# New Results on a General Class of Minimum Norm Optimization Problems

**Kuowen Chen**
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

**Jian Li**
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

**Yuval Rabani**
Computer Science and Engineering, The Hebrew University of Jerusalem, Israel

**Yiran Zhang**
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

## Abstract

We study the general norm optimization for combinatorial problems, initiated by Chakrabarty and Swamy (STOC 2019). We propose a general formulation that captures a large class of combinatorial structures: we are given a set $\mathcal{U}$ of $n$ weighted elements and a family of *feasible* subsets $\mathcal{F}$. Each subset $S \in \mathcal{F}$ is called a feasible solution/set of the problem. We denote the *value vector* by $\boldsymbol{v} = \{\boldsymbol{v}_i\}_{i\in[n]}$, where $\boldsymbol{v}_i \geq 0$ is the value of element $i$. For any subset $S \subseteq \mathcal{U}$, we use $\boldsymbol{v}[S]$ to denote the $n$-dimensional vector $\{v_e \cdot \mathbf{1}[e \in S]\}_{e\in\mathcal{U}}$ (i.e., we zero out all entries that are not in $S$). Let $f : \mathbb{R}^n \to \mathbb{R}_+$ be a symmetric monotone norm function. Our goal is to minimize the norm objective $f(\boldsymbol{v}[S])$ over feasible subset $S \in \mathcal{F}$. The problem significantly generalizes the corresponding min-sum and min-max problems.

We present a general equivalent reduction of the norm minimization problem to a multi-criteria optimization problem with logarithmic budget constraints, up to a constant approximation factor. Leveraging this reduction, we obtain constant factor approximation algorithms for the norm minimization versions of several covering problems, such as interval cover, multi-dimensional knapsack cover, and logarithmic factor approximation for set cover. We also study the norm minimization versions for perfect matching, $s$-$t$ path and $s$-$t$ cut. We show the natural linear programming relaxations for these problems have a large integrality gap. To complement the negative result, we show that, for perfect matching, it is possible to obtain a bi-criteria result: for any constant $\epsilon, \delta > 0$, we can find in polynomial time a nearly perfect matching (i.e., a matching that matches at least $1 - \epsilon$ proportion of vertices) and its cost is at most $(8 + \delta)$ times of the optimum for perfect matching. Moreover, we establish the existence of a polynomial-time $O(\log\log n)$-approximation algorithm for the norm minimization variant of the $s$-$t$ path problem. Specifically, our algorithm achieves an $\alpha$-approximation with a time complexity of $n^{O(\log\log n/\alpha)}$, where $9 \leq \alpha \leq \log\log n$.

**2012 ACM Subject Classification** Mathematics of computing → Approximation algorithms; Mathematics of computing → Combinatorial optimization; Mathematics of computing → Linear programming

**Keywords and phrases** Approximation Algorithms, Minimum Norm Optimization, Linear Programming

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2025.50

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2504.13489` [17]

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).
Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis
Article No. 50; pp. 50:1–50:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

In many optimization problems, a feasible solution typically induces a multi-dimensional value vector (e.g., by the subset of elements of the solution), and the objective of the optimization problem is to minimize either the total sum (i.e., $\ell_1$ norm) or the maximum (i.e., $\ell_\infty$ norm) of the vector entry. For example, in the minimum perfect matching problem, the solution is a subset of edges and the induced value vector is the weight vector of the matching (i.e., each entry of the vector is the weight of edge if the edge is in the matching and 0 for a non-matching edge) and we would like to minimize the total sum. Many of such problems are fundamental in combinatorial optimization but require different algorithms for their min-sum and min-max variants (and other possible variants). Recently there have been a rise of interests in developing algorithms for more general objectives, such as $\ell_p$ norms [4, 24], top-$\ell$ norms [22], ordered norms [9, 12] and more general norms [13, 14, 31, 19, 1, 33], as interpolation or generalization of min-sum and min-max objectives. The algorithmic study of such generalizations helps unify, interpolate and generalize classic objectives and algorithmic techniques.

The study of approximation algorithm for general norm minimization problems is initiated by Chakrabarty and Swamy [13]. They studied two fundamental problems, load balancing and $k$-clustering, and provided constant factor approximation algorithm for these problems. For load balancing, the induced value vector is the vector of machine loads and for $k$-clustering the vector is the vector of service costs. Subsequently, the norm minimization has been studied for a variety of other combinatorial problem such as general machine scheduling problem [19], stochastic optimization problems [31], online algorithms [38], parameterized algorithms [1] etc. In this paper, we study the norm optimization problem for a general set of combinatorial problems. In our problem, a feasible set is a subset of elements and the multi-dimensional value vector is induced by the subset of elements of the solution. Our problem is defined formally as follows:

▶ **Definition 1** (The Norm Minimization Problem (MinNorm)). *We are given a set $\mathcal{U} = [n]$ of $n$ weighted elements and a family of* feasible *subsets $\mathcal{F}$. Each subset $S \in \mathcal{F}$ is called a feasible solution/set of the problem. We denote the* value vector *by $\boldsymbol{v} = \{\boldsymbol{v}_i\}_{i\in[n]}$, where $\boldsymbol{v}_i \geq 0$ is the value of element $i$. We say a subset $S \subseteq \mathcal{U}$* feasible *if $S \in \mathcal{F}$. For any subset $S \subseteq \mathcal{U}$, we use $\boldsymbol{v}[S]$ to denote the $n$-dimensional vector $\{v_e \cdot \mathbf{1}[e \in S]\}_{e\in\mathcal{U}}$ (i.e., we zero out all entries that are not in $S$), and we call $\boldsymbol{v}[S]$* the value vector induced by *$S$. Let $f : \mathbb{R}^n \to \mathbb{R}_+$ be a symmetric monotone norm function. Given the norm function $f(\cdot)$, our goal is to find a feasible solution in $\mathcal{F}$ such that the norm of the value vector induced by the solution is minimized, i.e., we aim to solve the following optimization problem*

$$\textsf{MinNorm:} \qquad \text{minimize} \quad f(\boldsymbol{v}[S]) \qquad \text{subject to} \qquad S \in \mathcal{F}.$$

Note that the case $f(\boldsymbol{v}[S]) = \sum_{e\in S} v_e$ is the most studied min-sum objective and we call the corresponding problem the *original optimization problem*. Other interesting norms include $\ell_p$ norms, Top-$\ell$ norms (the sum of top-$\ell$ entries), ordered norms (see its definition in Section 3). Note that our general framework covers the $k$-clustering studied in [13]: in the $k$-clustering problem, the universe $\mathcal{U}$ is the set of edges and each feasible solution in $\mathcal{F}$ is a subset of edges that corresponds to a $k$-clustering. The load balancing problem does not directly fit into our framework, since one needs to first aggregate the processing times to machine loads, then apply the norm.

Before stating our results, we briefly mention some results that are either known or very simple to derive.

1. (Matroid) Suppose the feasible set $\mathcal{F}$ is a matroid and a feasible solution is a basis of this matroid. In fact, the greedy solution (i.e., the optimal min-sum solution) is the optimal solution for any monotone symmetric norm. This is a folklore result and can be easily seen as follows: First, it is easy to establish the following observation, using the exchange property of matroid: We use $\textsc{Top}_\ell(S)$ to denote the sum of largest $\ell$ elements of $S$. For any $\ell \in \mathbb{Z}_{\geq 1}$ and any basis $S \in \mathcal{F}$, $\textsc{Top}_\ell(S_{\text{greedy}}) \leq \textsc{Top}_\ell(S)$ where $S_{\text{greedy}}$ is the basis obtained by the greedy algorithm. Then using the majorization lemma by Hardy, Littlewood and Pòlya (Lemma 3), we can conclude $S_{\text{greedy}}$ is optimal for any monotone symmetric norm.
2. (Vertex Cover) We first relax the problem to the following convex program:

   $$\text{min.} \quad f(v_1x_1, \ldots, v_nx_n) \quad \text{s.t.} \quad x_i + x_j \geq 1 \text{ for any } (i,j) \in E.$$

   The objective is convex since $f$ is norm (in particular the triangle inequality of norm). Then, we solve the convex program and round all $x_i \geq 1/2$ to 1 and others to 0. It is easy to see this gives a 2-approximation (using the property $f(\alpha x) = \alpha f(x)$ for $\alpha \geq 0$).
3. (Set Cover) The norm-minimization set cover problem is a special case of the generalized load balancing problem introduced in [19]. Here is the reduction: each element corresponds to a job and each subset to a machine; if element $i$ is in set $S_j$, the processing time $p_{ij} = 1$, otherwise $p_{ij} = \infty$; the inner norm of each machine is the max norm (i.e., $\ell_\infty$) and the outer norm is $f(\cdot)$. Hence, this implies an $O(\log n)$-approximation for norm-minimization set cover problem using the general result in [19]. The algorithm in [19] is based on a fairly involved configuration LP. In the full version [17], we also provide a much simpler randomized rounding algorithm that is also an $O(\log n)$-approximation. Note this is optimal up to a constant factor given the approximation hardness of set cover [23, 21].
4. ($\textsc{Top}_\ell$ and Ordered Norms) If the min-sum problem can be solved or approximated efficiently, one can also solve or approximate the corresponding $\textsc{Top}_\ell$ and ordered norm optimization problems. This mostly follows from known techniques in [9, 13, 22].

