

Approximation Algorithms for Minimum Norm and Ordered Optimization Problems

Deeparnab Chakrabarty*

Chaitanya Swamy†

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. These functions are versatile and model a wide range 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.

- 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.
- In load balancing with unrelated machines, we give a 2-approximation for the problem of finding an assignment minimizing the sum of the largest ℓ loads, for any ℓ . We give a $(2+\varepsilon)$ -approximation for the so-called ordered load-balancing problem.
- For k -clustering, we give a $(5 + \varepsilon)$ -approximation for the ordered k -median problem significantly improving the constant factor approximations from Byrka, Sornat, and Spoerhase (STOC 2018) and Chakrabarty and Swamy (ICALP 2018).
- Our techniques also imply $O(1)$ approximations to the best *simultaneous optimization factor* for any instance of the unrelated machine load-balancing and the k -clustering setting. To our knowledge, these are the first *positive* simultaneous optimization results in these settings.

At a technical level, our main insight is connecting minimum-norm optimization to what we call min-max ordered optimization. The main ingredient in solving the min-max ordered optimization is *deterministic, oblivious rounding* of linear programming relaxations for load-balancing and clustering, and this technique may be of independent interest.

*Dartmouth College, Email: deeparnab@dartmouth.edu

†University of Waterloo, Email: cswamy@uwaterloo.edu

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 [34, 40, 20, 41, 15, 28]. 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 [26, 23, 18, 17, 27, 35, 12]. However, the techniques to study the sum-versions and max-versions 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 $f(\vec{\text{load}})$ where $\vec{\text{load}}$ 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 8.1 and Theorem 9.1). 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 (B-O) 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 point to the reader to [10, 11, 4] for a more comprehensive list of examples.

- **ℓ_p -norms.** Perhaps the most famous examples are ℓ_p norms where $f(\vec{v}) := (\sum_{i=1}^n v_i^p)^{1/p}$ 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 [34, 40] 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 [31, 37]. For the k -clustering setting, the $p = \{1, 2, \infty\}$ norms have been extensively studied over the years [23, 26, 18, 17, 27, 12, 1]. One can also derive an $O(1)$ -approximation for general ℓ_p -norms using most of the algorithms¹ 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

¹We 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 [24]. 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 [18] and Jain-Vazirani [27] 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.

by a non-increasing, non-negative vector $w \in \mathbb{R}_+^n$ with $w_1 \geq w_2 \geq \dots \geq w_n \geq 0$. 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 give a 2-approximation for the Top- ℓ -load balancing, and a $(2 + \varepsilon)$ -approximation for ordered load balancing (Theorem 8.2 and Theorem 8.3). Note that the case of $\ell = 1$ for Top- ℓ -load balancing corresponds to makespan minimization for which beating factor 2 is an 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 [16] 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 9.3).

- **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 $w^{(1)}, \dots, w^{(N)} \in \mathbb{R}_+^n$, 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 $g : \mathbb{R}^n \rightarrow \mathbb{R}_+$, 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 $g : \mathbb{R}^k \rightarrow \mathbb{R}_+$, 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 the clustering setting (this is partly motivated by the stochastic fanout model described in [29] 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 (apply part (d) operation above) and thus we can get a constant factor approximation for it. In fact, the *expected maximum* for the i.i.d case is an ordered-norm, and so we can get 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 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 [21] call a solution \vec{v} an α -approximate solution if $C(\vec{v}/\alpha) \leq \text{opt}$ where \vec{v} is the induced cost vector, \vec{v}/α is the coordinate-wise scaled vector, and $\text{opt} = \min_{\vec{w}} C(\vec{w})$. It is not hard to see² 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.

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

Connections and implications for simultaneous/fair optimization. In the minimum-norm optimization problem, we are given a fixed norm function f and we wish to find a solution minimizing $f(\vec{v})$ where \vec{v} is the cost-vector induced by the solution. In *simultaneous optimization* [30, 21], the goal is to find a solution \vec{v} , which *simultaneously* approximates all norms/convex functions. Such solutions are desirable as they possess certain fairness properties. More precisely, the goal is to find a solution inducing a cost vector \vec{v} which is simultaneous α -approximate, that is, $g(\vec{v}) \leq \alpha \cdot \text{opt}(g)$ for *all* monotone, symmetric norms $g : \mathbb{R}_+^n \rightarrow \mathbb{R}_+$, where $\text{opt}(g) := \min_{\vec{w}} g(\vec{w})$.

Simultaneous optimization is clearly a much stronger goal than what we are shooting for, in that, if one can find a solution which is simultaneous α -approximate, 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 the jobs have fixed load but can be allocated only on a subset of machines, one can always achieve [3, 8, 21] a simultaneous 2-approximate solution. Unfortunately, for unrelated (even related) machines [8] and k -clustering [30], 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, 21] are not particularly helpful when trying to optimize a *fixed* norm, which is the main focus in our paper.

Nevertheless, the techniques we develop give $O(1)$ approximations to the best simultaneous approximation factor possible in any instance of unrelated machines load-balancing and k -clustering (Theorem 10.5). Fix an unrelated machines load balancing instance \mathcal{I} . Let $\alpha_{\mathcal{I}}^*$ be the smallest α for which there is a solution to \mathcal{I} which is simultaneous α -approximate. Note that $\alpha_{\mathcal{I}}^*$ could be a constant for a nice instance \mathcal{I} ; the impossibility result mentioned above states $\alpha_{\mathcal{I}}^*$ can't be a constant for *all* instances. It is natural, and important, to ask whether for such nice instances can one get constant factor simultaneous approximate solutions? We answer this in the affirmative. We give an algorithm which, for any instance \mathcal{I} , returns a solution inducing a load vector \vec{v}' such that $g(\vec{v}') \leq O(\alpha_{\mathcal{I}}^*) \cdot \text{opt}(g)$ for all monotone, symmetric norms simultaneously. We can also obtain a similar result for the k -clustering setting. 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 an artifact 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 books [38, 32]); 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 [42] 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 [16] 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 [21]. 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. [31] and Makarychev and Sviridenko [37]. 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 [30] and Golovin et al. [22] 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 instead of k .

2 Technical overview and organization

We use this section to give an overview of the various technical ideas in this paper and point out the reader to where more details can be found.

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 \vec{v}^* could return a solution \vec{v} such that for every coordinate $\vec{v}_j \leq \rho \vec{v}_j^*$, 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 $\max_{w \in \mathcal{C}} \text{cost}(w; x)$ (Lemma 5.2) where \mathcal{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 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 $\text{POS} := \{1, 2, 4, 8, \dots\}$ and define a \tilde{w} -vector with $\tilde{w}_i = w_i$ if $i \in \text{POS}$, and $\tilde{w}_i = w_\ell$ where ℓ is the nearest power of 2 larger than i , then it is not too hard to see $\text{cost}(\tilde{w}; x) \leq \text{cost}(w; x) \leq 2\text{cost}(\tilde{w}; x)$. 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 the ordered norms, one can just focus on weight vectors which have breakpoints in *fixed* locations *independent* of what the weight vectors are. Note that other kinds of sparsification which round every coordinate of a weight vector to the nearest power of $(1 + \delta)$ don't have this *weight-independence* in the positions of breakpoints. This fixedness of the locations (and the fact that there are only logarithmically many of them) allows us to form a polynomial sized 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 multiplicative $(1 \pm \delta)$. In particular, this helps us bypass the problem of “infinitely many vectors” in \mathcal{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 the paper of Aouad and Segev [5], and then in different forms in Byrka et al [13] and our earlier paper [16].

The idea of this proxy cost is also simple. Suppose we knew what the ℓ th largest load would be in the optimal solution – suppose it was ρ . Then the Top- ℓ load can be written as $\ell \cdot \rho + \sum_{i: \text{all machines}} (\text{load}(i) - \rho)^+$, where we use $(z)^+ := \max(z, 0)$. 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 (see Claim 6.4). In particular, if we have the guesses of the ℓ th largest loads for all ℓ , then we could write the proxy cost of $\text{cost}(w; \vec{v})$. However, guessing n of the ρ_ℓ ’s would be infeasible. This is where the sparsification idea described above comes handy again. Since the only relevant positions of w to define \tilde{w} are the ones in POS, one just needs to guess approximations for ρ_ℓ ’s only in these positions to define the proxy function. And this again can be done in polynomial time. Once again, what is key is that positions are *independent* of the particular weight vector. This is *key for min-max ordered optimization*. Even though 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 (Sections 7 to 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 [16] 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 costs were subtly different but within $O(1)$ -factors). For load-balancing, the usual LP has a bad gap, and one needs to add additional constraints. After this, however, we can indeed show the LP has an integrality gap of ≤ 2 (this is established in Section 8.3).

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 [16] 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 8 and for k -clustering in Section 9. After reading Section 7, the sections on load balancing and clustering can be read in any order.

Extensions: connections to simultaneous optimization (Section 10). We end the paper by showing

the power of deterministic, weight-oblivious rounding to give constant factor approximations to instance-optimal algorithms for simultaneous optimization. The key idea stems from the result of Goel and Meyerson [21], which itself stems from the majorization theory of Hardy, Littlewood, and Polya [25], that if we want to simultaneously optimize all monotone, symmetric, norms, then it suffices to simultaneously optimize all the Top- ℓ norms. If the best simultaneous optimization for a given instance is $\alpha_{\mathcal{I}}^*$, then one can cast this as a multi-budgeted ordered optimization problem where we need to find a solution where the ordered-norm with respect to the r th weight vector is at most some budget B_r . Once again, if we have a good deterministic, weight-oblivious rounding algorithm for the LP relaxation, the multi-budgeted ordered optimization problem can also be easily solved. As a result, for any load-balancing and k -clustering instance, we get $O(1)$ -approximations to the best simultaneous optimization factor possible for that instance.

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 $\vec{\text{load}}$, or $\vec{\text{load}}_\sigma$ if σ is the assignment of jobs. In k -clustering, 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 $\{1, \dots, n\}$. 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 $\vec{v}_i^\downarrow = \vec{v}_{\pi(i)}$, where π is a permutation of $[n]$ such that $\vec{v}_{\pi(1)} \geq \vec{v}_{\pi(2)} \geq \dots \vec{v}_{\pi(n)}$.

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 $w_1 \geq w_2 \geq \dots \geq w_n \geq 0$. We use \tilde{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 $\text{cost}(w; \vec{v}) := \sum_{i=1}^n w_i \vec{v}_i^\downarrow$. This is the w -ordered norm, or simply ordered norm of \vec{v} . We denote the special case of when w is a $\{0, 1\}$ vector *Top- ℓ optimization*. That is, if $w_1 = \dots = w_\ell = 1$ and $w_i = 0$ otherwise, the problem asks to find a solution \vec{v} minimizing the sum of the ℓ largest entries. We use the notation $\text{cost}(\ell; \vec{v})$ 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 $w^{(1)}, \dots, w^{(N)} \in \mathbb{R}_+^n$, 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 $B_1, \dots, B_N \geq 0$. 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 the connections with simultaneous optimization [30, 21]; we discuss these connections more in Section 10.

Minimum norm optimization. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 value³. In *minimum norm optimization* problem we are given a monotone, symmetric

³Symmetric norms mayn't be monotone. For instance, consider the set $C \subseteq \mathbb{R}^2$, which is the convex hull of the points $\{(1, 1), (-1, -1), (0, 0.5), (0.5, 0), (0, -0.5), (-0.5, 0)\}$, and define $f(x)$ to be the smallest λ such that $x/\lambda \in C$. It is not hard to see that f is a symmetric norm over \mathbb{R}^2 , $f(0, 0.5) = 1$, but $f(0.5, 0.5) \leq 0.5$.

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 $p_{ij} \geq 0$ of job j on machine i . The solution to the problem is an assignment σ of jobs to machines. This induces a load $\text{load}_\sigma(i) := \sum_{j:\sigma(j)=i} p_{ij}$ on each machine. The vector $\vec{\text{load}}_\sigma$ of these loads is the cost-vector associated with the solution σ . Thus, the min-norm load balancing problem asks to find σ minimizing $f(\vec{\text{load}}_\sigma)$.

In the k -clustering setting, we have a metric space $(\mathcal{D}, \{c_{ij}\}_{i,j \in \mathcal{D}})$, and an integer $k \geq 0$. The 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 to the nearest open facility. The min-norm k -clustering problem asks to find the set F of facilities which 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 $\tilde{w} \in \mathbb{R}^n$ (with non-increasing coordinates) having $O(\log n / \delta)$ distinct weight values, such that for any vector \vec{v} , we have $\text{cost}(\tilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v}) \leq (1 + \delta) \text{cost}(\tilde{w}; \vec{v})$. Moreover, an important property we ensure is that the breakpoints of \tilde{w} —i.e., the indices where $\tilde{w}_i > \tilde{w}_{i+1}$ —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 $\tilde{w}_i = w_i$ 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 $\tilde{w}_i = w_{\min\{2^s, n\}} = \tilde{w}_{\min\{2^s, n\}}$. Note that $\tilde{w} \leq w$ coordinate wise, and $\tilde{w}_1 \geq \tilde{w}_2 \geq \dots \tilde{w}_n$.

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

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

Proof. Since $\tilde{w} \leq w$, it is immediate that $\text{cost}(\tilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v})$. The other inequality follows from a charging argument. Note that for any $s \geq 2$, we have $(\min\{2^s, n\} - 2^{s-1}) \leq 2(\min\{2^{s-1}, n\} - 2^{s-2})$; 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 $\text{cost}(\tilde{w}; \vec{v})$ from the indices $i \in \{2^{s-2} + 1, \dots, \min\{2^{s-1}, n\}\}$. The remaining cost $w_1 \vec{v}_1^\downarrow + w_2 \vec{v}_2^\downarrow$ is at most $2\tilde{w}_1 \vec{v}_1^\downarrow$. \square

For the refined sparsification that only loses a $(1 + \delta)$ -factor, we consider positions that are powers of $(1 + \delta)$. Let $\text{POS}_{n,\delta} := \{\min\{[(1 + \delta)^s], n\} : s \geq 0\}$. (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 $\tilde{w}_i = w_i$ if $i \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$ (note that $\ell < n$), set $\tilde{w}_i = w_{\text{next}(\ell)} = \tilde{w}_{\text{next}(\ell)}$. The following is a generalization of [Claim 4.1](#).

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

Not to detract the reader, we defer the proof of [Lemma 4.2](#) to [Appendix A](#). We once again stress that the, perhaps more natural, way of geometric bucketing (which is indeed used by [\[5, 13, 16\]](#)) where one ignores

small w_i s 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 \tilde{w} all lie in POS. As noted earlier, this non-dependence on w is extremely crucial for us.

5 Reducing minimum 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}_+$, 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 $\text{opt} = f(\vec{o})$.

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 $\mathbb{B}_+(f) := \{x \in \mathbb{R}_+^n : f(x) \leq 1\}$.

Oracle \mathcal{A} takes input $c \in \mathbb{R}_+^n$ returns a κ -approximation to $B_{\text{opt}}(c) := \max\{c^\top x : x \in \mathbb{B}_+(f)\}$ (B-O)
That is, \mathcal{A} returns $\hat{x} \in \mathbb{B}_+(f)$ such that $c^\top \hat{x} \geq B_{\text{opt}}(c)/\kappa$

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 $f(y) - f(x) \geq d^\top(y - x)$ 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. The proof can be found in [Appendix B](#).

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

- (i) *If d is a subgradient of f at x , then $f(x) = d^\top x$ and $f(y) \geq d^\top y$ 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 $\pi : [n] \rightarrow [n]$, the vector $d^{(\pi)} := \{d_{\pi(i)}\}_{i \in [n]}$ is a subgradient of f at $x^{(\pi)}$.*

Motivated by the above lemma, we define the following set of non-increasing subgradients over points on the unit norm-ball. This set is possibly infinite.

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

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. This establishes the reduction, however it is inefficient.

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

Proof. We first argue that $f(x) \leq \max_{w \in \mathcal{C}} \text{cost}(w; x)$. 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 $\hat{d} = d^\downarrow$ is a subgradient of f at x^\downarrow . By part (i), \hat{d} is also a subgradient of f at $x^\downarrow/f(x^\downarrow) \in \mathbb{B}_+(f)$. So $\hat{d} \in \mathcal{C}$. Also, $f(x) = \text{cost}(\hat{d}; x)$ (by part (iii)), and so $f(x) \leq \max_{w \in \mathcal{C}} \text{cost}(w; x)$.

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

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 \tilde{w} is “close” to a subgradient in \mathcal{C} , and this is where [\(B-O\)](#) is used. First note that $d \in \mathcal{C}$ iff⁴ $B_{\text{opt}}(d) = 1$. Thus to check if \tilde{w} is “close” to a subgradient in \mathcal{C} , it suffices to (approximately) solve for $B_{\text{opt}}(\tilde{w})$ 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 $\tilde{o}_1^\downarrow = 0$. If $\tilde{o}_1^\downarrow > 0$ (so $\text{opt} > 0$), then $\tilde{o}_1^\downarrow \geq 1$ (assuming integer data), and we can compute an estimate hi such that $\tilde{o}_1^\downarrow \leq \text{hi}$. In the sequel, assume that $\tilde{o}_1^\downarrow \geq 1$.
- (A2) We have bounds $\text{lb}, \text{ub} > 0$ such that $\text{lb} \leq \text{opt} \leq \text{ub}$. Then [\(A1\)](#) and [Lemma 5.1](#) (i) imply that $d_1 \leq \text{ub}$ 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 $\tilde{w} \in \mathbb{R}^n$ given by $\tilde{w}_i = w_i$ if $i \in \text{POS}$; and $\tilde{w}_i = w_{\text{next}(\ell)}$ otherwise, where $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$. Since \tilde{w} is completely specified by specifying the positions in POS , we define the $|\text{POS}|$ -dimensional vector $u := \{\tilde{w}_\ell\}_{\ell \in \text{POS}}$. We identify \tilde{w} with $u \in \mathbb{R}_+^{\text{POS}}$ and say that \tilde{w} is the *expansion* of u .

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

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

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

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 $\{0, \dots, N\}$.*

The following theorem establishes the reduction from the minimum norm problem to min-max ordered optimization. The proof idea is as sketched above; we defer the details of the proof to [Appendix B](#).

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

- (i) $\max_{w \in \mathcal{W}} \text{cost}(w; \vec{v}) \leq \max\left\{ \kappa(1 + \varepsilon)f(\vec{v}), \frac{\text{lb}}{n \cdot \text{hi}} \sum_{i \in [n]} \vec{v}_i \right\}$, (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 $f(\vec{v}) \leq \gamma \kappa(1 + 3\varepsilon) \text{opt}$.

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

⁴If $d \in \mathcal{C}$ 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 $B_{\text{opt}}(d) = 1$, then we have $d^T z = 1$ for some $f(z) \leq 1$ implying $f(z) + d^T (y - z) \leq d^T y$ for any y . If the LHS is $> f(y)$, then we would get $d^T (y / f(y)) > 1$ contradicting $B_{\text{opt}}(d) = 1$.

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 do 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 $\text{cost}(\ell; \vec{v}) = \min_{\rho \in \mathbb{R}} \ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{v}_i)$.

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

Proof. We have $\text{cost}(\ell; \vec{v}) = \sum_{i=1}^\ell \vec{v}_i^\downarrow \leq \ell \cdot \rho + \sum_{i=1}^\ell (\vec{v}_i^\downarrow - \rho)^+ \leq \ell \cdot \rho + \sum_{i=1}^n (\vec{v}_i^\downarrow - \rho)^+$. \square

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 $\ell \cdot \rho + \sum_{i=1}^n h_\rho(\vec{o}_i) \leq (1+\varepsilon)\text{cost}(\ell; \vec{o})$.*

Proof. We have $\sum_{i=1}^n (\vec{o}_i - \rho)^+ \leq \sum_{i=1}^\ell (\vec{o}_i^\downarrow - \vec{o}_\ell^\downarrow)$. Since $\rho \leq (1+\varepsilon)\vec{o}_\ell^\downarrow$, we have $\ell \cdot \rho + \sum_{i=1}^n (\vec{o}_i - \rho)^+ \leq (1+\varepsilon) \sum_{i=1}^\ell (\vec{o}_i^\downarrow + (\vec{o}_i^\downarrow - \vec{o}_\ell^\downarrow)) = (1+\varepsilon)\text{cost}(\ell; \vec{o})$. \square

The above claims indicate that if we obtain a good estimate ρ of \vec{o}_ℓ^\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 $\rho, x \leq y$; (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, $h_{\rho_1+\rho_2}(x+y) = x - \rho_1 + y - \rho_2 \leq (x - \rho_1)^+ + (y - \rho_2)^+$. \square

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 [\[13, 16\]](#). This strengthening (and its extension to ordered optimization) forms the basis of our significantly improved approximation guarantees of $(5 + \varepsilon)$ for ordered k -median ([Section 9.3](#)), which improves upon the prior-best guarantees for *both* ℓ -centrum and ordered k -median [\[16\]](#). Furthermore, this proxy function also leads to (essentially) a 2-approximation for Top- ℓ load balancing and ordered load balancing ([Section 8.3](#)).

