Approximation Algorithms for Minimum Norm and Ordered Optimization Problems*

Deeparnab Chakrabarty†
deeparnab@dartmouth.edu
Dartmouth
Hanover, New Hampshire, USA

Chaitanya Swamy‡
cswamy@uwaterloo.edu
University of Waterloo
Waterloo, Ontario, Canada

ABSTRACT

In many optimization problems, a feasible solution induces a multi-dimensional cost vector. For example, in load-balancing a schedule induces a load vector across the machines. In k-clustering, opening k facilities induces an assignment cost vector across the clients. Typically, one seeks a solution which either minimizes the sum- or the max- of this vector, and these problems (makespan minimization, k-median, and k-center) are classic NP-hard problems which have been extensively studied.

In this paper we consider the *minimum-norm* optimization problem. Given an arbitrary monotone, symmetric norm, the problem asks to find a solution which minimizes the norm of the induced cost-vector. Such norms are versatile and include ℓ_p norms, Top-ℓ norm (sum of the ℓ largest coordinates in absolute value), and ordered norms (non-negative linear combination of Top-ℓ norms), and consequently, the minimum-norm problem models a wide variety of problems under one umbrella, We give a general framework to tackle the minimum-norm problem, and illustrate its efficacy in the unrelated machine load balancing and k-clustering setting. Our concrete results are the following.

(a) We give constant factor approximation algorithms for the minimum norm load balancing problem in *unrelated* machines, and the minimum norm k-clustering problem. To our knowledge, our results constitute the *first* constant-factor approximations for such a general suite of objectives.

(b) For load balancing on unrelated machines, we give a $(2+\varepsilon)$-approximation for ordered load balancing (i.e., min-norm load-balancing under an ordered norm).

(c) For k-clustering, we give a $(5+\varepsilon)$-approximation for the ordered k-median problem, which significantly improves upon the previous-best constant-factor approximation (Chakrabarty and Swamy (ICALP 2018); Byrka, Sornat, and Spoerhase (STOC 2018)).

(d) Our techniques also imply $O(1)$ approximations to the instance-wise best *simultaneous approximation factor* for unrelated-machine load-balancing and k-clustering. To our knowledge, these are the first *positive* simultaneous approximation results in these settings.

At a technical level, one of our chief insights is that minimum-norm optimization can be reduced to a special case that we call *min-max ordered optimization*. Both the reduction, and the task of devising algorithms for the latter problem, require a sparsification idea that we develop, which is of interest for ordered optimization problems. The main ingredient in solving min-max ordered optimization is a *deterministic, oblivious rounding procedure* (that we devise) for suitable LP relaxations of the load-balancing and k-clustering problem; this may be of independent interest.

CCS CONCEPTS

• **Theory of computation → Approximation algorithms analysis**; **Scheduling algorithms**; **Facility location and clustering**.

KEYWORDS

approximation algorithm, minimum norm optimization, ordered optimization, clustering, load balancing

ACM Reference Format:
Deeparnab Chakrabarty and Chaitanya Swamy. 2019. Approximation Algorithms for Minimum Norm and Ordered Optimization Problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316322

1 INTRODUCTION

In many optimization problems, a feasible solution induces a multi-dimensional cost vector. For example, in the load balancing setting with machines and jobs, a solution is an assignment of jobs to machines, and this induces a *load* on every machine. In a clustering setting with facilities and clients, a solution is to open k facilities and connecting clients to the nearest open facilities, which induces an *assignment cost* on every client. This multi-dimensional vector dictates the quality of the solution. Depending on the application, oftentimes one minimizes either the sum of the entries of the cost vector, or the largest entry of the cost vector. For example, in the load balancing setting, the largest entry of the load vector is the *makespan* of the assignment, and minimizing makespan has been extensively studied [25, 31, 34, 35]. Similarly, in the clustering setting, the problem of minimizing the sum of assignment costs is the k-median problem, and the problem of minimizing the largest assignment cost is the k-center problem. Both of these are classic combinatorial optimization problems [12, 16, 20, 23, 24, 32]. However, the techniques to study the sum-versions and max-versions

*A full version with all omitted details is available on the CS arXiv.
†Supported by NSF CCF-1813165
‡Supported in part by NSERC grant 327620-09 and an NSERC DAS.

STOC '19, June 23–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6705-9/19/06…$15.00
https://doi.org/10.1145/3313276.3316322

are often different, and it is a natural and important to investigate what the complexity of these problems become if one is interested in a different statistic of the cost vector.

In this paper, we study a far-reaching generalization of the above two objectives. We study the *minimum norm optimization* problem, where given an arbitrary monotone, symmetric norm f, one needs to find a solution which minimizes the norm f evaluated on the induced cost vector. In particular, we study (a) the minimum norm load balancing problem which asks to find the assignment of jobs to (unrelated) machines which minimizes $f(\overrightarrow{\text{load}})$ where $\overrightarrow{\text{load}}$ is the induced load vector on the machines, and (b) the minimum norm k-clustering problem which asks to open k-facilities minimizing $f(\vec{c})$ where $\vec{c}$ is the induced assignment costs on the clients.

Our main contribution is a framework to study minimum norm optimization problems. Using this, we give constant factor approximation algorithms for the minimum norm unrelated machine load balancing and the minimum norm k-clustering problem (Theorem 9.1 and Theorem 8.1). To our knowledge our results constitute the *first* constant-factor approximations for a general suite of objectives in these settings. We remark that the above result is contingent on how f is given. We need a ball-optimization oracle (see (1) for more details), and for most norms it suffices to have access to a *first-order* oracle which returns the (sub)-gradient of f at any point.

Monotone, symmetric norms capture a versatile collection of objective functions. We list a few relevant examples below and refer the reader to [4, 10, 11] for a more comprehensive list.

- **ℓ_p-norms.** Perhaps the most famous examples are ℓ_p norms where $f(\vec{v}) := \left(\sum_{i=1}^{n} \vec{v}_i^{\,p}\right)^{1/p}$ for $p \geq 1$. Of special interest are $p = \{1, 2, \infty\}$. For unrelated machines load-balancing, the $p = 1$ case is trivial while the $p = \infty$ case is makespan minimization. This has a 2-approximation [31, 34] which has been notoriously difficult to beat. For the general ℓ_p norms, Azar and Epstein [7] give a 2-approximation, with improvements given by [28, 33]. For the k-clustering setting, the $p = \{1, 2, \infty\}$ norms have been extensively studied over the years [1, 12, 16, 20, 23, 24]. One can also derive an $O(1)$-approximation for general ℓ_p-norms using most of the algorithms[1] for the k-median problem.
- Top-ℓ **norms and ordered norms.** Another important class of monotone, symmetric norms is the Top-ℓ-*norm*, which given a vector $\vec{v}$ returns the sum of the largest ℓ elements. These norms are another way to interpolate between the ℓ_1 and the ℓ_∞ norm.

  A generalization of the Top-ℓ norm optimization is what we call the *ordered norms*. The norm is defined by a non-increasing, non-negative vector $w \in \mathbb{R}_+^n$ with $w_1 \geq w_2 \geq \cdots \geq w_n \geq 0$. Given these weights, the w-ordered, or simply, ordered norm of a vector $\vec{v} \in \mathbb{R}_+^n$ is defined as $\text{cost}(w; \vec{v}) := \sum_{i=1}^{n} w_i \vec{v}_i^{\downarrow}$ where $\vec{v}^{\downarrow}$ is the entries of $\vec{v}$ written in non-increasing order itself. It is not hard to see that the ordered norm is a non-negative linear combination of the Top-ℓ norms.

  For load balancing in unrelated machines, we are not aware of any previous works studying these norms. *We devise a $(2+\varepsilon)$-approximation for ordered load balancing* (Theorem 9.2). Note that the case of $\ell = 1$ for Top-ℓ-load balancing corresponds to makespan minimization for which improving upon the factor of 2 is a longstanding open problem.

  In k-clustering, the Top-ℓ optimization problem is called the ℓ-centrum problem, and the ordered-norm minimization problem is called the ordered k-median problem. Only recently, a 38-factor [13] and $18+\varepsilon$-factor [15] approximation algorithm was given for the ordered k-median problem. *We give a much improved $(5+\varepsilon)$-factor approximation algorithm for the ordered k-median problem* (Theorem 8.3).
- **Min-max ordered norm.** Of particular interest to us is what we call the *min-max ordered optimization* problem. In this, we are given N non-increasing, non-negative weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}_+^n$, and the goal is to find a solution $\vec{v}$ which minimizes $\max_{r=1}^{N} \text{cost}(w^{(r)}; \vec{v})$. This is a monotone, symmetric norm since it is a maximum over a finite collection of monotone, symmetric norms.

  One of the main insights of this paper is that the minimum norm problem reduces to min-max ordered optimization (Theorem 5.4). In particular, we show that the value of any monotone, symmetric norm can be written as the maximum of a collection of (possibly infinite) ordered norms; this result may be of independent interest in other applications involving such norms [4, 11].
- **Operations.** One can construct monotone, symmetric norms using various operations such as (a) taking a nonnegative linear combination of monotone, symmetric norms; (b) taking the maximum over any finite collection of monotone, symmetric norms; (c) given a (not-necessarily symmetric) norm $g : \mathbb{R}^n \to \mathbb{R}_+$, setting $f(v) := g(v^{\downarrow})$, or $f(v) := \mathbf{Exp}_{\pi}[g(\{v_{\pi(i)}\}_{i\in[n]})]$ where π is a random permutation of $[n]$; (d) given a monotone, symmetric norm $g : \mathbb{R}^k \to \mathbb{R}_+$, where $k \leq n$, setting $f(v) = \sum_{S\subseteq[n]:|S|=k} g(\{v_i\}_{i\in S})$. The richness of these norms makes the minimum-norm optimization problem a versatile and appealing model which captures a variety of optimization problems under one umbrella.

  As an illustration, consider the following stochastic optimization problem in clustering (this is partly motivated by the stochastic fanout model described in [26] for a different setting). We are given a universe of plausible clients, and a symmetric probability distribution over actual client instances. Concretely, say, each client materializes i.i.d with probability $p \in (0, 1)$. The problem is to open a set of k facilities such that the *expected maximum* distance of an instantiated client to an open facility is minimized. The expectation is indeed a norm (follows from part (d) above) and thus we can get a constant-factor approximation for this problem. In fact, the *expected maximum* for the i.i.d case is an ordered-norm, and so we can obtain a $(5+\varepsilon)$-approximation for this particular stochastic optimization problem.
- **General Convex Functions.** One could ask to find a solution minimizing a general convex function of the cost vector. In general, such functions can be arbitrarily sharp and this precludes

[1] We could not find an explicit reference for this. The only work which we found that explicitly studies the ℓ_p-norm minimization in the k-clustering setting is by Gupta and Tangwongsan [21]. They give a $O(p)$-approximation using local-search and prove that local-search can't do any better. However, ℓ_p^p-"distances" satisfy relaxed triangle inequality, in that, $d(u, v) \leq 2^p(d(u, w) + d(w, v))$. The algorithms of Charikar et al [16] and Jain-Vazirani [24] need triangle inequality with only "bounded hops" and thus give C^p-approximations for the ℓ_p^p "distances". In turn this implies a constant factor approximation for the ℓ_p-norm.

any non-trivial approximation. For instance in the clustering setting, consider the convex function $C(\vec{c})$ which takes the value 0 if the sum of $\vec{c}_j$'s (that is the k-median objective) is less than some threshold, and ∞ otherwise; for this function, it is NP-hard to get a finite solution. Motivated thus, Goel and Meyerson [18] call a solution $\vec{v}$ an α-approximate solution if $C(\vec{v}/\alpha) \leq opt$ where $\vec{v}$ is the induced cost vector, $\vec{v}/\alpha$ is the coordinate-wise scaled vector, and $opt = \min_{\vec{w}} C(\vec{w})$. It is not hard to see[2] that a constant factor approximation for monotone, symmetric norm-minimization implies a constant-approximate solution for any monotone, symmetric convex function. In particular, for the load-balancing and clustering setting we achieve this.

Connections and implications for simultaneous optimization. In the minimum-norm optimization problem, we are given a fixed norm f and we seek a solution whose cost vector $\vec{v}$ minimizes $f(\vec{v})$. In *simultaneous optimization* [18, 27], the goal is to find a solution whose cost vector $\vec{v}$ *simultaneously* approximates all norms/convex functions. Such solutions are desirable as they possess certain fairness properties. More precisely, we seek a solution inducing a cost vector $\vec{v}$ that is simultaneously an α-approximation for *all* monotone symmetric norms $g : \mathbb{R}^n \to \mathbb{R}_+$, i.e., $g(\vec{v}) \leq \alpha \cdot opt(g)$, where $opt(g) := \min_{\vec{w}} g(\vec{w})$.

