an associated non-negative opening cost f_i, and each client $j \in C$ can be served by assigning it to an open facility i, incurring connection cost $d(i,j)$. For each color class C_k, we are given a required demand b_k (where $1 \leq b_k \leq |C_k|$). The goal is to open facilities and assign clients to facilities such that at least b_k clients from C_k are serviced by a facility, and where the total facility and connection costs of opened facilities and their assigned clients is minimized.

FLMO can be naturally expressed as an instance of the CCF problem over an exponentially large set system, where the universe of elements is C and where each set corresponds to a *star* (i, S) defined by a facility $i \in F$ and a subset of clients $S \subseteq C$ assigned to it. The goal is to select a collection of such stars to cover at least b_k clients from each color class C_k, while minimizing the total cost, where the cost of a star is given by $c(i,S) := f_i + \sum_{j\in S} d(i,j)$. We can capture this problem via the following exponentially sized CCF-esque IP.

$$\begin{aligned}
\min \quad & \sum_{i\in F}\sum_{S\subseteq C} c(i,S)\cdot x(i,S) & & \\
\text{s.t.} \quad & \sum_{i\in F}\sum_{\substack{S\subseteq C\\ j\in S}} x(i,S) \geq z_j & \forall j \in C & \qquad \text{(IP-FLMO)}\\
& \sum_{j\in C_k} z_j \geq b_k & \forall k \in [r] & \\
& x(i,S),\ z_j \in \{0,1\} & \forall i \in F, S \subseteq C, j \in C &
\end{aligned}$$

Since there are exponentially many stars to be considered, we apply a refined version of the CCF framework that avoids explicitly enumerating all possible stars. The key change is that instead of guessing the high cost stars, we guess certain high-cost components of the optimal solutions.

We now describe each stage of our refined CCF-based algorithm for FLMO, beginning with a pre-processing step (Stage 0) that enables efficient guessing and separation oracle.

Stage 0: Scaling and pruning facility and connection costs. As is standard, we assume that the value of the optimal solution OPT is guessed up to a $(1 + o(1))$ factor (we overload OPT to denote both the true and guessed optimal cost). This introduces only a polynomial overhead in runtime and at most a $(1 + o(1))$ loss in the approximation ratio. In addition to guessing OPT, we define a polynomial scaling factor $B := n^3$.

Using the guessed OPT and B, we can now eliminate any unnecessary or negligible facility and connection costs. First, we disallow (i.e. remove from the instance) any facilities i for which $f_i > \mathsf{OPT}$ since we know they will never be selected in the optimal solution. We also disallow any connections between facilities i and clients j for which $d(i, j) > \mathsf{OPT}$. Next, for any facility i with $f_i \leq \mathsf{OPT}/B$, we define its scaled cost $\bar{f}_i := 0$; for any client-facility pair (i, j) where $d(i, j) \leq \mathsf{OPT}/B$, we define the scaled connection cost $\bar{d}(i, j) := 0$. For the remaining facility and connection costs, we scale them as follows: $\bar{f}_i := \lceil B/\mathsf{OPT} \cdot f_i \rceil$, $\bar{d}(i, j) := \lceil B/\mathsf{OPT} \cdot d(i, j) \rceil$. This will guarantee that all scaled facility costs will be integer-values in $[0, B]$. Thus, the scaled cost of each star $\bar{c}(i, S) := \bar{f}_i + \sum_{j \in S} \bar{d}(i, j)$ will be some polynomially bounded integer. The discretized distances $\bar{d}(i, j)$ may no longer form a metric (since they may violate triangle inequality), however, this is fine since our CCF framework did not require set costs to satisfy such properties. The metric property for distances will only be required in Stage 2', when we must solve the canonical facility location problem to cover heavy clients. For that stage, we will show that we can viably revert back to using the original metric d.

This pre-processing step will incur at most an additive $O(\mathsf{OPT}/B)$ term per cost term, and thus will result in a $1 + o(1)$ multiplicative increase in the total cost of the solution, but this can be absorbed into our final approximation factor. In the rest of the section we assume that we have guessed OPT correctly to within a $(1 + o(1))$ factor.

Stage 1: Guessing high-cost components. This stage consits of two parts: (1) Guess some high cost components (2) Create a residual instance accounting for the guess.

Guess high-cost components. In the CCF framework, we needed to guess the $L := \left(\frac{r}{\ell} \ln 1/\epsilon + r\right)$ most expensive stars from the optimal solution. This is because our rounding procedure in Lemma 10 and the greedy fix step later choose these many extra sets to satisfy the constraints and we account for these sets via the initial guesses. For the FLMO setting explicitly guessing a a star (i, S) would require exponential time. To circumvent this we instead guess partial information about many stars. However, we must now guess more information due to this partial knowledge.