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 of these thresholds. 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 $\delta, \varepsilon > 0$ be parameters. Let $\text{POS} = \text{POS}_{n,\delta} := \{\min\{\lceil (1 + \delta)^s \rceil, n\} : s \geq 0\}$. 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 $\tilde{w} \in \mathbb{R}^n$ by setting $\tilde{w}_i = w_i$ if $i \in \text{POS}$, and $\tilde{w}_i = w_{\text{next}(\ell)}$ 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}_ℓ^\downarrow for all $\ell \in \text{POS}$ within a multiplicative $(1 + \varepsilon)$ factor, and rewriting $\text{cost}(\tilde{w}; \vec{v})$ 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.

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

$$\text{where, } h_{\vec{t}}(\tilde{w}; a) := \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) h_{t_\ell}(a) \quad (2)$$

Note that the above proxy functions are strict generalizations of the case of the Top- ℓ optimization in which case $\text{POS} = \{\ell\}$, and the weights are 1 till ℓ and 0 afterwards.

Throughout the rest of this section, we work with the sparsified weight vector \tilde{w} . Observe that $h_{\vec{t}}(\tilde{w}; x)$ is a continuous, piecewise-linear, non-decreasing function of x . Our proxy for $\text{cost}(\tilde{w}; \vec{v})$ will be the function $\text{prox}_{\vec{t}}(\tilde{w}; \vec{v})$ for a suitably chosen threshold vector \vec{t} . To explain the above definition, notice that (1) 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 $\text{cost}(\tilde{w}; v)$ as a linear combination of $\text{cost}(\ell; v)$ terms.

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

Proof. We have

$$\text{cost}(\tilde{w}; \vec{v}) = \sum_{i=1}^n \tilde{w}_i \vec{v}_i^\downarrow = \sum_{i=1}^n \sum_{\ell=i}^n (\tilde{w}_\ell - \tilde{w}_{\ell+1}) \vec{v}_i^\downarrow = \sum_{\ell=1}^n (\tilde{w}_\ell - \tilde{w}_{\ell+1}) \sum_{i=1}^{\ell} \vec{v}_i^\downarrow = \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \text{cost}(\ell; \vec{v}).$$

The last equality follows since $\tilde{w}_\ell = \tilde{w}_{\ell+1}$ for all $\ell \in [n] \setminus \text{POS}$, and $\tilde{w}_{\ell+1} = \tilde{w}_{\text{next}(\ell)}$ for $\ell \in \text{POS}$. \square

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 $\text{cost}(\tilde{w}; \vec{v}) \leq \text{prox}_{\vec{t}}(\tilde{w}; \vec{v})$.

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

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, $\text{prox}_{\vec{t}}(\tilde{w}; \vec{o}) \leq (1 + \varepsilon) \text{cost}(\tilde{w}; \vec{o})$.

Proof. We have $\text{prox}_{\vec{t}}(\tilde{w}; \vec{o}) = \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\ell \cdot t_\ell + \sum_{i=1}^n h_{t_\ell}(\vec{o}_i^\downarrow))$. 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. \square

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, whose proof is in Appendix C, 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 $|\text{prox}_{\vec{t}}(\vec{w}; \vec{v}) - \text{prox}_{\vec{t'}}(\vec{w}; \vec{v})| \leq n\tilde{w}_1\Delta$.

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,

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

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 $\text{prox}_{\vec{t}}(\vec{w}; \vec{o}^\downarrow) \leq \text{prox}_{\vec{t'}}(\vec{w}; \vec{o}^\downarrow) + \varepsilon\tilde{w}_1\vec{o}_1^\downarrow$. The threshold vector $\vec{t'}$ satisfies the conditions of Claim 6.6, so $\text{prox}_{\vec{t'}}(\vec{w}; \vec{o}) \leq (1 + \varepsilon)\text{cost}(\vec{w}; \vec{o})$. So $\text{prox}_{\vec{t}}(\vec{w}; \vec{o}) \leq (1 + 2\varepsilon)\text{cost}(\vec{w}; \vec{o})$. \square

Lemma 6.9 (Polytime enumeration of threshold vectors). Suppose that 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 $O(|S| \cdot |\text{POS}| \cdot \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\}) = O(|S| \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{O(1/\delta)}\})$, 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_ℓ s 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 $1 + \log_{1+\varepsilon}(\frac{n}{\varepsilon}) = O(\frac{1}{\varepsilon} \log \frac{n}{\varepsilon})$ 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 $O(\frac{1}{\varepsilon} \log \frac{n}{\varepsilon})$, there are only $\exp(\max\{O(\frac{1}{\varepsilon} \log \frac{n}{\varepsilon}), |\text{POS}|\}) = O(\max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\})$ choices (by Claim 5.3). So the enumeration takes time $O(|S| \cdot |\text{POS}| \max\{(\frac{n}{\varepsilon})^{O(1/\varepsilon)}, n^{1/\delta}\})$, 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, \dots, t_{ℓ^*} such that $\vec{o}_\ell^\downarrow \leq t_\ell \leq (1 + \varepsilon)\vec{o}_\ell^\downarrow$ for each such ℓ . The remaining t_ℓ s 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 $(1 + \varepsilon)$. \square

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 whose proof is in Appendix C.

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

7 Approach towards 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 8.1](#) and [Section 9.1](#)) for the problem of minimizing $\sum_{i=1}^n h_t(\tilde{w}; \vec{v}_i)$, i.e., the \vec{v} -dependent part of our proxy function for $\text{cost}(\tilde{w}; \vec{v})$ (see [\(1\)](#) and [\(2\)](#)), where \tilde{w} is the sparsified version of w . Our LP-relaxation will have the property that only its objective depends on \tilde{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 [Section 8.2](#) and [Section 9.2](#)). To elaborate, we design a procedure that given an arbitrary feasible solution, say \bar{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 \tilde{w} , we have (loosely speaking) $\text{cost}(\tilde{w}, \vec{v}) = O(1) \cdot (\text{LP-objective-value of } \bar{x} \text{ under } \tilde{w})$. 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 \bar{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* $\text{cost}(w^{(i)}; \vec{v})$ under weight vectors $w^{(i)}$ in our collection is $O(\text{opt})$; 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 $\sum_{i=1}^n h_t(\tilde{w}; \vec{v}_i)$ can be equivalently written as $\sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \sum_{i=1}^n (\min\{\vec{v}_i, t_\ell\} - t_{\text{next}(\ell)})^+$. In our LP-relaxation, we introduce fractional variables to specify the quantities $\sum_{i=1}^n (\min\{\vec{v}_i, t_\ell\} - t_{\text{next}(\ell)})^+$. 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 $t_{\text{next}(\ell)}$ and t_ℓ , and we seek to 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 \tilde{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 [\[33\]](#)). Similar results are known in the literature, but we could not quite find a result that exactly fits our needs; we include a proof

in [Appendix D](#) for completeness.

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

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

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

8 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 $p_{ij} \geq 0$ required to process j on machine i . We have to output an assignment $\sigma : J \rightarrow [m]$ of jobs to machines. The load on machine i due to σ is $\text{load}_\sigma(i) := \sum_{j: \sigma(j)=i} p_{ij}$. Let $\vec{\text{load}}_\sigma := \{\text{load}_\sigma(i)\}_{i \in [m]}$ denote the load-vector induced by σ .

In the minimum-norm load-balancing problem, one seeks to minimize the norm of the load vector $\vec{\text{load}}_\sigma$ 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, $w_1 \geq w_2 \geq \dots \geq w_m \geq 0$), one seeks to minimize $\text{cost}(w; \vec{\text{load}}_\sigma) := w^T \vec{\text{load}}_\sigma^\downarrow = \sum_{i=1}^m w_i \vec{\text{load}}_\sigma^\downarrow(i)$. In the *Top- ℓ load balancing problem*, one seeks to minimize the sum of the ℓ largest loads in $\vec{\text{load}}_\sigma$.

Theorem 8.1. *Given any monotone, symmetric norm f on \mathbb{R}^m with a κ -approximate ball-optimization oracle for f (see [\(B-O\)](#)), and for any $\varepsilon > 0$, there is a $38\kappa(1+5\varepsilon)$ -approximation algorithm for the problem of finding an assignment $\sigma : J \rightarrow [m]$ which minimizes $f(\vec{\text{load}}_\sigma)$. The running time of the algorithm is $\text{poly}(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)})$.*

We have not optimized the constants in the above theorem. For the special cases of Top- ℓ and ordered load balancing, we can get much better results.

Theorem 8.2. *There is a polynomial time 2-approximation for the Top- ℓ -load balancing problem.*

Theorem 8.3. *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 $w^{(1)}, \dots, w^{(N)} \in \mathbb{R}_+^m$, and our goal is to find an assignment $\sigma : J \rightarrow [m]$ to minimize $\max_{r \in [N]} \text{cost}(w^{(r)}; \vec{\text{load}}_\sigma)$.

Theorem 8.4. *[Min-max ordered load balancing]*

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

As per the framework described in Section 7, in Section 8.1 we write an LP-relaxation for the (single) ordered optimization problem. Then in Section 8.2 we describe a *deterministic, weight-oblivious rounding* scheme which implies Theorem 8.4. Finally, in Section 8.3, we describe simpler and better rounding algorithms proving Theorem 8.2 and Theorem 8.3. These rounding algorithms are randomized (and oblivious), but their derandomizations are not. Nevertheless, we encourage the reader to first read Section 8.3 as a warm up to the deterministic, weight-oblivious rounding.

8.1 Linear programming relaxation

We begin by restating some definitions from Section 6 in the load balancing setting. As usual, $\vec{\sigma}$ will denote the load-vector induced by an optimal assignment for the problem under consideration. Recall that $\text{POS} = \text{POS}_{m,\delta} := \{\min\{\lceil(1+\delta)^s\rceil, m\} : s \geq 0\}$ is the sparse set of $O(\log m/\delta)$ indices. For $\ell \in \text{POS}$, $\text{next}(\ell)$ is the smallest index in POS larger than ℓ if $\ell < m$, and is $m+1$ otherwise. Given POS , recall the sparsified weight vector \tilde{w} of any weight vector w ; every $i \in [m]$, we set $\tilde{w}_i = w_i$ if $i \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$, we set $\tilde{w}_i = w_{\text{next}(\ell)}$.

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

$$\begin{aligned} \text{prox}_{\vec{t}}(\tilde{w}; \vec{\text{load}}_\sigma) &:= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \ell \cdot t_\ell + \sum_{i=1}^m h_{\vec{t}}(\tilde{w}; \text{load}_\sigma(i)), \quad \text{where} \\ h_{\vec{t}}(\tilde{w}; a) &:= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (a - t_\ell)^+. \end{aligned} \quad (\text{Prox-LB})$$

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 $\sigma : J \rightarrow [m]$ that approximately minimizes $\sum_{i=1}^m h_{\vec{t}}(\tilde{w}; \text{load}_\sigma(i))$ (see Lemma 6.10). We now describe the LP relaxation to minimize the proxy-cost. Our LP is parametrized by the vector \vec{t} .

Before describing the LP for the ordered load balancing, let us describe the LP for the special case of Top- ℓ load balancing. In this case, not that $\text{POS} = \{\ell\}$, and we have a guess $t = t_\ell$ of the ℓ th largest load. Also recall $h_t(\text{load}_\sigma(i))$ is simply $(\text{load}_\sigma(i) - t)^+$. The LP, as is usual, has variables x_{ij} to denote if j is assigned machine i . This x_{ij} is *split* into $y_{ij} + z_{ij}$ where z_{ij} denotes the fraction job j contributes to the load of machine i in the interval $[0, t)$. In the objective, for any machine i , we only consider the load “above the threshold”, that is, only the $p_{ij}y_{ij}$ portion.

$$\begin{aligned} \min \quad & \text{LP}_t(x, y, z) := \sum_i \sum_j p_{ij} y_{ij} & (\text{Top-}\ell\text{-LB}_{\vec{t}}) \\ \text{s.t.} \quad & \sum_i x_{ij} = 1 & \forall j \\ & x_{ij} = z_{ij} + y_{ij} & \forall i, j, \forall \ell \in \text{POS} \\ & \sum_j p_{ij} z_{ij} \leq t & \forall i, \forall \ell \in \text{POS} \\ & p_{ij} y_{ij} \geq (p_{ij} - t) x_{ij} & \forall i, j, \forall \ell \in \text{POS} \\ & x_{ij}, z_{ij}, y_{ij} \geq 0 & \forall i, j, \forall \ell \in \text{POS}. \end{aligned} \quad \begin{aligned} & (\text{T1}) \\ & (\text{T2}) \\ & (\text{T3}) \\ & (\text{T4}) \end{aligned}$$

The following lemma shows that the above LP is a valid relaxation.

Lemma 8.5. *For any $t > 0$ and any integral assignment σ , the value of the LP is at most $\sum_{i=1}^m h_t(\text{load}_\sigma(i))$.*

Proof. Given any assignment σ , set $x_{ij} = 1$ iff $\sigma(j) = i$. For each i, j with $x_{ij} = 1$, set

$$z_{ij} = \begin{cases} 1 & \text{if } \text{load}_\sigma(i) < t \\ \frac{t}{\text{load}_\sigma(i)} & \text{if } \text{load}_\sigma(i) \geq t \end{cases}$$

Set $y_{ij} = x_{ij} - z_{ij}$. We claim this (x, y, z) satisfies all constraints and has LP objective value equal to $\sum_{i=1}^m h_t(\tilde{w}; \text{load}_\sigma(i))$.

Constraint (T1) is satisfied since all jobs are assigned. Constraint (T2) is satisfied by definition. For any machine i , if $\text{load}_\sigma(i) < t$, then we get $\sum_j p_{ij} z_{ij} = \text{load}_\sigma(i) < t$. Otherwise, we get $\sum_j p_{ij} z_{ij} = t$. This implies (T3) is satisfied. We also satisfy (T4). To see this, note the inequality is vacuous if $z_{ij} = 1$, and otherwise it is satisfied with equality.

Finally note that $(\text{load}_\sigma(i) - t)^+ = \text{load}_\sigma(i) y_{ij}$, for all i and for all $j \in \sigma^{-1}(i)$. If $\text{load}_\sigma(i) < t$, then both sides are 0; otherwise, $y_{ij} = 1 - \frac{t}{\text{load}_\sigma(i)}$ for all j assigned to i by σ . Since the RHS is precisely $\sum_{i,j} p_{ij} y_{ij}$, the LP objective is precisely $\sum_{i \in [m]} h_t(\text{load}_\sigma(i))$. \square

At this point, we invite the reader to skip to [Section 8.3](#) to see a rounding for just the Top- ℓ load balancing problem. Next, we describe the LP for the general case by taking linear combinations of the above LP.

Now we write the LP for the ordered load balancing case. Again, 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 $(\text{load}_\sigma(i) - t_\ell)^+$. 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 .

$$\begin{aligned} \min \quad & \text{LP}_{\vec{t}}(\tilde{w}; x, y, z) := \sum_i \sum_{\ell \in \text{POS}} \sum_j (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) p_{ij} y_{ij}^{(\ell)} & (\text{OLB-P}_{\vec{t}}) \\ \text{s.t.} \quad & \sum_i x_{ij} = 1 & \forall j & (\text{OLB1}) \\ & x_{ij} = z_{ij}^{(\ell)} + y_{ij}^{(\ell)} & \forall i, j, \forall \ell \in \text{POS} & (\text{OLB2}) \\ & z_{ij}^{\text{next}(\ell)} \leq z_{ij}^{(\ell)} & \forall i, j, \forall \ell \in \text{POS} & (\text{OLB3}) \\ & \sum_j p_{ij} (z_{ij}^{(\ell)} - z_{ij}^{\text{next}(\ell)}) \leq t_\ell - t_{\text{next}(\ell)} & \forall i, \forall \ell \in \text{POS} & (\text{OLB4}) \\ & p_{ij} y_{ij}^{(\ell)} \geq (p_{ij} - t_\ell) x_{ij} & \forall i, j, \forall \ell \in \text{POS} & (\text{OLB5}) \\ & x_{ij}, z_{ij}^{(\ell)}, y_{ij}^{(\ell)} \geq 0 & \forall i, j, \forall \ell \in \text{POS}. \end{aligned}$$

Lemma 8.6. *For any valid threshold vector \vec{t} and any integral assignment σ , the value of the LP is at most $\sum_{i=1}^m h_{\vec{t}}(\tilde{w}; \text{load}_\sigma(i))$.*

Proof. Given any assignment σ , set $x_{ij} = 1$ iff $\sigma(j) = i$. For each $\ell \in \text{POS}$ and for i, j with $x_{ij} = 1$, set

$$z_{ij}^{(\ell)} = \begin{cases} 1 & \text{if } \text{load}_\sigma(i) < t_\ell \\ \frac{t_\ell}{\text{load}_\sigma(i)} & \text{if } \text{load}_\sigma(i) \geq t_\ell \end{cases}$$

Set $y_{ij}^{(\ell)} = x_{ij} - z_{ij}^{(\ell)}$. We claim this (x, y, z) satisfies all constraints and has LP objective value equal to $\sum_{i=1}^m h_{\vec{t}}(\tilde{w}; \text{load}_{\sigma}(i))$.

Constraint (OLB1) is satisfied since all jobs are assigned. Constraint (OLB2) is satisfied by definition, and (OLB3) is satisfied since $t_{\ell} \geq t_{\text{next}(\ell)}$. For any machine i and $\ell \in \text{POS}$, if $\text{load}_{\sigma}(i) < t_{\ell}$, then we get $\sum_j p_{ij} z_{ij}^{(\ell)} = \text{load}_{\sigma}(i) < t_{\ell}$. Otherwise, we get $\sum_j p_{ij} z_{ij}^{(\ell)} = t_{\ell}$. This implies (OLB4) is satisfied since $t_{\ell} \geq t_{\text{next}(\ell)}$. We also satisfy (OLB5). To see this, note the inequality is vacuous if $z_{ij}^{(\ell)} = 1$, and otherwise it is satisfied with equality.

Finally note that $(\text{load}_{\sigma}(i) - t_{\ell})^+ = \text{load}_{\sigma}(i) y_{ij}^{(\ell)}$, for all i and for all $j \in \sigma^{-1}(i)$. If $\text{load}_{\sigma}(i) < t_{\ell}$, then both sides are 0; otherwise, $y_{ij}^{(\ell)} = 1 - \frac{t_{\ell}}{\text{load}_{\sigma}(i)}$ for all j assigned to i by σ . Since the RHS is precisely $\sum_{i,j} p_{ij} y_{ij}^{(\ell)}$, the LP objective is precisely $\sum_{i \in [m]} h_{\vec{t}}(\tilde{w}; \text{load}_{\sigma}(i))$. \square

In Section 8.3 (which, as we encourage, can be read before moving further), we show a simple *randomized rounding* algorithm. 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 8.7. (Deterministic weight-oblivious rounding for load balancing.)

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 (OLB1)-(OLB5), and produces an assignment $\tilde{\sigma} : J \rightarrow [m]$ such that, for any sparsified weight vector \tilde{w} , we have that

$$\sum_{i=1}^m h_{10\vec{t}}(\tilde{w}; \text{load}_{\tilde{\sigma}}(i)) \leq 2 \cdot \text{LP}_{\vec{t}}(\tilde{w}; x, y, z) + 4 \sum_{\ell \in \text{POS}} \tilde{w}_{\ell} t_{\ell} \quad (3)$$

Note that the algorithm doesn't use the weights; rather the fixed output satisfies (3) for *all* weights simultaneously. We prove this theorem in Section 8.2 which can be directly skipped to. In the remainder of this section, we use the theorem to prove Theorem 8.4 and Theorem 8.1.

Proof of Theorem 8.4. We sparsify $w^{(r)}$ to $\tilde{w}^{(r)}$ for all $r \in [N]$; recall $\text{POS} = \text{POS}_{m,\delta}$. Let $\vec{\sigma}$ be the load-vector induced by an optimal solution. Let $\text{opt} := \max_{r \in [k]} \text{cost}(w^{(r)}; \vec{\sigma})$.

Using the enumeration procedure in Lemma 6.9 with $\varepsilon = 1$ and finding a ρ that is a power of 2 such that $\vec{\sigma}_1^{\downarrow} \leq \rho \leq 2\vec{\sigma}_1^{\downarrow}$, we may assume that we have obtained a valid threshold vector \vec{t} where all t_{ℓ} s are powers of 2 or 0, and which satisfies the conditions: $\vec{\sigma}_{\ell}^{\downarrow} \leq t_{\ell} \leq 2\vec{\sigma}_{\ell}^{\downarrow}$ if $\vec{\sigma}_{\ell}^{\downarrow} \geq \frac{\vec{\sigma}_1^{\downarrow}}{m}$, and $t_{\ell} = 0$ otherwise.