## Our Contributions

Our technical contribution can be summarized as follows:

1. (Theorem 5) We present a general reduction of the norm minimization problem to a multi-criteria optimization problem with logarithmic budget constraints, up to a constant approximation factor. This immediately implies an $O(\alpha \log n)$-approximation for the MinNorm problem if there is a poly-time $\alpha$-approximation for the corresponding weight minimization problem (See Theorem 6).
2. Leveraging the reduction in Theorem 5, we obtain constant factor approximation algorithms for the norm minimization versions of several covering problems, such as interval covering (Theorem 17), multi-dimensional knapsack cover (Theorem 9 and Theorem 10). These algorithms are based on rounding the natural linear programming relaxation of the multi-criteria optimization problem, possibly with a careful enumeration of partial solutions. For set cover, we obtain a simple randomized approximation algorithm with approximation factor $O(\log n)$ (See full version [17]), which is much simpler than the general algorithm in [19].
3. We also study the norm minimization versions for perfect matching, $s$-$t$ path and $s$-$t$ cut. We show the natural linear programming relaxations for these problems have a large integrality gap (Theorem 21 and a second theorem provided only in the full version [17]). This indicates that it may be difficult to achieve constant approximation factors for these problems.

4. To complement the above negative result, we show that, for perfect matching, it is possible to obtain a bi-criteria approximation: for any constant $\epsilon > 0$, we can find a nearly perfect matching that matches at least $1 - \epsilon$ proportion of vertices and the norm of this solution is at most $(8 + \delta)$ times of the optimum for perfect matching where $\delta$ is any positive real constant (Theorem 24).
5. We present an approximate dynamic programming approach that yields a $\alpha$-approximation $n^{O(\log\log n/\alpha)}$-time algorithm for the min-norm $s$-$t$ path problem for $9 \le \alpha \le \log\log n$ (Theorem 22), demonstrating an alternative technique for solving norm minimization problems beyond LP rounding.

## 2 Related Work

**Top-$\ell$ and Ordered Optimization.** As a special case of general norm optimization, ordered optimization for combinatorial optimization problems have received significant attention in the recent years. In fact, an ordered norm can be written as a conical combination of top-$\ell$ norms (see Claim 2). The ordered k-median problem was first studied by Byrka et al. [9] and Aouad and Segev [2]. Byrka et al. [9] obtained the first constant factor approximation algorithm (the factor is $38 + \epsilon$). Independently, Chakrabarty and Swamy [12] obtained an algorithm with approximation factor 18 for the top-$\ell$ norm), which can be combined with the enumeration procedure of Aouad and Segev [2] to get the same factor for the general ordered $k$-median. The current best known approximation is 5, by Chakrabarty and Swamy [13]. Deng and Zhang [20] studied ordered $k$-median with outliers and obtained a constant factor approximation algorithm. Maalouly and Wulf [22] studied the top-$\ell$ norm optimization for the matching problem and obtained an polynomial time exact algorithm (see full version [17]). Braverman et al. studied coreset construction for ordered clustering problems [7] which was motivated by applications in machine learning. Batra et al. [5] studied the ordered min-sum vertex cover problem and obtained the first poly-time approximation approximation with approximation factor $2 + \epsilon$.

**General Symmetric Norm Optimization.** Chakrabarty and Swamy [13] first studied general monotone symmetric norm objectives for clustering and unrelated machine load balancing and obtained constant factor approximation algorithms, substantially generalizing the results for $k$-Median and $k$-Center and makespan minimization for unrelated machine scheduling. In a subsequent paper [14], they obtained a simpler algorithm for load balancing that achieves an approximation factor of $2 + \epsilon$. Abbasi et al. [1] studied the parametrized algorithms for the general norm clustering problems and provided the first EPAS (efficient parameterized approximation scheme). Deng et al. [19] introduced the generalized load balancing problem, which further generalizes the problem studied by [14]. In the generalized load balancing problem, the load of a machine $i$ is a symmetric, monotone (inner) norm of the vector of processing times of jobs assigned to $i$. The generalized makespan is another (outer) norm aggregating the loads. The goal is to find an assignment of jobs to minimize the generalized makespan. They obtained a logarithmic factor approximation, which is optimal up to constant factor since the problem generalizes the set cover problem. For the special case where the inner norms are top-$k$ norms, Ayyadevara et al. [3] showed the natural configuration LP has a $\Omega(\log^{1/2} n)$ integrality gap.

**Submodular/Supermodular Optimization.** Optimizing submodular/supermodular function under various combinatorial constraints is another important class of optimization problems with general objectives and has been studied extensively in the literature. See e.g., [10, 35, 15, 8] and the survey [34]. However, note that results for submodular functions does not imply results for general symmetric monotone norms, since a general symmetric monotone norm is not necessarily a submodular function (see e.g., [19]).

Patton et al. [38] studied submodular norm objectives (i.e., norms that also satisfies continuous submodular property). They showed that it can approximate well-known classes of norms, such as $\ell_p$ norms, ordered norms, and symmetric norms and applied it to a variety of problems such as online facility location, stochastic probing, and generalized load balancing. Recently, Kesselheim et al. [33] introduced the notion of p-supermodular norm and showed that every symmetric norm can be approximated by a p-supermodular norm. Leveraging the result, they obtain new algorithms online load-balancing and bandits with knapsacks, stochastic probing and so on.

**Multi-budgeted Optimization.** There is a body of literature in the problem of optimizing linear or submodular objectives over a combinatorial structure with additional budget constraints (see e.g., [39, 11, 26, 27, 16, 6, 25]). For a single budget constraint, randomized or deterministic PTASes have been developed for various combinatorial optimization problems (e.g. spanning trees with a linear budget [39]). Assuming that a pseudopolynomial time algorithm for the exact version of the problems exists, Grandoni and Zenklusen showed that one can obtain a PTAS for the corresponding problem with any fixed number of linear budgets [27]. More powerful techniques such as randomized dependent rounding and iterative rounding have been developed to handle more general submodular objectives and/or other combinatorial structures such as matroid or intersection of matroid (e.g., [26, 27, 16, 25]). Iterative rounding technique [26, 37] has been used in general norm minimization problems [13, 14]. Our algorithms for matching (Section 9) and knapsack cover (Section 5) also adopt the technique.

## 3 Preliminaries

Throughout this paper, for vector $\boldsymbol{v} \in \mathbb{R}_+^n$, define $\boldsymbol{v}^{\downarrow}$ as the non-increasingly sorted version of $\boldsymbol{v}$, and $\boldsymbol{v}[S] = \{\boldsymbol{v}_j \cdot \mathbf{1}[j \in S]\}_{j\in[n]}$ for any $S \subseteq [n]$. Let $\textsc{Top}_k : \mathbb{R}^n \to \mathbb{R}_+$ be the top-$k$ norm that returns the sum of the $k$ largest *absolute values* of entries in any vector, $k \leq |n|$. Denote $[n]$ as the set of positive integers no larger than $n \in \mathbb{Z}$, and $a^+ = \max\{a, 0\}$, $a \in \mathbb{R}$.

We say function $f : \mathbb{R}^n \to \mathbb{R}_+$ is a *norm* if: (i) $f(\boldsymbol{v}) = 0$ if and only if $\boldsymbol{v} = 0$, (ii) $f(\boldsymbol{u}+\boldsymbol{v}) \leq f(\boldsymbol{u}) + f(\boldsymbol{v})$ for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$, (iii) $f(\theta\boldsymbol{v}) = |\theta| f(\boldsymbol{v})$ for all $\boldsymbol{v} \in \mathbb{R}^n, \theta \in \mathbb{R}$. A norm $f$ is *monotone* if $f(\boldsymbol{v}) \leq f(\boldsymbol{u})$ for all $0 \leq \boldsymbol{v} \leq \boldsymbol{u}$, and *symmetric* if $f(\boldsymbol{v}) = f(\boldsymbol{v}')$ for any permutation $\boldsymbol{v}'$ of $\boldsymbol{v}$. We are also interested in the following special monotone symmetric norms.

**Top-$\ell$ norms.** Let $\ell \in [n]$. A function is a Top-$\ell$ norm, denoted by $\textsc{Top}_\ell : \mathbb{R}^n \to \mathbb{R}_+$, if for each input vector $\boldsymbol{v} \in \mathbb{R}^n$ it returns the sum of the largest $\ell$ *absolute values* of entries in $\boldsymbol{v}$. For non-negative vectors, it simply returns the sum of the largest $\ell$ entries. We notice that by letting $\ell \in \{1, n\}$, $\textsc{Top}_\ell$ recovers the $\mathcal{L}_\infty$ and $\mathcal{L}_1$ norms, respectively, thus it generalizes the latter two.

**Ordered norms.** Let $\boldsymbol{v} \in \mathbb{R}_+^n$ be a non-increasing non-negative vector. For each vector $\boldsymbol{v} \in \mathbb{R}^{\mathcal{X}}$, let $\boldsymbol{v}^{\downarrow} \in \mathbb{R}^{|\mathcal{X}|}$ denote its non-increasingly sorted version and define $|\boldsymbol{v}| = \{|\boldsymbol{v}_i| : i \in \mathcal{X}\} \in \mathbb{R}_+^{\mathcal{X}}$. A function is a $\boldsymbol{w}$-ordered norm (or simply an ordered norm), denoted by $\textsc{Ord}_{\boldsymbol{w}} : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}_+$, if for each input vector $\boldsymbol{v} \in \mathbb{R}^{\mathcal{X}}$ it returns the inner product of $\boldsymbol{w}$ and

$|\boldsymbol{v}|^{\downarrow}$; we obtain $\textsc{Ord}_{\boldsymbol{w}}(\boldsymbol{v}) = \boldsymbol{w}^{\top}\boldsymbol{v}^{\downarrow}$ whenever $\boldsymbol{v} \in \mathbb{R}_{+}^{\mathcal{X}}$. It is easy to see that, by having $\boldsymbol{v}$ as a vector of $\ell$ 1s followed by $(|\mathcal{X}| - \ell)$ 0s, $\textsc{Ord}_{\boldsymbol{w}}$ recovers $\textsc{Top}_{\ell}$. On the other hand, it is known that each ordered norm can be written as a conical combination of Top-$\ell$ norms, as in the following claim.