Specifically, we guess tuples of the form (i_h, S_h, g_h) for $h \in [T]$ with $T = \lceil L/\epsilon \rceil$, where i_h is a facility from one of the T highest cost stars, $S_h \subseteq C$ is a set of the L farthest clients assigned to i_h in the optimal solution, and g_h is the total cost of the full optimal star at facility i_h, i.e., $\bar{c}(i_h, S_h^*)$, where S_h^* denotes the full client set served by i_h. We note that if there are fewer than T stars in the optimum solution, the problem becomes simpler; in that case we would have guessed all the optimum's facilities. This will be clear after a description of the remaining analysis and we give a remark at the end of Stage 3 as to how to deal with that case. Let OPT_{guess} be the cost of the guessed portion of the solution.

Create residual instance. Let $S_{pre} := \{(i_h, S_h, g_h)\}_{h \in [L]}$ denote the collection of guessed partial stars. We define $F_{pre} := \{i_h \mid (i_h, S_h, g_h) \in S_{pre}\}$ as the set of guessed high-cost facilities, and $C_{pre} := \bigcup_{h \in [L]} S_h$ as the set of clients guessed to be served by those stars. Let $G := \min_{h \in [L]} g_h$ be the smallest guessed star cost across all tuples.

Now, to describe the residual instance with respect to these guessed components, first update the color demands b_k to $b'_k := b_k - |C_k \cap C_{pre}|$. Next, restrict attention to clients in $C \setminus C_{pre}$ and define for each facility $i \in F$ a restricted family of allowable stars, denoted by $\mathcal{S}_i$, and updated star costs, denoted by $\bar{c}'$, as follows:

- For $i \in F_{pre}$, we allow only singleton stars $(i, \{j\})$ with $j \in C \setminus C_{pre}$ and $\bar{d}(i,j) \leq \min_{j' \in S_h} \bar{d}(i,j')$ (for the corresponding tuple $(i, S_h, g_h) \in S_{pre}$). For these stars, we define the updated cost as $\bar{c}'(i,\{j\}) := \bar{d}(i,j)$, since after guessing S_{pre}, we account $\bar{f}_i$ to open facility i and the connection costs for the clients in S_h.
- For $i \notin F_{pre}$, we allow any star (i,S) with $S \subseteq C \setminus C_{pre}$ and total cost at most G. For these stars, the cost remains as $\bar{c}'(i,S) := \bar{f}_i + \sum_{j \in S} \bar{d}(i,j)$.

Restricted Primal (LP-FLMO)

$$\min \sum_{i \in F} \sum_{(i,S) \in \mathcal{S}_i} \bar{c}'(i,S) \cdot x(i,S)$$

s.t.

$$\sum_{i \in F} \sum_{\substack{(i,S) \in \mathcal{S}_i: \\ j \in S}} x(i,S) \geq z_j \quad \forall j \in C \setminus C_{pre}$$

$$\sum_{j \in C_k \setminus C_{pre}} z_j \geq b'_k \quad \forall k \in [r]$$

$$z_j \leq 1 \quad \forall j \in C \setminus C_{pre}$$

$$0 \leq x(i,S) \leq 1 \quad \forall i \in F,\ (i,S) \in \mathcal{S}_i$$

$$0 \leq z_j \quad \forall j \in C \setminus C_{pre}$$

Dual

$$\max \sum_{k=1}^{r} b'_k \cdot \beta_k - \sum_{j \in C \setminus C_{pre}} \gamma_j$$

s.t.

$$\sum_{j \in S} \alpha_j \leq \bar{c}'(i,S) \quad \forall i \in F,\ (i,S) \in \mathcal{S}_i$$

$$\alpha_j - \gamma_j \leq \beta_k \quad \forall j \in C_k \setminus C_{pre},\ \forall k \in [r]$$

$$\alpha_j, \beta_k, \gamma_j \geq 0 \quad \forall j \in C \setminus C_{pre},\ k \in [r]$$

(LP-FLMO Primal & Dual)

Stage 2: Constructing a fractional solution via the dual. We now solve an LP relaxation for the residual instance defined in Stage 1. This may still contain exponentially many variables. To solve this LP efficiently, we apply the ellipsoid method to its dual (see LP-FLMO Primal & Dual), which has polynomially many variables but exponentially many constraints. This requires a polynomial-time separation oracle which, given a candidate dual solution (α, β, γ), either certifies feasibility or returns a violated constraint. That is, it identifies residual star $(i,S) \in \mathcal{S}_i$ such that $\sum_{j \in S} \alpha_j > \bar{c}'(i,S)$. Again, since facility costs $\bar{f}_i$ and distances $\bar{d}(i,j)$ are integral and polynomially bounded (from pre-processing in Stage 0), we can design such an oracle. We formalize this as the following lemma.