Simultaneous optimization is clearly a much stronger goal than what we are aiming for: if one can find a solution which is a simultaneous α-approximation, then this solution is clearly an α-approximation for a fixed norm. It is rather remarkable that in the setting of load balancing with *identical jobs*, and even in the *restricted assignment* setting where jobs have fixed processing times but can be allocated only on a subset of machines, one can always achieve a simultaneous 2-approximate solution [3, 8, 18]. Unfortunately, for unrelated (even related) machines [8] and k-clustering [27], there are impossibility results ruling out the *existence* of any simultaneous α-approximate solutions for constant α. These impossibilities also show that the techniques used in [3, 8, 18] are not particularly helpful when trying to optimize a *given, fixed* norm, which is the main focus in our paper.

The techniques we develop yield *an $O(1)$-approximation to the best simultaneous approximation factor possible for any instance of load-balancing on unrelated machines, and k-clustering*. Fix an unrelated-machine load-balancing instance $\mathcal{I}$. Let $\alpha^*_{\mathcal{I}}$ be the smallest α for which there is a solution to $\mathcal{I}$ that is a simultaneous α-approximation. Note that $\alpha^*_{\mathcal{I}}$ could be a constant for a specific instance $\mathcal{I}$; the impossibility result mentioned above states that $\alpha^*_{\mathcal{I}}$ cannot be a constant for *all* instances. It is natural, and pertinent, to ask whether one can obtain *instance-wise* guarantees: for example, can one obtain an $O(\alpha^*_{\mathcal{I}})$ simultaneous-approximation factor for every instance $\mathcal{I}$? Note that in settings where we are constrained to specify a single solution that is required to "work" for a multitude of norms, $\alpha^*_{\mathcal{I}}$ is a more meaningful benchmark to compare against (than $opt(g)$), as this explicitly captures the one-solution limitation, and so such instance-wise guarantees are particularly desirable. We design algorithms that yield $O(1)$-approximations to $\alpha^*_{\mathcal{I}}$ both for load balancing and k-clustering. These seem to be the first *positive* results on simultaneous optimization in these settings. We remark that our algorithm is not a generic reduction to the minimum norm optimization, but is consequence of our techniques developed to tackle the problem.

Other related work. The ordered k-median and the ℓ-centrum problem have been extensively studied in the Operations Research literature for more than two decades (see, e.g. the book [29]); we point the interested reader to these books, or the paper by Aouad and Segev [5], and references within for more information on this perspective. From an approximation algorithms point of view, Tamir [36] gives the first $O(\log n)$-approximation for the ℓ-centrum problem, and Aouad and Segev [5] give the first $O(\log n)$ approximations for the ordered k-median problem. Very recently, Byrka, Sornat, and Spoerhase [13] and our earlier paper [15] give the first constant-factor approximations for the ℓ-centrum and ordered k-median problems. Another recent relevant work is of Alamdari and Shmoys [2] who consider the *k-centridian* problem where the objective is a weighted average of the k-center and the k-median objective (a special case of the ordered k-median problem); [2] give a constant-factor approximation algorithm for this problem.

In the load balancing setting, research has mostly focused on ℓ_p norms; we are not aware of any work studying the Top-ℓ optimization question in load balancing. For the ℓ_p-norm Awerbuch et al. [6] give a $\Theta(p)$-approximation for unrelated machines; their algorithm is in fact an *online* algorithm. Alon et al. [3] give a PTAS for the case of identical machines. This paper [3] also shows a polynomial time algorithm in the case of restricted assignment (jobs have fixed processing times but can't be assigned everywhere) with unit jobs which is optimal *simultaneously* in all ℓ_p-norms. Azar et al. [8] extend this result to get a 2-approximation algorithm *simultaneously* in all ℓ_p norms in the restricted assignment case. This is generalized to a simultaneous 2-approximation in all symmetric norms (again in the restricted assignment situation) by Goel and Meyerson [18]. As mentioned in the previous subsection, Azar et al. [8] also note that even in the related machine setting, no constant factor approximation is possible simultaneously even with the ℓ_1 and ℓ_∞ norm. For unrelated machines, for any fixed ℓ_p norm Azar and Epstein [7] give a 2-approximation via convex programming. The same paper also gave a $\sqrt{2}$-approximation for the $p = 2$ case. These factors have been improved (in fact for any constant p the approximation factor is < 2) by Kumar et al. [28] and Makarychev and Sviridenko [33]. We should mention that the techniques in these papers are quite different from ours and in particular these strongly use the fact that the ℓ_p^p cost is separable. Finally, in the clustering setting, Kumar and Kleinberg [27] and Golovin et al. [19] give simultaneous constant factor approximations in all ℓ_p norms, but their results are *bicriteria results* in that they open $O(k \log n)$ and $O(k\sqrt{\log n})$ facilities respectively.

2 TECHNICAL OVERVIEW, ORGANIZATION

First approach and its failure. Perhaps the first thing one may try for the minimum-norm optimization problem is to write a *convex program* min $f(\vec{v})$ where $\vec{v}$ ranges over *fractional* cost vectors, ideally, convex combinations of integral cost vectors. If there were a *deterministic* rounding algorithm which given an optimal solution

[2] Consider the monotone, symmetric norm $f(x) := \min\{t : C(|x|/t) \leq opt\}$. By definition $f(\vec{o}) = 1$, and so a α-approximate min-norm solution $\vec{v}$ satisfies $f(\vec{v}) \leq \alpha$, implying $C(\vec{v}/\alpha) \leq opt$. The definition requires knowing the value of opt which can be guessed using binary search.

$\vec{v}^*$ could return a solution $\vec{v}$ such that for every coordinate $\vec{v}_j \le \rho\vec{v}^*_j$, then by homogeneity of f, we would get a ρ-approximation. Indeed, for some optimization problems such a rounding is possible. Unfortunately, for both unrelated load balancing and k-clustering, this strategy is a failure as there are simple instances for both problems, where even when $\vec{v}^*$ is a convex combination of integer optimum solutions, no such rounding, with constant ρ, exists. In particular, the *integrality gaps* of these convex programs are unbounded.

Reduction to min-max ordered optimization (Section 5). Given the above failure, at first glance, it may seem hard to be able to reason about a general norm. One of the main insights of this paper is that the monotone, symmetric norm minimization problem reduces to min-max ordered optimization. This is a key conceptual step since it allows us a foothold in arguing about the rather general problem. Our result may also be of interest in other settings dealing with symmetric norms. In particular, we show that given any monotone, symmetric norm f, the function value at any point $f(x)$ is equal to $\max_{w\in C} \text{cost}(w;x)$ (Lemma 5.2) where C is a potentially infinite family of non-increasing subgradients on the unit-norm ball. That is, $f(x)$ equals the maximum over a collection of ordered norms. Thus, finding the x minimizing $f(x)$ boils to the min-max ordered-optimization problem. The snag is that this collection of weight vectors could be infinite. This is where the next simple, but extremely crucial, technical observation helps us.

Sparsification idea (Section 4). Given a non-increasing, non-negative weight vector $w \in \mathbb{R}^n_+$, the ordered norm of a vector $\vec{v} \in \mathbb{R}^n_+$ is $\text{cost}(w;\vec{v}) := \sum_{i=1}^n w_i \vec{v}_i^{\downarrow}$. The main insight is that although w may have all its n-coordinates distinct, only a few *fixed* coordinates matter. More precisely, if we focus only on the coordinates $\text{POS} := \{1, 2, 4, 8, \cdots, \}$ and define a $\widetilde{w}$-vector with $\widetilde{w}_i = w_i$ if $i \in \text{POS}$, and $\widetilde{w}_i = w_\ell$ where ℓ is the nearest power of 2 larger than i, then it is not too hard to see $\text{cost}(\widetilde{w};x) \le \text{cost}(w;x) \le 2\text{cost}(\widetilde{w};x)$. Indeed, one can increase the granularity of the coordinates to (ceilings of) powers of $(1+\delta)$ to get arbitrarily close approximations where the number of relevant coordinates is $O(\log n/\delta)$.

The above sparsification shows that for ordered norms, one can focus on weight vectors whose breakpoints are in *fixed* positions that are *independent* of the weight vector. In contrast, the natural sparsification idea that rounds each w_i to the nearest power of $(1+\delta)$ does not yield this *weight-independence* property in the positions of breakpoints. The uniformity of positions (and the fact that we only have logarithmically many positions) allows us to form a polynomial-size ε-net of weight vectors. More precisely, for any weight vector w, there is another weight vector w' in this net such that for any vector $\vec{v}$, $\text{cost}(w;\vec{v})$ and $\text{cost}(w';\vec{v})$ are within a multiplicative $(1\pm\delta)$-factor. In particular, this helps us bypass the problem of having "infinitely many vectors" in the collection C described above.

Ordered optimization and proxy costs (Section 6). Now we focus on min-max ordered optimization. First let us consider just simple ordered optimization, and in particular, just Top-ℓ optimization. To illustrate the issues, let us fix the optimization problem to be load balancing on unrelated machines. One of the main technical issues in tackling the Top-ℓ optimization problem is that one needs to find an assignment such that sum of loads on a set of ℓ machines is minimized, but this set of machines itself depends on the assignment. Intuitively, the problem would be easier (indeed, trivial) if we could sum the loads over all machines. Or perhaps sum some *function* of the loads, but over *all* machines. Then perhaps one could write a linear/convex program to solve this problem fractionally, and the objective function would be clear. This is where the idea of *proxy costs* comes handy. We mention that this idea was already present (in different forms) in [5, 13, 15].

The idea of this proxy cost is also simple. Suppose we knew the ℓ-th largest load, say ρ, in the optimal solution. Then the Top-ℓ norm of the load vector can be written as $\ell\cdot\rho + \sum_{\text{all machines } i}(\text{load}(i)-\rho)^+$, where $z^+ := \max\{z, 0\}$. This is the *proxy-cost* of the Top-ℓ norm given parameter ρ. Note that the summation is over *all* machines; however, the summand is not the load of the machine but a function $h_\rho(\text{load}(i))$ of the load. Furthermore, we could assume by binary search that we have a good guess of ρ.

For ordered optimization, first we observe that $\text{cost}(w;\vec{v})$ can be written as a non-negative linear combination of the Top-ℓ norms (Claim 6.4). In particular, if we have the guesses of the ℓ-th largest load for all ℓ, then we could write the proxy cost of $\text{cost}(w;\vec{v})$. However, guessing all n ρ_ℓ's would be infeasible. This is where our earlier sparsification idea comes in handy again. Since the w_ℓs for only indices $\ell \in \text{POS}$ are used to define $\widetilde{w}$, one only needs to guess the (approximations for) ρ_ℓs for the indices $\ell \in \text{POS}$ to define the proxy function. And this again can be done in polynomial time. Also, what is *crucial for min-max ordered optimization* is that the positions are *independent* of the particular weight vector: although there are N different weight functions, their sparsified versions have the *same* break points, and their proxy functions are defined using these same, logarithmically many break points.

LP relaxations and deterministic oblivious rounding (Sec.7-9) One can use the proxy costs to write linear programming relaxations for the problems at hand (in our case, load balancing and k-clustering). Indeed, for k-clustering, this was the approach taken by Byrka et al. [13] and our earlier work [15] for ordered k-median. With proxy costs, the LP relaxation for ordered k-median is the usual LP but the objective has *non-metric* costs. Nevertheless, both the papers showed constant integrality gaps for these LPs. (Our proxy cost here is subtly different, but is within $O(1)$-factor of the expression in [13, 15].) For load-balancing, the natural LP has a bad gap, and one needs to add additional constraints, using which we can indeed show the LP has an integrality gap of roughly 2.

However, it is not at all clear how to use this LP for *min-max ordered problems* with multiple weight functions. The algorithms of Byrka et al [13] are randomized which bound the *expected* cost of the ordered k-median; with multiple weights, this won't help solve the min-max problem unless one can argue very sharp concentration properties of the algorithm. The same is true for our load-balancing algorithm. These algorithms can be derandomized, but these derandomizations lead to algorithms which use the (single) weight function crucially, and it is not clear at all how to minimize the max of even two weight functions. The primal-dual algorithm in [15] suffers from the same problem. Our approach in this paper is to consider *deterministic* rounding of the LP solution which are *oblivious to the weight* vectors. We can achieve this for the LP relaxations we write for load balancing and k-clustering

(although we need to strengthen the latter furthermore). We defer further technical overview to Section 7, and then give details for load-balancing in Section 9 and for k-clustering in Section 8; the latter two sections can be read in any order.