We now solve an LP similar to (OLB-P $_{\vec{t}}$) with the objective modified to encode the min-max-ness.

$$\min \quad \lambda : \quad (x, y, z) \text{ satisfies (OLB1) - (OLB5)} \quad (4)$$

$$\sum_{\ell \in \text{POS}} (\tilde{w}_{\ell}^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) t_{\ell} + \text{LP}_{\vec{t}}(\tilde{w}^{(r)}; x, y, z) \leq \lambda \quad \forall r \in [N] \quad (5)$$

Claim 8.8. Let λ^* be the optimum solution to the LP above. Then, $\lambda^* \leq 3\text{opt}$.

Proof. Let σ^* be the optimal integral assignment attaining opt and (x, y, z) be the assignment described by this integral assignment as in the proof of Lemma 8.6. For any $r \in [N]$, we therefore get $\text{LP}_{\vec{t}}(\tilde{w}^{(r)}; x, y, z) = \sum_{i=1}^m h_{\vec{t}}(\tilde{w}^{(r)}; \vec{\sigma})$. Thus, from the definition of prox and using (5), we get $\lambda^* \leq \max_{r \in [N]} \text{prox}_{\vec{t}}(\tilde{w}^{(r)}; \vec{\sigma}^{\downarrow})$. Finally, Lemma 6.8 (with $\varepsilon = 1$) gives that for any $r \in [N]$, $\text{prox}_{\vec{t}}(\tilde{w}^{(r)}; \vec{\sigma}^{\downarrow}) \leq 3\text{cost}(\tilde{w}^{(r)}; \vec{\sigma})$. \square

Given the optimal solution (x, y, z) to the above LP, we use Theorem 8.7 (since we have ensured that the t_{ℓ} 's are powers of 2 or 0) to obtain an assignment $\tilde{\sigma} : J \rightarrow [m]$. We get for *any* $r \in [N]$,

$$\begin{aligned}
\text{prox}_{10\vec{t}}(\tilde{w}^{(r)}; \overrightarrow{\text{load}_{\vec{\sigma}}}) &= \sum_{\ell \in \text{POS}} (\tilde{w}_{\ell}^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell \cdot 10t_{\ell} + \sum_{i=1}^m h_{10\vec{t}}(\tilde{w}^{(r)}; \text{load}_{\vec{\sigma}}(i)) \\
&\leq 10 \sum_{\ell \in \text{POS}} (\tilde{w}_{\ell}^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell \cdot t_{\ell} + 2 \cdot \text{LP}_{\vec{t}}(\tilde{w}^{(r)}; x, y, z) + 4 \sum_{\ell \in \text{POS}} \tilde{w}_{\ell}^{(r)} t_{\ell} \\
&\leq 10\lambda^* + 4 \sum_{\ell \in \text{POS}} \tilde{w}_{\ell}^{(r)} t_{\ell} \leq 30\text{opt} + 4 \sum_{\ell \in \text{POS}} \tilde{w}_{\ell}^{(r)} t_{\ell}
\end{aligned} \tag{6}$$

where the first inequality follows from the *obliviousness property* of the rounding in [Theorem 8.7](#). The same rounded assignment works for all the weights simultaneously. The second inequality follows from [\(5\)](#). The last inequality invokes [Claim 8.8](#). Now we use the fact that $t_{\ell} \leq 2\vec{\sigma}_{\ell}$ to get $\sum_{\ell \in \text{POS}} \tilde{w}_{\ell}^{(r)} t_{\ell} \leq 2\text{cost}(\tilde{w}^{(r)}; \vec{\sigma}) \leq 2\text{cost}(w^{(r)}; \vec{\sigma}) \leq 2 \cdot \text{opt}$. The second-last inequality follows from the sparsification property ([Lemma 4.2](#)). Substituting in [\(6\)](#), we get that for any $r \in [N]$, $\text{prox}_{10\vec{t}}(\tilde{w}^{(r)}; \overrightarrow{\text{load}_{\vec{\sigma}}}) \leq 38\text{opt}$. From [Claim 6.5](#) and [Lemma 4.2](#), we get for any $r \in [N]$,

$$\text{cost}(w^{(r)}; \overrightarrow{\text{load}_{\vec{\sigma}}}) \leq (1 + \delta)\text{cost}(w^{(r)}; \overrightarrow{\text{load}_{\vec{\sigma}}}) \leq (1 + \delta)\text{prox}_{10\vec{t}}(\tilde{w}^{(r)}; \overrightarrow{\text{load}_{\vec{\sigma}}}) \leq 38(1 + \delta)\text{opt}$$

□

Proof of Theorem 8.1. This follows by combining [Theorem 5.4](#) and [Theorem 8.4](#) (taking $\delta = \varepsilon$). We only need to show that we can obtain the estimates hi, lb, ub in [\(A1\)](#), [\(A2\)](#), and they lead to the stated running time. The approximation guarantee obtained is $38(1 + \varepsilon)\kappa(1 + 3\varepsilon) \leq 38\kappa(1 + 5\varepsilon)$.

Let $\vec{\sigma}^{\downarrow}$ be the sorted cost vector induced by an optimal assignment. Let $e_i \in \mathbb{R}^m$ denote the vector with 1 in coordinate i , and 0s everywhere else. We can determine in polytime if $\vec{\sigma}_1^{\downarrow} = 0$; if not, since the p_{ij} s are integers, we have $\vec{\sigma}_1^{\downarrow} \geq 1$, and $\text{opt} \geq f(\vec{\sigma}_1^{\downarrow} e_1) \geq \text{lb} := f(e_1)$ since f is monotone. Consider the assignment where $\sigma(j) := \text{argmin}_{i \in [m]} p_{ij}$ for each job j . We have

$$\text{opt} \leq f(\overrightarrow{\text{load}_{\sigma}}) \leq \sum_j f(p_{\sigma(j)j} e_{\sigma(j)}) = \text{ub} := \sum_j f(p_{\sigma(j)j} e_i).$$

The second inequality follows from the triangle inequality; the third equality follows from symmetry. This also means that $\vec{\sigma}_1^{\downarrow} \leq \text{hi} := \sum_j p_{\sigma(j)j} = \sum_j \min_{i \in [m]} p_{ij}$, since by monotonicity, we have $\text{opt} = f(\vec{\sigma}^{\downarrow}) \geq f(\vec{\sigma}_1^{\downarrow} e_1)$. So $\text{ub/lb} = \text{hi}$ and $\log(\frac{n \cdot \text{ub} \cdot \text{hi}}{\text{lb}}) = \text{poly}(\text{input size})$. So the running time of the reduction in [Theorem 5.4](#), and the size of the simultaneous ordered-load-balancing problem it creates, are $\text{poly}(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)})$, and the entire algorithm runs in time $\text{poly}(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)})$. □

8.2 Deterministic weight oblivious rounding : proof of [Theorem 8.7](#)

We are given a solution (x, y, z) which satisfy constraints [\(OLB1\)](#)-[\(OLB5\)](#). It is convenient to do a change of variables. First, define $z_{ij}^{(m+1)} = 0$ and $z_{ij}^{(0)} = x_{ij}$, and let $y_{ij}^{(\ell)} = x_{ij} - z_{ij}^{(\ell)}$ for $\ell = 0, m + 1$. For all $\ell \in \{0\} \cup \text{POS}$ and all i, j , define

$$\bar{q}_{ij}^{(\ell)} := z_{ij}^{(\ell)} - z_{ij}^{(\text{next}(\ell))} = y_{ij}^{(\text{next}(\ell))} - y_{ij}^{(\ell)}$$

which is nonnegative due to [\(OLB3\)](#). Now for any \tilde{w} , we can rewrite $\text{LP}_{\vec{t}}(\tilde{w}; x, y, z)$ as follows

$$\begin{aligned}
\sum_{i,j} \sum_{\ell \in \text{POS}} (\tilde{w}_{\ell} - \tilde{w}_{\text{next}(\ell)}) p_{ij} y_{ij}^{(\ell)} &= \sum_{i,j,\ell \in \text{POS}} (\tilde{w}_{\ell} - \tilde{w}_{\text{next}(\ell)}) \sum_{\ell' \in \{0\} \cup \text{POS}: \ell' < \ell} p_{ij} \bar{q}_{ij}^{(\ell')} \\
&= \sum_{i,j,\ell' \in \{0\} \cup \text{POS}} p_{ij} \bar{q}_{ij}^{(\ell')} \tilde{w}_{\text{next}(\ell')} =: \text{LP}_{\vec{t}}(\tilde{w}; \bar{q})
\end{aligned} \tag{7}$$

We first give an overview of the rounding procedure. We begin by *filtering* \bar{q} to obtain $\hat{q} \leq 2\bar{q}$ with the property that $\hat{q}_{ij}^{(\ell)} = 0$ if $p_{ij} > 2t_\ell$ for all i, j and all $\ell \in \text{POS}$ (Lemma 8.9). This relies on the constraints (OLB5). Next, we set up an auxiliary LP (IR) similar to (OLB-P $_{\bar{t}}$) using the same modified set of variables $q_{ij}^{(\ell)}$, which have the same intended meaning. We include constraints (OLB1), and (OLB4) but with the RHS multiplied by 2. We also include constraints $\sum_{i,j} p_{ij} q_{ij}^{(\ell)} \leq 2 \sum_{i,j} p_{ij} \bar{q}_{ij}^{(\ell)}$ for all $\ell \in \text{POS}$; the objective of (IR) is to minimize $\sum_{i,j} p_{ij} q_{ij}^{(0)}$. The latter budget constraints and the objective of (IR) serve to ensure that the objective values of q and \bar{q} under (7) are comparable. Notice that \hat{q} yields a feasible solution to this auxiliary LP. We next use *iterative rounding* (that is, Theorem 7.1) on this system to produce an integral point \bar{q}^{int} that assigns every job, satisfies the other budget constraints approximately, and whose objective value under (IR) is at most that of \hat{q} . We argue that the integral point \bar{q}^{int} yields the desired assignment $\tilde{\sigma} : J \rightarrow [m]$, where $\tilde{\sigma}(j)$ is set to the unique i such that $\sum_{\ell \in \{0\} \cup \text{POS}} \bar{q}_{ij}^{\text{int}(\ell)} = 1$. We now describe the algorithm in detail and proceed to analyze it.

Algorithm.

- L1. **Filtering.** For every job j and machine i , we do the following. If $p_{ij} \leq 2t_\ell$ for all $\ell \in \text{POS}$, then set $\hat{q}_{ij}^{(\ell)} = \bar{q}_{ij}^{(\ell)}$ for all $\ell \in \{0\} \cup \text{POS}$. Otherwise, let $\bar{\ell} \in \text{POS}$ be the smallest index for which $p_{ij} > 2t_{\bar{\ell}}$. For every index $\ell \in \{0\} \cup \text{POS}$, we set $\hat{q}_{ij}^{(\ell)} = 0$ if $p_{ij} > 2t_\ell$, and $\hat{q}_{ij}^{(\ell)} = \bar{q}_{ij}^{(\ell)} \cdot x_{ij} / y_{ij}^{(\bar{\ell})}$ otherwise (where $0/0$ is defined as 0). Lemma 8.9 shows that $\hat{q} \leq 2\bar{q}$, and \hat{q} satisfies (8). We will produce an integral point \bar{q}^{int} whose support is contained in that of \hat{q} , so $\bar{q}_{ij}^{\text{int}(\ell)} = 1$ will imply that $p_{ij} \leq 2t_\ell$.

- L2. **Iterative rounding.** Consider the following auxiliary LP.

$$\begin{aligned}
\min \quad & \sum_{i,j} p_{ij} q_{ij}^{(0)} & (\text{IR}) \\
\text{s.t.} \quad & \sum_i \sum_{\ell \in \{0\} \cup \text{POS}} q_{ij}^{(\ell)} = 1 & \forall j \\
& \sum_j p_{ij} q_{ij}^{(\ell)} \leq 2(t_\ell - t_{\text{next}(\ell)}) & \forall i, \forall \ell \in \text{POS} \\
& \sum_{i,j} p_{ij} q_{ij}^{(\ell)} \leq 2 \sum_{i,j} p_{ij} \bar{q}_{ij}^{(\ell)} & \forall \ell \in \text{POS} \\
& q_{ij}^{(\ell)} \geq 0 & \forall i, j, \forall \ell \in \{0\} \cup \text{POS}.
\end{aligned} \tag{8}$$

We call the constraints, except for (8) and the non-negativity constraints, budget constraints. By Lemma 8.9, \hat{q} is a feasible solution to (IR).

We round \hat{q} to an integral point \bar{q}^{int} using Theorem 7.1, taking $A_1 = A_2$ to be the constraint matrix formed by constraints (8), where each equality constraint is written as a pair of \leq - and \geq - inequalities. Define $\tilde{\sigma} : J \rightarrow [m]$ by setting $\tilde{\sigma}(j)$ to be the unique i such that $\sum_{\ell \in \{0\} \cup \text{POS}} \bar{q}_{ij}^{\text{int}(\ell)} = 1$. Return $\tilde{\sigma}$.

Analysis.

Lemma 8.9. *The solution \hat{q} obtained after step L1 satisfies $\hat{q} \leq 2\bar{q}$ and constraints (8).*

Proof. Fix a job j and a machine i . If $x_{ij} = \sum_{\ell \in \{0\} \cup \text{POS}} \bar{q}_{ij}^{(\ell)} = 0$, or $p_{ij} \leq 2t_\ell$ for all $\ell \in \text{POS}$, then we have $\hat{q}_{ij}^{(\ell)} = \bar{q}_{ij}^{(\ell)}$ for all $\ell \in \text{POS}$. So suppose otherwise. Let $\bar{\ell} \in \text{POS}$ be the smallest index for which $p_{ij} > 2t_{\bar{\ell}}$. Constraint (OLB5) for $i, j, \bar{\ell}$ implies that $y_{ij}^{(\bar{\ell})} \geq 0.5x_{ij} > 0$. It follows that $\hat{q}_{ij}^{(\ell)} \leq 2\bar{q}_{ij}^{(\ell)}$ for all

$\ell \in \{0\} \cup \text{POS}$. Also, since $p_{ij} \leq 2t_\ell$ for $\ell \in \{0\} \cup \text{POS}$ iff $\ell < \bar{\ell}$, we have $y_{ij}^{(\bar{\ell})} = \sum_{\ell \in \{0\} \cup \text{POS} : p_{ij} \leq 2t_\ell} \bar{q}_{ij}^{(\ell)}$. Therefore, $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_{ij}^{(\ell)} = x_{ij}$, and so \hat{q} satisfies (8). \square

We summarize the properties satisfied by the integral point \bar{q} obtained by rounding \hat{q} using [Theorem 7.1](#).

Lemma 8.10. *The $\{0, 1\}$ -point $\bar{q} \geq 0$ obtained in step L2 satisfies $\sum_{i,j} p_{ij} \bar{q}_{ij}^{(0)} \leq 2 \sum_{i,j} p_{ij} \bar{q}_{i,j}^{(0)}$, constraints (8), and*

$$\sum_j p_{ij} \bar{q}_{ij}^{(\ell)} \leq 0 \quad \forall i, \forall \ell \in \text{POS} : t_\ell = t_{\text{next}(\ell)} \quad (9)$$

$$\sum_j p_{ij} \bar{q}_{ij}^{(\ell)} \leq 6t_\ell - 2t_{\text{next}(\ell)} \quad \forall i, \forall \ell \in \text{POS} : t_\ell > t_{\text{next}(\ell)} \quad (10)$$

$$\sum_{i,j} p_{ij} \bar{q}_{ij}^{(\ell)} \leq 2 \sum_{i,j} p_{ij} \bar{q}_{i,j}^{(\ell)} + 4t_\ell \quad \forall \ell \in \text{POS}. \quad (11)$$

Proof. These are all direct consequences of [Theorem 7.1](#). Part (a) (of [Theorem 7.1](#)) shows that $\sum_{i,j} p_{ij} \bar{q}_{ij}^{(0)} \leq \sum_{i,j} p_{ij} \hat{q}_{ij}^{(0)} \leq 2 \sum_{i,j} p_{ij} \bar{q}_{i,j}^{(0)}$. Since (8) is encoded via the constraints involving A_1, A_2 in the setup of [Theorem 7.1](#), part (c) shows that (8) holds.

Every $\bar{q}_{ij}^{(\ell)}$ variable appears in at most 2 budget constraints of (IR). If $t_\ell = t_{\text{next}(\ell)}$, then $\bar{q}_{ij}^{(\ell)} = \hat{q}_{ij}^{(\ell)} = 0$ for all i, j . So since the support of \bar{q} is a subset of the support of \hat{q} (part (b)), we have $\sum_j p_{ij} \bar{q}_{ij}^{(\ell)} = 0$. So suppose $t_\ell > t_{\text{next}(\ell)}$, and consider the budget constraint $\sum_j p_{ij} \bar{q}_{ij}^{(\ell)} \leq 2(t_\ell - t_{\text{next}(\ell)})$ for a given machine i and $\ell \in \text{POS}$. If $\bar{q}_{ij}^{(\ell)} > 0$, we know that $p_{ij} \leq 2t_\ell$, so applying part (d), shows that (10) holds. Part (d) then also shows that (11) holds. \square

Finishing up the proof of [Theorem 8.7](#). We first show that for any i and any $\ell \in \text{POS}$, we have that $\sum_{\ell' \in \text{POS} : \ell' \geq \ell} \sum_j p_{ij} \bar{q}_{ij}^{(\ell')} \leq 10t_\ell$. By [Lemma 8.10](#), we have that $\sum_{\ell' \in \text{POS} : \ell' \geq \ell} \sum_j p_{ij} \bar{q}_{ij}^{(\ell')}$ is at most $\sum_{\ell' \in \text{POS} : \ell' \geq \ell, t_{\ell'} > t_{\text{next}(\ell')}} (6t_{\ell'} - 2t_{\text{next}(\ell')})$. Suppose that $\ell_1 < \ell_2 < \dots < \ell_a \in \text{POS}$ are all the indices $\ell' \in \text{POS}$ satisfying $\ell' \geq \ell, t_{\ell'} > t_{\text{next}(\ell_1)}$. Then,

$$\sum_{\ell' \in \text{POS} : \ell' \geq \ell, t_{\ell'} > t_{\text{next}(\ell')}} (6t_{\ell'} - 2t_{\text{next}(\ell')}) \leq (6t_{\ell_1} - 2t_{\text{next}(\ell_1)}) + \dots + (6t_{\ell_a} - 2t_{\text{next}(\ell_a)}) \leq 6t_{\ell_1} + 4(t_{\ell_2} + t_{\ell_3} + \dots + t_{\ell_a})$$

Recall that the t_ℓ s are all powers of 2, or 0. So $t_{\ell_2} \leq t_{\ell_1}/2$, $t_{\ell_3} \leq t_{\ell_2}/2$, and so on. So the RHS above is at most $6t_{\ell_1} + 4t_{\ell_1} \leq 10t_\ell$. This implies that $(\text{load}_{\bar{\sigma}}(i) - 10t_\ell)^+ \leq \sum_{\ell' \in \{0\} \cup \text{POS} : \ell' < \ell} \sum_j p_{ij} \bar{q}_{ij}^{(\ell')}$. Therefore,

$$\begin{aligned} h_{10\bar{\ell}}(\tilde{w}; \text{load}_{\bar{\sigma}}(i)) &= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\text{load}_{\bar{\sigma}}(i) - 10t_\ell)^+ \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_{\ell' \in \{0\} \cup \text{POS} : \ell' < \ell} \sum_j p_{ij} \bar{q}_{ij}^{(\ell')} = \sum_{\ell' \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell')} \sum_j p_{ij} \bar{q}_{ij}^{(\ell')}. \end{aligned}$$

It follows that $\sum_{i=1}^m h_{10\bar{\ell}}(\tilde{w}; \text{load}_{\bar{\sigma}}(i)) \leq \sum_{\ell \in \{0\} \cup \text{POS}} \sum_{i,j} \tilde{w}_{\text{next}(\ell)} p_{ij} \bar{q}_{ij}^{(\ell)}$. Using [Lemma 8.10](#), we can bound the RHS by

$$2 \sum_{\ell \in \{0\} \cup \text{POS}} \sum_{i,j} \tilde{w}_{\text{next}(\ell)} p_{ij} \bar{q}_{i,j}^{(\ell)} + 4 \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} t_\ell \leq 2 \cdot \text{LP}(\tilde{w}; \bar{q}) + 4 \sum_{\ell \in \text{POS}} \tilde{w}_\ell t_\ell. \quad \square$$

8.3 Improved approximation for Top- ℓ and ordered load balancing problems

In this section we prove [Theorem 8.2](#) and [Theorem 8.3](#). Recall, in the (single) ordered load balancing problem, we have only one non-increasing weight vector w and we wish to find an assignment σ minimizing $\text{cost}(w; \text{load}_\sigma)$. In the Top- ℓ problem, this weight vector is a 0, 1 vector.