▶ **Lemma 20.** *Assuming that all distances $\bar{d}(i,j)$ and facility costs f_i are integral and polynomially bounded, there exists a polynomial-time separation oracle for the dual LP given in* LP-FLMO *Primal & Dual.*

Note that the LP may not be feasible if our guess was incorrect. In this case we can discard the guess. For a correct guess, we denote the cost of this LP solution by $\mathsf{OPT}_{\mathsf{LP}}$. Note that $\mathsf{OPT}_{\mathsf{LP}} \leq \mathsf{OPT} - \mathsf{OPT}_{guess}$.

Stage 2': Handling Heavy vs. Shallow Clients. Given the fractional solution $(\mathbf{x}^*, \mathbf{z}^*)$ to the residual LP, we now have a polynomial-sized support of stars. We begin by partitioning the clients into *heavy* clients C_{he} and *shallow* clients C_{sh} based on their fractional coverage: let $C_{he} := \{j \in C \setminus C_{pre} \mid z_j > \tau\}$ and $C_{sh} := \{j \in C \setminus C_{pre} \mid z_j \leq \tau\}$, where we set $\tau := (1 - 1/e)(1 - \epsilon)$ as in the CCF framework. To cover the heavy clients, we prove the following lemma.

▶ **Lemma 21.** *Given* $(\mathbf{x}^*, \mathbf{z}^*)$*, a feasible solution to the residual LP with cost* $\mathsf{OPT}_{\mathsf{LP}}$*, there is an efficient algorithm to cover the heavy clients* C_{he} *with cost at most* $\beta_{FL} \cdot \left(\frac{e}{e-1}\right) \cdot \left(\frac{1}{1-\epsilon}\right) \cdot \mathsf{OPT}_{\mathsf{LP}}$*, where* β_{FL} *is the approximation factor of the underlying LP-based approximation algorithm for UCFL.*

Now, to proceed with Stages 3 and 4 for the shallow clients, we perform the following steps.

- **Restrict the instance to shallow clients.** In our fractional solution for the primal in LP-FLMO Primal & Dual, we update the stars to only have shallow clients and remove all the heavy clients. The variables $x(i, S)$ remain unchanged. The z variables are restricted to only shallow clients. We also update the covering requirements to reflect that heavy clients have already been selected: $b'_k := b'_k - |C_k \cap C_{he}|$ for each color class $k \in [r]$.
- **Scale the solution**. We scale the solution by $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right)$. Note that the new scaled solution is a feasible solution for the primal in LP-FLMO Primal & Dual with a cost $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right) \cdot \mathsf{OPT}_{\mathsf{LP}}$ and each color's residual requirement is over-covered by a factor of $\frac{e}{(e-1)(1-\epsilon)}$.

Stage 3: Rounding the fractional solution. We work with the residual instance restricted to shallow clients and scaled as mentioned in the previous stage. We now want to apply Lemma 10 to round this fractional solution. The algorithm in the lemma potentially picks L stars to ensure that randomized rounding satisfies the constraints with high probability. Algorithmically, we still perform the same step. For each color class $k \in [r]$, we select (up to) the top $\frac{1}{\ell} \ln(1/\epsilon)$ stars *from the support* of the fractional solution $\mathbf{x}^*$. The proof of Lemma 10 easily extends to only picking from the support. Finally, we randomly round the remaining stars with probabilities given by $\mathbf{x}^*$. Suppose the set of stars chosen in this stage is $\mathcal{S}(C_{sh})$. We can prove the following key lemma.

▶ **Lemma 22.** *After selecting the stars* $\mathcal{S}(C_{sh})$*, each constraint is satisfied with probability at least* $1 - \epsilon/r$ *and the expected cost of* $\mathcal{S}(C_{sh})$ *is bounded by* $\left(\frac{e}{e-1}\right)\left(\frac{1}{1-\epsilon}\right)\mathsf{OPT}_{\mathsf{LP}} + \epsilon OPT + \mathsf{OPT}_{guess}$.