Extensions: multi-budgeted ordered optimization and connections to simultaneous optimization. Finally, we showcase the versatility of deterministic, weight-oblivious rounding by utilizing it to obtain $O(1)$-approximations for multi-budgeted ordered optimization, and the instance-wise best simultaneous-approximation factor, for our applications of load balancing and k-clustering.

Multi-budgeted ordered optimization is the variant of min-max ordered optimization, wherein we are given multiple ordered norms and a budget for each ordered norm, and we seek a solution whose induced cost vector satisfies these budgets. As with min-max ordered optimization, our oblivious rounding procedures easily lead to $O(1)$-approximations for the multi-budgeted ordered {load balancing, k-clustering} problems.

Let $\alpha^*_{\mathcal{I}}$ be the smallest α such that there *exists* a solution whose cost-vector $\vec{v}$ satisfies $g(\vec{v}) \leq \alpha opt(g)$ for all monotone, symmetric norms g; here, $opt(g) = \min_{\vec{w}} g(\vec{w})$, where $\vec{w}$ ranges over the cost vectors induced by feasible solutions. A γ-approximation algorithm for the *best simultaneous-approximation factor* takes an instance $\mathcal{I}$ and returns a solution $\vec{v}$ such that $g(\vec{v}) \leq \gamma\alpha^*_{\mathcal{I}} opt(g)$ for any monotone, symmetric norm g. We devise $O(1)$-approximation algorithms for the best simultaneous-approximation factor for load balancing and k-clustering. The key idea stems from Goel and Meyerson [18], who use the majorization theory of Hardy, Littlewood, and Polya [22], to show that in order to simultaneously approximate all monotone, symmetric, norms, then it suffices to simultaneously approximate all Top-ℓ norms. If the best simultaneous approximation for a given instance is $\alpha^*_{\mathcal{I}}$, then we can cast the latter problem as a multi-budgeted ordered optimization problem with a budget of $\alpha^*_{\mathcal{I}} opt_\ell$ for each Top-ℓ-norm, where opt_ℓ is the optimal value for the Top-ℓ norm. Using our sparsification ideas, we can argue that we only need to consider the logarithmically many positions in POS, so although we do not know opt_ℓ, we can "guess" this for all $\ell \in$ POS. Using our $O(1)$-approximation results for multi-budgeted ordered optimization coupled with a binary search for $\alpha^*_{\mathcal{I}}$, then leads to an $O(1)$-approximation for $\alpha^*_{\mathcal{I}}$.

3 PRELIMINARIES

Solutions to the optimization problems we deal with in this paper induce cost vectors. We use $\vec{v}$ to denote them when talking about problems in the abstract. In load-balancing, the vector of the loads on machines is denoted by $\overrightarrow{\text{load}}$, or $\overrightarrow{\text{load}}_\sigma$ if σ is the assignment of jobs. In k-clustering, we the vector of assignment costs of clients is denoted as $\vec{c}$. We always use $\vec{o}$ to denote the cost vector in the optimum solution.

For an integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For a vector $\vec{v} \in \mathbb{R}^n$, we use $\vec{v}^{\downarrow}$ to denote the vector v with coordinates sorted in non-increasing order. That is, we have $\vec{v}^{\downarrow}_i = \vec{v}_{\pi(i)}$, where π is a permutation of $[n]$ such that $\vec{v}_{\pi(1)} \geq \vec{v}_{\pi(2)} \geq \ldots \vec{v}_{\pi(n)}$.

Throughout the paper, we use w (with or without superscripts) to denote a non-increasing, non-negative weight vector. The dimension of this vector is the dimension of the cost vector. In the abstract, we use n to denote this dimension; so $w \in \mathbb{R}^n_+$ and $w_1 \geq w_2 \geq \cdots \geq w_n \geq 0$. We use $\widetilde{w}$ to denote the "sparsified" version of the weight vector w which is defined in Section 4.

Ordered and top-ℓ optimization. Given a weight vector w as above, the *ordered optimization* problem asks to find a solution with induced cost vector $\vec{v}$ which minimizes $\text{cost}(w; \vec{v}) := \sum_{i=1}^n w_i \vec{v}^{\downarrow}_i$. This is the w-ordered norm, or simply ordered norm of $\vec{v}$. The special case where w is a $\{0,1\}$ vector—so $w_1 = \cdots w_\ell = 1$ (for some $\ell \in [n]$) and $w_i = 0$ otherwise—is called Top-ℓ *optimization*: we seek a solution $\vec{v}$ minimizing the sum of the ℓ largest entries. We use $\text{cost}(\ell; \vec{v})$ to denote the cost of the Top-ℓ optimization problem. In the literature in the k-clustering setting, the Top-ℓ optimization problem is called the *ℓ-centrum problem*, and the ordered optimization problem is called the *ordered k-median* problem.

Min-max and multi-budgeted ordered optimization. In a significant generalization of ordered optimization, we are given multiple non-increasing weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}^n_+$, and min-max ordered optimization asks to find a solution with induced cost vector $\vec{v}$ which minimizes $\max_{r \in [N]} \text{cost}(w^{(r)}; \vec{v})$. A related problem called *multi-budget ordered optimization* has the same setting as min-max ordered optimization, but one is also given N budgets $B_1, \ldots, B_N \geq 0$. The objective is to find a solution inducing cost vector $\vec{v}$ such that $\text{cost}(w; \vec{v}) \leq B_r$, for all r. This problem leads to connections with simultaneous optimization [18, 27].

Minimum norm optimization. A function $f : \mathbb{R}^n \to \mathbb{R}$ is a norm if (i) $f(x) = 0$ iff $x = 0$; (ii) $f(x + y) \leq f(x) + f(y)$ for all $x, y \in \mathbb{R}^n$ (triangle inequality); and (iii) $f(\lambda x) = |\lambda| f(x)$ for all $x \in \mathbb{R}^n, \lambda \in \mathbb{R}$ (homogeneity). Properties (ii) and (iii) imply that f is convex. f is *symmetric* if permuting the coordinates of x does not affect its value, i.e., $f(x) = f(x^{\downarrow})$ for all $x \in \mathbb{R}^n$. f is monotone if increasing its coordinate cannot decrease its value In *minimum norm optimization* problem we are given a monotone, symmetric norm f, and we have to find a solution inducing a cost vector $\vec{v}$ which minimized $f(\vec{v})$. Notice that Top-ℓ optimization, ordered optimization, and min-max ordered optimization are special cases of this problem.

Load balancing and k-clustering problems. In the load balancing setting, we have m machines, n jobs, and a processing time $p_{ij} \geq 0$ for job j on machine i. A solution to the problem is an assignment σ of jobs to machines. This induces a load vector $\overrightarrow{\text{load}}_\sigma \in \mathbb{R}^m$, with $\text{load}_\sigma(i) := \sum_{j:\sigma(j)=i} p_{ij}$ for all $i \in [m]$, which is the cost-vector associated with σ. Thus, the min-norm load balancing problem asks to find σ minimizing $f(\overrightarrow{\text{load}}_\sigma)$.

In the k-clustering setting, we have a metric space $\left(\mathcal{D}, \{c_{ij}\}_{i,j \in \mathcal{D}}\right)$, and an integer $k \geq 0$. A solution to the problem is a set $F \subset \mathcal{D}$, $|F| = k$ of k *open* facilities. This induces a cost-vector $\vec{c}$, where $\vec{c}_j := \min_{i \in F} c_{ij}$ is the assignment cost of j. In minimum-norm k-clustering, we seek a set F of facilities that minimizes $f(\vec{c})$.

4 SPARSIFYING WEIGHTS

Let $\delta > 0$ be a parameter. We show how to sparsify $w \in \mathbb{R}^n$ to a weight vector $\widetilde{w} \in \mathbb{R}^n$ (with non-increasing coordinates) having $O(\log n/\delta)$ distinct weight values, such that for any vector $\vec{v}$, we have $\text{cost}(\widetilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v}) \leq (1 + \delta)\text{cost}(\widetilde{w}; \vec{v})$. Moreover, an important property we ensure is that the breakpoints of $\widetilde{w}$—i.e., the indices where $\widetilde{w}_i > \widetilde{w}_{i+1}$—lie in a set that depends only by n and δ and is *independent* of w. As explained in Section 2, sparsification

in two distinct places; one, to give a polynomial time reduction from min-norm optimization to min-max ordered optimization (Section 5), and two, to specify proxy costs which allow us to tackle min-max ordered optimization.

For simplicity, we first describe a sparsification that leads to a factor-2 loss (instead of $1+\delta$), and then refine this. For every index $i \in [n]$, we set $\widetilde{w}_i = w_i$ if $i = \min\{2^s, n\}$ for some integer $s \geq 0$; otherwise, if $s \geq 1$ is such that $2^{s-1} < i < \min\{2^s, n\}$, set $\widetilde{w}_i = w_{\min\{2^s,n\}} = \widetilde{w}_{\min\{2^s,n\}}$. Note that $\widetilde{w} \leq w$ coordinate wise, and $\widetilde{w}_1 \geq \widetilde{w}_2 \geq \ldots \widetilde{w}_n$.

Observe that, unlike a different sparsification based on, say, geometric bucketing of the w_is, the sparsified vector $\widetilde{w}$ is *not* component-wise close to w; in fact $\widetilde{w}_i$ could be substantially smaller than w_i for an index i. Despite this, Claim 4.1 shows that $\text{cost}(\widetilde{w}; \vec{v})$ and $\text{cost}(w; \vec{v})$ are close to each other.

Claim 4.1. *For any $\vec{v} \in \mathbb{R}^n_+$, we have $\text{cost}(\widetilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v}) \leq 2\text{cost}(\widetilde{w}; \vec{v})$.*

Proof. Since $\widetilde{w} \leq w$, it is immediate that $\text{cost}(\widetilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v})$. The other inequality follows from a charging argument. Note that for any $s \geq 2$, we have $\left(\min\{2^s, n\} - 2^{s-1}\right) \leq 2\left(\min\{2^{s-1}, n\} - 2^{s-2}\right)$; hence, the cost contribution $\sum_{i=2^{s-1}+1}^{\min\{2^s,n\}} w_i \vec{v}_i^{\downarrow}$ is at most twice the cost contribution in $\text{cost}(\widehat{w}; v)$ from the indices $i \in \left\{2^{s-2}+1, \ldots, \min\{2^{s-1}, n\}\right\}$. The remaining cost $w_1\vec{v}_1^{\downarrow} + w_2\vec{v}_2^{\downarrow}$ is at most $2\widetilde{w}_1\vec{v}_1^{\downarrow}$. □

For the refined sparsification that only loses a $(1+\delta)$-factor, we consider positions that are powers of $(1+\delta)$. Let $\text{POS}_{n,\delta} := \left\{\min\{\lceil(1+\delta)^s\rceil, n\} : s \geq 0\right\}$. (Note that $\{1, n\} \subseteq \text{POS}_{n,\delta}$.) Observe that $\text{POS}_{n,\delta}$ depends *only* on n, δ and is oblivious of the weight vector. We abbreviate $\text{POS}_{n,\delta}$ to POS in the remainder of this section, and whenever n, δ are clear from the context. For $\ell \in \text{POS}$, $\ell < n$, define $\text{next}(\ell)$ to be the smallest index in POS larger than ℓ. For every index $i \in [n]$, we set $\widetilde{w}_i = w_i$ if $i \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$ (note that $\ell < n$), set $\widetilde{w}_i = w_{\text{next}(\ell)} = \widetilde{w}_{\text{next}(\ell)}$. Lemma 4.2 generalizes Claim 4.1.

Lemma 4.2. *For any $\vec{v} \in \mathbb{R}^n_+$, we have $\text{cost}(\widetilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v}) \leq (1+\delta)\text{cost}(\widetilde{w}; \vec{v})$.*

We once again stress that the, perhaps more natural, way of geometric bucketing (which is indeed used by [5, 13, 15]) where one ignores small w_is and rounds down each remaining w_i to the nearest power of 2 (or $(1+\varepsilon)$), doesn't work for our purposes. With geometric bucketing, the resulting sparsified vector w' is component-wise close to w (and so $\text{cost}(w'; \vec{v})$ is close to $\text{cost}(w; \vec{v})$). But the breakpoints of w' depend heavily on w, whereas the breakpoints of $\widetilde{w}$ all lie in POS. As noted earlier, this non-dependence on w is extremely crucial for us.

5 REDUCING MIN-NORM OPTIMIZATION TO MIN-MAX ORDERED OPTIMIZATION

In this section we show our reduction of the minimum norm optimization problem to min-max ordered optimization. We are given a monotone, symmetric norm $f : \mathbb{R}^n \to \mathbb{R}_+$, and we want to find a solution to the underlying optimization problem which minimizes the f evaluated on the induced cost vector. Let $\vec{o}$ denote the optimal cost vector and let $opt = f(\vec{o})$.