▷ Claim 2. (See, e.g., [13]). For each $\boldsymbol{v} \in \mathbb{R}_{+}^{\mathcal{X}}$ and another non-increasing vector $\boldsymbol{w} \in \mathbb{R}_{+}^{|\mathcal{X}|}$, one has

$$\textsc{Ord}_{\boldsymbol{w}}(\boldsymbol{v}) = \sum_{\ell=1}^{|\mathcal{X}|}(\boldsymbol{w}_{\ell} - \boldsymbol{w}_{\ell+1})\textsc{Top}_{\ell}(\boldsymbol{v}),$$

where we define $\boldsymbol{v}_{|\mathcal{X}|+1} = 0$.

The following lemma is due to Hardy, Littlewood and Pòlya. [29].

▶ **Lemma 3.** *([29]). If $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{R}_{+}^{\mathcal{X}}$ and $\alpha \geq 0$ satisfy $\textsc{Top}_{\ell}(\boldsymbol{v}) \leq \alpha \cdot \textsc{Top}_{\ell}(\boldsymbol{u})$ for each $\ell \in [|\mathcal{X}|]$, one has $f(\boldsymbol{v}) \leq \alpha \cdot f(\boldsymbol{u})$ for any symmetric monotone norm $f : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}_{+}$.*

## 4 A General Reduction to Multi-Budgeted Optimization Problem

In this section, we provide an equivalent formulation for the general symmetric norm minimization problem MinNorm (up to constant approximation factor). Recall that as defined in Section 1, we are given a set $\mathcal{U}$ of $n$ elements, and $\mathcal{F}$ represents a family of feasible subsets of $\mathcal{U}$. The goal of MinNorm is to find a feasible subset $S \in \mathcal{F}$ to minimize $f(\boldsymbol{v}[S])$, where $f$ is a symmetric monotone norm function. We say that we find a $c$-approximation for the problem for some $c \geq 1$, if we can find an $S$ such that $f(\boldsymbol{v}[S]) \leq c \cdot f(\boldsymbol{v}[S^*])$, where $S^*$ is the optimal solution. Since a general norm function is quite abstract and hard to deal with, we formulate the following (equivalent, up to constant approximation factor) optimization problem which is more combinatorial in nature.

▶ **Definition 4** (Logarithmic Budgeted Optimization (LogBgt)). *The input of a Logarithmic Budgeted Optimization Problem is a tuple $\eta = (\mathcal{U}; S_1, S_2, \ldots, S_T; \mathcal{F})$, where:*

- *$\mathcal{U}$ is a finite set with $n$ elements.*
- *$S_1, S_2, \cdots, S_T$ are disjoint subsets of $\mathcal{U}$, where $T = \lceil \log n \rceil$ is the number of sets. For $1 \leq i \leq T$, We refer to $S_i$ as the $i$-th group, and for any $u \in S_i$, we call $i$ the group index of $u$.*
- *$\mathcal{F}$ is a family of feasible subsets of $\mathcal{U}$. The size of $|\mathcal{F}|$ may be exponentially large in $n$, but we ensure that there exists a polynomial-time algorithm to decide whether $D \in \mathcal{F}$ for a subset $D \subseteq \mathcal{U}$.*

*For any $c' > 0$, we say a subset $D \subseteq \mathcal{U}$ is a **$c'$-valid** solution if and only if:*

1. *$D$ satisfies the feasibility constraint, i.e., $D \in \mathcal{F}$, and*
2. *$|D \cap S_i| \leq c' \times 2^i$ for all $1 \leq i \leq T$.*

*For any $c \geq c_0 \geq 1$, we define $(c, c_0)$-LogBgt problems as follows: Given an input $\eta$, the goal is to find a $c$-valid solution or certify that there is no $c_0$-valid solution. In particular, we denote $(c, 1)$-LogBgtas $c$-LogBgt.*

Notice that the structure of a problem is defined by $\mathcal{U}$ and $\mathcal{F}$ (for example, the vertex cover problem is given by vertex set $\mathcal{U}$ and $\mathcal{F}$ contains all subsets of $\mathcal{U}$ corresponding to a vertex cover), each problem corresponds to a MinNorm version and an LogBgt version. We show that solving LogBgt is equivalent to approximating MinNorm, up to constant approximation factors. In fact, the reduction from norm approximation to optimization problem with multiple budgets has been implicitly developed in prior work [13, 32]. For generality and ease of usage, we encapsulate the reduction in the following general theorem.

▶ **Theorem 5.** *For any $c \geq 1$ ($c$ can depend on $n$) and $\epsilon > 0$, if we can solve $c$-LogBgt in polynomial time, we can approximate the MinNorm problem within a factor of $(4c + \epsilon)$ in polynomial time. On the other hand, if we can find a $c$-approximation for MinNorm in polynomial time, we can solve the $47c^2$-LogBgt in polynomial time.*

We defer the full details to the full version [17] and now outline the main ideas behind the proof.

**Reducing MinNorm to LogBgt.** Suppose we are given an instance of the MinNorm problem and have access to an algorithm for LogBgt. Let $S^*$ denote an optimal solution for MinNorm, and let $\boldsymbol{v}^{\downarrow}[S^*]$ be the vector of values sorted in nonincreasing order. Our strategy is to "guess" an approximation of $\boldsymbol{v}^{\downarrow}[S^*]$ by enumerating candidate threshold vectors (each threshold vector is of the form $\{t_\ell\}_{\ell \in \mathrm{POS}}$, i.e., we guess the values at positions $\mathrm{POS} = \{\min\{2^s, n\} : s \geq 0\}$. ) One can show (The details can be found in the full version [17]) that this enumeration can be done in poly-time and is guaranteed to include a candidate that is "close" to the true vector $\boldsymbol{v}^{\downarrow}[S^*]$.

For each candidate enumeration $\{t_\ell\}_{\ell \in \mathrm{POS}}$, we construct an associated LogBgt instance as follows. For each index $i$ and for each candidate threshold indexed by $\ell$ (with $\ell = 2^j < n$), if $\max\{t_{\min\{2\ell,n\}}, \varepsilon t_1^{\downarrow}/n\} < v_i \leq t_\ell$, we assign index $i$ to a set $S_{j+1}$. Additionally, every index $i$ for which $v_i \leq \max\{t_n, \varepsilon t_1^{\downarrow}/n\}$ is placed in the last set $S_T$. With this construction, when the guessed threshold vector is "close" to $\boldsymbol{v}^{\downarrow}[S^*]$, solving the resulting LogBgt instance produces a solution that is within a constant factor of the optimal MinNorm value.

**Reducing LogBgt to MinNorm.** In the reverse direction, suppose we are given an instance of the LogBgt problem and a $c$-approximation algorithm for MinNorm. In this case, we construct a corresponding MinNorm instance by setting, for each element $i$, a value that reflects its membership in one of the sets $S_j$ (or its exclusion from all such sets). The idea is that the structure of the LogBgt instance is encoded in the chosen values $\{v_i\}$ so that a $c$-approximation for the resulting MinNorm instance yields a solution for the original LogBgt problem. In particular, one can show that a $c$-approximation for MinNorm leads to a $47c^2$-approximation for LogBgt.

**A Logarithmic Approximation.** Based on Theorem 5, we can easily deduce the following general theorem. We use $\mathfrak{A}$ to denote a general combinatorial optimization problem with the min-sum objective function $\min_{S \in \mathcal{F}} \boldsymbol{v}(S)$, where we write $\boldsymbol{v}(S) = \sum_{e \in S} v_e$ and $\mathcal{F}$ is the set of feasible solutions.

▶ **Theorem 6.** *If there is a poly-time approximation algorithm for the min-sum problem $\mathfrak{A}$ (with approximation factor $\alpha \geq 1$), there is a poly-time factor $(4\alpha\lceil \log n \rceil + \epsilon)$ approximation algorithm for the corresponding MinNorm problem for any fixed constant $\epsilon > 0$.*

**Proof.** By Theorem 5, we just need to find a poly-time $O(\log n)$ approximation algorithm for the LogBgt version. Consider the input $\mathcal{U}, \mathcal{F}, S_1, \cdots, S_T$ where $T = \lceil \log n \rceil$ (recall the definitions in Section 4). Then we construct $v_e$ for each $e \in \mathcal{U}$ by:

1. $v_e = 0$ if $e \notin S_1, S_2, \cdots, S_T$;
2. $v_e = 1/2^i$ if $e \in S_i$.

Then, if there is a 1-valid solution for the LogBgt problem, we can see that there is a feasible set $S^\star \in \mathcal{F}$ with $\boldsymbol{v}(S^\star) \leq T$. Then, by the assumption of the theorem, the approximation algorithm for $\mathfrak{A}$ can output a feasible solution $S \in \mathcal{F}$ with $\boldsymbol{v}(S) \le \alpha T$. This further implies that $S$ is a $\alpha T$-valid solution, because

$$|S \cap S_i| = \boldsymbol{v}(S \cap S_i) \cdot 2^i \le \alpha T \cdot 2^i, \text{ for each } i \in [T].$$

This means that the $\alpha T$-LogBgt problem can be solved in polynomial time. By Theorem 5, we complete the proof. ◀

## 5 Multi-dimensional Knapsack Cover Problem

In this section, we consider the multi-dimensional knapsack cover problem defined as follows.