Stage 4: Greedily fixing remaining unsatisfied constraints. If any color class constraint remains unsatisfied after Stage 3, we fix it by greedily selecting stars until the constraint becomes satisfied. As in the original CCF framework, we can employ the greedy fix from the Max-Budgeted-Cover heuristic. The LP-MBC for this problem is to select a subset of stars that cover the maximum number of clients at a cost bounded by $\frac{e}{(e-1)(1-\epsilon)}\mathsf{OPT}_{\mathsf{LP}}$. Further, as per the remark after Lemma 18, this Greedy can be executed by only looking at the support of the LP solution. Therefore the step can be performed in polynomial time. Similar to the analysis in CCF, this greedy fix is applied to the r^{th} covering class with a probability bounded by ϵ/r. Further we can show that the total expected cost to fix all constraints is bounded by $\epsilon \cdot \mathsf{OPT} + rc_{\max}$ where c_{max} is the cost of highest cost star in the support. Each star in the support has cost at most $\epsilon\mathsf{OPT}/r$ due to the guessing in Stage 1 (as discussed in proof of Lemma 22). Hence the expected fixing cost is $(1 + O(\epsilon))\mathsf{OPT}$.

Putting together, we obtain the following result. The analysis is similar to that of CCF. The running time is polynomial in $n^{O(r/\epsilon^2)}$.

▶ **Theorem 4.** *Let* β *be the approximation ratio for* UCFL *via the natural LP relaxation. Then there is a randomized polynomial-time algorithm that given an instance of* FLMO *with fixed* r *and* $\epsilon > 0$ *yields a* $\frac{e}{e-1}(\beta + 1 + \epsilon)$*-approximate solution.*

References

1 Sayan Bandyapadhyay, Aritra Banik, and Sujoy Bhore. 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 47*, pages 39–51. Springer, 2021. doi:10.1007/978-3-030-86838-3_4.

2 Suman K Bera, Shalmoli Gupta, Amit Kumar, and Sambuddha Roy. Approximation algorithms for the partition vertex cover problem. *Theoretical Computer Science*, 555:2–8, 2014. doi:10.1016/J.TCS.2014.04.006.

3 Marcin Bienkowski, Jaroslowaw Byrka, Marek Chrobak, Neil Dobbs, Tomasz Nowicki, Maxim Sviridenko, Grzegorz Świrszcz, and Neal E Young. Approximation algorithms for the joint replenishment problem with deadlines. *Journal of Scheduling*, 18(6):545–560, 2015. doi:10.1007/S10951-014-0392-Y.

4 Nader H Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *STACS 98: 15th Annual Symposium on Theoretical Aspects of Computer Science Paris, France, February 25–27, 1998 Proceedings 15*, pages 298–308. Springer, 1998. doi:10.1007/BFB0028569.

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

6 Timothy M Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1576–1585. SIAM, 2012. doi:10.1137/1.9781611973099.125.

7 Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, volume 1, pages 642–651. Citeseer, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365555.

8 Chandra Chekuri, Tanmay Inamdar, Kent Quanrud, Kasturi Varadarajan, and Zhao Zhang. Algorithms for covering multiple submodular constraints and applications. *Journal of Combinatorial Optimization*, 44(2):979–1010, 2022. doi:10.1007/S10878-022-00874-X.

9 Chandra Chekuri, T.S. Jayram, and Jan Vondrak. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 201–210, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2688073.2688086.

10 Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 575–584, 2010. doi:10.1109/FOCS.2010.60.

11 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

12 Sariel Har-Peled and Mitchell Jones. Few cuts meet many point sets. *Algorithmica*, 85(4):965–975, 2023. doi:10.1007/S00453-022-01059-Y.

13 Tanmay Inamdar and Kasturi Varadarajan. On Partial Covering For Geometric Set Systems. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.47.

14 David S Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49, 1973. doi:10.1145/800125.804034.

15 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. doi:10.1016/J.IC.2012.01.007.

16 Varun Suriyanarayana, Varun Sivashankar, Siddharth Gollapudi, and David B Shmoys. Improved approximation algorithms for the joint replenishment problem with outliers, and with fairness constraints. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2793–2828. SIAM, 2024. doi:10.1137/1.9781611977912.99.

17 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004. doi:10.1016/S0167-6377(03)00062-2.

18 Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 641–648, 2010. doi:10.1145/1806689.1806777.

19 Jan Vondrák. *Submodularity in combinatorial optimization*. PhD thesis, Charles University, 2007. Avaulable at https://theory.stanford.edu/~jvondrak/data/KAM_thesis.pdf.

20 Jan Vondrák. A note on concentration of submodular functions. *arXiv preprint arXiv:1005.2791*, 2010. arXiv:1005.2791.

21 Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982. doi:10.1007/BF02579435.