We assume the following *(approximate) ball-optimization oracle*. Given any cost vector $c \in \mathbb{R}^n$, we can (approximately) optimize $c^\top x$ over the ball $\mathbb{B}_+(f) := \{x \in \mathbb{R}^n_+ \ : \ f(x) \leq 1\}$.

$$\text{Oracle } \mathcal{A} \text{ takes input } c \in \mathbb{R}^n_+ \text{ returns a } \kappa\text{-approximation to } Bopt(c) := \max\{c^\top x : x \in \mathbb{B}_+(f)\} \tag{1}$$

Note that under mild assumptions, the ball-optimization oracle can be obtained, via the ellipsoid method, using a first-order oracle for f that returns the subgradient (or even approximate subgradient) of f. Recall, $d \in \mathbb{R}^n$ is a *subgradient* of f at $x \in \mathbb{R}^n$ if we have $f(y) - f(x) \geq d^T(y - x)$ for all $y \in \mathbb{R}^n$. It is well known that a convex function has a subgradient at every point in its domain.

We begin by stating some preliminary properties of norms, monotone norms, and symmetric norms.

Lemma 5.1. *Let $f : \mathbb{R}^n \to \mathbb{R}_+$ be a norm and $x \in \mathbb{R}^n_+$.*

(i) If d is a subgradient of f at x, then $f(x) = d^T x$ and $f(y) \geq d^T y$ for all $y \in \mathbb{R}^n$. Also, d is a subgradient of f at any point λx, where $\lambda \geq 0$.

(ii) If f is monotone, there exists a subgradient d of f at x such that $d \geq 0$.

(iii) Let f be symmetric, and d be a subgradient of f at x. Then, d and x are similarly ordered, i.e., if $d_i < d_j$ then $x_i \leq x_j$, and $f(x) = \text{cost}(d^{\downarrow}; x)$. Moreover, for any permutation $\pi : [n] \to [n]$, the vector $d^{(\pi)} := \left\{d_{\pi(i)}\right\}_{i \in [n]}$ is a subgradient of f at $x^{(\pi)}$.

Motivated by the above lemma, we define the following (possibly infinite) set of non-increasing subgradients at points in $\mathbb{B}_+(f)$.

$$C = \Big\{d \in \mathbb{R}^n_+ : \quad d_1 \geq d_2 \geq \ldots \geq d_n, \quad d \text{ is a subgradient of } f \text{ at some } x \in \mathbb{B}_+(f)\Big\}.$$

As a warm up, Lemma 5.2 shows that min-norm optimization is *equivalent* to min-max ordered optimization with an *infinite* collection of weight vectors.

Lemma 5.2. *Let $x \in \mathbb{R}^n_+$. We have $f(x) = \max_{w \in C} \text{cost}(w; x)$.*

Proof. We first argue that $f(x) \leq \max_{w \in C} \text{cost}(w; x)$. By part (ii) (of Lemma 5.1), there is a subgradient $d \geq 0$ of f at x. By part (iii), there is a common permutation π that defines $d^{\downarrow}$ and $x^{\downarrow}$, and $\widehat{d} = d^{\downarrow}$ is a subgradient of f at $x^{\downarrow}$. By part (i), $\widehat{d}$ is also a subgradient of f at $x^{\downarrow}/f(x^{\downarrow}) \in \mathbb{B}_+(f)$. So $\widehat{d} \in C$. Also, $f(x) = \text{cost}(\widehat{d}; x)$ (by part (iii)), and so $f(x) \leq \max_{w \in C} \text{cost}(w; x)$.

Conversely, consider any $w \in C$, and let it be a subgradient of f at $z \in \mathbb{B}_+(f)$. We have $f(x) = f(x^{\downarrow}) \geq w^T x^{\downarrow}$ (by part (i) of Lemma 5.1), and so $f(x) \geq \text{cost}(w; x)$. Therefore, $f(x) \geq \max_{w \in C} \text{cost}(w; x)$. □

To reduce to min-max ordered optimization, we need to find a polynomial-sized collection of weight vectors. Next, we show how to leverage the *weight sparsification* idea in Section 4 and achieve this taking a slight hit in the approximation factor. Let $0 < \varepsilon \leq 0.5$ be a parameter. The sparsification procedure (Lemma 4.2) shows that, with an $(1+\varepsilon)$-loss, we can focus on a set of $O(\log n/\varepsilon)$ coordinates and describe the weight vectors by their values at these

coordinates. For the ordered-optimization objective $\text{cost}(w;x)$, moving to the sparsified weight incurs only a $(1+\varepsilon)$-loss. Furthermore, again taking a loss of $(1+\varepsilon)$, we can assume these coordinates are set to powers of $(1+\varepsilon)$. Our goal (roughly speaking) is then only to consider the collection consisting of the sparsified, rounded versions of vectors in $\mathcal{C}$. Claim 5.3 implies that we can enumerate all sparsified, rounded weight vectors in polynomial time.

But we also need to be able to determine if such a vector $\widetilde{w}$ is "close" to a subgradient in $\mathcal{C}$, and this is where (1) is used. First note that $d \in \mathcal{C}$ iff[3] $Bopt(d) = 1$. Thus to check if $\widetilde{w}$ is "close" to a subgradient in $\mathcal{C}$, it suffices to (approximately) solve for $Bopt(\widetilde{w})$ and check if the answer is within $(1 \pm \varepsilon)$ (or scaled by κ if we only have an approximate oracle). We give the details next.

To make the enumeration go through we need to make the following mild assumptions. These assumptions need to be checked for the problems at hand, and are often easy to establish.

(A1) We can determine in polytime if $\vec{o}_1^{\downarrow} = 0$. If $\vec{o}_1^{\downarrow} > 0$ (so $opt > 0$), then $\vec{o}_1^{\downarrow} \geq 1$ (assuming integer data), and we can compute an estimate hi such that $\vec{o}_1^{\downarrow} \leq \mathsf{hi}$. In the sequel, assume that $\vec{o}_1^{\downarrow} \geq 1$.

(A2) We have bounds $\mathsf{lb}, \mathsf{ub} > 0$ such that $\mathsf{lb} \leq opt \leq \mathsf{ub}$. Then (A1) and Lemma 5.1 (i) imply that $d_1 \leq \mathsf{ub}$ for all $d \in \mathcal{C}$.

We take $\delta = \varepsilon$ in the sparsification procedure in Section 4. Let $\text{POS} = \text{POS}_{n,\varepsilon} := \{\min\{\lceil (1+\varepsilon)^s \rceil, n\} : s \geq 0\}$. Recall that $\text{next}(\ell)$ is the smallest index in POS larger than ℓ. The sparsified version of $w \in \mathbb{R}^n$ is the vector $\widetilde{w} \in \mathbb{R}^n$ given by $\widetilde{w}_i = w_i$ if $i \in \text{POS}$; and $\widetilde{w}_i = w_{\text{next}(\ell)}$ otherwise, where $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$. Since $\widetilde{w}$ is completely specified by specifying the positions in POS, we define the $|\text{POS}|$-dimensional vector $u := \{\widetilde{w}_\ell\}_{\ell \in \text{POS}}$. We identify $\widetilde{w}$ with $u \in \mathbb{R}_+^{\text{POS}}$ and say that $\widetilde{w}$ is the *expansion of* u.

$$\text{Define} \quad \mathcal{W}' \subseteq \mathbb{R}_+^n := \Big\{\text{expansion of } u \in \mathbb{R}_+^{\text{POS}} : \exists \ell^* \in \text{POS s.t. } u_\ell = 0 \ \forall \ell \in \text{POS with } \ell > \ell^*,$$
$$u_1, u_2, \ldots, u_{\ell^*} \text{ are powers of } (1+\varepsilon) \text{ (possibly smaller than 1)}$$
$$u_1 \in \Big[\tfrac{\mathsf{lb}}{n \cdot \mathsf{hi}},\ \mathsf{ub}(1+\varepsilon)\Big), \quad u_1 \geq u_2 \geq \ldots \geq u_{\ell^*} \geq \tfrac{\varepsilon u_1}{n(1+\varepsilon)}\Big\}.$$

Let $\mathbf{1}^n$ denote the all 1s vector in $\mathbb{R}^n$. Now define

$$\mathcal{W} := \Big\{w \in \mathcal{W}' : \text{ oracle } \mathcal{A} \text{ run on } w \text{ returns } \hat{x} \in \mathbb{B}_+(f) \text{ s.t. } w^T\hat{x} \in \Big[(1-\varepsilon)/\kappa, 1+\varepsilon\Big]\Big\} \cup \Big\{\tfrac{\mathsf{lb}}{n \cdot \mathsf{hi}} \cdot \mathbf{1}^n\Big\}.$$

The extra scaled all ones vector is added for a technical reason. We use the following enumeration claim.

Claim 5.3. *There are at most $(2e)^{\max\{N,k\}}$ non-increasing sequences of k integers chosen from $\{0,\ldots,N\}$.*

The following theorem establishes the reduction from the minimum norm problem to min-max ordered optimization.

Theorem 5.4. *For any $\vec{v} \in \mathbb{R}_+^n$, the following hold.*

(i) $\max_{w \in \mathcal{W}} \text{cost}(w;\vec{v}) \leq \max\Big\{\kappa(1+\varepsilon)f(\vec{v}), \frac{\mathsf{lb}}{n \cdot \mathsf{hi}} \sum_{i \in [n]} \vec{v}_i\Big\}$,

(ii) $f(\vec{v}) \leq (1-\varepsilon)^{-1} \max_{w \in \mathcal{W}} \text{cost}(w;\vec{v})$.

Hence, a γ-approximate solution $\vec{v}$ for the min-max ordered-optimization problem with objective $\max_{w \in \mathcal{W}} \text{cost}(w;\vec{v})$ (where $\gamma \geq 1$) satisfies $f(\vec{v}) \leq \gamma\kappa(1+3\varepsilon)opt$.

Constructing $\mathcal{W}$ requires $O\big(\frac{\log n}{\varepsilon^2} \log(\frac{n \cdot \mathsf{ub} \cdot \mathsf{hi}}{\mathsf{lb}})(\frac{n}{\varepsilon})^{O(1/\varepsilon)}\big)$ calls to $\mathcal{A}$, which is also a bound on $|\mathcal{W}|$.

[3] If $d \in \mathcal{C}$ is the subgradient of f at $y \in \mathbb{B}_+(f)$, $d^T x \leq f(x) \leq 1 \ \forall x \in \mathbb{B}_+(f)$, and $d^T y/f(y) = 1$, so $\max_{x \in \mathbb{B}_+(f)} d^T x = 1$. Alternately, if $Bopt(d) = 1$, then we have $d^\top z = 1$ for some $f(z) \leq 1$ implying $f(z) + d^\top(y-z) \leq d^\top y$ for any y. If the LHS is $> f(y)$, then we would get $d^\top(y/f(y)) > 1$ contradicting $Bopt(d) = 1$.

6 PROXY COSTS

As mentioned in Section 2, the key to tackling ordered optimization is to view the problem of minimizing the sum of a suitably devised proxy-cost function over all coordinates. We describe this proxy in this section. We first so so for Top-ℓ optimization. This will serve to motivate and illuminate the proxy-cost function that we use for (general) ordered optimization. As usual, we use $\vec{o}$ to denote the cost vector corresponding to an optimal solution, and *opt* to denote the optimal cost. Recall, $\text{cost}(\ell;\vec{v})$ is the cost of the Top-ℓ optimization.

Define $z^+ := \max\{0, z\}$ for $z \in \mathbb{R}$. For any scalar $\rho > 0$, define $h_\rho(z) := (z-\rho)^+$. The main insight is that for any $\vec{v} \in \mathbb{R}^n$, we have $\text{cost}(\ell;\vec{v}) = \min_{\rho \in \mathbb{R}}\big(\ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{v}_i)\big)$.

Claim 6.1. *For any $\ell \in [n]$, any $\vec{v} \in \mathbb{R}^n$, and any $\rho \in \mathbb{R}$, we have $\text{cost}(\ell;\vec{v}) \leq \ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{v}_i)$.*

Claim 6.2. *Let $\ell \in [n]$, and ρ be such that $\vec{o}_\ell^{\downarrow} \leq \rho \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow}$. Then $\ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{o}_i) \leq (1+\varepsilon)\text{cost}(\ell;\vec{o})$.*

The above claims indicate that if we obtain a good estimate ρ of $\vec{o}_\ell^{\downarrow}$, then $\ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{v}_i)$ can serve as a good proxy for $\text{cost}(\ell;\vec{v})$, and we can focus on the problem of finding v minimizing $\sum_{i=1}^n h_\rho(\vec{v}_i)$. The following properties will be used many times.