▶ **Definition 7** (Min-norm $d$-dimensional Knapsack Cover Problem (MinNorm-KnapCov)). *Let $d$ be a positive integer. We are given a set of items $\mathcal{U} = \{1, 2, \ldots, n\}$, where each item $i \in \mathcal{U}$ has a weight vector $w_i \in \mathbb{R}^d$. The feasible set $\mathcal{F}$ is defined as:*

$$\mathcal{F} = \left\{ D \subseteq \mathcal{U} : \sum_{v \in D} w_{v,i} \ge 1 \quad \forall i \in \{1, 2, \ldots, d\} \right\}.$$

*Now, given a symmetric monotone norm $f$ and a value vector $\boldsymbol{v} \in \mathbb{R}^{\mathcal{U}}_{\ge 0}$, we can define the norm minimization problem for $d$-dimensional Knapsack Cover and denote it as MinNorm-KnapCov.*

In light of Theorem 5, we introduce $T = \lceil \log n \rceil$ disjoint sets $S_1, S_2, \ldots, S_T$ and consider the LogBgt problem with $(\mathcal{U}; S_1, \ldots, S_T; \mathcal{F})$, which We denote as LogBgt-KnapCov. We consider LogBgt-KnapCov for two cases: (1) $d = O(1)$ and (2) $d = O(\sqrt{\log n / \log\log W})$ ($W$ will be defined in Section 5.2). For both cases, we use the following natural linear programming formulation for LogBgt-KnapCov :

$$\begin{aligned} \min \quad & \qquad 0 \\ s.t. \quad & \sum_{v \in \mathcal{U}} x_v w_{v,i} \ge 1 \quad \forall 1 \le i \le d \\ & \sum_{v \in S_j} x_v \le 2^j \quad \forall 1 \le j \le T \\ & 0 \le x_v \le 1 \quad \forall v \in S_j, 1 \le j \le T \end{aligned} \tag{LP-KnapCover-1}$$

For both cases, we develop a method called **partial enumeration**. Partial enumeration lists a subset of possible partial solutions for the first several groups. Here is the complete definition:

▶ **Definition 8** (Partial Enumeration). *For a $(c, c_0)$-LogBgt problem with set $S_1, S_2, \cdots S_T$, the* ***partial enumeration*** *algorithm first determine a quantity $T_0$ (depending on the problem at hand). The partial enumeration algorithm returns a subset $X \subseteq 2^{S_1} \times 2^{S_2} \times \cdots \times 2^{S_{T_0}}$. Each element of $X$ is a* ***partial solution*** *$(D_1, D_2, \cdots, D_{T_0})$ (Recall the definition of partial solution: $D_i \subseteq S_i$), and this algorithm ensures:*

1. *If there exists a $c_0$-valid solution, then at least one partial solution $(D_1, D_2, \cdots, D_{T_0}) \in X$ satisfies that there exists an $c$-valid* ***extended solution*** *(A solution $D$ is called an extended solution of a partial solution $(D_1, \cdots, D_{T_0})$ if $D \cap S_i = D_i$ for all $i = 1, 2, \cdots, T_0$).*
2. *The size of $X$ is polynomial, and this partial enumeration algorithm runs in polynomial time.*

## 5.1 An Algorithm for $d = O(1)$

When $d = O(1)$, for any $\varepsilon > 0$, we choose $T_0 = \log d + \log(1/\varepsilon) + O(1)$ and apply partial enumeration. Recall that a partial solution can be described as a vector $(D_1, D_2, \cdots, D_{T_0})$, where $D_i \subseteq S_i$ for $i = 1, 2, \cdots, T_0$. Our partial enumeration algorithm in this section simply outputs the set of all possible **partial solutions** (i.e., $X = X_1^{all} \times X_2^{all} \times \ldots \times X_{T_0}^{all}$ where $X_i^{all} = \{X' \subseteq S_i : |X'| \leq 2^i\}$).

For each partial solution, we use it to modify the Linear Program (LP-KnapCover-1) and perform a rounding algorithm. The rounding algorithm do the following steps:

1. Modify (LP-KnapCover-1) by removing $S_1, \ldots, S_{T_0}$ according to the partial solution. Then confirm there is no solution or find an extreme point $x^*$.
2. Round all $x_u^* > 0$ to 1 for $u \in \bigcup_{T_0 < j \leq T} S_j$

We can prove that the rounding algorithm can obtain a $(1 + \varepsilon)$-valid solution based on a specific partial solution. Ultimately, we obtain a $(1 + \varepsilon)$-valid solution or ensure no 1-valid solution for this enumeration result, and this algorithm runs in $\exp(O(d \log n/\varepsilon))$-time. The details are provided in the full version [17].

▶ **Theorem 9.** *If $d$ is a constant, then for any constant $\varepsilon > 0$, there exists an polynomial-time algorithm which can solve $(1 + \varepsilon)$-LogBgt-KnapCov. Thus we have a polynomial-time $(4 + \varepsilon)$-approximation algorithm for MinNorm-KnapCov when $d = O(1)$.*

## 5.2 An Algorithm for Larger d

In this subsection, we provide a polynomial-time constant-factor approximation algorithm for $d = O\left(\sqrt{\frac{\log n}{\log \log W}}\right)$, where $W$ is defined as

$$W = \max_{1 \leq i \leq d} \frac{\max_{u \in \mathcal{U}} w_{u,i}}{\min_{u \in \mathcal{U}, w_{u,i} > 0} w_{u,i}}.$$

To ensure there exist valid solutions, $\{u \in \mathcal{U} : w_{u,i} > 0\}$ must be a non-empty set.

The second algorithm employs the same rounding procedure but modifies the partial enumeration method. The new algorithm choose $T_0 = \lceil \log d \rceil$. For $1 \leq j \leq T_0$, it partitions $S_j$ into multiple subsets based on vectors of size $d$, which represent the logarithms of weights. Instead of enumerating all subsets, we only enumerate the number of elements within each subset and then take double the number of any elements in this subset. Further details are provided in the full version [17].

We ensure that this partial enumeration algorithm runs in polynomial time if $d = O\left(\sqrt{\frac{\log n}{\log \log W}}\right)$. In addition, the rounding procedure can find a 2-valid solution based on at least one partial solution. The complete details are presented in the full version [17].

▶ **Theorem 10.** *There exists a poly$(n, \log(W))$ algorithm that can solve 2-LogBgt-KnapCov when $d = O\left(\sqrt{\frac{\log n}{\log \log W}}\right)$. Thus we have a $(4 + \varepsilon)$-approximation algorithm for MinNorm-KnapCov with $d = O\left(\sqrt{\frac{\log n}{\log \log W}}\right)$.*

## 6 Interval Cover Problem

In this section, we study the norm minimization for the interval cover problem, which is defined as follows:

▶ **Definition 11** (Min-norm Interval Cover Problem (MinNorm-IntCov)). *Given a set $\mathcal{U}^{\text{int}}$ of intervals and a target interval $\Gamma$ on the real axis, a feasible solution of this problem is a subset $D \subseteq \mathcal{U}^{\text{int}}$ such that $D$ fully covers the target interval $\Gamma$ (i.e., $\Gamma \subseteq \bigcup_{I \in D} I$). Suppose $\boldsymbol{v} \in \mathbb{R}_{\geq 0}^{\mathcal{U}^{\text{int}}}$ is a value vector and $f(\cdot)$ is a monotone symmetric norm function. Our goal is to find a feasible subset $D$ such that $f(\boldsymbol{v}[D])$ is minimized. We denote the problem as MinNorm-IntCov.*

In light of Theorem 5, we can focus on obtaining a constant-factor approximation algorithm for $(c, c_0)$-LogBgt-IntCov. The input of a LogBgt-IntCov problem is a tuple $\eta^{\text{int}} = (\mathcal{U}^{\text{int}}; S_1^{\text{int}}, \ldots, S_T^{\text{int}}; \Gamma)$, which is the LogBgt problem with input $(\mathcal{U}^{\text{int}}; S_1^{\text{int}}, \ldots, S_T^{\text{int}}; \mathcal{F}^{\text{int}})$ where the set of feasible solutions is $\mathcal{F}^{\text{int}} = \{D \subseteq \mathcal{U}^{\text{int}} : \Gamma \subseteq \bigcup_{I \in D} I\}$.

In this section, we begin by transforming the interval cover problem into a new problem called the *tree cover* problem. The definitions of both problems are provided in Section 6.1. These transformation results in only a constant-factor loss in the approximation factor (i.e., if a polynomial-time algorithm can solve $c$-LogBgt-TreeCov for some constant $c$, then there exists a polynomial-time constant-factor approximation algorithm for LogBgt-IntCov).

Next, we focus on LogBgt-TreeCov. We first employ the **partial enumeration** algorithm, as defined in Section 5, to list partial solutions for the first $T_0 = \lfloor \log\log\log n \rfloor$ sets. The details of this process are provided in Section 6.2. Following partial enumeration, we apply a rounding algorithm to evaluate each partial solution. The entire rounding process is detailed in Section 6.3.