We prove this by rounding (OLB- P_ℓ) for a particular valid threshold vector. As usual, let \vec{o} be the load vector for the optimal assignment. For the ordered problem, we first sparsify w to get \tilde{w} using [Lemma 4.2](#) with $\delta = \varepsilon$. Next, we use [Lemma 6.9](#) to get threshold vector \vec{t} satisfying (a) $\vec{o}_\ell^\downarrow \leq t_\ell \leq (1 + \varepsilon)\vec{o}_\ell^\downarrow$ if $\vec{o}_\ell^\downarrow \geq \varepsilon \vec{o}_1^\downarrow / n$, and $t_\ell = 0$ otherwise. This enumeration is what leads to the $(1 + \varepsilon)$ loss. For the Top- ℓ load balancing problem, we can in fact exactly guess the \vec{o}_ℓ^\downarrow , that is, the ℓ th largest cost in the optimum solution.

Our improved approximation algorithms follow from *oblivious, randomized rounding* algorithm for (OLB- P_ℓ). In fact, the randomized algorithm would be oblivious of the guesses of t_ℓ 's (the t_ℓ 's will be used only to solve (OLB- P_ℓ)). However, being randomized, this algorithm doesn't suffice to give good algorithms for the min-max problem. Indeed, the derandomized version of these algorithms are *not* oblivious. Without further ado, we state and analyze the randomized algorithm in the proof of the following lemma.

Lemma 8.11. *There is an algorithm which takes as input $x_{ij} \in [0, 1]$ for all i, j pairs, and returns a random assignment $\tilde{\sigma} : J \rightarrow [m]$ with the following property. For any t and any (y_{ij}, z_{ij}) satisfying (a) $y_{ij} + z_{ij} = x_{ij}$ for all i, j , (b) $\sum_j p_{ij} z_{ij} \leq t$ for all i , and (c) $p_{ij} y_{ij} \geq (p_{ij} - t)x_{ij}$ for all i, j , we get $\text{Exp}[\sum_{i=1}^m (\text{load}_{\tilde{\sigma}}(i) - 2t)^+] \leq 2 \sum_{i,j} p_{ij} y_{ij}$*

Proof. The algorithm is a randomized version of the Shmoys-Tardos algorithm [40] for the generalized assignment problem. More precisely, for every machine i , we make $n_i = \left\lceil \sum_{j \in J} x_{ij} \right\rceil$ copies. Let I_c be the union of the copies. Now we define a bipartite graph on the vertex set $I_c \cup J$ and define a fractional (sub)-matching \bar{x} on it. Fix a machine i and consider the n_i copies. Arrange the jobs J in *non-increasing* order of p_{ij} . We start with the first copy and call it active. Each job in the order tries to send x_{ij} units of mass to the active copy till the total \bar{x} -mass faced by the active copy equals 1. We then move to the next copy and the job sends the remainder unit of its fraction to that copy. We continue till all jobs in J distribute a total of $\sum_j x_{ij}$ on the n_i copies, and all but perhaps one of the n_i copies face a fractional x_{ij} -mass of exactly 1. In sum, at the end of this procedure for all machine, for every job we have $\sum_{k \in I_c} \bar{x}_{kj} = 1$ while for every machine copy $k \in I_c$, we have $\sum_{j \in J} \bar{x}_{kj} \leq 1$. For each machine i , we let $J_r^{(i)}$ be the set of jobs j which have $\bar{x}_{kj} > 0$ for the r th copy of machine i . A standard result from matching theory [36] gives us the following claim.

Claim 8.12. *There is a distribution D on matchings in this bipartite graph such that for any copy $k \in I_c$ and any job $j \in J$, we have*

$$\Pr_{M \leftarrow D}[(k, j) \in M] \leq \bar{x}_{kj} \leq x_{ij}$$

The randomized rounding algorithm for load balancing samples a matching M from D described in [Claim 8.12](#), and then allocates to machine i all the jobs j such that $(k, j) \in M$ for any copy k of machine i . Let $\tilde{\sigma}$ be this random assignment.

Analysis. For each machine i , let Z_i denote the *random variable* indicating the load p_{ij} of the job $j \in J_1^{(i)}$ allocated to the *first* copy of machine i .

Claim 8.13. $\text{load}_{\tilde{\sigma}}(i) \leq \sum_{j \in J} p_{ij} x_{ij} + Z_i$

Proof. Since the jobs are in descending order, the load of the random job allocated to the $r + 1$ th copy of machine i is at most $\sum_{j \in J_r^{(i)}} p_{ij} \bar{x}_{ij}$. Thus, the load on machine i due to all but the job allocated to its first copy is at most $\sum_{j \in J} p_{ij} x_{ij}$. The claim follows now from the definition of Z_i . \square

Thus,

$$(\text{load}_{\tilde{\sigma}}(i) - 2t)^+ \leq \left(\sum_j p_{ij}x_{ij} - t \right)^+ + (Z_i - t)^+ \leq \sum_j p_{ij}y_{ij} + (Z_i - t)^+$$

where the last inequality uses assumptions (a) and (b) of the lemma. The proof of the lemma follows from the following claim.

Claim 8.14. *For any machine i , $\mathbf{Exp}[(Z_i - t)^+] \leq \sum_j p_{ij}y_{ij}$*

Proof. If k is the first copy of machine i , then we get $\mathbf{Exp}[(Z_i - t)^+] = \sum_{j \in J_1^{(i)}} \Pr_{M \leftarrow \mathcal{D}}[(k, j) \in M] \cdot (p_{ij} - t)^+ \leq \sum_j (p_{ij} - t)^+ x_{ij} \leq \sum_j p_{ij}y_{ij}$. The first inequality uses [Claim 8.12](#), and the second uses assumption (c) of the Lemma. \square

\square

Proof of Theorem 8.3. As described above, using [Lemma 4.2](#) and [Lemma 6.9](#), we have a vector \vec{t} with which we solve (OLB- $\mathbf{P}_{\vec{t}}$). Given the solution x , we apply [Lemma 8.11](#). Note that for all ℓ , the tuple $(t_\ell, y_{ij}^{(\ell)}, z_{ij}^{(\ell)})$ satisfies the conditions of the lemma. Part (a) follows from (OLB2), part (b) follows from adding up (OLB4) for all $1 \leq \ell' \leq \ell$, and part (c) follows from (OLB5). Therefore, we get that for each ℓ , $\mathbf{Exp}[\sum_{i=1}^m (\text{load}_{\tilde{\sigma}}(i) - 2t_\ell)^+] \leq 2 \sum_{i,j} p_{ij}y_{ij}^{(\ell)}$. This in turn implies $\mathbf{Exp}[h_{2\vec{t}}(\tilde{w}; \text{load}_{\tilde{\sigma}})] \leq 2\text{LP}_{\vec{t}}(\tilde{w}; x, y, z) \leq 2 \sum_{i=1}^m h_{\vec{t}}(\tilde{w}; \vec{\sigma})$, where the last inequality follows from [Lemma 8.6](#). Using [Lemma 6.10](#), we get that $\mathbf{Exp}[\text{cost}(w; \text{load}_{\tilde{\sigma}})] \leq (2 + \varepsilon)\text{opt}$. \square

Proof of Theorem 8.2. Note that the ε -loss over 2 in the previous theorem came from two sources: one is in moving to the sparsified weight vector, and the other in the guess of $\vec{\sigma}_\ell^\downarrow$'s. For the Top- ℓ version of the problem, the position set $\text{POS} = \{\ell\}$ is the singleton position ℓ . The 0, 1 weight vector, in this case, coincides with the sparsified vector. Indeed, we can just focus on the simpler (Top- ℓ -LB $_{\vec{t}}$). Furthermore, as we show below, we can guess $\vec{\sigma}_\ell^\downarrow$ “exactly” via binary search. In particular, for any guess t of $\vec{\sigma}_\ell^\downarrow$, we solve (Top- ℓ -LB $_{\vec{t}}$) of value LP_t . As per the previous proof, the algorithm described in [Lemma 8.11](#) gives a randomized algorithm with expected Top- ℓ cost $\leq 2(\ell t + \text{LP}_t)$ for any t . Therefore, via binary search, we find the t which minimizes $(\ell t + \text{LP}_t)$; this minimum value is $\leq \text{opt}$ since for $t = \vec{\sigma}_\ell^\downarrow$, the value is $\leq \text{opt}$. For this t , the randomized algorithm described in the proof of [Theorem 8.3](#) returns an assignment with expected cost $\leq 2\text{opt}$. \square

Derandomization. Both the above algorithms above can be easily derandomized, but this comes at the cost of obliviousness. We describe the derandomization for the Top- ℓ problem and the derandomization of the ordered problem is similar. In particular, for any t we give a deterministic algorithm which returns an assignment $\tilde{\sigma}$ with $\sum_{i=1}^m (\text{load}_{\tilde{\sigma}}(i) - 2t)^+ \leq 2\text{LP}_t$; this will imply a deterministic 2-approximation using [Lemma 6.10](#) as it did in the proof of [Theorem 8.2](#).

In the proof of [Lemma 8.11](#), when we construct the bipartite graph between jobs and the copies of the machines, introduce a cost $(p_{ij} - t)^+$ on the edges of the form (k, j) where k is the first copy of machine i and $j \in J_1^{(i)}$. Every other (k, j) edge has cost 0. Subsequently, find a *minimum cost* matching which matches every job, and every copy of any machine which was also fractionally fully matched. The deterministic assignment $\tilde{\sigma}$ is given by this matching as in the proof of the lemma. Also as in the proof, we get that for any machine i , $(\text{load}_{\tilde{\sigma}}(i) - 2t)^+ \leq \sum_j p_{ij}y_{ij} + (Z_i - t)^+$ where Z_i is the p_{kj} of the job assigned to the first copy. Since we have found the matching precisely minimizing this cost, the minimum value is at most the expected value (given by any distribution, in particular, the distribution of [Claim 8.12](#)), which was shown to be $\leq \sum_{i,j} p_{ij}y_{ij}$ in [Claim 8.14](#). In sum, we can deterministically find an assignment σ with $\sum_{i=1}^m (\text{load}_{\tilde{\sigma}}(i) - 2t)^+ \leq 2\text{LP}_t$.

9 k -Clustering

In this section, we 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.⁵ 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 *minimum-norm k -clustering*, the goal is to minimize the norm of \vec{c} under a given monotone, symmetric norm. The *ordered k -median*⁶ problem is the special case where we are given non-increasing weights $w_1 \geq w_2 \geq \dots \geq w_n \geq 0$, 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 $w_1 = 1 = \dots = w_\ell$ and the remaining w_i s are 0. That is, we want to minimize the sum of the ℓ largest assignment costs.

Theorem 9.1. *Given any monotone, symmetric norm f on \mathbb{R}^m with a κ -approximate ball-optimization oracle for f (see (B-O)), and for any $\varepsilon > 0$, there is a $\kappa(408 + O(\varepsilon))$ -approximation algorithm for the problem of finding $F \subseteq \mathcal{F}$ with $|F| \leq k$ such that the resulting assignment-cost vector \vec{c} minimizes $f(\vec{c})$. The running time of the algorithm is $\text{poly}(\text{input size}, (\frac{n}{\varepsilon})^{O(1/\varepsilon)})$.*

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 $w^{(1)}, \dots, w^{(N)} \in \mathbb{R}_+^m$, and our goal is to find an assignment $\sigma : J \rightarrow [m]$ to minimize $\max_{r \in [N]} \text{cost}(w^{(r)}; \vec{\text{load}}_\sigma)$.

Theorem 9.2. [*Min-max ordered k -median*]

Given any non-increasing weight vectors $w^{(1)}, \dots, w^{(N)} \in \mathbb{R}_+^n$, we can find a $(408 + O(\varepsilon))$ -approximation algorithm for the Min-Max Ordered k -median problem of finding a $F \subseteq \mathcal{F}$ with $|F| \leq k$ such that the resulting assignment-cost vector \vec{c} minimizes $\max_{r \in [N]} \text{cost}(w^{(r)}; \vec{c})$. The running time of the algorithm is $\text{poly}(\text{input size}, n^{O(1/\varepsilon)})$.

As per the framework described in Section 7, we first write (in Section 9.1) an LP-relaxation for the (single) ordered k -median problem. We then show a *deterministic, weight-oblivious rounding* scheme (in Section 9.2) which implies Theorem 9.2.

We have not optimized the constant in the approximation factor for easier exposition of ideas. For the special case of the (single) ordered k -median problem we can obtain a much better approximation factor. Specifically, this improves upon the factors from [16, 13]. Our technique for this, however, is different from LP-rounding. Instead we give a combinatorial, primal-dual algorithm for the LP (as in our previous work [16]) and our improvement stems from the better notion of proxy costs.

Theorem 9.3. *There is a polynomial time $(5 + \varepsilon)$ -approximation for the ordered k -median problem, for any constant $\varepsilon > 0$.*

9.1 Linear programming relaxation

We begin by restating some notions from Sections 4 and 6 in the clustering setting. As always, we let \vec{o} denote the costs induced by an optimal solution. For convenience, we use $\delta = 1$ in the sparsification described in Section 4. Therefore, the relevant positions for us is $\text{POS} = \text{POS}_{n,1} := \{\min\{2^s, n\} : s \geq 0\}$. 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.

⁵Our results either directly extend, or can be adapted, to the setting where $\mathcal{F} \neq \mathcal{D}$.

⁶Ideally, we would have called this the ordered k -clustering problem since k -median is a special case. We stick to the ordered median name since this is what it is called in the literature.

Given a weight vector $w \in \mathbb{R}_+^n$ (with non-increasing coordinates), we sparsify it to \tilde{w} , that is, for every $r \in [n]$, we set $\tilde{w}_r = w_r$ if $r \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < r < \text{next}(\ell)$, we set $\tilde{w}_r = w_{\text{next}(\ell)}$. Recall from [Claim 4.1](#) that for any vector $v \in \mathbb{R}_+^n$, we have $\text{cost}(\tilde{w}; v) \leq \text{cost}(w; v) \leq 2\text{cost}(\tilde{w}; v)$.

Given any valid *threshold vector* $\vec{t} \in \mathbb{R}_+^{\text{POS}}$ with non-increasing coordinates, we have the proxy function

$$\begin{aligned} \text{prox}_{\vec{t}}(\tilde{w}; v) &:= (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)})\ell \cdot t_\ell + \sum_{j \in \mathcal{D}} h_{\vec{t}}(\tilde{w}; v_j), \quad \text{where} \\ h_{\vec{t}}(\tilde{w}; a) &:= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)})(a - t_\ell)^+ \end{aligned} \quad (12)$$

From [Section 6](#), we know that for the right choice of \vec{t} , the above proxies well-approximate opt . In particular, 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 . As per [Lemma 6.10](#), we focus on the problem of finding an assignment-cost vector \vec{c} that (approximately) minimizes $\sum_{j \in \mathcal{D}} h_{\vec{t}}(\tilde{w}; \vec{c}_j)$. Our LP relaxation below for this is parametrized by the threshold vector \vec{t} .

We augment the standard k -median LP for this *non-metric* k -median problem. A key extra feature is the set of valid constraints [\(OCI-4\)](#). These are crucially exploited in the rounding algorithm. In the sequel, we always use i to index \mathcal{F} and j to index \mathcal{D} .

$$\begin{aligned} \min \quad & \text{CLP}_{\vec{t}}(\tilde{w}; y) := \sum_{j, i} h_{\vec{t}}(\tilde{w}; c_{ij}) x_{ij} & (\text{OCI-P}_{\vec{t}}) \\ \text{s.t.} \quad & \sum_i x_{ij} \geq 1 & \text{for all } j & (\text{OCI-1}) \\ & 0 \leq x_{ij} \leq y_i & \text{for all } i, j & (\text{OCI-2}) \\ & \sum_i y_i \leq k. & & (\text{OCI-3}) \\ & \sum_{i: c_{ij} \leq r} y_i \geq 1 & \forall j, r : \exists \ell \in \text{POS} \text{ s.t. } |\{k \in \mathcal{D} : c_{jk} \leq r - t_\ell\}| > \ell & (\text{OCI-4}) \end{aligned}$$

Remark 9.4. We note that the fractional setting of the y -variables implies the setting of the x -variables: if $c_{ij} < c_{i'j}$, then we use i fully before using i' , that is, if $\bar{x}_{i'j} > 0$ then $\bar{x}_{ij} = \bar{y}_i$. Given y , this is the optimal setting of x since the order of the c_{ij} 's and $h_{\vec{t}}(c_{ij})$'s are the same.

Lemma 9.5. *Let \vec{t} be threshold vector that well-estimates \vec{o}^\downarrow . Then $\text{CLP}_{\vec{t}}(\tilde{w}; y) \leq h_{\vec{t}}(\tilde{w}; \vec{o})$.*

Proof. Consider the optimal solution whose assignment costs are \vec{o} . Consider the solution $y_i^* = 1$ for every opened facility, and $y_i^* = 0$ otherwise. $x_{ij}^* = 1$ if client j is assigned facility i . Note that $\text{CLP}_{\vec{t}}(\tilde{w}; y^*)$ is precisely $h_{\vec{t}}(\tilde{w}; \vec{o})$. Constraints [\(OCI-1\)](#)–[\(OCI-3\)](#), the standard k -median constraints, are clearly satisfied.

We now show that [\(OCI-4\)](#) are also satisfied by y^* . whenever \vec{t} well-estimates \vec{o}^\downarrow . Consider any j, r , and index $\ell \in \text{POS}$. Since $t_\ell \geq \vec{o}_\ell^\downarrow$, at most ℓ clients have assignment cost larger than t_ℓ in this optimal solution. If no facility is opened within the ball $\{i : c_{ij} \leq r\}$, then all the clients k with $c_{jk} \leq r - t_\ell$ will incur assignment cost larger than t_ℓ ; if there are more than ℓ such clients then this cannot happen for this optimal solution, so [\(OCI-4\)](#) holds for this optimal solution. \square

As discussed in Section 7, our approach to min-max ordered optimization is via deterministic, weight-oblivious rounding of an LP for the ordered optimization problem. The theorem below formalizes this for the clustering problem.

Theorem 9.6. (Deterministic weight-oblivious rounding for k -clustering.)

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 (\vec{x}, \vec{y}) satisfying (OCI-1)–(OCI-4), produces a set $F \subseteq \mathcal{F}$ with $|F| \leq k$ and a resulting assignment-cost vector \vec{c} which has the property that for any sparsified weight vector \tilde{w} , we have $\sum_{j \in \mathcal{D}} h_{44\vec{t}}(\tilde{w}; \vec{c}_j) \leq 44 \cdot \text{CLP}_{\vec{t}}(\tilde{w}; \vec{y}) + 40 \sum_{\ell \in \text{POS}} \tilde{w}_{\ell} \text{next}(\ell) t_\ell$.

The theorem implies that (\vec{x}, \vec{y}) is an optimal solution to (OCI-P $_{\vec{t}}$), then we obtain an $O(1)$ -approximation for ordered k -median. We remark that Byrka et al. [13] show that a *randomized* rounding procedure of Charikar and Li [19] for the standard k -median LP has the property that it produces an assignment-cost vector \vec{c} satisfying $\text{Exp}[\sum_{j \in \mathcal{D}} (\vec{c}_j - 19\rho)^+] \leq \sum_{j,i} (c_{ij} - \rho)^+ \vec{x}_{ij}$ for every $\rho \in \mathbb{R}_+$; that is, it gives a threshold-oblivious rounding for ℓ -centrum. Since $\text{cost}(w; v)$ is a nonnegative linear combination of $\text{cost}(\ell; v)$ terms, this also gives a *randomized* weight-oblivious rounding 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). The deterministic property in Theorem 9.6 is crucial and is a key distinction between our guarantee and the one in [13]. Indeed, we need to develop various new ideas to obtain our result.

We prove Theorem 9.6 in Section 9.2. In the remainder of this section, we show how this leads to an $O(1)$ -approximation for both min-max ordered k -median, and minimum-norm k -clustering.

Proof of Theorem 9.2. We let $\text{opt} = \max_{r \in [N]} \text{cost}(w^{(r)}; \vec{o}^\downarrow)$. We sparsify each $w^{(r)}$ to $\tilde{w}^{(r)}$ for all $r \in [k]$, where recall that we set $\delta = 1$ in the procedure of Section 4; so every $\ell \in \text{POS} = \text{POS}_{n,1}$ is of the form $\min\{2^s, n\}$. As described above (before the description of the LP), in polynomial time we have access to a threshold vector \vec{t} which well-estimates the optimal assignment-cost vector \vec{o}^\downarrow . More precisely, we have a polynomial sized set of guesses which contains a well-estimating vector, and for each such guess we do what we describe next, and return the best solution.