Claim 6.3. *We have: (i) $h_\rho(x) \leq h_\rho(y)$ for any ρ, $x \leq y$; (ii) $h_{\rho_1}(x) \leq h_{\rho_2}(x)$ for any $\rho_1 \geq \rho_2$, and any x; (iii) $h_{\rho_1+\rho_2}(x+y) \leq h_{\rho_1}(x) + h_{\rho_2}(y)$ for any ρ_1, ρ_2, x, y.*

Proof. Part (iii) is the only part that is not obvious. If $h_{\rho_1+\rho_2}(x+y) = 0$, then the inequality clearly holds; otherwise, $h_{\rho_1+\rho_2}(x+y) = x - \rho_1 + y - \rho_2 \leq (x-\rho_1)^+ + (y-\rho_2)^+$. □

We remark that our proxy function for Top-ℓ optimization is similar to, but subtly stronger than, the proxy function utilized in recent prior works on the ℓ-centrum and ordered k-median clustering problems [13, 15]. This strengthening (and its extension to ordered optimization) forms the basis of our significantly improved approximation guarantees of $(5+\varepsilon)$ for ordered k-median, which improves upon the prior-best guarantees for *both* ℓ-centrum and ordered k-median [15]. Furthermore, this proxy function also leads to (essentially) a 2-approximation for Top-ℓ load balancing and ordered load balancing.

Ordered optimization. We now build upon our insights for Top-ℓ optimization. Let $w \in \mathbb{R}^n$ be the weight vector (with non-increasing coordinates) underlying the ordered-optimization problem. So, $opt = \text{cost}(w;\vec{o})$ is the optimal cost. The intuition underlying our proxy function comes from the observation that we can write $\text{cost}(w;\vec{v}) = \sum_{i=1}^n (w_i - w_{i+1})\text{cost}(i;\vec{v})$, where we define $w_{n+1} := 0$. Plugging in the proxy functions for $\text{cost}(i;\vec{v})$ in this expansion immediately leads to a proxy function for $\text{cost}(w;\vec{v})$. The $\text{cost}(i;\vec{v})$ terms that appear with positive coefficients in the above linear combination

are those where $w_i > w_{i+1}$, i.e., corresponding to the breakpoints of w. Thus, the proxy function that we obtain for ordered optimization will involve multiple ρ-thresholds, which are intended to be the estimates of the $\vec{o}_i^{\downarrow}$ values corresponding to breakpoints. However, we cannot afford to "guess" so many thresholds. So an important step to make this work is to first *sparsify* the weight vector w to control the number of breakpoints, and then utilize the above expansion. As mentioned in Section 4, while geometric bucketing of weights would reduce the number of breakpoints for a single weight function, for our applications to min-max ordered optimization, we need the uniform way of sparsifying multiple weight vectors, and we therefore use the sparsification procedure in Section 4.

Let $\text{POS} = \text{POS}_{n,\delta} := \{\min\{\lceil(1+\delta)^s\rceil, n\} : s \geq 0\}$, where $\delta, \varepsilon > 0$ are parameters. Recall that $\text{next}(\ell)$ is the smallest index in POS larger than ℓ. For notational convenience, we define $\text{next}(n) := n+1$, and for $\vec{v} \in \mathbb{R}^n$, define $\vec{v}_{n+1} := 0$. We sparsify w to $\widetilde{w} \in \mathbb{R}^n$ by setting $\widetilde{w}_i = w_i$ if $i \in \text{POS}$, and $\widetilde{w}_i = w_{\text{next}(\ell)}$ otherwise, where $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$.

Our proxy function is obtained by guessing (roughly speaking) the thresholds $\vec{o}_\ell^{\downarrow}$ for all $\ell \in \text{POS}$ within a multiplicative $(1+\varepsilon)$ factor, and rewriting $\text{cost}(\widetilde{w}; \vec{v})$ in terms of these thresholds. Let $\vec{t} := \{t_\ell\}_{\ell \in \text{POS}}$ be a *threshold vector*. Define $\vec{t}_{n+1} := 0$. We say that $\vec{t}$ is *valid* if $t_\ell \geq t_{\text{next}(\ell)}$ for all $\ell \in \text{POS}$. (So this implies that $\vec{t} \geq 0$.) A valid threshold vector $\vec{t}$, defines the proxy function.

$$\text{prox}_{\vec{t}}(\widetilde{w}; \vec{v}) := \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\Big[\ell \cdot t_\ell + \sum_{i=1}^{n} h_{t_\ell}(\vec{v}_i)\Big]$$

$$= \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\ell \cdot t_\ell + \sum_{i=1}^{n} h_{\vec{t}}\,(\widetilde{w}; \vec{v}_i), \qquad (2)$$

$$\text{where,} \quad h_{\vec{t}}\,(\widetilde{w}; a) := \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)h_{t_\ell}(a) \qquad (3)$$

Throughout the rest of this section, we work with the sparsified weight vector $\widetilde{w}$. Observe that $h_{\vec{t}}\,(\widetilde{w}; x)$ is a continuous, piecewise-linear, non-decreasing function of x. Our proxy for $\text{cost}(\widetilde{w}; \vec{v})$ will be the function $\text{prox}_{\vec{t}}(\widetilde{w}; \vec{v})$ for a suitably chosen threshold vector $\vec{t}$. To explain the above definition, notice that (2) is the expression obtained by plugging in the proxy functions $(\ell \cdot \rho + \sum_{i=1}^{n}(v_i - \rho)^+)$ defined for the $\text{cost}(\ell; \cdot)$-objectives in the expansion of $\text{cost}(\widetilde{w}; v)$ as a linear combination of $\text{cost}(\ell; v)$ terms.

Claim 6.4. *For any $\vec{v} \in \mathbb{R}^n$, we have $\text{cost}(\widetilde{w}; \vec{v}) = \sum_{\ell \in \text{POS}}\big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\text{cost}(\ell; \vec{v})$.*

Claim 6.5. *For any valid threshold vector $\vec{t} \in \mathbb{R}^{\text{POS}}$, and any $\vec{v} \in \mathbb{R}^n$, we have $\text{cost}(\widetilde{w}; \vec{v}) \leq \text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{v})$.*

Proof. We have $\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{v}) = \sum_{\ell \in \text{POS}}\big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\big(\ell \cdot t_\ell + \sum_{i=1}^{n} h_{t_\ell}(\vec{v}_i)\big)$. The statement now follows by combining Claim 6.4 and Claim 6.1, taking $t = t_\ell$ for each $\ell \in \text{POS}$. □

Claim 6.6. *Let $\vec{t} \in \mathbb{R}^{\text{POS}}$ be a valid threshold vector such that $\vec{o}_\ell^{\downarrow} \leq t_\ell \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow}$ for all $\ell \in \text{POS}$. Then, $\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{o}) \leq (1+\varepsilon)\text{cost}(\widetilde{w}; \vec{o})$.*

Proof. We have $\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{o}) = \sum_{\ell \in \text{POS}}\big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\big(\ell \cdot t_\ell + \sum_{i=1}^{n} h_{t_\ell}(\vec{o}_i^{\downarrow})\big)$. The statement now follows by combining Claim 6.2, where we take $t = t_\ell$ for each $\ell \in \text{POS}$, and Claim 6.4. □

Claim 6.5 and Claim 6.6 imply that: (1) if we can obtain in polytime a valid threshold vector $\vec{t} \in \mathbb{R}^{\text{POS}}$ satisfying the conditions of Claim 6.6, and (2) obtain a cost vector v that approximately minimizes $\sum_{i=1}^{n} h_{\vec{t}}\,(v_i)$, then we would obtain an approximation guarantee for the ordered-optimization problem. We will not quite be able to satisfy (1). Instead, we will obtain thresholds that will satisfy a somewhat weaker condition (see Lemma 6.8), which we show is still sufficient. The following claim will be useful.

Claim 6.7. *Let $\vec{t}, \vec{t}' \in \mathbb{R}^{\text{POS}}$ be two valid threshold vectors with $\vec{t} \leq \vec{t}'$ and $\|\vec{t} - \vec{t}'\|_\infty \leq \Delta$. Then, for any $\vec{v} \in \mathbb{R}^n$, we have $\big|\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{v}) - \text{prox}_{\vec{t}'}\,(\widetilde{w}; \vec{v})\big| \leq n\widetilde{w}_1\Delta$.*

Lemma 6.8. *Let $\vec{t} \in \mathbb{R}^{\text{POS}}$ be a valid threshold vector satisfying the following for all $\ell \in \text{POS}$: $\vec{o}_\ell^{\downarrow} \leq t_\ell \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow}$ if $\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon\vec{o}_1^{\downarrow}}{n}$, and $t_\ell = 0$ otherwise. Then,*

$$\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{o}) = \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\ell \cdot t_\ell + \sum_{i=1}^{n} h_{\vec{t}}\,(\widetilde{w}; \vec{o}_i)$$
$$\leq (1+2\varepsilon)\text{cost}(\widetilde{w}; \vec{o}).$$

Proof. For $\ell \in \text{POS}$, define $t'_\ell = t_\ell$ if $\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon\vec{o}_1^{\downarrow}}{n}$, and $t'_\ell = \vec{o}_\ell^{\downarrow}$ otherwise. Clearly, $\vec{t} \leq \vec{t}'$ and $\|\vec{t} - \vec{t}'\|_\infty \leq \frac{\varepsilon\vec{o}_1^{\downarrow}}{n}$, so by Claim 6.7, we have $\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{o}^{\downarrow}) \leq \text{prox}_{\vec{t}'}\,(\widetilde{w}; \vec{o}^{\downarrow}) + \varepsilon\widetilde{w}_1\vec{o}_1^{\downarrow}$. The threshold vector $\vec{t}'$ satisfies the conditions of Claim 6.6, so $\text{prox}_{\vec{t}'}\,(\widetilde{w}; \vec{o}) \leq (1+\varepsilon)\text{cost}(\widetilde{w}; \vec{o})$. So $\text{prox}_{\vec{t}}\,(\widetilde{w}; \vec{o}) \leq (1+2\varepsilon)\text{cost}(\widetilde{w}; \vec{o})$. □

Lemma 6.9 (Polytime enumeration of threshold vectors). *Suppose we can obtain in polynomial time a (polynomial-size) set $S \subseteq \mathbb{R}$ containing a value ρ satisfying $\vec{o}_1^{\downarrow} \leq \rho \leq (1+\varepsilon)\vec{o}_1^{\downarrow}$. Then, in time $O\big(|S| \cdot |\text{POS}| \cdot \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\}\big) = O\big(|S| \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{O(1/\delta)}\}\big)$, we can obtain a set $A \subseteq \mathbb{R}_+^{\text{POS}}$ that contains a valid threshold vector $\vec{t}$ satisfying the conditions of Lemma 6.8.*

If $\vec{o}$ is integral, $\vec{o}_1^{\downarrow} > 0$, and ρ is a power of $(1+\varepsilon)$, then this $\vec{t}$ satisfies: for every $\ell \in \text{POS}$, either $t_\ell = 0$ or $t_\ell \geq 1$ and is a power of $(1+\varepsilon)$.

Proof. We first guess the largest index $\ell^* \in \text{POS}$ such that $\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon\vec{o}_1^{\downarrow}}{n}$. For each such ℓ^*, and each $t_1 \in S$, we do the following. We guess t_ℓ for $\ell \in \text{POS}, 2 \leq \ell \leq \ell^*$, where all the t_ℓs are of the form $t_1/(1+\varepsilon)^j$ for some integer $j \geq 0$ and are at least $\frac{\varepsilon t_1}{n(1+\varepsilon)}$, and the j-exponents are non-decreasing with ℓ. For $\ell \in \text{POS}$ with $\ell > \ell^*$, we set $t_\ell = 0$, and add the resulting threshold vector $\vec{t}$ to A. Note that there are at most $1 + \log_{1+\varepsilon}\big(\frac{n}{\varepsilon}\big) = O\big(\frac{1}{\varepsilon}\log\frac{n}{\varepsilon}\big)$ choices for the exponent j. So since we need to guess a non-decreasing sequence of at most $|\text{POS}| = O(\log n/\delta)$ exponents from a range of size $O\big(\frac{1}{\varepsilon}\log\frac{n}{\varepsilon}\big)$, there are only $\exp\big(\max\{O(\frac{1}{\varepsilon}\log(\frac{n}{\varepsilon})), |\text{POS}|\}\big) = O\big(\max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\}\big)$ choices (by Claim 5.3). So the enumeration takes time $O\big(|S| \cdot |\text{POS}| \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\}\big)$, which is also an upper bound on $|A|$.