## 6.1 From Interval Cover to Tree Cover

We first introduce some notations for the tree cover problem. Denote a rooted tree as $G = (V, E, r)$, where $(V, E)$ forms an undirected tree and $r$ is the root. For each node $u \in V$, let $\text{Ch}(u)$ be the set of children of $u$, and $\text{Des}(u)$ be the set of all descendants of $u$ (including $u$). It is easy to see that $\text{Des}(u) = \{u\} \cup \text{Des}(\text{Ch}(u))$. For a subset of vertices $P \subseteq V$, we define $\text{Des}(P) = \bigcup_{u \in P} \text{Des}(u)$. We also define $\text{Par}(u)$ as the parent node of $u$ and define $\text{Anc}(u)$ as the set of ancestors of $u$ ($\text{Anc}(r) = \{r\}$, and for any $u \in V \backslash \{r\}$, $\text{Anc}(u) = \{u\} \cup \text{Anc}(\text{Par}(u))$). In addition, we define the set of leaves $\text{Leaf}(G) = \{u \in V : \text{Ch}(u) = \emptyset\}$.

▶ **Definition 12** (LogBgt Tree Cover Problem (LogBgt-TreeCov)). *We are given a tuple $\eta^{\text{tr}} = (\mathcal{U}^{\text{tr}}; S_1^{\text{tr}}, \ldots, S_T^{\text{tr}}; G)$, where $G = (V, E, r)$ is a rooted tree and $\mathcal{U}^{\text{tr}} = V \setminus \{r\}$. $T = \lceil \log n \rceil$, and $S_1^{\text{tr}}, S_2^{\text{tr}}, \cdots S_T^{\text{tr}}$ is a partition of $V \backslash \{r\}$ (and $S_i^{\text{tr}}$ is called the ith group), and $r$ is not in any group. The partition $S_1^{\text{tr}}, S_2^{\text{tr}}, \cdots S_T^{\text{tr}}$ satisfies the following property: For any node $u \in S_i^{\text{tr}}$ and an arbitrary child $v$ of $u$, $v$ belongs to group $S_j^{\text{tr}}$ with $j > i$. For each $u \in V \setminus \{r\}$, we define $\text{Id}(u) = j$ if $u \in S_j^{\text{tr}}$. In particular, we denote $\text{Id}(r) = 0$. So $\text{Id}(u) > \text{Id}(\text{Par}(u))$ for all $u \in \mathcal{U}^{\text{tr}}$.*

*A feasible solution for the tree cover problem is a subset $D \subseteq \mathcal{U}^{\text{tr}}$ such that the descendants of $D$ covers all leaves. Formally, the feasible set is defined as*

$$\mathcal{F}^{\text{tr}} = \{D \subseteq \mathcal{U}^{\text{tr}} : \text{Leaf}(G) \subseteq \text{Des}(D) = \bigcup_{u \in D} \text{Des}(u)\}.$$

We prove the following theorem to reduce the interval cover problem to the tree cover problem. The proof of the theorem can be found in the full version [17].

▶ **Theorem 13.** *If there exists a polynomial-time algorithm for the $(c, 8c_0)$-LogBgt-TreeCov problem, then there exists a polynomial-time algorithm for the $(3c, c_0)$-LogBgt-IntCov problem.*

Based on this theorem, we mainly need to deal with the tree cover problem in the following subsections.

## 6.2 Partial Enumeration Method for Tree Cover Problem

In this subsection, we present a partial enumeration algorithm for the LogBgt-TreeCov problem. Recall that we introduced the concept of partial enumeration in Section 5. For a LogBgt-TreeCov problem with input $(\mathcal{U}^{\text{tr}}; S_1^{\text{tr}}, S_2^{\text{tr}}, \ldots, S_T^{\text{tr}}; G)$, where $G = (V, E, r)$ is a rooted tree, and $n = |\mathcal{U}^{\text{tr}}|$. we set $T_0 = \lfloor \log\log\log n \rfloor$ and perform partial enumeration for the first $T_0$ sets. The goal is to find a set $X \subseteq 2^{S_1^{\text{tr}}} \times 2^{S_2^{\text{tr}}} \times \cdots \times 2^{S_{T_0}^{\text{tr}}}$ such that there exists a partial solution $(D_1, D_2, \ldots, D_{T_0}) \in X$ satisfying: at least one of the partial solution can be extended to a $c$-valid solution for some constant $c$.

In this subsection, we define $\text{Id}(u)$ as the group index of $u$ for $u \in \mathcal{U}^{\text{tr}}$. Now we focus on the LogBgt-TreeCov problem. For each $u \in \mathcal{U}^{\text{tr}}$, define the first type of cost $C_1(u) = \frac{1}{2^{\text{Id}(u)}}$. We then define the second type of cost:

$$C_2(u) = \begin{cases} C_1(u) & \text{if } u \in \text{Leaf}(G) \\ \min\{C_1(u), \sum\limits_{v \in \text{Ch}(u)} C_2(v)\} & \text{if } u \notin \text{Leaf}(G) \end{cases}$$

Intuitively, the cost $C_1(u)$ represents the "cost" of selecting $u$, as it indicates the proportion of the group that $u$ occupies. Meanwhile, $C_2(u)$ denotes the minimum cost required to cover $u$ using its descendants.

We now present the partial enumeration algorithm. The pseudo-code can be found in the full version [17]. Here, we briefly describe the main idea of the partial enumeration algorithm.

We employ a depth-first search (DFS) strategy to explore most of the states in the search space. During the search process, we maintain two sets:

- $P \subseteq \mathcal{U}^{\text{tr}}$, representing the set of candidate elements that can still be explored, i.e., $\text{Des}(P)$ contains all uncovered leaves.
- $D \subseteq \mathcal{U}^{\text{tr}}$, storing the elements that have already been selected as part of the partial solution.

Initially, $P = \text{Ch}(r)$ is the child set of the root, and $D = \emptyset$. At each recursive step, we select $u \in P$ with the smallest group index. The recursion proceeds by exploring two possibilities:

1. Adding $u$ to the partial solution, i.e., including $u$ in $D$ and continuing the search.
2. Excluding $u$ from the partial solution, i.e., replacing $u$ with its child nodes while keeping $D$ unchanged. (If $u$ is a leaf, this option is not applicable.)

The search terminates when $(P, D)$ fails to satisfy at least one of the following conditions:

1. $\exists u \in P$, $\text{Id}(u) \le T_0$
2. $\forall u \in D$, $C_2(u) > \frac{1}{\log n}$
3. $\left(\sum_{v \in D} C_2(v)\right) + \left(\sum_{u \in P} C_2(u)\right) \le 2c_0 T$
4. $\forall 1 \le i \le T_0, \quad |D \cap S_i^{\text{tr}}| \le 2c_0 \cdot 2^i$

The first and fourth conditions are derived from the objective of the Partial Enumeration Method. Regarding the second condition, we observe that for all $u \in \mathcal{U}^{\text{tr}}$, it holds that $\text{Id}(u) \le T_0$ and $C_1(u) > \frac{1}{\log n}$. Furthermore, if $C_2(u) \le \frac{1}{\log n}$, the impact of ignoring $u$ is negligible.

The third condition is based on the property that for any $2c_0$-valid solution $D^* \subseteq \mathcal{U}^{\text{tr}}$, it satisfies:

$$\sum_{u \in D^*} C_2(u) \le 2c_0 T.$$

Due to these conditions, for each group $S_j$ where $1 \le j \le T_0$, we only need to determine at most $2c_0T^2$ items. Consequently, we can establish that our partial enumeration algorithm runs in polynomial time.

We then present the following theorem:

▶ **Theorem 14.** *There exists a polynomial-time partial enumeration algorithm for the* LogBgt-TreeCov *problem with the following guarantee: If the input $\eta^{\mathrm{tr}}$ has a $c_0$-valid solution, then at least one of the output partial solutions has a $2c_0$-valid extended solution.*

The complete proof can be found in the full version [17].

## 6.3 A Rounding Algorithm for Tree Cover Problem

We now focus on the $(c, c_0)$-LogBgt-TreeCov problem with input $\eta^{\mathrm{tr}} = (\mathcal{U}^{\mathrm{tr}}; S_1^{\mathrm{tr}}, \ldots, S_T^{\mathrm{tr}}; G)$, where $G = (V, E, r)$ is a rooted tree. Let $L = \mathrm{Leaf}(G)$ be the set of leaves. Recall that $\mathrm{Anc}(u)$ represents the set of ancestors of node $u$. For sets $\mathcal{V}, \mathcal{L} \subseteq \mathcal{U}^{\mathrm{tr}}$ and $c \ge 1$, we express the formulation of the linear program as follows:

$$\begin{aligned}
\min \quad & 0 \\
s.t. \quad & \sum_{v \in \mathrm{Anc}(u) \cap \mathcal{V}} x_v = 1 \quad \forall u \in \mathcal{L} \\
& \sum_{v \in S_i^{\mathrm{tr}} \cap \mathcal{V}} x_v \le c \cdot 2^i \quad \forall T_0 + 1 \le i \le T \\
& x_v \ge 0 \quad \forall v \in \mathcal{V}
\end{aligned} \qquad \text{(LP-Tree-Cover(c,}\mathcal{V}\text{,}\mathcal{L}\text{))}$$

We call $\sum_{v \in S_i^{\mathrm{tr}} \cap \mathcal{V}} x_v \le c \cdot 2^i$ *cardinality constraints*, and call $\sum_{v \in \mathrm{Anc}(u) \cap \mathcal{V}} x_v = 1$ *feasibility constraints.* Recall that $T_0 = \lfloor \log\log\log n \rfloor$. Also, define $T_1 = \lfloor \log\log n \rfloor$.