We solve an LP similar to (OCI-P $_{\vec{t}}$) with the objective modified to encode the min-max-ness.

$$\begin{aligned} \min \quad & \lambda : \quad (x, y) \text{ satisfies (OCI-1) - (OCI-4)} \\ & \sum_{\ell \in \text{POS}} (\tilde{w}_\ell^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell t_\ell + \text{CLP}_{\vec{t}}(\tilde{w}^{(r)}; y) \leq \lambda \quad \forall r \in [N] \end{aligned}$$

Let (\vec{x}, \vec{y}) be an optimal solution to the above LP. Let \vec{c} be the assignment-cost vector obtained by applying Theorem 9.6 to round (\vec{x}, \vec{y}) . Then, for every $r \in [N]$, we have

$$\begin{aligned} \text{prox}_{44\vec{t}}(\tilde{w}^{(r)}; \vec{c}) &= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell \cdot 44t_\ell + \sum_{j \in \mathcal{D}} h_{44\vec{t}}(\tilde{w}^{(r)}; \vec{c}_j) \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell \cdot 44t_\ell + 44 \cdot \text{CLP}_{\vec{t}}(\tilde{w}^{(r)}; \vec{y}) + 40 \sum_{\ell \in \text{POS}} \tilde{w}_\ell^{(r)} \text{next}(\ell) t_\ell. \end{aligned} \quad (13)$$

The next claim bounds the third term in (13).

Claim 9.7. $\sum_{\ell \in \text{POS}} \tilde{w}_\ell^{(r)} \text{next}(\ell) t_\ell \leq (4 + 10\varepsilon) \text{opt}$.

Proof. We first show $\sum_{\ell \in \text{POS}} \tilde{w}_\ell^{(r)} \text{next}(\ell) t_\ell \leq 4(1 + \varepsilon) \text{cost}(\tilde{w}^{(r)}; \vec{o}^\downarrow) + 3\varepsilon \tilde{w}_1^{(r)} t_1$. This is because every $\ell \in \text{POS}$ is of the form $\min\{2^s, n\}$; so if ℓ' is such that $\text{next}(\ell') = \ell$, we have $\text{next}(\ell) \leq 4(\ell - \ell')$. Furthermore, \vec{t} well-estimates \vec{o}^\downarrow . Therefore, for any ℓ ,

$$\tilde{w}_\ell^{(r)} \text{next}(\ell) t_\ell \leq 4(1 + \varepsilon)(\ell - \ell') \tilde{w}_\ell^{(r)} \vec{o}_\ell^\downarrow + \frac{\varepsilon t_1}{n} \cdot \tilde{w}_1^{(r)} \text{next}(\ell) \quad (14)$$

When we add for all $\ell \in \text{POS}$, the second terms add up to $\leq 3\varepsilon t_1 \tilde{w}_1^{(r)}$ since $\sum_{\ell \in \text{POS}} \text{next}(\ell) \leq 3n$ (again, we use every ℓ is a power of 2 except one in n). Since $t_1 \leq (1 + \varepsilon) \tilde{o}_1^\downarrow$, we get the second terms add up to $\leq 3\varepsilon(1 + \varepsilon) \text{opt} \leq 6\varepsilon \text{opt}$ since $\varepsilon \leq 1$. To argue about the first terms, note

$$(\ell - \ell') \tilde{w}_\ell^{(r)} \tilde{o}_\ell^\downarrow = \sum_{j=\ell'+1}^{\ell} \tilde{w}_\ell^{(r)} \tilde{o}_\ell^\downarrow \leq \sum_{j=\ell'+1}^{\ell} \tilde{w}_j^{(r)} \tilde{o}_j^\downarrow$$

where we have used the non-increasing property of both $\tilde{w}^{(r)}$ and \tilde{o}^\downarrow . Therefore, the first terms of (14) telescope to $\leq 4(1 + \varepsilon) \text{cost}(\tilde{w}^{(r)}; \tilde{o}^\downarrow) \leq 4(1 + \varepsilon) \text{opt}$. \square

Plugging the above in (13) and combining with Lemmas 6.5 and 6.8, we obtain that

$$\begin{aligned} \max_{r \in [N]} \text{cost}(\tilde{w}^{(r)}; \tilde{c}) &\leq \max_{r \in [N]} \text{prox}_{44t}(\tilde{w}^{(r)}; \tilde{c}) \\ &\leq 44 \cdot \max_{r \in [N]} \left(\sum_{\ell \in \text{POS}} (\tilde{w}_{\text{next}(\ell)}^{(r)} - \tilde{w}_\ell^{(r)}) \ell \cdot t_\ell + \text{CLP}_t(\tilde{w}^{(r)}; \tilde{y}) \right) + (160 + 400\varepsilon) \text{opt} \\ &\leq 44 \cdot \max_{r \in [N]} \left(\sum_{\ell \in \text{POS}} (\tilde{w}_{\text{next}(\ell)}^{(r)} - \tilde{w}_\ell^{(r)}) \ell \cdot t_\ell + h_t(\tilde{w}^{(r)}; \tilde{o}^\downarrow) \right) + (160 + 400\varepsilon) \text{opt} \\ &= 44 \cdot \max_{r \in [N]} \text{prox}_t(\tilde{w}^{(r)}; \tilde{o}^\downarrow) + (160 + 400\varepsilon) \text{opt} \\ &\leq 44(1 + 2\varepsilon) \cdot \max_{r \in [N]} \text{cost}(\tilde{w}^{(r)}; \tilde{o}^\downarrow) + (160 + 400\varepsilon) \text{opt} \leq (204 + O(\varepsilon)) \cdot \text{opt}. \end{aligned}$$

The first inequality above is due to Claim 6.5; the second follows by expanding prox and using (13). The third inequality follows from Lemma 9.5. The next equality is simply the definition of prox; the last two inequalities follow from Lemmas 6.8 and 4.2 respectively. Again applying Lemma 4.2 (with $\delta = 1$) gives that $\max_{r \in [L]} \text{cost}(w^{(r)}; \tilde{c}) \leq 2(204 + O(\varepsilon)) \cdot \text{opt}$. \square

Proof of Theorem 9.1. We combine Theorem 5.4 and Theorem 9.2. We only need to show that we can obtain the estimates hi, lb, ub in (A1), (A2), and they lead to the stated running time. The approximation guarantee obtained is $(408 + O(\varepsilon)) \kappa(1 + 3\varepsilon) = \kappa(408 + O(\varepsilon))$.

By scaling, we may assume that $c_{ij} \geq 1$ for every non-zero c_{ij} . Let \tilde{o}^\downarrow be the sorted cost vector induced by an optimal solution. Let $e_i \in \mathbb{R}^m$ denote the vector with 1 in coordinate i , and 0s everywhere else. We can determine in polytime if $\tilde{o}_1^\downarrow = 0$; if not, we have $\tilde{o}_1^\downarrow \geq 1$, and $\text{opt} \geq f(\tilde{o}_1^\downarrow e_1) \geq \text{lb} := f(e_1)$ since f is monotone. In any solution, the assignment cost of any client j is at most $\max_{i \in \mathcal{F}} c_{ij}$. So $\text{opt} \leq f(\{\max_i c_{ij}\}_{j \in \mathcal{D}}) \leq \text{ub} := \sum_j f((\max_i c_{ij}) e_1)$. This also means that $\tilde{o}_1^\downarrow \leq \text{hi} := \sum_j \max_{i \in \mathcal{F}} c_{ij}$, since by monotonicity, we have $\text{opt} = f(\tilde{o}^\downarrow) \geq f(\tilde{o}_1^\downarrow e_1)$. So $\text{ub}/\text{lb} = \text{hi}$ and $\log(\frac{n \cdot \text{ub} \cdot \text{hi}}{\text{lb}}) = \text{poly}(\text{input size})$. So the running time of the reduction in Theorem 5.4, and the size of the min-max ordered- k -median problem it creates, are $\text{poly}(\text{input size}, (\frac{n}{\varepsilon})^{O(1/\varepsilon)})$, and the entire running time is $\text{poly}(\text{input size}, (\frac{n}{\varepsilon})^{O(1/\varepsilon)})$. \square

9.2 Deterministic weight oblivious rounding : proof of Theorem 9.6

Fix a sparsified vector \tilde{w} . This is used *only* in the analysis. Define $\tilde{C}_j := \sum_i c_{ij} \tilde{x}_{ij}$, and $\text{CLP}_j := \sum_i h_t(\tilde{w}; c_{ij}) x_{ij}$ 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 $c(p, S) := \min_{r \in S} c_{pr}$.

Overview. We proceed by initially following the template of the k -median LP-rounding algorithm by Charikar et al. [18, 19], with some subtle but important changes. We cluster clients around nearby centers (which are also clients) as in [18, 19] to ensure that every non-cluster center k is close to some cluster center $j = \text{ctr}(k)$ (step C1). 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, $\text{nbr}(j)$ be the cluster-center (other than itself) nearest to j , and let $a_j := c_{j\text{nbr}(j)}$. 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 $a_{\text{ctr}(k)} = O(\bar{C}_k)$ we have $c(k, F) = O(\bar{C}_k)$, and this will also imply that $h_{\alpha t}(\tilde{w}; c(k, F)) = O(1) \cdot \text{CLP}_k$ for some constant α (see Lemma 9.10). This turns out to be true for non-cluster centers k which are “far away” from their respective cluster centers j . So we can focus on the point that are “near” to their corresponding cluster centers; in the algorithm we use N_j (for “near”) the points near the center j .

Moving each “near” non-cluster center k to $\text{ctr}(k)$ yields a consolidated instance, where at each $j \in D$, we have d_j clients (including j) co-located at j . Clearly, (\bar{x}, \bar{y}) also induces a fractional k -median solution to this consolidated instance. However, *unlike in standard k -median*, it is not in general true that the LP-objective-value $\sum_{j \in D, i} d_j h_t(\tilde{w}; c_{ij}) \bar{x}_{ij}$ of the solution to the consolidated instance is at most the LP-objective-value of (\bar{x}, \bar{y}) . The reason is that while the clustering ensures that $\bar{C}_j \leq \bar{C}_k$ if $j = \text{ctr}(k)$, this does not imply that $\sum_i h_t(\tilde{w}; c_{ij}) \bar{x}_{ij} \leq \sum_i h_t(\tilde{w}; c_{ik}) \bar{x}_{ik}$. Nevertheless, 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 (see Lemma 9.9).

We now focus on rounding the solution to the consolidated instance. As in [18, 19], we can obtain a more-structured fractional solution to this consolidated instance, where every cluster-center j is served to an extent of $\hat{y}_j = \bar{y}(F_j) \geq 0.5$ by itself, and to an extent of $1 - \hat{y}_j$ by $\text{nbr}(j)$. We now perform another clustering step (step C2), where we select some $(j, \text{nbr}(j))$ pairs with the property that every $k \in D$ that is not part of a pair is close to a some j that belongs to a pair, and $a_j \leq a_k$. 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 [18, 19]. Define $t_0 := \infty$ and $\text{next}(0) = 1$. Note that we want to compare the cost of the rounded solution for the consolidated instance to the cost $\sum_{j \in D} d_j h_{\alpha t}(\tilde{w}; a_j)(1 - \hat{y}_j)$ 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 $\ell \in \{0\} \cup \text{POS}$, where $a_j \hat{q}_j^{(\ell)}$ is intended to represent (roughly speaking) $(1 - \hat{y}_j) \times (\min\{a_j, \alpha t_\ell\} - \alpha t_{\text{next}(\ell)})^+$, so that $\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)}$ is $O(h_{\alpha t}(\tilde{w}; a_j)(1 - \hat{y}_j))$. This latter term can be charged to the LP-cost (see Lemma 9.11).

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 $t_\ell = \Omega(a_j)$, $t_{\text{next}(\ell)} \leq a_j$, so that: (3) for every $\ell \in \{0\} \cup \text{POS}$, 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 (OCI-4). Properties (1)–(3) will imply that, for a suitable constant α , the resulting assignment-cost vector \vec{c} for the consolidated instance satisfies $\sum_{j \in D} d_j h_{\alpha t}(\tilde{w}; \vec{c}_j)$ is $O(\text{cost of fractional solution for consolidated instance})$. Finally, Lemma 9.9 (iii) transfers this guarantee to the original instance. We now give the details.

Algorithm.

C1. Clustering I. Let $S \leftarrow \mathcal{D}$, and $D \leftarrow \emptyset$. While $S \neq \emptyset$, we do the following. We pick $j \in S$ with smallest \bar{C}_j . We add j to D . For every $k \in S$ (including j) such that $c_{jk} \leq 4 \max\{\bar{C}_j, \bar{C}_k\}$, we remove k from S , and set $\text{ctr}(k) = j$.

At the end of the above loop, for every $j \in D$, define the following quantities. Let $F_j = \{i : c_{ij} = \min_{j' \in D} c_{ij'}\}$ with ties broken arbitrarily, and $\hat{y}_j := \min\{1, \overline{y}(F_j)\}$.

Define $\text{nbr}(j) = \arg\min_{k \in D: k \neq j} c_{jk}$ if $\hat{y}_j < 1$, again with arbitrary tie-breaking, and $\text{nbr}(j) = j$ otherwise. Let $a_j := c_{j\text{nbr}(j)}$ denote the distance of j to $\text{nbr}(j)$. We define the “near” set $N_j := \{k \in D : \text{ctr}(k) = j, c_{jk} \leq 3a_j/10\}$, and let $d_j := |N_j|$. Let $N := \bigcup_{j \in D} N_j$. The consolidated instance consists of the clients in D , each of which has demand d_j . That is, in the consolidated instance, for every $j \in D$, we move each $k \in N$ to $\text{ctr}(k)$, and drop all other clients.

C2. Clustering II for consolidated instance. We create a collection \mathcal{C} of disjoint clusters, where each cluster consists of at most two nodes of D , as follows. Initialize $S \leftarrow D, \mathcal{C} \leftarrow \emptyset$. While $S \neq \emptyset$, pick $j \in S$ with smallest a_j ; break ties in favor of nodes with $\hat{y}_j = 1$. Add $\{j, \text{nbr}(j)\}$ to \mathcal{C} , and remove every $k \in S$ with $\{k, \text{nbr}(k)\} \cap \{j, \text{nbr}(j)\} \neq \emptyset$.

C3. Auxiliary LP, iterative Rounding, and facility opening. Recall that $t_0 = \infty$ and $\text{next}(0) = 1$. For every $j \in D$, do the following. If $a_j \leq 20t_\ell$ for all $\ell \in \{0\} \cup \text{POS}$, set $\hat{q}_j^{(\ell)} = (1 - \hat{y}_j)(\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ / a_j$ for all $\ell \in \text{POS}$. Otherwise, let $\bar{\ell} \in \text{POS}$ be the smallest index such that $a_j > 20t_{\bar{\ell}}$. For every $\ell \in \{0\} \cup \text{POS}$, set $\hat{q}_j^{(\ell)} = 0$ if $a_j > 20t_\ell$, and $\hat{q}_j^{(\ell)} = (1 - \hat{y}_j)(\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ / (a_j - 10t_{\bar{\ell}})$ otherwise.

Next we consider the following auxiliary LP which we round to open our facilities.

$$\min \sum_{j \in D} d_j a_j q_j^{(0)} \quad (\text{IR2})$$

$$\text{s.t.} \quad \sum_{\ell \in \{0\} \cup \text{POS}} q_j^{(\ell)} \leq 1 \quad \forall j \quad (15)$$

$$\sum_{j \in C} \sum_{\ell \in \{0\} \cup \text{POS}} q_j^{(\ell)} \leq 1 \quad \forall C \in \mathcal{C} \quad (16)$$

$$\sum_{j \in D} \sum_{\ell \in \{0\} \cup \text{POS}} q_j^{(\ell)} \geq |D| - k \quad (17)$$

$$\sum_{j \in D} d_j a_j q_j^{(\ell)} \leq \sum_{j \in D} d_j a_j \hat{q}_j^{(\ell)} \quad \forall \ell \in \text{POS} \quad (18)$$

$$q \geq 0.$$

Later, in [Lemma 9.11](#) we show that \hat{q} is a feasible solution the above LP. We next use [Theorem 7.1](#) to round \hat{q} to an integral point \hat{q}^{int} taking A_1, A_2 to be the constraint matrix of the constraints (15)–(17). We open centers at $F = \{j \in D : \sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_j^{\text{int}(\ell)} = 0\}$. This ends the description of our algorithm.

Analysis. The analysis proceeds in a few steps. In each step we state the main lemmas and prove them later in [Section 9.2.1](#). The first step is to show that moving to the consolidated instance doesn’t cost is much, We start with a standard claim from [18] and its implication on \hat{y}_j ’s.

Lemma 9.8. *If $j, k \in D$, then $c_{jk} \geq 4 \max\{\bar{C}_j, \bar{C}_k\}$.*

This implies that for any $j \in D$ and $i \notin F_j$, $c_{ij} > 2\bar{C}_j$, which in turn implies $\hat{y}_j \geq 1/2$. The next lemma shows that consolidating the clients doesn’t cost much; again note that *unlike the standard k -median case*, the LP-cost of a non-cluster point mayn’t be less than of the cluster center. Nevertheless, the following lemma shows a charging is possible.

Lemma 9.9. *If $k \in \mathcal{D}$ and $j = \text{ctr}(k)$, then $\sum_i h_{5t}(\tilde{w}; c_{ij}) \bar{x}_{ij} \leq 5 \cdot \text{CLP}_k$.*

In our consolidation step, we dropped the “far” away clients. The first statement in the following lemma justifies this; as we show later, our algorithm eventually open a subset $F \subseteq \mathcal{F}$ such that for every client $j \in D$, $c(j, F) \leq 2a_j$ (Lemma 9.12). The second statement shows that if the consolidated instance has a “good” solution, then the clients in N also have “small” connection costs.

Lemma 9.10. *Let $F \subseteq \mathcal{F}$ be such that $c(j, F) \leq 2a_j$ for all $j \in D$. Then, for any $k \in \mathcal{D} \setminus N$, we have $h_{31t}(\tilde{w}; c(k, F)) \leq 31 \cdot \text{CLP}_k$. Also, for any $\theta \geq 0$, we have $\sum_{k \in N} h_{(\theta+4)t}(\tilde{w}; c(k, F)) \leq \sum_{j \in D} d_j h_{\theta t}(\tilde{w}; c(j, F)) + 4 \cdot \sum_{k \in N} \text{CLP}_k$.*

Thus, we need to bound the connection costs of the consolidated instance. This is the heart of the proof. First, we show that the $\hat{q}_j^{(\ell)}$ variables defined in C3 satisfies two properties. The first property is that it is a feasible solution to the auxiliary LP (IR2). The second property shows how the w -weighted combination of these variables corresponding to a client $j \in D$, can be upper bounded by its fractional contribution to the original linear program.

Lemma 9.11. *The vector \hat{q} satisfies the following two conditions. For any $j \in D$, we have*

1. $\hat{q}_j^{(\ell)}$'s are a feasible solution to (IR2).
2. $\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)} \leq 4 \cdot \sum_i h_{5t}(\tilde{w}; c_{ij}) \bar{x}_{ij}$

Since \hat{q}^{int} is obtained by rounding \hat{q} using Theorem 7.1, constraint (17) ensures that the number of facilities we finally open $|F| \leq k$. We first establish that every client in D is at bounded distance from F (as promised earlier). For brevity, we use \vec{c}_j to denote $c(j, F)$.

Lemma 9.12. *We have $F \cap C \neq \emptyset$ for every $C \in \mathcal{C}$, and hence, $\vec{c}_j \leq 2a_j$ for every $j \in D$.*

Proof. Since \hat{q}^{int} in step C3 is obtained by rounding \hat{q} using Theorem 7.1, it satisfies (16), and its support is contained in that of \hat{q} . Since \hat{q}^{int} satisfies (16), if $C \in \mathcal{C}$ is such that $|C| = 2$, then it is immediate that $F \cap C \neq \emptyset$. If $|C| = 1$, say $C = \{k\}$, then we must have $\hat{y}_k = 1$, and so $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_k^{(\ell)} = 0$. Therefore, $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_k^{(\ell)} = 0$, and so $k \in F$.

Consider $j \in D$, and suppose $j \notin F$. Then, there is some $C = \{j', \text{nbr}(j')\} \in \mathcal{C}$ with $a_{j'} \leq a_j$ and $\{j, \text{nbr}(j)\} \cap \{j', \text{nbr}(j')\} \neq \emptyset$. There is some $i \in F \cap C$, and $c_{ij} \leq c_{j \text{nbr}(j)} + c_{j' \text{nbr}(j')} \leq 2a_j$. So $c(j, F) \leq 2a_j$. \square

The next lemma upper bounds the connection cost $\sum_{j \in D} d_j h_{\theta t}(\tilde{w}; \vec{c}_j)$ for some suitable constant θ , by the w -weighted cost of the solution \hat{q}^{int} . Then using Lemma 9.11, as a corollary, this is bounded by the LP-cost.