We now argue that A contains a desired valid threshold vector. First, note that by construction A only contains valid threshold vectors. Consider the iteration when we consider $t_1 = \rho$, and have guessed ℓ^* correctly. For $\ell \in \text{POS}$ with $2 \leq \ell \leq \ell^*$, we know that

$\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon \vec{o}_1^{\downarrow}}{n} \geq \frac{\varepsilon t_1}{n(1+\varepsilon)}$ and $\vec{o}_\ell^{\downarrow} \leq \vec{o}_1^{\downarrow} \leq t_1$. So we will enumerate non-increasing values $t_2, \ldots, t_{\ell^*}$ such that $\vec{o}_\ell^{\downarrow} \leq t_\ell \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow}$ for each such ℓ. The remaining t_ℓs are set to 0, so $\vec{t}$ satisfies the conditions of Lemma 6.8.

Finally, suppose $\vec{o} \in \mathbb{Z}_+^n$ and ρ is a power of $(1+\varepsilon)$. If $t_\ell < 1$, then $\ell \geq \ell^*$, but $\vec{o}_\ell^{\downarrow} \leq t_\ell < 1$, which means that $\vec{o}_\ell^{\downarrow} = 0$ contradicting that $\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon \vec{o}_1^{\downarrow}}{n}$. Also, $t_\ell = \rho/(1+\varepsilon)^j$, so it is a power of $(1+\varepsilon)$. □

The upshot of the above discussion is that it suffices to focus on the algorithmic problem of minimizing $\sum_{i=1}^n h_{\vec{t}}(v_i)$ for a given valid threshold vector. This is formalized by the following lemma.

Lemma 6.10. *Let $\vec{t} \in \mathbb{R}^{\mathrm{POS}}$ be a valid threshold vector satisfying the conditions of Lemma 6.8. Let $\vec{v} \in \mathbb{R}_+^n$ be such that $\sum_{i=1}^n h_{\theta\vec{t}}(\widetilde{w}; \vec{v}_i) \leq \gamma \cdot \sum_{i=1}^n h_{\vec{t}}(\widetilde{w}; \vec{o}_i) + M$, where $\gamma, \theta \geq 1$, $M \geq 0$. Then, $\mathrm{cost}(\widetilde{w}; \vec{v}) \leq \max\{\theta, \gamma\}(1+2\varepsilon)\mathrm{cost}(\widetilde{w}; \vec{o}) + M$, and hence $\mathrm{cost}(w; \vec{v}) \leq (1+\delta)\max\{\theta, \gamma\}(1+2\varepsilon)opt + (1+\delta)M$.*

7 OUR APPROACH FOR MIN-MAX ORDERED OPTIMIZATION

Given the reduction Theorem 5.4 in Section 5, we now discuss our approach for solving min-max-ordered load balancing and clustering. Eventually, we will need to take a problem-dependent approach, but at a high level, there are some common elements to our approaches for the two problems as we now elucidate.

As a stepping stone, we first consider ordered optimization (i.e., where we have one weight vector w), and formulate a suitable LP-relaxation (see Section 9.1 and Section 8.1) for the problem of minimizing $\sum_{i=1}^n h_{\vec{t}}(\widetilde{w}; \vec{v}_i)$, i.e., the $\vec{v}$-dependent part of our proxy function for $\mathrm{cost}(\widetilde{w}; \vec{v})$ (see (2) and (3)), where $\widetilde{w}$ is the sparsified version of w. Our LP-relaxation will have the property that only its objective depends on $\widetilde{w}$ and not its constraints. The LP for min-max ordered optimization is obtained by modifying the objective in the natural way.

The technical core of our approach involves devising a deterministic, weight-oblivious rounding procedure for this LP (see Theorems 8.4 and 9.5). To elaborate, we design a procedure that given an arbitrary feasible solution, say $\overline{x}$, to this LP, rounds it *deterministically*, without any knowledge of w, to produce a solution to the underlying optimization problem whose induced cost vector $\vec{v}$ satisfies the following: for *every* sparsified weight vector $\widetilde{w}$, we have (loosely speaking) $\mathrm{cost}(\widetilde{w}, \vec{v}) = O(1)\cdot$(LP-objective-value of $\overline{x}$ under $\widetilde{w}$). We call this a *deterministic, weight-oblivious* rounding procedure. To achieve this, we need to introduce some novel constraints in our LP, beyond the standard ones for load balancing and k-clustering. The benefit of such an oblivious guarantee is clear: if $\overline{x}$ is an optimal solution to the LP-relaxation for min-max ordered optimization, then the above guarantee yields $O(1)$-approximation for the min-max ordered-optimization problem. Indeed, this also will solve the multi-budgeted ordered optimization problem.

We point out that it is important that the oblivious rounding procedures we design are *deterministic*, which is also what makes them noteworthy, and we need to develop various new ideas to obtain such guarantees. Using a randomized $O(1)$-approximation oblivious rounding procedure in min-max ordered optimization would yield that the *maximum expected cost* $\mathrm{cost}(w^{(i)}; \widetilde{v})$ under weight vectors $w^{(i)}$ in our collection is $O(opt)$; but what we need is a bound on the *expected maximum cost*. Therefore, without a sharp concentration result, a randomized oblivious guarantee is insufficient for the purposes of utilizing it for min-max ordered optimization. Also, note that derandomizing an oblivious randomized-rounding procedure would typically cause it to lose its obliviousness guarantee. (We also remark that if we allow randomization, then it is well-known that *any* LP-relative approximation algorithm can be used to obtain a randomized oblivious rounding procedure (see [14].)

To obtain our deterministic oblivious rounding procedure, we first observe that $\sum_{i=1}^n h_{\vec{t}}(\widetilde{w}; \vec{v}_i)$ can be equivalently written as $\sum_{\ell \in \mathrm{POS}} \widetilde{w}_{\mathrm{next}(\ell)} \sum_{i=1}^n \left(\min\{\vec{v}_i, t_\ell\} - t_{\mathrm{next}(\ell)}\right)^+$. In our LP-relaxation, we introduce fractional variables to specify the quantities $\sum_{i=1}^n \left(\min\{\vec{v}_i, t_\ell\} - t_{\mathrm{next}(\ell)}\right)^+$. If we can round the fractional solution while roughly preserving these quantities (up to constant factors), then we can get the desired oblivious guarantee. This is what we achieve (allowing for an $O(1)$ violation of the thresholds) by, among other things, leveraging our new valid constraints that we add to the LP. For instance, in load balancing, $\vec{v}_i$ denotes the load on machine i and the above quantity represents the portion of the total load on a machine between thresholds $t_{\mathrm{next}(\ell)}$ and t_ℓ, and we seek to be preserve this in the rounding.

Preserving the aforementioned quantities amounts to having multiple knapsack constraints, and rounding them so as to satisfy them with as little violation as possible. We utilize the following technical tool to achieve this. We emphasize that the objective $c^T q$ below is *not* related to $\widetilde{w}$, but encodes quantities that arise in our rounding procedure. Theorem 7.1 is proved using *iterative rounding*, by combining ideas from [9], which considered directed network design, and the ideas involved in an iterative-rounding based 2-approximation algorithm for the generalized assignment problem (see Section 3.2 of [30]). Similar results are known in the literature, but we could not quite find a result that exactly fits our needs.

Theorem 7.1. *Let $\hat{q}$ be a feasible solution to the following LP:*

$$\min \quad c^T q \quad A_1 q \leq b_1, \; A_2 q \geq b_2, \quad Bq \leq d, \quad q \in \mathbb{R}_+^M. \qquad \text{(Q)}$$

Suppose that: (i) $A_1, A_2, B, b_1, b_2, d \geq 0$; (ii) A_1, A_2 are $\{0,1\}$-matrices, and the supports of the rows of $\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ form a laminar family; (iii) b_1, b_2 are integral; and (iv) $q_j \leq 1$ is an implicit constraint implied by $A_1 q \leq b_1$, $A_2 q \geq b_2$. Let k be the maximum number of constraints of $Bq \leq d$ that a variable appears in.

We can round $\hat{q}$ to an integral (hence $\{0,1\}$) solution $\overset{int}{q}$ satisfying: (a) $c^T \overset{int}{q} \leq c^T \hat{q}$; (b) the support of $\overset{int}{q}$ is contained in the support of $\hat{q}$; (c) $A_1 \overset{int}{q} \leq b_1$, $A_2 \overset{int}{q} \geq b_2$; and (d) $(B\overset{int}{q})_i \leq d_i + k(\max_{j:\hat{q}_j>0} B_{ij})$ for all i ranging over the rows of B.

8 k-CLUSTERING

We now use our framework to design constant factor approximation algorithms for the minimum-norm k-clustering problem. We are given a metric space $(\mathcal{D}, \{c_{ij}\}_{i,j\in\mathcal{D}})$, and an integer $k \geq 0$. Let $n = |\mathcal{D}|$. For notational similarity with facility-location problems, let $\mathcal{F} := \mathcal{D}$, denote the candidate set of facilities. (Our results either directly extend, or can be adapted, to the setting where $\mathcal{F} \neq \mathcal{D}$.) A feasible solution opens a set $F \subseteq \mathcal{F}$ of at most k facilities, and

assigns each client $j \in \mathcal{D}$ to a facility $i(j) \in F$. This results in the assignment-cost vector $\vec{c} := \{c_{i(j)j}\}_{j\in\mathcal{D}}$.

In the *minimum-norm k-clustering* problem, the goal is to minimize $f(\vec{c})$ under a given monotone, symmetric norm f. The *ordered k-median* problem is the special case where we are given weights $w_1 \geq w_2 \geq \ldots \geq w_n \geq 0$, and the goal is to minimize $\text{cost}(w;\vec{c}) = w^T\vec{c}^{\downarrow}$. The ℓ*-centrum* problem is the further special case, where $w_1 = 1 = \ldots = w_\ell$ and the remaining w_is are 0.

Theorem 8.1. *Given any monotone, symmetric norm specified via a κ-approximate ball-optimization oracle (see (1)), and any $\varepsilon > 0$, there is a $\kappa\big(408+O(\varepsilon)\big)$-approximation algorithm for minimum-norm k-clustering with running time* $\text{poly}\big(\textit{input size}, (\frac{n}{\varepsilon})^{O(1/\varepsilon)}\big)$.

As shown by the reduction in Section 5, the key component needed to tackle the norm-minimization problem is an algorithm for the *min-max ordered k-median problem*, wherein we are given *multiple* non-increasing weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}^n_+$, and the goal is to find $F \subseteq \mathcal{F}$ with $|F| \leq k$ such that the resulting assignment-cost vector $\vec{c}$ minimizes $\max_{r\in[L]} \text{cost}(w^{(r)};\vec{c})$.

Theorem 8.2. *Given non-increasing weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}^n_+$, for any $\varepsilon > 0$, there is a $\big(408+O(\varepsilon)\big)$-approximation algorithm for min-max ordered k-median with running time* $\text{poly}\big(\textit{input size}, n^{O(1/\varepsilon)}\big)$.

As described in Section 7, the key technical component is a *deterministic, weight-oblivious rounding procedure* for our LP-relaxation for (single) ordered k-median (Section 8.1). The approximation factor of this rounding procedure then translates to the above guarantees. Furthermore, notably, the oblivious rounding can be exploited to obtain guarantees for multi-budgeted ordered k-median and the best simultaneous approximation achievable for k-clustering.

We have not optimized the constant in the approximation factor for easier exposition of ideas. For the special case of ordered k-median, we can obtain a much better approximation factor of $5+\varepsilon$, which significantly improves upon the guarantees in [13, 15]. Our technique here is *combinatorial*, based on the *primal-dual method* and Lagrangian relaxation, and our improvement stems from our better notion of proxy costs.