The algorithm is as follows, and the pseudocode is provided in the full version [17]:

1. Check if LP-Tree-Cover($2c_0$, $V_0$, $L_0$) has a feasible solution. If so, obtain an extreme point $x^*$. Otherwise, confirm that there is no such integral solution.
2. Remove the leaves $u$ with $x_u^* = 0$, and delete all the descendants of their parents. Then $\mathrm{Par}(u)$ becomes a leaf. Repeat this process until $x_u^* \ne 0$ for each leaf $u$. Let the modified node set and leaf set be $V_1$ and $L_1$, respectively.
3. For $u \in V_1$, attempt to round $x_u^*$. If $x_u^* \ge 1/2$, round it to 1. If $x_u^* > 0$, and $u$ is not a leaf in $S_{T_1+1}^{\mathrm{tr}} \cup \cdots \cup S_T^{\mathrm{tr}}$, also round it to 1. In all other cases, round $x_u^*$ to 0. Let $D'$ be the set of nodes $u$ for which $x_u^*$ was rounded to 1. Note that $D'$ may not cover $L_1$.
4. Remove all descendants in $D'$, and attempt to choose another set from $S_{T_0+1}^{\mathrm{tr}} \cup \cdots \cup S_{T_1}^{\mathrm{tr}}$ to cover all leaves. Formalize this objective as LP-Tree-Cover. Specifically,
   - $V_2 = (V_1 \setminus \mathrm{Des}(D')) \cap \{u \in \mathcal{U}^{\mathrm{tr}} : T_0 + 1 \le \mathrm{Id}(u) \le T_1\}$, and
   - $L_2 = (V_2 \cap L_1) \cup \{u \in V_2 : \exists v \in \mathrm{Ch}(u) \cap (V_1 \setminus V_2), (V_2 \setminus \mathrm{Des}(D')) \cap \mathrm{Des}(v) \cap L_1 \ne \emptyset\}$.

   To understand this, observe that $V_2$ consists of the nodes in the $(T_0 + 1)$th to $T$th groups that remain uncovered. The set $L_2$ includes nodes in $V_2$ that are either leaves or have at least one uncovered child with a group index greater than $T_0$ (i.e., at least one descendant leaf remains uncovered).

   Then solve LP-Tree-Cover($2c_0$, $V_2$, $L_2$). The fact that this problem must have feasible solutions is proved later, so we do not need to consider the case of no solution.
5. Let $x^{**}$ be an extreme point of LP-Tree-Cover($2c_0$, $V_2$, $L_2$). For each $u \in V_2$, round it to 1 if and only if $x_u^{**} > 0$. Let $D'' = \{u \in V_2 : x_u^{**} > 0\}$, then $D''$ covers $L_2$.
6. Combine the three parts of the solution. That is, return $\left(\bigcup_{i=1}^{T_0} D_i\right) \cup D' \cup D''$.

We prove the following lemma for the above algorithm:

▶ **Lemma 15.** *There exists a polynomial-time algorithm such that, if $T_0 = \lfloor \log\log\log n \rfloor$ and a partial solution $(D_1, \dots, D_{T_0})$ has a $2c_0$-valid extended solution, then the algorithm returns a $(4c_0 + 1)$-valid solution.*

By combining Theorem 14 and Lemma 15, we establish the following theorem:

▶ **Theorem 16.** *For any $c_0 \geq 1$, there exists a polynomial-time algorithm for $(4c_0 + 1, c_0)$-LogBgt-TreeCov.*

Furthermore, applying Theorem 13 and Theorem 16, we obtain the following result:

▶ **Theorem 17.** *There exists a polynomial-time algorithm that solves the $(3(32c_0 + 1), c_0)$-LogBgt-IntCov. Consequently, we obtain a polynomial-time constant-factor approximation algorithm for MinNorm-IntCov.*

The complete proofs are provided in the full version [17].

## 7 Integrality Gap for Perfect Matching, s-t Path, and s-t Cut

In this section, we argue that it may be challenging to achieve constant approximations for the norm optimization problems for perfect matching, $s$-$t$ path, and $s$-$t$ cut just by LP rounding. We show that the natural linear programs have large integrality gaps.

▶ **Definition 18** (Min-Norm $s$-$t$ Path Problem (MinNorm-Path)). *Given a directed graph $G^{\text{path}} = (V^{\text{path}}, \mathcal{U}^{\text{path}})$ (here $V^{\text{path}}$ is the set of vertices and $\mathcal{U}^{\text{path}}$ is the set of edges) and nodes $s, t \in V^{\text{path}}$, define the feasible set:*

$$\mathcal{F}^{\text{path}} = \{D \subseteq \mathcal{U}^{\text{path}} : D \text{ forms a path from } s \text{ to } t\}.$$

*For the MinNorm version, we are also given a monotone symmetric norm $f$ and a value vector $\boldsymbol{v} \in \mathbb{R}_{\geq 0}^{\mathcal{U}^{\text{path}}}$. The goal is to select an $s$-$t$ path $D \in \mathcal{F}^{\text{path}}$ that minimizes $f(\boldsymbol{v}[D])$.*

In light of Theorem 5, we can define the LogBgt-Path problem with input tuple $\eta^{\text{path}} = (\mathcal{U}^{\text{path}}; S_1^{\text{path}}, \dots, S_T^{\text{path}}; G^{\text{path}}; s; t)$, which is the LogBgt problem defined in Definition 4 with input $(\mathcal{U}^{\text{path}}; S_1^{\text{path}}, \dots, S_T^{\text{path}}; \mathcal{F}^{\text{path}})$.

▶ **Definition 19** (Min-Norm Perfect Matching Problem (MinNorm-PerMat)). *Given a bipartite graph $G^{\text{pm}} = (L, R, \mathcal{U}^{\text{pm}})$ with $|L| = |R|$, define the feasible set:*

$$\mathcal{F}^{\text{pm}} = \{D \subseteq \mathcal{U}^{\text{pm}} : D \text{ forms a perfect matching in } G^{\text{pm}}\}.$$

*For the MinNorm version, we are also given a monotone symmetric norm $f$ and a value vector $\boldsymbol{v} \in \mathbb{R}_{\geq 0}^{\mathcal{U}^{\text{pm}}}$. The goal is to select $D \in \mathcal{F}^{\text{pm}}$ that minimizes $f(\boldsymbol{v}[D])$.*

We define the LogBgt-PerMat problem with $\eta^{\text{pm}} = (\mathcal{U}^{\text{pm}}; S_1^{\text{pm}}, \dots, S_T^{\text{pm}}; G^{\text{cut}}; s; t)$ as the LogBgt problem with $(\mathcal{U}^{\text{pm}}; S_1^{\text{pm}}, \dots, S_T^{\text{pm}}; \mathcal{F}^{\text{pm}})$.

Due to Theorem 5, we establish the equivalence between approximating the MinNorm problem and the LogBgt problem. Hence, if we can show that the LogBgt version is hard to approximate, then the same hardness (up to constant factor) also applies to the MinNorm version.

Now, we consider using the linear programming rounding approach to approximate the LogBgt problem with $\eta = (\mathcal{U}; S_1, \dots, S_T; \mathcal{F})$. Such an algorithm proceeds according to the following pipeline:

- First, we formulate the natural linear program (for $c \geq 1$) as the following:

$$\begin{array}{llll} \min & 0 & & \\ s.t. & x \text{ satisfies} & \text{relaxed constraints of } \mathcal{F}, & \\ & \sum_{u \in S_i} x_u \leq c \cdot 2^i & \forall 1 \leq i \leq T, & \\ & x_u \geq 0 & \forall u \in \mathcal{U}. & \end{array} \qquad \text{(LP-LBO-Original}(\eta,\ c))$$

- Then, to solve the $(c, c_0)$-LogBgt problem, we first check whether LP-LBO-Original($\eta$, $c_0$) has feasible solutions. If feasible, we use a rounding algorithm to find an integral solution for (LP-LBO-Original($\eta$, $c$)).

Since the factor $c$ in LP-LBO-Original($\eta$, $c$) determines the approximation factor, we study the following linear program.

$$\begin{array}{llll} \min & z & & \\ s.t. & x \text{ satisfies} & \text{relaxed constraints of } \mathcal{F}, & \\ & \sum_{u \in S_i} x_u \leq z \cdot 2^i & \forall 1 \leq i \leq T, & \\ & x_u \geq 0 & \forall u \in \mathcal{U}. & \end{array} \qquad \text{(LP-LBO}(\eta))$$

Clearly, if LP-LBO-Original($\eta$, $c$) has a feasible solution, the optimal value of LP-LBO($\eta$) is at most $c$. Suppose we can round a fractional solution LP-LBO-Original($\eta$, $c$) to an integral feasible $c'$-valid solution $\bar{x}$ for some constant $c'$ (i.e., $\bar{x}$ satisfies $\mathcal{F}$ and $\sum_{u \in S_i} \bar{x}_u \leq c' \cdot 2^i \ \forall 1 \leq i \leq T$). Then, we get an integral solution for LP-LBO($\eta$) with objective value $c'$, contradicting the fact that the integrality gap of LP-LBO($\eta$) is $\omega(1)$. Hence, we can conclude that if the integrality gap of LP-LBO($\eta$) is $\omega(1)$, it would be difficult to derive a constant factor approximation algorithm for both LogBgt and MinNorm-version of the problem using the LP LP-LBO-Original($\eta, c$).