Lemma 9.13.

$$\sum_{j \in D} d_j h_{40t}(\tilde{w}; \vec{c}_j) \leq 2 \cdot \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \cdot \sum_{j \in D} d_j a_j \hat{q}_j^{(\ell)} + 40 \cdot \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \text{next}(\ell) t_\ell.$$

Corollary 9.14.

$$\sum_{j \in D} d_j h_{40t}(\tilde{w}; \vec{c}_j) \leq 40 \sum_{k \in N} \text{CLP}_k + 40 \cdot \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \text{next}(\ell) t_\ell.$$

Proof. Using [Lemma 9.11](#).(ii) for all $j \in D$, we get

$$\sum_{j \in D} d_j \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)} \leq 4 \sum_{j \in D} \sum_{k \in N_j} \sum_i h_{5t}(\tilde{w}; c_{ij}) \bar{x}_{ij} \leq 20 \sum_{j \in D} \sum_{k \in N_j} \text{CLP}_k$$

where in the first inequality we have used $d_j = |N_j|$ and [Lemma 9.11](#).(ii), and in the second we have used [Lemma 9.9](#). \square

Proof of Theorem 9.6. We need to upper bound $\sum_{j \in D} h_{44t}(\tilde{w}; \vec{c}_j)$. We start by splitting the clients in D into those in N and those not in N , and apply [Lemma 9.10](#) to get the following.

$$\begin{aligned} \sum_{j \in D} h_{44t}(\tilde{w}; \vec{c}_j) &\leq \sum_{k \in N} h_{44t}(\tilde{w}; \vec{c}_k) + \sum_{k \in D \setminus N} h_{31t}(\tilde{w}; \vec{c}_k) \\ &\leq \sum_{j \in D} d_j h_{40t}(\tilde{w}; \vec{c}_j) + 4 \sum_{k \in N} \text{CLP}_k + 31 \cdot \sum_{k \in D \setminus N} \text{CLP}_k \\ &\leq 44 \sum_{k \in N} \text{CLP}_k + 31 \sum_{k \notin N} \text{CLP}_k + 40 \cdot \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \text{next}(\ell) t_\ell. \end{aligned}$$

where the last inequality follows from [Corollary 9.14](#). \square

9.2.1 Proofs of the Lemmas

Proof of Lemma 9.8. This is standard: suppose that j was added to D before k . If $c_{jk} < 4 \max\{\bar{C}_j, \bar{C}_k\}$, then k would have been removed from S at this point, and would never have been added to D . \square

Proof of Lemma 9.9. For the proof of this lemma, and indeed that of [Lemma 9.10](#), one inequality that we will use repeatedly is that for any client k , and any $\rho \geq 0$, we have $(\bar{C}_k - \rho)^+ \leq \sum_i (c_{ik} - \rho)^+ \bar{x}_{ik}$, since x_{ik} 's (ranging over i) form a probability distribution and the $(z)^+$ function is convex. In particular, this implies

$$\sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\bar{C}_k - t_\ell)^+ \leq \text{CLP}_k. \quad (19)$$

Now, since $j = \text{ctr}(k)$, we have $c_{jk} \leq 4\bar{C}_k$. From [Remark 9.4](#), we get that for any $\rho \geq 0$, $\sum_i (c_{ij} - \rho)^+ \bar{x}_{ij} \leq \sum_i (c_{ij} - \rho)^+ \bar{x}_{ik}$. So we have

$$\begin{aligned} \sum_i h_{5t}(\tilde{w}; c_{ij}) \bar{x}_{ij} &= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_i (c_{ij} - 5t_\ell)^+ \bar{x}_{ij} \leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_i (c_{ij} - 5t_\ell)^+ \bar{x}_{ik} \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_i (c_{ik} + 4\bar{C}_k - 5t_\ell)^+ \bar{x}_{ik} \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_i (c_{ik} - t_\ell)^+ \bar{x}_{ik} + 4 \cdot \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\bar{C}_k - t_\ell)^+. \end{aligned}$$

The penultimate and final inequalities follow from [Claim 6.3](#). Using (19), the final expression above is at most $5 \cdot \text{CLP}_k$. \square

Proof of Lemma 9.10. First consider $k \in D \setminus N$ with $j = \text{ctr}(k)$. By definition, we have $\frac{3a_j}{10} \leq c_{jk} \leq 4\bar{C}_k$. Since $c(j, F) \leq 2a_j$, we get $c(j, F) \leq \frac{40}{3}\bar{C}_k$. In turn, this implies $c(k, F) \leq 4\bar{C}_k + c(j, F) \leq \frac{92}{3}\bar{C}_k \leq 31 \cdot \bar{C}_k$. So $h_{31t}(\tilde{w}; c(k, F)) \leq 31 \cdot \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\bar{C}_k - t_\ell)^+ \leq 31 \cdot \text{CLP}_k$, where the first inequality follows from [Claim 6.3](#) and the last inequality follows from (19).

Now consider $j \in D$, and $k \in N$ with $\text{ctr}(k) = j$. Then, $c(k, F) \leq c(j, F) + 4\bar{C}_k$, so again utilizing [Claim 6.3](#), we have

$$\begin{aligned} h_{(\theta+4)\bar{t}}(\tilde{w}; c(k, F)) &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (c(j, F) + 4\bar{C}_k - (\theta+4)t_\ell)^+ \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (c(j, F) - \theta t_\ell)^+ + 4 \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\bar{C}_k - t_\ell)^+ \\ &\leq h_{\theta\bar{t}}(\tilde{w}; c(j, F)) + 4 \cdot \text{CLP}_k. \end{aligned} \quad (20)$$

Adding up these inequalities for all $k \in N$ with $\text{ctr}(k) = j$, and then over all $j \in D$ gives $\sum_{k \in N} h_{(\theta+4)\bar{t}}(\tilde{w}; c(k, F))$ on the LHS and $\sum_{j \in D} d_j h_{\theta\bar{t}}(\tilde{w}; c(j, F)) + \sum_{k \in N} 4 \cdot \text{CLP}_k$. \square

Proof of Lemma 9.11. For every client $j \in D$, we show that $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_j^{(\ell)} = (1 - \hat{y}_j)$. This would imply \hat{q} satisfies (15)-(17), since $\hat{y}_j \geq 1/2$, and y satisfies (23). $\hat{q}_k^{(\ell)}$ satisfies (18) trivially. We also show that for every $j \in D$, $\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)} \leq 2h_{10\bar{t}}(\tilde{w}; a_j)(1 - \hat{y}_j)$. Part (b) of the lemma will follow from [Claim 9.15](#) which is stated and proved below.

Fix $j \in D$. If $a_j \leq 20t_\ell$ for all $\ell \in \{0\} \cup \text{POS}$, then $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_j^{(\ell)} = \frac{1-\hat{y}_j}{a_j} \cdot \sum_{\ell \in \{0\} \cup \text{POS}} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+$. Noting that $\sum_{\ell \in \{0\} \cup \text{POS}} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ = a_j$ (recall that $t_{n+1} = 0$) proves part (a) in this case. Also, $\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)} = (1 - \hat{y}_j) \cdot \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ = (1 - \hat{y}_j) h_{10\bar{t}}(\tilde{w}; a_j)$. The last equality uses the equivalent way of writing $h_{\bar{t}}(\cdot)$ alluded to in [Section 7](#).

In the other case, let $\bar{\ell} \in \text{POS}$ be the smallest index such that $a_j > 20t_{\bar{\ell}}$. We have $a_j \leq 20t_\ell$ for $\ell \in \{0\} \cup \text{POS}$ iff $\ell < \bar{\ell}$. So $\sum_{\ell \in \{0\} \cup \text{POS}} \hat{q}_j^{(\ell)} = \frac{1-\hat{y}_j}{a_j - 10t_{\bar{\ell}}} \cdot \sum_{\ell \in \{0\} \cup \text{POS}: \ell < \bar{\ell}} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+$ and $\sum_{\ell \in \{0\} \cup \text{POS}: \ell < \bar{\ell}} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ = a_j - 10t_{\bar{\ell}}$, so part (a) holds in this case as well. Since $\frac{a_j}{a_j - 10t_{\bar{\ell}}} \leq 2$, we also have $\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} a_j \hat{q}_j^{(\ell)} \leq 2(1 - \hat{y}_j) \cdot \sum_{\ell \in \{0\} \cup \text{POS}: \ell < \bar{\ell}} \tilde{w}_{\text{next}(\ell)} (\min\{a_j, 10t_\ell\} - 10t_{\text{next}(\ell)})^+ \leq 2(1 - \hat{y}_j) h_{10\bar{t}}(\tilde{w}; a_j)$.

As mentioned earlier, the lemma follows from the following easy claim.

Claim 9.15. We have $h_{10\bar{t}}(\tilde{w}; a_j)(1 - \hat{y}_j) \leq 2 \cdot \sum_i h_{5\bar{t}}(\tilde{w}; c_{ij}) \bar{x}_{ij}$ for all $j \in D$.

Proof of Claim 9.15. Fix $j \in D$. For every $i \in F_k$, where $k \in D$, $k \neq j$, we have $c_{jk} \leq c_{ij} + c_{ik} \leq 2c_{ij}$, and so $a_j = c_{j\text{nbr}(j)} \leq 2c_{ij}$. Also $\hat{y}_j \geq \sum_{i \in F_j} \bar{x}_{ij}$, so $1 - \hat{y}_j \leq \sum_{i \notin F_j} \bar{x}_{ij}$. So $h_{10\bar{t}}(\tilde{w}; a_j)(1 - \hat{y}_j) \leq \sum_{i \notin F_j} h_{10\bar{t}}(\tilde{w}; 2c_{ij}) \bar{x}_{ij} \leq 2 \cdot \sum_{i \notin F_j} h_{5\bar{t}}(\tilde{w}; c_{ij}) \bar{x}_{ij}$. \square

\square

Proof of Lemma 9.13. Consider any $\ell \in \text{POS}$, and any $j \in D \setminus F$. [Lemma 9.12](#) implies $\vec{c}_j \leq 2a_j$. By definition (step C3), we have that $\hat{q}_j^{(\ell)} = 0 = \hat{q}_j^{\text{int}(\ell)}$ if $a_j > 20t_\ell$. So we have $\sum_{\ell' \in \text{POS}: \ell' \geq \ell} \vec{c}_j^{\text{int}(\ell')} \hat{q}_j^{\text{int}(\ell')} \leq 40t_\ell$. Since $j \notin F$, we have $\sum_{\ell' \in \{0\} \cup \text{POS}} \hat{q}_j^{\text{int}(\ell')} = 1$. So $(\vec{c}_j - 40t_\ell)^+ \leq \sum_{\ell' \in \{0\} \cup \text{POS}: \ell' < \ell} \vec{c}_j^{\text{int}(\ell')} \hat{q}_j^{\text{int}(\ell')}$. Therefore,

$$\begin{aligned} h_{40\bar{t}}(\tilde{w}; \vec{c}_j) &= \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) (\vec{c}_j - 40t_\ell)^+ \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_{\ell' \in \{0\} \cup \text{POS}: \ell' < \ell} \vec{c}_j^{\text{int}(\ell')} \hat{q}_j^{\text{int}(\ell')} = \sum_{\ell' \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \vec{c}_j^{\text{int}(\ell')} \hat{q}_j^{\text{int}(\ell')}. \end{aligned}$$

Note that the above bound also clearly holds if $j \in F$. [Lemma 9.12](#) shows that $\vec{c}_j \leq 2a_j$ for all $j \in D$. It follows that $\sum_{j \in D} d_j h_{40t}(\tilde{w}; \vec{c}_j) \leq 2 \cdot \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \cdot \sum_{j \in D} d_j a_j q_j^{\text{int}(\ell)}$.

If $\hat{q}_j^{(\ell)} > 0$, we have $\hat{y}_j < 1$ and $10t_{\text{next}(\ell)} < a_j \leq 20t_\ell$. We exploit constraint [\(OCI-4\)](#) to show that if $\hat{q}_j^{(\ell)} > 0$, then $d_j \leq \text{next}(\ell)$. Suppose not. Consider constraint [\(OCI-4\)](#) for client j , $r = \frac{4a_j}{10}$, and consider index $\text{next}(\ell)$. Notice that $k \in N_j$ implies that $c_{jk} \leq \frac{4a_j}{10} - t_{\text{next}(\ell)}$. Since $d_j = |N_j| > \text{next}(\ell)$, [\(OCI-4\)](#) enforces that $\sum_{i: c_{ij} \leq r} \bar{y}_i \geq 1$. But $c_{ij} \leq r$ implies that $i \in F_j$ (otherwise, we would have $a_j \leq 2r$); this means that $\bar{y}(F_j) \geq 1$, and so $\hat{y}_j = 1$, which yields a contradiction.

So $\hat{q}_j^{(\ell)} > 0$ implies that $d_j a_j \leq 20\text{next}(\ell)t_\ell$. By [Theorem 7.1](#), we have that $\sum_{j \in D} d_j a_j q_j^{\text{int}(\ell)}$ is at most $\sum_{j \in D} d_j a_j \hat{q}_j^{(0)}$ if $\ell = 0$, and at most $\sum_{j \in D} d_j a_j \hat{q}_j^{(\ell)} + 20\text{next}(\ell)t_\ell$ otherwise. Therefore

$$\sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \cdot \sum_{j \in D} d_j a_j q_j^{\text{int}(\ell)} \leq \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \cdot \sum_{j \in D} d_j a_j \hat{q}_j^{(\ell)} + 20 \cdot \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \text{next}(\ell) t_\ell.$$

Combining everything, we obtain that

$$\sum_{j \in D} d_j h_{40t}(\tilde{w}; \vec{c}_j) \leq 2 \cdot \sum_{\ell \in \{0\} \cup \text{POS}} \tilde{w}_{\text{next}(\ell)} \cdot \sum_{j \in D} d_j a_j \hat{q}_j^{(\ell)} + 40 \cdot \sum_{\ell \in \text{POS}} \tilde{w}_{\text{next}(\ell)} \text{next}(\ell) t_\ell. \quad \square$$

9.3 Improved primal-dual algorithm for ordered k -median

We now devise a much-improved $(5 + \varepsilon)$ -approximation algorithm for ordered k -median. We sparsify the weight vector $w \in \mathbb{R}_+^n$ to \tilde{w} taking $\delta = \varepsilon$ in [Section 4](#), where $0 < \varepsilon \leq 1$. Let $\text{POS} = \text{POS}_{n, \varepsilon} = \{\min\{\lceil (1 + \varepsilon)^s \rceil, n\} : s \geq 0\}$. By [Lemma 6.9](#), we may assume that we have $\vec{t} \in \mathbb{R}^{\text{POS}}$ such that $\vec{o}_\ell^\downarrow \leq t_\ell \leq (1 + \varepsilon)\vec{o}_\ell^\downarrow$ 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 $\ell \in \text{POS}$. By [Lemma 6.10](#), we can then focus on the problem of finding an assignment-cost vector \vec{c} minimizing $\sum_{i=1}^n h_{\vec{t}}(\tilde{w}; \vec{c}_i)$. We now consider the standard- k -median LP [\(P _{\$\rho\$} \)](#) (i.e., we will not need constraints [\(OCI-4\)](#)), and its dual [\(D _{\$\rho\$} \)](#). Since \tilde{w} is fixed throughout, we abbreviate $h_{\vec{t}}(\tilde{w}; \cdot)$ to $h_{\vec{t}}(\cdot)$.

$$\min \sum_{j,i} h_{\vec{t}}(c_{ij}) x_{ij} \quad (\text{P}_\rho) \quad \max \sum_j \alpha_j - k \cdot \lambda \quad (\text{D}_\rho)$$

$$\text{s.t.} \quad \sum_i x_{ij} \geq 1 \quad \text{for all } j \quad (21) \quad \text{s.t.} \quad \alpha_j \leq h_{\vec{t}}(c_{ij}) + \beta_{ij} \quad \forall i, j \quad (24)$$

$$0 \leq x_{ij} \leq y_i \quad \text{for all } i, j \quad (22) \quad \sum_j \beta_{ij} \leq \lambda \quad \forall i \quad (25)$$

$$\sum_i y_i \leq k. \quad (23) \quad \alpha, \lambda \geq 0.$$

Let $\text{OPT} = \text{OPT}_{\vec{t}}$ denote the common optimal value of [\(P _{\$\rho\$} \)](#) and [\(D _{\$\rho\$} \)](#). So $\text{OPT} \leq \sum_j h_{\vec{t}}(\vec{o}_j^\downarrow)$. Let lb denote a lower bound on opt such that $\log \text{lb}$ is polynomially bounded (e.g., we can take lb to be $\tilde{w}_1 \cdot (\text{estimate of optimal } k\text{-center objective})$). We will be using the following claim which makes simple observations about the $h_{\vec{t}}(\cdot)$ function.

Claim 9.16. *We have: (i) $h_{\vec{t}}(x) \leq h_{\vec{t}}(y)$ for any $x \leq y$; (ii) $h_{\theta_1 \vec{t}}(x) \leq h_{\theta_2 \vec{t}}(x)$ for any $\theta_1 \geq \theta_2$, and any x ; (iii) $h_{(\theta_1 + \theta_2) \vec{t}}(x + y) \leq h_{\theta_1 \vec{t}}(x) + h_{\theta_2 \vec{t}}(y)$ for any θ_1, θ_2, x, y .*

Proof. Part (iii) is the only part that is not obvious. For any $\ell \in \text{POS}$, by part (iii) of [Claim 6.3](#), we If $h_{(\theta_1 + \theta_2) \vec{t}}(x + y) = 0$, then the inequality clearly holds; otherwise, $h_{(\theta_1 + \theta_2) \vec{t}}(x + y) = \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)})(x - x - \rho_1 + y - \rho_2 \leq (x - \rho_1)^+ + (y - \rho_2)^+$. \square

Our algorithm is based on the primal-dual schema coupled with Lagrangian relaxation. For each $\lambda \geq 0$, we describe a primal-dual algorithm to open a good set of facilities, and then vary λ to obtain a convex combination of at most two solutions, called a *bi-point solution* that opens k facilities. Finally, we round this bi-point solution. The primal-dual process for a fixed $\lambda \geq 0$ is very similar to the Jain-Vazirani primal-dual process for k -median, which was also used in [16].

P1. Dual-ascent. Initialize $\mathcal{D}' = \mathcal{D}$, $\alpha_j = \beta_{ij} = 0$ for all $i, j \in \mathcal{D}$, $T = \emptyset$. The clients in \mathcal{D}' are called *active clients*. If $\alpha_j \geq h_t(c_{ij})$, we say that j reaches i . (So if $c_{ij} \leq t_n$, then j reaches i from the very beginning.)

Repeat the following until all clients become inactive. Uniformly raise the α_j s of all active clients, and the β_{ij} s for (i, j) such that $i \notin T$, j is active, and can reach i until one of the following events happen.

- Some client $j \in \mathcal{D}$ reaches some i (and previously could not reach i): if $i \in F$, we *freeze* j , and remove j from \mathcal{D}' .
- Constraint (25) becomes tight for some $i \notin T$: we add i to T ; for every $j \in \mathcal{D}'$ that can reach i , we freeze j and remove j from \mathcal{D}' .

P2. Pruning. Initialize $F \leftarrow \emptyset$. We consider facilities in T in *non-decreasing order of when they were added to T* . When considering facility i , we add i to F if for every $j \in \mathcal{D}$ with $\beta_{ij} > 0$, we have $\beta_{i'j} = 0$ for all other facilities i' currently in F .

P3. Return F as the set of centers. Let $i(j)$ denote the point nearest to j (in terms of c_{ij}) in F .

Define $P(i) := \{j \in \mathcal{D} : \beta_{ij} > 0\}$; for a set $S \subseteq \mathcal{F}$, define $P(S) := \bigcup_{i \in S} P(i)$. The following theorem states the key properties obtained from the primal-dual algorithm.

Theorem 9.17. *The solution returned by the primal-dual algorithm satisfies the following.*

- (i) $3\lambda|F| + \sum_{j \in P(F)} 3h_t(c_{i(j)j}) + \sum_{j \notin P(F)} h_{3t}(c_{i(j)j}) \leq 3 \sum_j \alpha_j$
- (ii) For any $j \in \mathcal{D}$, there is a facility $i \in F$ such that $h_{3t}(c_{ij}) \leq 3h_t(c_{ij}) \leq 3\alpha_j$, and $\alpha_j \geq \alpha_k$ for all $k \in P(i)$.

Proof. Part (i) follows from the analysis in [16], which we can simplify slightly using part (ii). For every $i \in F$ and $j \in P(i)$, we have $i(j) = i$. So $\sum_{j \in P(F)} 3\alpha_j = 3 \sum_{j \in P(F)} (\beta_{i(j)j} + h_t(c_{i(j)j})) = 3\lambda|F| + \sum_{j \in P(F)} 3h_t(c_{i(j)j})$. Consider a client $j \notin P(F)$. By part (ii), which we prove below, there is some $i \in F$ such that $h_{3t}(c_{ij}) \leq 3\alpha_j$, and so $h_{3t}(c_{i(j)j}) \leq 3\alpha_j$ (by Claim 9.16, (i)). This completes the proof of part (i).