Theorem 8.3. *There is a polynomial time $(5+\varepsilon)$-approximation for the ordered k-median problem, for any constant $\varepsilon > 0$.*

8.1 LP Relaxation and Deterministic Oblivious Rounding

As always, let $\vec{o}$ denote the costs induced by an optimal solution. To avoid trivial settings, assume that $\vec{o}_1^{\downarrow} > 0$. For convenience, we use $\delta = 1$ in the sparsification described in Section 4. So $\text{POS} = \text{POS}_{n,1} := \{\min\{2^s, n\} : s \geq 0\}$. For $\ell \in \text{POS}$, recall that $\text{next}(\ell)$ is the smallest index in POS larger than ℓ if $\ell < n$, and is $n+1$ otherwise. Given a weight vector $w \in \mathbb{R}^n_+$ (with non-increasing coordinates), we sparsify it to $\widetilde{w}$, that is, for every $r \in [n]$, we set $\widetilde{w}_r = w_r$ if $r \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < r < \text{next}(\ell)$, we set $\widetilde{w}_r = w_{\text{next}(\ell)}$. Given a *threshold vector* $\vec{t} \in \mathbb{R}^{\text{POS}}_+$ with non-increasing coordinates, we have the proxy function

$$\text{prox}_{\vec{t}}(\widetilde{w}; v) := \sum_{\ell\in\text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\ell \cdot t_\ell + \sum_{j\in\mathcal{D}} h_{\vec{t}}(\widetilde{w}; v_j)$$

where $v \in \mathbb{R}^{\mathcal{D}}$ and $h_{\vec{t}}(\widetilde{w}; a) := \sum_{\ell\in\text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)(a - t_\ell)^+$.

Since $\vec{o}_1^{\downarrow}$ takes at most n^2 values, we may assume that we know $\rho = \vec{o}_1^{\downarrow}$; so by Lemma 6.9, we may assume that we have $\vec{t}$ that satisfies: $\vec{o}_\ell^{\downarrow} \leq t_\ell \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow}$ for all $\ell \in \text{POS}$ with $\vec{o}_\ell^{\downarrow} \geq \frac{\varepsilon\vec{o}_1^{\downarrow}}{n}$, and $t_\ell = 0$ for all other $\ell \in \text{POS}$. *In this section*, it will be convenient to set $t_\ell = \frac{\varepsilon t_1}{n}$ whenever $t_\ell = 0$. Then, we have $\vec{o}_\ell^{\downarrow} \leq t_\ell \leq (1+\varepsilon)\vec{o}_\ell^{\downarrow} + \frac{\varepsilon t_1}{n}$ for all $\ell \in \text{POS}$, and $\vec{o}_1^{\downarrow} \leq t_1 \leq (1+\varepsilon)\vec{o}_1^{\downarrow}$ (in particular); if these conditions hold then we say that $\vec{t}$ *well-estimates* $\vec{o}^{\downarrow}$. By Lemma 6.10, we can therefore focus on the problem of finding an assignment-cost vector $\vec{c}$ that (approximately) minimizes $\sum_{j\in\mathcal{D}} h_{\vec{t}}(\widetilde{w};\vec{c}_j)$.

Our LP-relaxation is parametrized by $\vec{t}$, and augments the standard k-median LP for this *non-metric* k-median problem with constraints (7) that are crucially exploited in the rounding algorithm. Define $t_{n+1} := 0$. Throughout, we use i to index $\mathcal{F}$ and j to index $\mathcal{D}$.

$$\min \quad \text{CLP}_{\vec{t}}(\widetilde{w}; y) := \sum_{j,i} h_{\vec{t}}(\widetilde{w}; c_{ij}) x_{ij} \tag{OCl-P}$$

$$\text{s.t.} \quad \sum_i x_{ij} \geq 1 \qquad \text{for all } j \tag{4}$$

$$0 \leq x_{ij} \leq y_i \qquad \text{for all } i, j \tag{5}$$

$$\sum_i y_i \leq k. \tag{6}$$

$$\sum_{i:c_{ij}\leq r} y_i \geq 1 \qquad \forall j, r:\ \exists \ell \in \text{POS s.t. } \big|\{k \in \mathcal{D} : c_{jk} \leq r - t_\ell\}\big| > \ell \tag{7}$$

Variables x_{ij} and y_i, and constraints (4)–(6), have the same meaning as in the standard k-median LP. We argue that constraints (7) are satisfied by the optimal solution corresponding to $\vec{o}^{\downarrow}$, whenever $\vec{t}$ well-estimates $\vec{o}^{\downarrow}$. For any j, r, and index $\ell \in \text{POS}$ that gives rise to constraint (7), if no facility is opened in the ball $\{i : c_{ij} \leq r\}$, then all the clients k with $c_{jk} \leq r - t_\ell$ will incur assignment cost (under $\vec{o}^{\downarrow}$) larger than t_ℓ. But there cannot be more than ℓ such clients in this optimal solution since $t_\ell \geq \vec{o}_\ell^{\downarrow}$, and so the choice of ℓ implies that (7) must be satisfied. As discussed in Section 7, our approach to min-max ordered optimization is via a deterministic, weight-oblivious rounding procedure for an LP-relaxation for the ordered optimization problem. The theorem below formalizes this.

Theorem 8.4. *Let $\vec{t}$ be a valid threshold vector that well-estimates $\vec{o}^{\downarrow}$. There is a deterministic, weight-oblivious rounding procedure which given a solution $(\overline{x}, \overline{y})$ satisfying (4)–(7), produces a set $F \subseteq \mathcal{F}$ with $|F| \leq k$ such that the resulting assignment-cost vector $\vec{c}$ satisfies: for* any *sparsified weight vector $\widetilde{w}$, we have $\sum_{j\in\mathcal{D}} h_{44\vec{t}}(\widetilde{w};\vec{c}_j) \leq 44 \cdot \text{CLP}_{\vec{t}}(\widetilde{w};\overline{y}) + 40\sum_{\ell\in\text{POS}} \widetilde{w}_\ell \text{next}(\ell) t_\ell$.*

The theorem implies that if $(\overline{x}, \overline{y})$ is an optimal solution to the analogue of (OCl-P) for min-max ordered k-median, then we obtain an $O(1)$-approximation for min-max ordered k-median. (It is easy to bound $40\sum_{\ell\in\text{POS}} \widetilde{w}_\ell \text{next}(\ell) t_\ell$ by $O(opt)$.)

We remark that Byrka et al. [13] show that a *randomized* rounding procedure of [17] for the standard k-median LP yields a *randomized* oblivious rounding procedure for ordered k-median. However, as noted earlier, this randomized guarantee is insufficient for the purposes of utilizing it for min-max ordered k-median (and consequently min-norm k-clustering).

Proof sketch of Theorem 8.4. Fix a sparsified vector $\widetilde{w}$. This is used *only* in the analysis. Define $\bar{C}_j := \sum_i c_{ij}\bar{x}_{ij}$, and $\mathsf{CLP}_j := \sum_i h_{\vec{t}}(\widetilde{w}; c_{ij})x_{ij}$ for every client j. For a set $S \subseteq \mathcal{F}$, and a vector $v \in \mathbb{R}^F$, we define $v(S) := \sum_{i\in S} v_i$. For any $p \in \mathcal{F} \cup \mathcal{D}$ and $S \subseteq \mathcal{F} \cup \mathcal{D}$, define $c(p, S) := \min_{r\in S} c_{pr}$.

We proceed by initially following the template of the k-median LP-rounding algorithm by Charikar et al. [16], with some subtle but important changes. We cluster clients around nearby centers (which are also clients) as in [16] to ensure that every non-cluster center k is close to some cluster center $j = \mathrm{ctr}(k)$. Let D be the set of cluster centers. For $j \in D$, let F_j be the set of facilities that are nearer to j than to any other cluster center, $\mathrm{nbr}(j)$ be the cluster-center (other than itself) nearest to j, and let $a_j := c_{j\mathrm{nbr}(j)}$. We will eventually ensure that we open a set F of facilities such that $c(j, F) = O(a_j)$ for every $j \in D$. So for a non-cluster center k for which $a_{\mathrm{ctr}(k)} = O(\bar{C}_k)$ we have $c(k, F) = O(\bar{C}_k)$, and this will also imply that $h_{O(1)\cdot\vec{t}}\big(\widetilde{w}; c(k, F)\big) = O(1)\cdot\mathsf{CLP}_k$ So we focus on the non-cluster centers that are "near" their corresponding cluster centers; let N_j (for "near") denote such clients that are "near" center j.

Moving each near non-cluster center k to $\mathrm{ctr}(k)$ yields a consolidated instance, where at each $j \in D$, we have some d_j clients (including j) co-located at j. *Unlike in standard k-median*, the solution induced by $(\bar{x}, \bar{y})$ for the consolidated instance may not have cost at most the LP-objective-value of $(\bar{x}, \bar{y})$, because $\bar{C}_j \le \bar{C}_k$ for $j = \mathrm{ctr}(k)$ does not imply that $\sum_i h_{\vec{t}}(\widetilde{w}; c_{ij})\bar{x}_{ij} \le \sum_i h_{\vec{t}}(\widetilde{w}; c_{ik})\bar{x}_{ik}$; however, we show that an approximate form of this inequality holds, and a good solution to the consolidated instance does translate to a good solution to the original instance.

We now focus on rounding the solution to the consolidated instance. As in [16], we can obtain a more-structured fractional solution to this consolidated instance, where every cluster-center j is served to an extent of $\hat{y}_j = \bar{y}(F_j) \ge 0.5$ by itself, and to an extent of $1 - \hat{y}_j$ by $\mathrm{nbr}(j)$. We now perform another clustering step, where we select some $(j, \mathrm{nbr}(j))$ pairs such that every $k \in D$ that is not part of a pair is close to a some j that belongs to a pair, and $a_j \le a_k$. For standard k-median, it suffices to ensure that: (1) we open at most k facilities, and (2) we open at least one facility in each pair.

However, for the oblivious guarantee, we need to impose more constraints, and this is where we diverge substantially from [16]. Define $t_0 := \infty$ and $\mathrm{next}(0) = 1$. We want to compare the cost of the rounded solution for the consolidated instance to the cost $\sum_{j\in D} d_j h_{\alpha\vec{t}}(\widetilde{w}; a_j)(1 - \hat{y}_j)$ of the above structured fractional solution, where α is a suitable constant. The LP solution can be used to define variables $\hat{q}_j^{(\ell)}$ for all $\ell \in \{0\} \cup \mathrm{POS}$, where $a_j\hat{q}_j^{(\ell)}$ is intended to represent (roughly speaking) $(1-\hat{y}_j)\times\big(\min\{a_j, \alpha t_\ell\} - \alpha t_{\mathrm{next}(\ell)}\big)^+$, so that $\sum_{\ell\in\{0\}\cup\mathrm{POS}} \widetilde{w}_{\mathrm{next}(\ell)} a_j\hat{q}_j^{(\ell)}$ is $O\big(h_{\alpha\vec{t}}(\widetilde{w}; a_j)(1 - \hat{y}_j)\big)$.

Now in addition to properties (1), (2), following the template in Section 7, we also seek to assign each $j \in D$ where a center is not opened to a *single* threshold t_ℓ where $t_\ell = \Omega(a_j)$, $t_{\mathrm{next}(\ell)} \le a_j$, so that: (3) for every $\ell \in \{0\} \cup \mathrm{POS}$, the total $d_j a_j$ cost summed over all $j \in D$ that are not open and assigned to t_ℓ is (roughly speaking) comparable to $\sum_{j\in D} d_j a_j \hat{q}_j^{(\ell)}$. We apply Theorem 7.1 on a suitable system to round $\hat{q}$ to an integral solution (which specifies both the open facilities and the assignment of clients to thresholds) satisfying the above properties. An important property that we need in order to achieve this is, is an upper bound on d_j, and this is the key place where we exploit constraint (7). Properties (1)–(3) will imply that, for a suitable constant α, the resulting assignment-cost vector $\vec{c}$ for the consolidated instance satisfies $\sum_{j\in D} d_j h_{\alpha\vec{t}}(\widetilde{w}; \vec{c}_j)$ is O(cost of fractional solution for consolidated instance). □

9 LOAD BALANCING

In this section, we use our framework to design constant factor approximation algorithms for the minimum-norm load balancing problem. Let us recall the problem. We are given a set J of n jobs, a set of m machines, and for each job j and machine i, the processing times $p_{ij} \ge 0$ required to process j on machine i. We have to output an assignment $\sigma : J \to [m]$ of jobs to machines. The load on machine i due to σ is $\mathsf{load}_\sigma(i) := \sum_{j:\sigma(j)=i} p_{ij}$. Let $\overrightarrow{\mathsf{load}}_\sigma := \{\mathsf{load}_\sigma(i)\}_{i\in[m]}$ denote the load-vector induced by σ.

In the minimum-norm load-balancing problem, one seeks to minimize the norm of the load vector $\overrightarrow{\mathsf{load}}_\sigma$ for a given monotone, symmetric norm. In the special case of *ordered load-balancing* problem, given a non-negative, non-increasing vector $w \in \mathbb{R}^m_+$ (that is, $w_1 \ge w_2 \ge \cdots \ge w_m \ge 0$), one seeks to minimize $\mathrm{cost}\big(w; \overrightarrow{\mathsf{load}}_\sigma\big) := w^T\overrightarrow{\mathsf{load}}^{\downarrow}_\sigma = \sum_{i=1}^m w_i\overrightarrow{\mathsf{load}}^{\downarrow}_\sigma(i)$. In the Top-$\ell$ *load balancing problem*, one seeks to minimize the sum of the ℓ largest loads in $\overrightarrow{\mathsf{load}}_\sigma$.