## 7.1 Reduction from Perfect Matching to s-t Path

For an LogBgt perfect matching problem with $\eta^{\text{pm}} = (\mathcal{U}^{\text{pm}}; S_1^{\text{pm}}, \dots, S_T^{\text{pm}}; G^{\text{pm}})$, where $G^{\text{pm}} = (L, R, E)$ is a bipartite graph, we consider the following LP (on the left). The LP on the right is for LogBgt-Path with $\eta^{\text{path}} = (\mathcal{U}^{\text{path}}; S_1^{\text{path}}, \dots, S_T^{\text{path}}; G^{\text{path}}; s; t)$.

$$\begin{array}{lll} \min & z & \\ \text{s.t.} & \sum_{i \in L, (i,j) \in E} x_{i,j} = 1 & \forall j \in R, \\ & \sum_{j \in R, (i,j) \in E} x_{i,j} = 1 & \forall i \in L, \\ & \sum_{e \in S_i^{\text{pm}}} x_e \leq z \cdot 2^i & \forall 1 \leq i \leq T, \\ & 0 \leq x_{i,j} \leq 1 & \forall (i,j) \in \mathcal{U}^{\text{pm}}. \end{array}$$

(LP-LBO-PM($\eta^{\text{pm}}$))

$$\begin{array}{lll} \min & z & \\ \text{s.t.} & \sum_{j \in V^{\text{path}}, (i,j) \in \mathcal{U}^{\text{path}}} x_{i,j} = 0 & \forall i \in V^{\text{path}} \setminus \{s, t\}, \\ & \sum_{j \in V^{\text{path}}, (s,j) \in \mathcal{U}^{\text{path}}} x_{s,j} = 1 & \\ & \sum_{i \in V^{\text{path}}, (i,t) \in \mathcal{U}^{\text{path}}} x_{i,t} = 1 & \\ & \sum_{e \in S_i} x_e \leq z \cdot 2^i & \forall 1 \leq i \leq T, \\ & 0 \leq x_e \leq 1 & \forall e \in \mathcal{U}^{\text{path}}. \end{array}$$

(LP-LBO-Path($\eta^{\text{path}}$))

We have the following theorem showing that LogBgt-Path problem is not harder than LogBgt-PerMat problem.

▶ **Theorem 20.** *For any arbitrary function $\alpha(n) \geq 1$, We have the following conclusions:*

**(a)** *If the integrality gap of* (LP-LBO-PM($\eta^{\text{pm}}$)) *is no more than $\alpha(n)$ for all instances $\eta^{\text{pm}}$ of the* LogBgt-PerMat *problem, then the integrality gap of* (LP-LBO-Path($\eta^{\text{path}}$)) *is also $O(\alpha(n))$ for all instances of the* LogBgt-Path *problem.*

**(b)** *If we have a polynomial-time $\alpha(n)$-approximation algorithm for the* MinNorm-PerMat *problem, then we have a polynomial-time $O(\alpha(n))$-approximation algorithm for the* MinNorm-Path *problem.*

## 7.2 Integrality Gaps for Min-Norm s-t Path and Min-Norm Perfect Matching

▶ **Theorem 21.** *For infinitely many $n$, there exists an instance $\eta^{\text{cut}}$ of size $n$ such that the integrality gap of* (LP-LBO-Path($\eta^{\text{path}}$)) *can be $\Omega(\log n)$. Thus, the relaxations of the natural linear programming of both* LogBgt-Path *and* LogBgt-PerMat *have integrality gaps of $\Omega(\log n)$.*

The proof of the above theorem can be found in the full version [17]. Also, we have the result about the integrality gap for $s$-$t$ cut, which is also omitted and can be found in the full version.

*Remark.* In the example in the proof (see the full version [17] for details), the gap between any feasible subset of $\mathcal{U}' = \{e \in \mathcal{U} : x_e > 0\}$ ($\{x_e\}_{e\in\mathcal{U}}$ is the fractional solution) and the fractional solution is larger than any given constant. Thus any rounding algorithm that deletes zero-value variables (including the rounding algorithm we developed in this paper and the iterative rounding method in [13]) cannot successfully yield a constant-factor approximation.

## 8 An Algorithm for Min-Norm s-t Path Problem

Recall that we define the MinNorm-Path problem and prove that the natural linear program has a large integrality gap in Section 7. In this section, we provide a factor $\alpha$ approximation algorithm that runs in $n^{O(\log\log n/\alpha)}$ time, for any $9 \leq \alpha \leq \log\log n$. In particular, this implies an $O(\log\log n)$-factor polynomial-time approximation algorithm and a constant-factor quasi-polynomial $n^{O(\log\log n)}$-time algorithm for MinNorm-Path. Note that this does not contradict Theorem 21, since we do not use the LP rounding approach in this section.

In light of Theorem 5 with $\varepsilon = 1$, we consider $\frac{\alpha-1}{4}$-LogBgt-Path problem, with input tuple $\eta = (\mathcal{U}; S_1, S_2, \ldots, S_T; G; s; t)$, where $n = |\mathcal{U}|$ and $T = \lceil \log n \rceil$, where $\alpha$ is the approximation factor we aim to achieve.

We first provide an overview of our main ideas. A natural approach to solve LogBgt-Path is to employ dynamic programming, in which the states keep track of the number of edges used from each group. However, since we have $T = \lceil \log n \rceil$ groups, the number of states may be as large as $n^{O(T)}$. To resolve this issue, we perform an *approximation dynamic programming*, in which we only approximate the number of edges in each group. In particular, the numbers are rounded to the nearest power of $p$ after each step, for carefully chosen value $p > 1$ (to ensure that we do not lose too much in the rounding). This rounding technique is inspired by a classic approximation algorithm for the subset sum problem [30] Now, the dynamic programming state is a vector that approximates the number of edges used from each group in a path from $x$ to $y$ with at most $2^i$ edges, where $x, y \in V$ and $0 \leq i \leq \lceil \log n \rceil$. The dynamic programming process involves storing the number of selected items in each group and rounding them to the nearest power of $p$ at each iteration. However, this method results in a state space of size $(\log_p n)^T = n^{\Omega(\log\log n)}$, which is better than $n^{O(T)}$, but still super-polynomial. To resolve the above issue, we need the second idea, which is to trade off the approximation factor and the running time. In particular, we introduce an integer parameter $\beta$, defined as $\beta = \frac{\alpha-1}{4(1+\delta)}$ for some $\delta \in \left[\frac{1}{2}, 2\right]$. We then partition the original $T$ groups into $T/\beta$ *supergroups*, each containing $\beta$ groups. This reduces the number of states to $(\log_p n)^{O(T/\beta)}$, but incurs a loss of $O(\beta)$ in the approximation factor.

Now, we present the details of our algorithm. Let $K = \lceil T/\beta \rceil$, and $B_i = \min(T, i \cdot \beta)$ for all $0 \le i \le K$. For $1 \le i \le K$ and $D \subseteq \mathcal{U}$, define $C_i(D) = \sum_{j=B_{i-1}+1}^{B_i} \frac{1}{2^j} |D \cap S_j|$ (specifically, $C_i(\emptyset) = 0$). Furthermore, we define the vector $C(D) = (C_1(D), \dots C_K(D)) \in \mathbb{R}^K$. It is important to notice that:

- If $D \subseteq \mathcal{U}$ is a $c$-valid solution ($c > 0$), then $C_i(D) \le c\beta$ for all $1 \le i \le K$.
- If $C_i(D) \le c\beta$ for all $1 \le i \le K$, then $D \subseteq \mathcal{U}$ is a $c\beta$-valid solution.

In iteration $i$ ($1 \le i \le \lceil \log n \rceil$), for each pair of vertices $x, y \in V$, we define $Q_{i,x,y}$ as a set of vectors in $\mathbb{R}^K$ that encodes information about paths from $x$ to $y$ containing at most $2^i$ edges. Specifically, for any path $D \subseteq \mathcal{U}$ from $x$ to $y$ with at most $2^i$ edges, the set $Q_{i,x,y}$ includes a corresponding vector that approximates $C(D)$.

- Initially, $Q_{0,x,x} = \{C(\emptyset)\}$. For $Q_{0,x,y}$, it is set to $\{C(\{(x, y)\})\}$ if $(x, y)$ is an edge, and $\emptyset$ if $(x, y)$ is not an edge.
- In the $i$-th iteration ($1 \le i \le \lceil \log n \rceil$), we begin by initializing $Q_{i,x,y} = \emptyset$ for all $x, y$. Then, for each pair $x, y$, we enumerate all vertices $z$ and add the sum of $Q_{i-1,x,z}$ and $Q_{i-1,z,y}$ to $Q_{i,x,y}$. Here, the sum of two sets is defined as the set of all pairwise sums of elements from the two sets.
- To reduce the size of $Q_{i,x,y}$ , we round the components of these vectors in $Q_{i,x,y}$ to 0 or powers of $p = 1 + \frac{\delta/2}{\log n}$ (recall that $\delta \in [1/2, 2]$).

Our main result in this section is the following theorem:

▶ **Theorem 22.** *For any $9 \le \alpha \le \log \log n$, there exists a $n^{O(\log \log n/\alpha)}$-time algorithm for $\frac{\alpha-1}{4}$-LogBgt-Path. Thus we have an approximation approximation which runs in $n^{O(\log \log n/\alpha)}$-time and achieves an approximation factor of $\alpha$ for MinNorm-Path.*

The details of the proof are provided in the full version [17].

## 9 A Bi-criterion Approximation for Matching

In this section, we consider the matching problem. While we do not know how to design a constant factor approximation algorithm for the MinNorm version of perfect matching problem yet, we demonstrate that it is possible to find a nearly perfect matching within a constant approximation factor – a bi-criterion approximation algorithm. In particular, for any given norm, we can find a matching that matches $1 - \epsilon$ fraction of nodes and its corresponding norm is at most $c$ (a constant) times the norm of the optimal integral perfect matching. Note that a constant factor approximation algorithm (without relaxing the perfect matching requirement) is impossible using the natural linear program, due to Theorem 21.