Part (ii) is new and follows from our pruning step. Fix $j \in \mathcal{D}$. Consider the facility $i' \in T$ that caused j to freeze. If $i' \in F$, we can take $i = i'$ and we are done. Otherwise, there must be some facility $i \in T$ that was *added before i' to T* such that $P(i) \cap P(i') \neq \emptyset$. Let $\tau_{i'}$ and τ_i denote the times when i' and i were added to T . Then, $\alpha_j \geq \tau_{i'} \geq \tau_i$, and for any client $k \in P(i)$, we have $\alpha_k \leq \tau_i$. \square

For $\lambda = \text{ub} := (n+1)h_{\overline{p}}(\max_{i,j} c_{ij})$, the primal-dual algorithm opens only one facility. We now perform binary search in $[0, \text{ub}]$ to find the “right” λ . If during the binary search, we find some λ such that the above primal-dual algorithm returns F with $|F| = k$, then part (i) of Theorem 9.17 shows that $\sum_j h_{3t}(c_{i(j)j}) \leq 3\text{OPT} \leq 3 \sum_j h_t(\vec{o}^\downarrow)$, and so by Lemma 6.10, we have $\text{cost}(w; \{c_{i(j)j}\}_j) \leq 3(1 + \varepsilon)(1 + 2\varepsilon)\text{opt}$.

Otherwise, we find two sufficiently close values $\lambda_1 < \lambda_2$, primal solutions $(F_1, i_1 : \mathcal{D} \rightarrow F_1)$, $(F_2, i_2 : \mathcal{D} \rightarrow F_2)$, and dual solutions (α^1, β^1) , (α^2, β^2) , obtained for $\lambda = \lambda_1$ and $\lambda = \lambda_2$ respectively, such that $|F_1| > k > |F_2|$. We show here how to utilize F_1 and F_2 to obtain a simpler 9-approximation, and defer the proof of the following improved guarantee to Appendix E.

Theorem 9.18. *Let $0 < \varepsilon \leq 1$. If we continue the binary search until $\lambda_2 - \lambda_1 < \frac{\varepsilon \text{lb}}{n^2 2^n}$, then there is a way of opening k facilities from $F_1 \cup F_2$ so that the resulting solution has $\text{cost}(w; \cdot)$ -cost at most $(5 + O(\varepsilon))\text{opt}$.*

Obtaining a $(9 + O(\varepsilon))$ -approximation. We continue the binary search until $\lambda_2 - \lambda_1 \leq \varepsilon \text{lb}/n$ (assuming we do not find λ for which $|F| = k$). Let $a, b \geq 0$ be such that $ak_1 + bk_2 = k$, $a + b = 1$. A convex combination of F_1 and F_2 yields a feasible bi-point solution that we need to round to a feasible solution. Let $d_{1,j} = h_{3t}(c_{i_1(j)j})$ and $d_{2,j} = h_{3t}(c_{i_2(j)j})$. Let $C_1 := \sum_j d_{1,j}$ and $C_2 := \sum_j d_{2,j}$. Then,

$$\begin{aligned} aC_1 + bC_2 &\leq 3a \left(\sum_j \alpha_{1,j} - k_1 \lambda_1 \right) + 3b \left(\sum_j \alpha_{2,j} - k_2 \lambda_2 \right) \\ &\leq 3a \left(\sum_j \alpha_{1,j} - k \lambda_2 \right) + 3b \left(\sum_j \alpha_{2,j} - k \lambda_2 \right) + 3ak_1(\lambda_2 - \lambda_1) \leq 3OPT + 3\varepsilon \text{lb}. \end{aligned}$$

If $b \geq 0.5$, then $C_2 = \sum_j h_{3t}(c_{i_2(j)j}) \leq 6 \sum_j h_t(\vec{d}_j^\downarrow) + 6\varepsilon \text{lb}$, so F_2 yields a solution of $\text{cost}(\tilde{w}; \cdot)$ -cost at most $6(1 + \varepsilon)(1 + 2\varepsilon) \text{opt} + 6\varepsilon \text{lb}(1 + \varepsilon)$. So suppose $a \geq 0.5$. The procedure for rounding the bi-point solution is similar to that in the Jain-Vazirani algorithm for k -median, except that we derandomize their randomized-rounding step by solving an LP.

- B1. For every $i \in F_2$, let $\sigma(i) \in F_1$ denote the facility in F_1 closest to i (under the c_{ij} distances). If $|\sigma(F_2)| < k_2$, add facilities from F_1 to it to obtain $\overline{F}_1 \subseteq F_1$ such that $\sigma(F_2) \subseteq \overline{F}_1$ and $|\overline{F}_1| = k_2$.
- B2. **Opening facilities.** We will open either all facilities in \overline{F}_1 , or all facilities in F_2 . Additionally, we will open $k - k_2$ facilities from $F_1 \setminus \overline{F}_1$. We formulate the following LP to determine how to do this. Variable θ indicates if we open the facilities in \overline{F}_1 , and variables z_i for every $i \in F_1 \setminus \overline{F}_1$ indicate if we open facility i .

$$\begin{aligned} \min \quad & \sum_{j: i_1(j) \in \overline{F}_1} (\theta d_{1,j} + (1 - \theta) d_{2,j}) + \sum_{k: i_1(k) \notin \overline{F}_1} (z_{i_1(k)} d_{1,k} + (1 - z_{i_1(k)}) (2d_{2,k} + d_{1,k})) \quad (\text{R-P}) \\ \text{s.t.} \quad & \sum_{i \in F_1 \setminus \overline{F}_1} z_i \leq k - k_2, \quad \theta \in [0, 1], \quad z_i \in [0, 1] \quad \forall i \in F_1 \setminus \overline{F}_1. \end{aligned} \quad (26)$$

The above LP is integral, and we open the facilities specified by an integral optimal solution (as discussed above), and assign each client to the nearest open facility.

We prove that: (1) (R-P) has a fractional solution of objective value at most $2(aC_1 + bC_2)$, and (2) any integral solution $(\tilde{\theta}, \tilde{z})$ to (R-P) yields a feasible solution with assignment-cost vector \vec{c} such that $\sum_j h_{9t}(\vec{c}_j)$ is at most the objective value of $(\tilde{\theta}, \tilde{z})$. Together with the bound on $aC_1 + bC_2$, using Lemma 6.10, these imply that the solution returned has $\text{cost}(w; \cdot)$ -cost at most $(1 + \varepsilon)(1 + 2\varepsilon) \cdot 9 \cdot \text{opt} + 6\varepsilon \text{lb}(1 + \varepsilon) \leq (9 + O(\varepsilon)) \text{opt}$.

For the former, consider the solution where we set $\theta = a$, $z_i = a$ for all $i \in F_1 \setminus \overline{F}_1$. We have $\sum_{i \in F_1 \setminus \overline{F}_1} z_i = a(k_1 - k_2) = k - k_2$. Every client j with $i_1(j) \in \overline{F}_1$ contributes $ad_{1,j} + bd_{2,j}$ to the objective value of (R-P), which is also its contribution to $aC_1 + bC_2$. Consider a client k with $i_1(k) \notin \overline{F}_1$. Its contribution to the objective value of (R-P) is $ad_{1,k} + b(2d_{2,k} + d_{1,k}) \leq d_{1,k} + 2bd_{2,k}$, which is at most twice its contribution to $aC_1 + bC_2$ (since $a \geq 0.5$).

For the latter, suppose we have an integral solution $(\tilde{\theta}, \tilde{z})$ to (R-P). Let \vec{c}_j denote the assignment cost of client j under the resulting solution. For every j with $i_1(j) \in \overline{F}_1$, either $i_1(j)$ or $i_2(j)$ is opened, so $h_{9t}(\vec{c}_j) \leq h_{3t}(\vec{c}_j) \leq \tilde{\theta} d_{1,j} + (1 - \tilde{\theta}) d_{2,j}$. Now consider k with $i_1(k) \notin \overline{F}_1$. If $\tilde{z}_{i_1(k)} = 1$, then $h_{3t}(\vec{c}_k) \leq d_{1,k}$. Otherwise, \vec{c}_k is at most $c_{i_2(k)k} + c_{i_2(k)\sigma(i_2(k))} \leq c_{i_2(k)k} + (c_{i_2(k)k} + c_{i_1(k)k})$ since $\sigma(i_2(k))$ is the facility in F_1 closest to $i_2(k)$. Applying Claim 9.16, we then have

$$h_{9t}(\vec{c}_k) \leq h_{6t}(2c_{i_2(k)k}) + h_{3t}(c_{i_1(k)k}) = 2d_{2,k} + d_{1,k}$$

which is the contribution of k to the objective value of $(\tilde{\theta}, \tilde{z})^{\text{int}}$. Therefore, $\sum_j h_{9t}(\tilde{c}_j)$ is at most the objective value of $(\tilde{\theta}, \tilde{z})^{\text{int}}$.

10 Multi-budgeted ordered optimization and simultaneous optimization

In this section we show how the deterministic, weight-oblivious rounding can be used to obtain results for multi-budgeted ordered optimization, which in turn, using the results of Goel and Meyerson [21], implies *constant-factor approximations* to the best simultaneous optimization factor possible for any instance of the unrelated load-balancing and k -clustering problem. We begin by formally defining these problems.

Definition 10.1 (Multi-budgeted ordered optimization). Given a optimization problem where a solution induces a cost vector \vec{v} , given N non-negative, non-increasing weights functions $w^{(1)}, \dots, w^{(N)}$, and N budgets $B_1, \dots, B_N \in \mathbb{R}_+$, the multi-budgeted ordered optimization problems asks whether there exists a solution inducing a cost vector \vec{v} such that $\text{cost}(w^{(r)}; \vec{v}) \leq B_r$ for all $1 \leq r \leq n$.

A ρ -approximation algorithm for this problem would either assert no such solution exists, or furnish a solution inducing a cost vector \vec{v} such that $\text{cost}(w^{(r)}; \vec{v}) \leq \rho \cdot B_r$ for all $1 \leq r \leq n$.

Theorem 10.2. *There are $O(1)$ -factor approximation algorithms for the multi-budgeted ordered (unrelated machines) load-balancing problem and for the multi-budgeted ordered k -clustering problem.*

The following is a slight modification of the definition given in [21] where they used general monotone, symmetric *convex* functions but their notion of approximation scaled the cost-vector by a factor and applied the function on it. As discussed earlier, the definition below implies the same for the original [21] notion.

Definition 10.3 (Optimal simultaneous optimization factor). Given an *instance* \mathcal{I} of an optimization problem, an simultaneous α -approximate solution induces a cost vector \vec{v} such that $g(\vec{v}) \leq \alpha \text{opt}(g)$ where $\text{opt}(g) = \min_{\vec{w}} g(\vec{w})$ where \vec{w} ranges over cost vectors induced by all feasible solutions. Let $\alpha_{\mathcal{I}}^*$ be the smallest α for which an simultaneous α -approximate solution exists for the instance I . This is the best simultaneous optimization factor for this instance.

A ρ -approximation to the best simultaneous optimization factor takes an instance \mathcal{I} and returns a solution \vec{v} such that $g(\vec{v}) \leq \rho \alpha_{\mathcal{I}} \text{opt}(g)$ for any monotone, symmetric norm g .

The following theorem establishes the connections between the two problems via the work of Goel and Meyerson [21].

Theorem 10.4. *A ρ -approximation algorithm for the multi-budgeted ordered optimization problem implies a $\rho(1 + \varepsilon)$ -approximation to the best simultaneous optimization factor for any instance.*

Proof. Using the terminology of Goel and Meyerson [21], a vector $v \in \mathbb{R}_+^n$ is α -submajorized by $w \in \mathbb{R}_+^n$ if and only if for all $1 \leq \ell \leq n$, $\text{Top-}\ell(v) \leq \alpha \cdot \text{Top-}\ell(w)$. That is, for any ℓ , the sum of the ℓ largest entries of v are at most α times the sum of the ℓ largest entries of w . A cost vector v is *globally α -balanced* if it is α -submajorized by any other feasible cost-vector w . Modifying the theory of majorization by Hardy, Littlewood, and Polya [25], Goel and Meyerson [21] establish the following.

Theorem GM (Theorem 2.3, [21] (Paraphrased)). *A solution inducing a cost vector v is simultaneous α -approximate if and only if v is globally α -balanced.*

Fix an instance \mathcal{I} of an optimization problem. For any $1 \leq \ell \leq n$, let $\text{opt}_\ell := \min_w \text{Top-}\ell(w)$ where the minimization is over feasible cost vectors for this instance \mathcal{I} . Let $\alpha_{\mathcal{I}}^*$ be the smallest α for which an α -simultaneous approximate solution exists for the instance I . By **Theorem GM**, this means that there is a solution inducing a cost vector v^* such that for all $1 \leq \ell \leq d$, we have $\text{Top-}\ell(v^*) \leq \alpha_{\mathcal{I}}^* \cdot \text{opt}_\ell$.

Now suppose we knew opt_ℓ for all ℓ . Then we can use the ρ -approximate multi-budgeted optimization algorithm to obtain a ρ -approximate instance optimal solution. There are n weight vectors where $w^{(\ell)}$ has ℓ ones and rest zeros. Via binary search, we find the smallest A such that setting budgets $B_\ell := A \cdot opt_\ell$ and running our ρ -approximation algorithm, we get a feasible solution v . Clearly, $A \leq \alpha_{\mathcal{I}}^*$ since v^* is the certificate for it; and v is globally ρA -balanced. This implies v is a ρ -approximate instance-optimal solution to the simultaneous optimization problem.

However, we don't know opt_ℓ . But once again we can use the sparsification tricks done throughout the paper. First we observe that we need know only estimates of opt_ℓ , and that only for the $\ell \in \text{POS}_{n,\varepsilon} := \{[(1+\varepsilon)^s], n\}$. The latter is because for any $\ell < i < \text{next}(\ell)$, we have $opt_\ell \leq opt_i \leq (1+\varepsilon)opt_\ell$. The first inequality follows from definition and the second inequality follows since in the solution inducing the opt_ℓ solution, the contribution of the coordinates from ℓ to i is at most εopt_ℓ . So any vector v which satisfies $\text{Top-}\ell(v) \leq \alpha \text{Top-}\ell(w)$ for all w only for $\ell \in \text{POS}$, is in fact also a global $\alpha(1+\varepsilon)$ -balanced vector. Therefore, it suffices therefore to know opt_ℓ only for the ℓ s in POS . Furthermore, with another $(1+\varepsilon)$ -loss, we need only know a non-increasing (valid) threshold vector \vec{t} such that $opt_\ell \leq \vec{t}_\ell \leq (1+\varepsilon)opt_\ell$ for $\ell \in \text{POS}$. By Claim 5.3, there are only polynomially many guesses, and for each we perform the binary search procedure described above (but only for $|\text{POS}|$ many weight vectors.) \square

As a corollary, using Theorem 10.2, we get

Theorem 10.5. *There is a constant factor approximation algorithm to the best simultaneous optimization factor of any instance of the unrelated machines load balancing and the k -clustering problem.*

We now prove Theorem 10.2.

Proof of Theorem 10.2. The theorem is a corollary of Theorem 8.7 and Theorem 9.6. We show the proof for load balancing and the proof for clustering is analogous and is omitted from the extended abstract. First we sparsify each weight to \tilde{w} using Lemma 4.2. Suppose there is indeed an assignment $\vec{\sigma}$ which matches all the budgets. Using the enumeration procedure in Lemma 6.9 with $\varepsilon = 1$ and finding a ρ that is a power of 2 such that $\vec{\sigma}_1^\downarrow \leq \rho \leq 2\vec{\sigma}_1^\downarrow$, we assume that we have obtained a valid threshold vector \vec{t} where all t_ℓ s are powers of 2 or 0, and which satisfies the conditions: $\vec{\sigma}_\ell^\downarrow \leq t_\ell \leq 2\vec{\sigma}_\ell^\downarrow$ if $\vec{\sigma}_\ell^\downarrow \geq \frac{\vec{\sigma}_1^\downarrow}{m}$, and $t_\ell = 0$ otherwise.

For each such guess, we try to find a feasible solution to the LP (which is very similar to (4))

$$(x, y, z) \text{ satisfies (OLB1) - (OLB5)} \quad (27)$$

$$\sum_{\ell \in \text{POS}} (\tilde{w}_\ell^{(r)} - \tilde{w}_{\text{next}(\ell)}^{(r)}) \ell t_\ell + \text{LP}_{\vec{t}}(\tilde{w}^{(r)}; x, y, z) \leq 3B_r \quad \forall r \in [N] \quad (28)$$

From the proof of Claim 8.8, we know that if there is an assignment σ^* matching all the budgets, then for some \vec{t} the above LP is feasible. So, if all the LPs return infeasible, we can answer infeasible. Otherwise, we get a solution $(\vec{x}, \vec{y}, \vec{z})$ satisfying (OLB1) - (OLB5), and the threshold vector \vec{t} satisfies the powers of 2 condition. Now we apply Theorem 8.7. We get an assignment $\vec{\sigma}$, and as in the proof of Theorem 8.4, we get for all $r \in [N]$, $\text{cost}(w^{(r)}; \vec{\sigma}) \leq 38(1+\delta)B_r$. \square

11 Acknowledgements

This work started when both authors were visiting the Simons Institute for the Theory of Computing, Berkeley, in their Fall 2017 program of “Bridging Discrete and Continuous Optimization.” We gratefully acknowledge their support and hospitality. DC also thanks David Eisenstat for pointing him to the stochastic fan-out model and its connections with ordered optimization.

Appendices

A Refined sparsification: proof of Lemma 4.2

Recall, $\text{POS}_{n,\delta} := \{\min\{\lceil(1+\delta)^s\rceil, n\} : s \geq 0\}$ and 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 . Similarly we define $\text{prev}(\ell)$. For every index $i \in [n]$, we set $\tilde{w}_i = w_i$ if $i \in \text{POS}$; otherwise, if $\ell \in \text{POS}$ is such that $\ell < i < \text{next}(\ell)$ (note that $\ell < n$), set $\tilde{w}_i = w_{\text{next}(\ell)} = \tilde{w}_{\text{next}(\ell)}$. For notational convenience, we often extend POS to $\{0\} \cup \text{POS} \cup \{n+1\}$; in that case, $\text{next}(0) := 1$ and $\text{next}(n) := n+1$. The weights are extended as $w_0 := \tilde{w}_0 := \infty$ and $w_{n+1} := \tilde{w}_{n+1} = 0$. Lemma 4.2 states that for any $\vec{v} \in \mathbb{R}_+^n$, we have $\text{cost}(\tilde{w}; \vec{v}) \leq \text{cost}(w; \vec{v}) \leq (1+\delta)\text{cost}(\tilde{w}; \vec{v})$. The simple direction $\text{cost}(\tilde{w}; v) \leq \text{cost}(w; v)$ follows since $\tilde{w} \leq w$.

To prove the other direction, fix a cost vector \vec{v} . Let us make a few notation-simplifying definitions. For every $i \in [n]$, let $\alpha_i := w_i \vec{v}_i$ and let $\beta_i := \tilde{w}_i \vec{v}_i$. Thus, both α 's and β 's are non-increasing, and $\alpha_\ell := \beta_\ell$, for all $\ell \in \text{POS}$. For each $\ell \in \text{POS}$, we define the set $J_\ell := \{1, \dots, \ell-1\}$, and so $J_1 := \emptyset$. Also note, $J_{n+1} := [n]$. The proof follows from this simple observation about ceilings.

Claim A.1. For any $\ell \in \text{POS}$, $|J_{\text{next}(\ell)}| \leq (1+\delta)\ell$.

Proof. We need to show that $\text{next}(\ell) - 1 \leq (1+\delta)\ell$ since the LHS is the size of $J_{\text{next}(\ell)}$. First observe that the claim trivially holds for $\ell = n$. So we may assume $\ell := \lceil(1+\delta)^s\rceil$ for some $s \geq 0$. We will use the following observation, $\lceil(1+\delta)z\rceil < 1 + (1+\delta)\lceil z\rceil$ for any non-negative z . This follows since the ceiling of a number is at most one more than it, and the ceiling monotonically increases value. Now, note that $\text{next}(\ell) \leq \lceil(1+\delta)^t\rceil$ where t is the smallest integer $> s$ such that $\lceil(1+\delta)^t\rceil \neq \ell$. (The inequality may occur if $\text{next}(\ell) = n$ instead). Now apply the observation with $z := (1+\delta)^{t-1}$; $\lceil z\rceil = \ell$ by definition. So we get, $\text{next}(\ell) \leq 1 + (1+\delta)\ell$. \square

The proof of Lemma 4.2 now follows easily. First note,

$$\text{cost}(w; \vec{v}) = \sum_{i=1}^n \alpha_i \leq \sum_{\ell \in \text{POS}} |J_{\text{next}(\ell)} \setminus J_\ell| \alpha_\ell = \sum_{\ell \in \text{POS}} \alpha_\ell (|J_{\text{next}(\ell)}| - |J_\ell|) = \sum_{\ell \in \text{POS}} |J_{\text{next}(\ell)}| (\alpha_\ell - \alpha_{\text{next}(\ell)}),$$

where the inequality above follows since α 's are non-increasing. Now we use the fact that $\alpha_\ell = \beta_\ell$ for $\ell \in \text{POS}$, and Claim A.1, to get

$$\text{cost}(w; \vec{v}) \leq (1+\delta) \sum_{\ell \in \text{POS}} \ell (\beta_\ell - \beta_{\text{next}(\ell)}) = (1+\delta) \sum_{\ell \in \text{POS}} \beta_{\text{next}(\ell)} (\text{next}(\ell) - \ell)$$

Using the fact that β 's are non-increasing, we get that the last summand in the RHS is at most the sum of all the β_i 's which is $\text{cost}(\tilde{w}; \vec{v})$. Together, we get $\text{cost}(w; \vec{v}) \leq (1+\delta)\text{cost}(\tilde{w}; \vec{v})$.