Theorem 9.1. *Given any monotone, symmetric norm f on $\mathbb{R}^m$ with a κ-approximate ball-optimization oracle for f (see (1)), and for any $\varepsilon > 0$, there is a $38\kappa(1 + 5\varepsilon)$-approximation algorithm for the problem of finding an assignment $\sigma : J \to [m]$ which minimizes $f\big(\overrightarrow{\mathsf{load}}_\sigma\big)$. The running time of the algorithm is $(\frac{m}{\varepsilon})^{O(1/\varepsilon)}$.*

We have not optimized the constants in the above theorem. For the special case of ordered load balancing, we can get much better results.

Theorem 9.2. *There is a polynomial time $(2 + \varepsilon)$-approximation for the ordered load balancing problem, for any constant $\varepsilon > 0$.*

As shown by the reduction in Section 5, the key component needed to tackle the norm-minimization problem is an algorithm for the *min-max multi-ordered load-balancing problem*, wherein we are given *multiple* non-increasing weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}^m_+$, and our goal is to find an assignment $\sigma : J \to [m]$ to minimize $\max_{r\in[N]} \mathrm{cost}(w^{(r)}; \overrightarrow{\mathsf{load}}_\sigma)$.

Theorem 9.3. *Given any non-increasing weight vectors $w^{(1)}, \ldots, w^{(N)} \in \mathbb{R}^m_+$, we can find $38(1 + \delta)$-approximation algorithm to the min-max ordered load balancing problem of finding an assignment $\sigma : J \to [m]$ minimizing $\max_{r\in[N]} \mathrm{cost}(w^{(r)}; \overrightarrow{\mathsf{load}}_\sigma)$. The algorithm runs in time $\mathrm{poly}\big(\text{input size}, m^{O(1/\delta)}\big)$.*

9.1 Linear Programming Relaxation

We begin by restating some definitions from Section 6 in the load balancing setting. As usual, $\vec{o}$ will denote the load-vector induced by an optimal assignment for the problem under consideration. Recall that $\mathrm{POS} = \mathrm{POS}_{m,\delta} := \{\min\{\lceil(1 + \delta)^s\rceil, m\} : s \ge 0\}$ is the sparse set of $O(\log m/\delta)$ indices. For $\ell \in \mathrm{POS}$, $\mathrm{next}(\ell)$ is the smallest index in POS larger than ℓ if $\ell < m$, and is $m + 1$ otherwise. Given POS,

recall the sparsified weight vector $\widetilde{w}$ of any weight vector w; every $i \in [m]$, we set $\widetilde{w}_i = w_i$ if $i \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$, we set $\widetilde{w}_i = w_{\text{next}(\ell)}$.

Given a valid threshold vector $\vec{t} \in \mathbb{R}^{\text{POS}}$ (i.e., t_ℓ is non-increasing in ℓ) we move from $\text{cost}(\widetilde{w}; \overrightarrow{\text{load}}_\sigma)$ to the proxy

$$\text{prox}_{\vec{t}}(\widetilde{w}; \overrightarrow{\text{load}}_\sigma) := \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)\ell \cdot t_\ell + \sum_{i=1}^{m} h_{\vec{t}}\big(\widetilde{w}; \text{load}_\sigma(i)\big)$$

where $h_{\vec{t}}(\widetilde{w}; a) := \sum_{\ell \in \text{POS}} \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big)(a - t_\ell)^+$.

Again, from Section 6, we know that for the right choice of $\vec{t}$, this change of objective does not incur much loss, and so our goal is to find $\sigma : J \to [m]$ that approximately minimizes $\sum_{i=1}^{m} h_{\vec{t}}\big(\widetilde{w}; \text{load}_\sigma(i)\big)$ (see Lemma 6.10). We now describe the LP relaxation to minimize the proxy-cost. Our LP is parametrized by the vector $\vec{t}$.

We use variables x_{ij} to denote if job j is assigned to machine i. Now for every i, j, *and every* $\ell \in \text{POS}$, we have variables $z_{ij}^{(\ell)}, y_{ij}^{(\ell)}$ to denote respectively the portions of job j that lie "below" and "above" the t_ℓ threshold on machine i. More precisely, given an *integral* assignment σ and an ordering of the jobs in $\sigma^{-1}(i)$, $z_{ij}^{(\ell)}$ denotes the fraction of j that contributes to the load in the interval $[0, t_\ell)$ on machine i, and $y_{ij}^{(\ell)}$ denotes the fraction of j that contributes to the load interval $[t_\ell, \infty)$. Thus, for every ℓ, we have $x_{ij} = z_{ij}^{(\ell)} + y_{ij}^{(\ell)}$, and $\sum_j p_{ij} y_{ij}^{(\ell)}$ represents $\big(\text{load}_\sigma(i) - t_\ell\big)^+$. Throughout i indexes the set $[m]$ of machines, and j indexes the job-set J. To keep notation simple, define $z_{ij}^{(m+1)} = 0$ for all i, j.

$$\min \quad \text{LP}_{\vec{t}}(\widetilde{w}; x, y, z) := \sum_i \sum_{\ell \in \text{POS}} \sum_j \big(\widetilde{w}_\ell - \widetilde{w}_{\text{next}(\ell)}\big) p_{ij} y_{ij}^{(\ell)} \tag{8}$$

$$\text{s.t.} \quad \sum_i x_{ij} = 1 \qquad \forall j \tag{9}$$

$$x_{ij} = z_{ij}^{(\ell)} + y_{ij}^{(\ell)} \qquad \forall i, j, \ \forall \ell \in \text{POS} \tag{10}$$

$$z_{ij}^{\text{next}(\ell)} \le z_{ij}^{(\ell)} \qquad \forall i, j, \ \forall \ell \in \text{POS} \tag{11}$$

$$\sum_j p_{ij}\big(z_{ij}^{(\ell)} - z_{ij}^{(\text{next}(\ell))}\big) \le t_\ell - t_{\text{next}(\ell)} \qquad \forall i, \ \forall \ell \in \text{POS} \tag{12}$$

$$p_{ij} y_{ij}^{(\ell)} \ge \big(p_{ij} - t_\ell\big) x_{ij} \qquad \forall i, j, \ \forall \ell \in \text{POS} \tag{13}$$

$$x_{ij}, z_{ij}^{(\ell)}, y_{ij}^{(\ell)} \ge 0 \qquad \forall i, j, \ \forall \ell \in \text{POS}.$$

Lemma 9.4. *For any valid threshold vector $\vec{t}$ and any integral assignment σ, the value of the LP is at most $\sum_{i=1}^{m} h_{\vec{t}}\,(\widetilde{w}; \text{load}_\sigma(i))$.*

As discussed in Section 7, we need a *deterministic, weight-oblivious* rounding algorithm. The main technical contribution of this section is precisely such a rounding procedure.

Theorem 9.5. *Let $\vec{t}$ be a valid threshold vector such that every t_ℓ is either a power of 2 or 0. There is a* deterministic *algorithm which takes any solution (x, y, z) satisfying constraints (9)-(13), and produces an assignment $\widetilde{\sigma} : J \to [m]$ such that, for any sparsified weight vector $\widetilde{w}$, we have that*

$$\sum_{i=1}^{m} h_{10\vec{t}}\,\big(\widetilde{w}; \text{load}_{\widetilde{\sigma}}(i)\big) \le 2 \cdot \text{LP}_{\vec{t}}\,(\widetilde{w}; x, y, z) + 4 \sum_{\ell \in \text{POS}} \widetilde{w}_\ell t_\ell$$

REFERENCES

[1] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. 2017. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In *Proc., FOCS.*
[2] S. Alamdari and D. B. Shmoys. 2017. A bicriteria approximation algorithm for the k-center and k-median problems. In *Proc., WAOA.*
[3] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. 1998. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling* 1, 1 (1998), 55–66.
[4] A. Andoni, H. Nguyen, A. Nikolov, I. Razenshteyn, and E. Waingarten. 2017. Approximate near neighbors for general symmetric norms. In *Proc., STOC.*
[5] A. Aouad and D. Segev. 2018. The ordered k-median problem: surrogate models and approximation algorithms. *Math. Programming* (2018), 1–29.
[6] B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan, and J. S. Vitter. 1995. Load balancing in the L_p norm. In *Proc., FOCS.*
[7] Y. Azar and A. Epstein. 2005. Convex programming for scheduling unrelated parallel machines. In *Proc., STOC.*
[8] Y. Azar, L. Epstein, Y. Richter, and G. J. Woeginger. 2004. All-norm approximation algorithms. *J. Algorithms* 52, 2 (2004), 120–133.
[9] N. Bansal, R. Khandekar, and V. Nagarajan. 2009. Additive guarantees for degree-bounded directed network design. *SICOMP* 39, 4 (2009), 1413–1431.
[10] R. Bhatia. 2013. *Matrix analysis.* Vol. 169. Springer.
[11] J. Błasiok, V. Braverman, S. Chestnut, R. Krauthgamer, and L. Yang. 2017. Streaming symmetric norms via measure concentration. In *Proc., STOC.*
[12] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. 2014. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proc., SODA.*
[13] J. Byrka, K. Sornat, and J. Spoerhase. 2018. Constant-factor approximation for ordered k-median. In *Proc., STOC.*
[14] R. Carr and S. Vempala. 2002. Randomized metarounding. *Random Structures Algorithms* 20, 3 (2002), 343–352.
[15] D. Chakrabarty and C. Swamy. 2018. Interpolating between k-median and k-center: Approximation Algorithms for Ordered k-median. In *Proc., ICALP.*
[16] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. 2002. A constant-factor approximation algorithm for the k-median problem. *J. Comput. System Sci.* 65, 1 (2002), 129–149.
[17] M. Charikar and S. Li. 2012. A dependent LP-rounding approach for the k-median problem. In *Proc., ICALP.*
[18] A. Goel and A. Meyerson. 2006. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica* 44, 4 (2006), 301–323.
[19] D. Golovin, A. Gupta, A. Kumar, and K. Tangwongsan. 2008. All-norms and all-ℓ_p-norms approximation algorithms. In *Proc., FSTTCS.*
[20] T. F. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293 – 306.
[21] A. Gupta and K. Tangwongsan. 2008. Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554* (2008).
[22] G. H. Hardy, J. E. Littlewood, and G. Pólya. 1934. *Inequalities.* Camb. Univ Press.
[23] D. S. Hochbaum and D. B. Shmoys. 1985. A Best Possible Heuristic for the k-Center Problem. *Math. Oper. Res.* 10, 2 (1985), 180–184.
[24] K. Jain and V. V. Vazirani. 2001. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM (JACM)* 48, 2 (2001), 274–296.
[25] K. Jansen and L. Rohwedder. 2017. On the configuration-LP of the restricted assignment problem. In *Proc., SODA.*
[26] I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, and A. Shalita. 2017. Social hash partitioner: a scalable distributed hypergraph partitioner. *Proc., Very Large Databases (VLDB) Endowment* 10, 11 (2017), 1418–1429.
[27] A. Kumar and J. Kleinberg. 2006. Fairness measures for resource allocation. *SIAM Journal on Computing (SICOMP)* 36, 3 (2006), 657–680.
[28] V. S. Kumar, Madhav V Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. 2009. A unified approach to scheduling on unrelated parallel machines. *Journal of the ACM (JACM)* 56, 5 (2009), 28.
[29] G. Laporte, S. Nickel, and F. S. da Gama. 2015. *Location Science.* Springer.
[30] L. C. Lau, R. Ravi, and M. Singh. 2011. *Iterative methods in combinatorial optimization.* Vol. 46. Cambridge University Press.
[31] J. Lenstra, D. Shmoys, and E. Tardos. 1990. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming* 46, 1-3 (1990), 259–271.
[32] S. Li and O. Svensson. 2016. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing (SICOMP)* 45, 2 (2016), 530–547.
[33] K. Makarychev and M. Sviridenko. 2014. Solving optimization problems with diseconomies of scale via decoupling. In *Proc., FOCS.*
[34] D. B. Shmoys and E. Tardos. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical programming* 62, 1-3 (1993), 461–474.
[35] O. Svensson. 2012. Santa Claus schedules jobs on unrelated machines. *SIAM J. Comput.* 41, 5 (2012), 1318–1341.
[36] A. Tamir. 2001. The k-centrum multi-facility location problem. *Discrete Applied Mathematics* 109, 3 (2001), 293–307.