First, we introduce some notations. We are given a bipartite graph $G = (L, R, E)$, where $L$ is the set of nodes of the first color class, $R$ is the set of node of the other color class, and $E$ is the set of edges. Let $m = |L| = |R|$. We define $\mathcal{U} = E, n = |\mathcal{U}|$. We also study the LogBgt version and we let $S_1, S_2 \cdots , S_T$ be the disjoint subsets of $\mathcal{U}$.

▶ **Definition 23** ($\epsilon$-nearly Matching)**.** *Let $0 < \epsilon < 1$. We define the following problem as $\epsilon$-nearly matching. Given a bipartite graph $G = (L, R, E)$, a set $S \subseteq E$ is a nearly matching if it is a matching with at least $(1 - \epsilon)m$ edges.*

Again, we use the natural linear program for the LogBgt version of the problem. Through an iterative rounding method (similar to that in [13]), we can obtain the following bi-criterion approximation result.

▶ **Theorem 24.** *If there exists a 1-valid solution for a LogBgt-PerMat problem, then there exists an $O(n^{\frac{18}{\delta\epsilon}})$-time algorithm to obtain a $(2+\delta)$-valid solution for the corresponding LogBgt $\epsilon$-nearly matching problem for any $\delta, \epsilon < 1$.*

This theorem implies the following result for MinNorm-PerMat.

▶ **Theorem 25.** *Given a MinNorm-PerMat problem with a monotone symmetric norm $f(\cdot)$ and a value vector $\boldsymbol{v}$, let $D^*$ be an optimal solution for this problem. For any constants $\epsilon, \delta < 1$, there exists a polynomial-time algorithm to obtain a $\epsilon$-nearly matching $D$ such that $\frac{f(\boldsymbol{v}[D])}{f(\boldsymbol{v}[D^*])} \le (8+\delta)$.*

The details and variants of the problem are omitted and can be found in the full version [17].

## 10 Concluding Remarks

In this paper, we propose a general formulation for general norm minimization in combinatorial optimization. Our formulation captures a broad class of combinatorial structures, encompassing various fundamental problems in discrete optimization. Via a reduction of the norm minimization problem to a multi-criteria optimization problem with logarithmic budget constraints, we develop constant-factor approximation algorithms for multiple important covering problems, such as interval cover, multi-dimensional knapsack cover, and set cover (with logarithmic approximation factors). We also provide a bi-criteria approximation algorithm for min-norm perfect matching, and an $O(\log\log n)$-approximation algorithm for the min-norm $s$-$t$ path problem, via a nontrivial approximate dynamic programming approach.

Our results open several intriguing directions for future research. First, one can explore other combinatorial optimization problems, such as Steiner trees and other network design problems, within our general framework. Additionally, our formulation could be extended to encompass the min-norm load balancing problem studied in [13] (where job processing times are first summed into machine loads before applying a norm), and even the generalized load balancing [19] and cascaded norm clustering problems [18, 1] (which allow for two levels of cost aggregation via norms). Second, obtaining a nontrivial true approximation algorithm for perfect matching – rather than a bi-criterion approximation – remains an important open problem. Third, it is an important open problem whether a polynomial-time constant-factor approximation exists for the min-norm $s$-$t$ path problem. Lastly, it would be interesting to study other general objective functions beyond symmetric monotone norms and submodular functions (such as general subadditive functions [28] and those studied in [36]).

## References

1 Fateme Abbasi, Sandip Banerjee, Jarosław Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1377–1399. IEEE, 2023. doi:10.1109/FOCS57990.2023.00085.

2 Ali Aouad and Danny Segev. The ordered $k$-median problem: surrogate models and approximation algorithms. *Math. Program.*, 177(1-2):55–83, 2019. doi:10.1007/S10107-018-1259-3.

3 Nikhil Ayyadevara, Nikhil Bansal, and Milind Prabhu. On minimizing generalized makespan on unrelated machines. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 21, 2023.

4 Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 331–337, 2005. doi:10.1145/1060590.1060639.

5 Jatin Batra, Syamantak Das, and Agastya Vibhuti Jha. Tight approximation algorithms for ordered covering. In *Algorithms and Data Structures Symposium*, pages 120–135. Springer, 2023. doi:10.1007/978-3-031-38906-1_9.

6 André Berger, Vincenzo Bonifaci, Fabrizio Grandoni, and Guido Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128:355–372, 2011. doi:10.1007/S10107-009-0307-4.

7 Vladimir Braverman, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for ordered weighted clustering. In *International Conference on Machine Learning*, pages 744–753. PMLR, 2019. URL: http://proceedings.mlr.press/v97/braverman19a.html.

8 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. SIAM, 2014. doi:10.1137/1.9781611973402.106.

9 Jaroslaw Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered $k$-median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 620–631, 2018. doi:10.1145/3188745.3188930.

10 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.

11 Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Random pseudo-polynomial algorithms for exact matroid problems. *Journal of Algorithms*, 13(2):258–273, 1992. doi:10.1016/0196-6774(92)90018-8.

12 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between $k$-median and $k$-center: Approximation algorithms for ordered $k$-median. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs*, pages 29:1–29:14, 2018. doi:10.4230/LIPICS.ICALP.2018.29.

13 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 126–137, 2019. doi:10.1145/3313276.3316322.

14 Deeparnab Chakrabarty and Chaitanya Swamy. Simpler and better algorithms for minimum-norm load balancing. In *27th Annual European Symposium on Algorithms*, volume 144 of *LIPIcs*, pages 27:1–27:12, 2019. doi:10.4230/LIPICS.ESA.2019.27.

15 Chandra Chekuri and Alina Ene. Submodular cost allocation problem and applications. In *Automata, Languages and Programming - 38th International Colloquium*, volume 6755 of *Lecture Notes in Computer Science*, pages 354–366, 2011. doi:10.1007/978-3-642-22006-7_30.

16 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1080–1097. SIAM, 2011. doi:10.1137/1.9781611973082.82.

17 Kuowen Chen, Jian Li, Yuval Rabani, and Yiran Zhang. New results on a general class of minimum norm optimization problems, 2025. arXiv:2504.13489.

18 Eden Chlamtáč, Yury Makarychev, and Ali Vakilian. Approximating fair clustering with cascaded norm objectives. In *Proceedings of the 2022 annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 2664–2683. SIAM, 2022. doi:10.1137/1.9781611977073.104.

19 Shichuan Deng, Jian Li, and Yuval Rabani. Generalized unrelated machine scheduling problem. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2898–2916. SIAM, 2023. doi:10.1137/1.9781611977554.CH110.

20 Shichuan Deng and Qianfan Zhang. Ordered k-median with outliers. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 34:1–34:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.APPROX/RANDOM.2022.34.

21 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing*, pages 624–633, 2014. doi:10.1145/2591796.2591884.

22 Nicolas El Maalouly. Exact matching: Algorithms and related problems. In *40th International Symposium on Theoretical Aspects of Computer Science*, 2023.

23 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.

24 Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-norms and all-l_p-norms approximation algorithms. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (2008)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2008.

25 Fabrizio Grandoni, R. Ravi, Mohit Singh, and Rico Zenklusen. New approaches to multi-objective optimization. *Math. Program.*, 146(1-2):525–554, 2014. doi:10.1007/S10107-013-0703-7.

26 Fabrizio Grandoni, Ramamoorthi Ravi, and Mohit Singh. Iterative rounding for multi-objective optimization problems. In *European Symposium on Algorithms*, pages 95–106. Springer, 2009.

27 Fabrizio Grandoni and Rico Zenklusen. Approximation schemes for multi-budgeted independence systems. In *European Symposium on Algorithms*, pages 536–548. Springer, 2010. doi:10.1007/978-3-642-15775-2_46.

28 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: submodular and xos functions. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1688–1702, USA, 2017. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611974782.111.

29 G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities.* Cambridge University Press, 1934.

30 Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, October 1975. doi:10.1145/321906.321909.

31 Sharat Ibrahimpur and Chaitanya Swamy. Approximation algorithms for stochastic minimum-norm combinatorial optimization. In *61st IEEE Annual Symposium on Foundations of Computer Science*, pages 966–977, 2020. doi:10.1109/FOCS46700.2020.00094.

32 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *48th International Colloquium on Automata, Languages, and Programming*, volume 198 of *LIPIcs*, pages 81:1–81:20, 2021. doi:10.4230/LIPICS.ICALP.2021.81.

33 Thomas Kesselheim, Marco Molinaro, and Sahil Singla. Supermodular approximation of norms and applications. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1841–1852, 2024. doi:10.1145/3618260.3649734.

34 Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3(71-104):3, 2014.

35 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010. doi:10.1287/MOOR.1100.0463.

36 Jian Li and Samir Khuller. Generalized machine activation problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 80–94. SIAM, 2011. doi:10.1137/1.9781611973082.7.

37 André Linhares, Neil Olver, Chaitanya Swamy, and Rico Zenklusen. Approximate multi-matroid intersection via iterative refinement. *Mathematical Programming*, 183:397–418, 2020. doi:10.1007/S10107-020-01524-Y.

38 Kalen Patton, Matteo Russo, and Sahil Singla. Submodular norms with applications to online facility location and stochastic probing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.

39 Ram Ravi and Michel X Goemans. The constrained minimum spanning tree problem. In *Algorithm Theory—SWAT'96: 5th Scandinavian Workshop on Algorithm Theory Reykjavík, Iceland, July 3–5, 1996 Proceedings 5*, pages 66–75. Springer, 1996.