B Proofs from Section 5

Proof of Lemma 5.1.

Part (i). Since f is a norm, we have $f(2x) = 2f(x)$ and $f(x/2) = f(x)/2$. Since d is a subgradient at x , we have $f(x) = f(2x) - f(x) \geq d^\top x$ implying, $d^\top x \leq f(x)$. On the other hand, $-f(x)/2 = f(x/2) - f(x) \geq d^\top (-x/2)$, implying $d^\top x \geq f(x)$. Hence $f(x) = d^\top x$.

For any $y \in \mathbb{R}^n$, since $f(y) - f(x) \geq d^\top (y - x)$, using $f(x) = d^\top x$ we get that $f(y) \geq d^\top y$. Also, for any $\lambda \geq 0$, $f(y) - f(\lambda x) = f(y) - f(x) + f(x)(1 - \lambda) \geq d^\top (y - x) + d^\top x(1 - \lambda) = d^\top (y - \lambda x)$. Therefore d is a subgradient of f at λx .

Part (ii). Let \widehat{d} be a subgradient of f at x . Define $d = (\widehat{d})^+ := \{(\widehat{d}_i)^+\}_{i \in [n]}$. We first claim that if $x_i > 0$, then $\widehat{d}_i \geq 0$. Suppose not. Let $x^{(-i)}$ denote the vector with $x_j^{(-i)} = x_j$ for all $j \neq i$, and $x_i^{(-i)} = 0$. So $f(x) \geq f(x^{(-i)})$ by monotonicity. But $f(x^{(-i)}) - f(x) \geq \widehat{d}_i(-x_i) > 0$, which yields a contradiction.

Consider $y \in \mathbb{R}^n$. Let $I = \{i \in [n] : \widehat{d}_i \geq 0\}$. Define $y'_i = y_i$ if $i \in I$ and 0 otherwise. By the above, we know that $x_i = 0$ for $i \notin I$. By monotonicity, we have $f(y) \geq f(y')$, so

$$f(y) - f(x) \geq f(y') - f(x) \geq \sum_{i \in [n]} \widehat{d}_i(y'_i - x_i) = \sum_{i \in I} \widehat{d}_i(y'_i - x_i) = \sum_{i \in I} d_i(y'_i - x_i) = \sum_{i \in [n]} d_i(y_i - x_i).$$

The last equality follows since $d_i = 0$ for all $i \notin I$, and $y_i = y'_i$ for all $i \in I$.

Part (iii). Suppose there are $i, j \in [n]$ such that $d_i < d_j$ but $x_i > x_j$. Let x' be the vector obtained from x by swapping x_j and x_i : i.e., $x'_k = x_k$ for all $k \in [n] \setminus \{i, j\}$, $x'_i = x_j$, $x'_j = x_i$. Then, $f(x') - f(x) \geq d^T(x' - x) = (d_i - d_j)(x_j - x_i) > 0$, but $f(x') = f(x)$ due to symmetry, which gives a contradiction. This also implies that if $d_i > d_j$, then $x_i \geq x_j$. It follows that there is a common permutation $\kappa : [n] \rightarrow [n]$ such that $d_{\kappa(1)} \geq \dots \geq d_{\kappa(n)}$ and $x_{\kappa(1)} \geq \dots \geq x_{\kappa(n)}$. Hence, $f(x) = d^T x = \sum_{i \in [n]} d_{\kappa(i)} x_{\kappa(i)} = d^\downarrow \cdot x^\downarrow = \text{cost}(d^\downarrow; x)$.

We have $f(x^{(\pi)}) = f(x) = d^T x = d^{(\pi)} \cdot x^{(\pi)}$. For any $y \in \mathbb{R}_n$, we have $f(y) = f(y^{(\pi^{-1})}) \geq d^T y^{(\pi^{-1})} = d^{(\pi)} \cdot y$, so $f(y) - f(x) \geq d^{(\pi)} \cdot (y - x)$. This shows that $d^{(\pi)}$ is a subgradient of f at $x^{(\pi)}$. \square

Proof of Claim 5.3. Any non-decreasing sequence $a_1 \geq a_2 \geq \dots \geq a_k$, where $a_i \in \{0\} \cup [N]$ for all $i \in [k]$, can be mapped bijectively to the set of $k+1$ integers $N - a_1, a_1 - a_2, \dots, a_{k-1} - a_k, a_k$ from $\{0\} \cup [N]$ that add up to N . The number of such sequences of $k+1$ integers is equal to the coefficient of x^N in the generating function $(1 + x + \dots + x^N)^k$. This is equal to the coefficient of x^N in $(1 - x)^{-k}$, which is $\binom{N+k-1}{N}$ using the binomial expansion. Let $M = \max\{N, k-1\}$. We have $\binom{N+k-1}{N} = \binom{N+k-1}{M} \leq \left(\frac{e(N+k-1)}{M}\right)^M \leq (2e)^M$. \square

Proof of Theorem 5.4. We first bound the number of oracle calls to \mathcal{A} . By Claim 5.3, since the enumeration of u_1, \dots, u_{ℓ^*} involved in \mathcal{W}' requires guessing a non-increasing sequence of $O(\log n/\varepsilon)$ exponents from a range of size $O(\frac{1}{\varepsilon} \log(\frac{n}{\varepsilon}))$, we have

$$|\mathcal{W}'| = O(|\text{POS}| \cdot \frac{1}{\varepsilon} \log(\frac{n \cdot \text{ub} \cdot \text{hi}}{\text{lb}}) (\frac{n}{\varepsilon})^{O(1/\varepsilon)}) = O(\frac{\log n}{\varepsilon^2} \log(\frac{n \cdot \text{ub} \cdot \text{hi}}{\text{lb}}) (\frac{n}{\varepsilon})^{O(1/\varepsilon)}).$$

The latter quantity is thus a bound on the number of calls to \mathcal{A} and $|\mathcal{W}|$.

We now prove parts (i) and (ii), from which the final guarantee will follow easily. For part (i), consider any $w \in \mathcal{W}$. If $w = \frac{\text{lb}}{n \cdot \text{hi}} \mathbb{1}^n$, then $\text{cost}(w; \vec{v}) \leq \frac{\text{lb}}{n \cdot \text{hi}} \sum_{i \in [n]} \vec{v}_i$. Otherwise, we know that $\text{Bopt}(w) \leq \kappa(1 + \varepsilon)$. So $w^T \vec{v}^\downarrow / f(\vec{v}^\downarrow) \leq \text{Bopt}(w) \leq \kappa(1 + \varepsilon)$. Hence, $w^T \vec{v}^\downarrow \leq \kappa(1 + \varepsilon) f(\vec{v}^\downarrow)$, or equivalently, $\text{cost}(w; \vec{v}) \leq \kappa(1 + \varepsilon) f(\vec{v})$.

For part (ii), it suffices to show, due to Lemma 5.2, that $\text{cost}(d; \vec{v})$ is at most the stated bound, for every $d \in \mathcal{C}$. So fix $d \in \mathcal{C}$. If $d_1 < \frac{\text{lb}}{n \cdot \text{hi}}$, then $d < \frac{\text{lb}}{n \cdot \text{hi}} \cdot \mathbb{1}^n$, so $\text{cost}(d; \vec{v}) < \text{cost}(\frac{\text{lb}}{n \cdot \text{hi}} \cdot \mathbb{1}^n; \vec{v}) < (1 - \varepsilon)^{-1} \max_{w \in \mathcal{W}} \text{cost}(w; \vec{v})$. So suppose otherwise.

Let $\ell^* \in \text{POS}$ be the largest index for which $d_\ell \geq \frac{\varepsilon d_1}{n}$. Since $d_1 \in [\frac{\text{lb}}{n \cdot \text{hi}}, \text{ub}]$, there is some \tilde{w}_1 that is a power of $(1 + \varepsilon)$ such that $d_1 \leq \tilde{w}_1 \leq (1 + \varepsilon)d_1$. For every $\ell \in \text{POS}$, $\ell \leq \ell^*$, we have $d_\ell \geq \frac{\varepsilon \tilde{w}_1}{n(1 + \varepsilon)}$. Hence, there are non-increasing \tilde{w}_ℓ values that are powers of $(1 + \varepsilon)$ satisfying $\tilde{w}_\ell \leq d_\ell \leq (1 + \varepsilon)\tilde{w}_\ell$ for all $\ell \in \text{POS}$, $\ell \leq \ell^*$. Thus, there is some $\tilde{w} \in \mathcal{W}'$ such that $d_\ell \leq \tilde{w}_\ell \leq (1 + \varepsilon)d_\ell$ for all $\ell \geq \ell^*$ in POS , and $\tilde{w}_\ell = 0$ for all other $\ell \in \text{POS}$.

Next, we claim that $\tilde{w} \in \mathcal{W}$. For every $x \in \mathbb{R}_+^n$, we have $\text{cost}(d; x) \leq \sum_{i: d_i \geq \varepsilon d_1/n} \tilde{w}_i x_i^\downarrow + \frac{\varepsilon d_1}{n} \cdot n x_1^\downarrow$, so $(1 - \varepsilon)\text{cost}(d; x) \leq \text{cost}(\tilde{w}; x)$. Also, since $\tilde{w} \leq (1 + \varepsilon)d$, we have $\text{cost}(\tilde{w}; x) \leq (1 + \varepsilon)\text{cost}(d; x)$. Since $d \in \mathcal{C}$, we have $\max_{x \in \mathbb{B}_+(f)} d^T x = \max_{x \in \mathbb{B}_+(f)} \text{cost}(d; x) = 1$. It follows that

$$\max_{x \in \mathbb{B}_+(f)} \tilde{w}^T x = \max_{x \in \mathbb{B}_+(f)} \text{cost}(\tilde{w}; x) \in \left[(1 - \varepsilon) \max_{x \in \mathbb{B}_+(f)} \text{cost}(d; x), (1 + \varepsilon) \max_{x \in \mathbb{B}_+(f)} \text{cost}(d; x) \right] = [1 - \varepsilon, 1 + \varepsilon].$$

Therefore, the point $\hat{x} \in \mathbb{B}_+(f)$ returned by \mathcal{A} on \tilde{w} satisfies $\tilde{w}^T \hat{x} \in [(1 - \varepsilon)/\kappa, 1 + \varepsilon]$ showing that $\tilde{w} \in \mathcal{W}$.

Finally, since $\text{cost}(d; \vec{v}) \leq \text{cost}(\tilde{w}; \vec{v})/(1 - \varepsilon)$, this implies that $\text{cost}(d; \vec{v}) \leq (1 - \varepsilon)^{-1} \max_{w \in \mathcal{W}} \text{cost}(w; \vec{v})$.

The final approximation guarantee of the theorem now follows easily. The optimum of the min-max ordered-optimization problem is at most $\max_{w \in \mathcal{W}} \text{cost}(w; \vec{o})$, which by part (i) is at most $\max\{\kappa(1 + \varepsilon)\text{opt}, \frac{\text{lb}}{n \cdot \text{hi}} \cdot n \cdot \vec{o}_1^\downarrow\} \leq \kappa(1 + \varepsilon)\text{opt}$. Therefore, $\max_{w \in \mathcal{W}} \text{cost}(w; \tilde{v}) \leq \gamma\kappa(1 + \varepsilon)\text{opt}$. By part (ii), this implies that $f(\tilde{v}) \leq \gamma\kappa \cdot \frac{1 + \varepsilon}{1 - \varepsilon} \cdot \text{opt} \leq \gamma\kappa(1 + 3\varepsilon)\text{opt}$. \square

C Proofs from Section 6

Proof of Claim 6.7. Consider the difference $\text{prox}_{\vec{t}}(\tilde{w}; v) - \text{prox}_{\vec{t}'}(\tilde{w}; v)$. Since $\vec{t} \leq \vec{t}'$, only the second term in (1) has a nonnegative contribution to this difference, and only indices $\ell \in \text{POS}$ for which $t_\ell \leq t'_\ell$ contribute non-negatively. The total contribution from such indices is at most $\sum_{\ell \in \text{POS}: t_\ell \leq t'_\ell} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \sum_{i=1}^n \Delta \leq n\tilde{w}_1\Delta$. Similarly, only the first (i.e., constant) term in (1) has a nonnegative contribution to the difference $\text{prox}_{\vec{t}'}(\tilde{w}; v) - \text{prox}_{\vec{t}}(\tilde{w}; v)$, and this contribution is at most $\sum_{\ell \in \text{POS}: t_\ell \geq t'_\ell} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \ell \Delta \leq n\tilde{w}_1\Delta$. \square

Proof of Lemma 6.10. The second inequality follows immediately from the first one and Lemma 4.2, so we focus on showing the first inequality. By Claim 6.5, we have

$$\begin{aligned} \text{cost}(\tilde{w}; v) &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \ell \cdot \theta t_\ell + \sum_{i=1}^n h_{\theta \vec{t}}(\tilde{w}; v_i) \\ &\leq \sum_{\ell \in \text{POS}} (\tilde{w}_\ell - \tilde{w}_{\text{next}(\ell)}) \theta \ell t_\ell + \gamma \sum_{i=1}^n h_{\vec{t}}(\tilde{w}; \vec{o}_i^\downarrow) + M \\ &\leq \max\{\theta, \gamma\} \text{prox}_{\vec{t}}(\tilde{w}; \vec{o}^\downarrow) + M \leq \max\{\theta, \gamma\} (1 + 2\varepsilon) \text{cost}(\tilde{w}; \vec{o}) + M. \end{aligned}$$

The last inequality follows from Lemma 6.8. \square

D Iterative rounding of linear system: proof of Theorem 7.1

We first prove some properties of an extreme point of (Q). We call the constraints $Bq \leq d$, budget constraints. Let N be the number of budget constraints.

Lemma D.1. *Let q' be an extreme point of (Q). Then either q' is integral, or there is some tight budget constraint $(*)$ with support S such that $\sum_{j \in S: q'_j > 0} (1 - q'_j) \leq k$.*

Proof. Let T denote the support of q' . It is well known (see, e.g., [39]) that then there is an invertible submatrix A' of the constraint-matrix of (Q), whose columns correspond to the support T , and rows correspond to $|T|$ linearly-independent constraints that are tight at q' . So if q'' denotes the vector comprising the q_j variables for $j \in T$, and g denotes the right-hand-sides of these tight constraints, then q' is the unique solution to the system $A'q'' = g$.

If A' does not consist of any budget constraints, then the supports of the rows of A' form a laminar family, and it is well known that such a matrix is totally unimodular (TU). So since $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ is integral, q' is integral. So if q' is not integral, then A' contains at least one budget constraint.

Let \mathcal{L} denote the laminar family formed by the supports of the rows of A' corresponding to the $A_1q \leq b_1$, $A_2q \geq b_2$ constraints. Consider the following token-assignment scheme. Every $j \in T$ supplies q'_j tokens to the row of A' corresponding to the smallest set of \mathcal{L} containing j (if such a row exists), and $(1 - q'_j)/k$ tokens to the at most k budget constraints of A' where it appears. Thus, every $j \in T$ supplies at most one token unit overall, and the total supply of tokens is at most $|T|$.

Notice that every row i of A' corresponding to a constraint from $A_1q \leq b_1$ or $A_2q \geq b_2$ consumes at least 1 token unit: let $L \in \mathcal{L}$ be the support of row i , and $L' \subsetneq L$ be the largest set of $\mathcal{L} \cup \{\emptyset\}$ strictly contained in L . If $L' \neq \emptyset$, let i' be the row of A' corresponding to set L' . Row i consumes $\sum_{j \in L \setminus L'} q'_j$ tokens, which is equal to $(A'q')_i - (A'q')_{i'}$ if $L' \neq \emptyset$, and equal to $(A'q')_i$ otherwise. This quantity is an integer, and strictly positive (since all q'_j s are positive), so is at least 1. Suppose for a contradiction that, for every row i corresponding to a budget constraint of A' , $\sum_{j \in T: A'_{ij} > 0} (1 - q'_j) > k$. Then every constraint of A' consumes at least 1 token unit, and at least one constraint consumes more than 1 token unit. This yields a contradiction since the total consumption of tokens is larger than (number of constraints of A') $= |T|$.

Hence, if q' is not integral, there must be some tight budget constraint $(*)$ (in fact, a budget constraint of A') with support S such that $\sum_{j \in S: q'_j > 0} (1 - q'_j) \leq k$. \square

The iterative-rounding algorithm for rounding \hat{q} is as follows. We initialize $q = \hat{q}$, and our current system of constraints to the constraints of **(Q)**. We repeat the following until we obtain an integral solution.

- I1. Move from q to an extreme-point q' of the current system of constraints no greater objective value (under **(Q)**) whose support is contained in the support of q . If q' is not integral, by [Lemma D.1](#) there is some tight budget constraint $(*)$ with support S such that $\sum_{j \in S: q'_j > 0} (1 - q'_j) \leq k$.
- I2. Set $q \leftarrow q'$. If q is not integral then update the system of constraints by dropping $(*)$ (and go to [step 8.10](#)); otherwise, return $\overset{\text{int}}{q} := q$.

We prove that the above process terminates, and the point $\overset{\text{int}}{q}$ returned satisfied the stated properties. In each iteration, we drop a budget constraint, and there are N budget constraints, so we terminate in at most N iterations. By definition, we terminate with an integral point. We never increase the objective value, and always stay within the support of \hat{q} , so properties (a) and (b) hold. We never drop a constraint from $A_1q \leq b_1$, $A_2q \geq b_2$ from our system, so the final point $\overset{\text{int}}{q}$ satisfies these constraints. Since $q_j \leq 1$ is an implicit constraint implied by these constraints (and $\overset{\text{int}}{q}$ is integral), this implies that $\overset{\text{int}}{q} \in \{0, 1\}^n$.

Finally, we prove part (d). Consider a budget constraint $(Bq)_i \leq d_i$. If we never drop this budget constraint during iterative rounding, then $\overset{\text{int}}{q}$ satisfies this constraint. Otherwise, consider the iteration when we drop this constraint and the extreme point q' obtained in **I1** just before we drop this constraint. Then, if S is the support of this budget constraint, it must be that $(Bq')_i \leq d_i$ and $\sum_{j \in S: q'_j > 0} (1 - q'_j) \leq k$. Also, the support of $\overset{\text{int}}{q}$ is contained in the support of q' . Therefore,

$$\begin{aligned} \sum_j B_{ij} \overset{\text{int}}{q}_j &\leq \sum_{j \in S: q'_j > 0} B_{ij} = \sum_{j \in S: q'_j > 0} B_{ij} q'_j + \sum_{j \in S: q'_j > 0} B_{ij} (1 - q'_j) \\ &\leq (Bq')_i + k \left(\max_{j \in S: q'_j > 0} B_{ij} \right) \leq (Bq')_i + k \left(\max_{j: q'_j > 0} B_{ij} \right) = d_i + k \left(\max_{j: q'_j > 0} B_{ij} \right). \end{aligned}$$

The last inequality follows since $q'_j > 0$ implies that $\hat{q}_j > 0$. \square

E Improved $(5 + O(\varepsilon))$ -approximation for ordered k -median

In this section, we prove [Theorem 9.18](#). Recall that we continue the binary search until $\lambda_2 - \lambda_1 < \frac{\varepsilon \text{lb}}{n^2 2^n}$. For $r = 1, 2$, and $i \in \mathcal{F}$, define $P^r(i) := \{j \in \mathcal{D} : \beta_{ij}^r > 0\}$; for a set $S \subseteq \mathcal{F}$, define $P^r(S) := \bigcup_{i \in S} P^r(i)$. A continuity argument from [\[17\]](#) shows the following; we defer the proof to the end of this section.

Lemma E.1 ([\[17\]](#)). *We have $\|\alpha^1 - \alpha^2\|_\infty \leq 2^n(\lambda_2 - \lambda_1) \leq \frac{\varepsilon \text{lb}}{n^2}$. Hence, for any $i \in F_1 \cup F_2$, and any $r \in \{1, 2\}$, we have $\sum_j \beta_{ij}^r \geq \lambda_2 - \frac{\varepsilon \text{lb}}{n}$.*

For every $i \in \mathcal{F}$, $j \in \mathcal{D}$, define $\alpha_j := \max\{\alpha_j^1, \alpha_j^2\}$, and $\beta_{ij} := \max\{\beta_{ij}^1, \beta_{ij}^2\}$; note that $\beta_{ij} = (\alpha_j - h_{\bar{t}}(c_{ij}))^+$.

To obtain the improvement, we utilize insights from the 4-approximation algorithm for k -median in [\[17\]](#). The idea is to first augment F_1 using facilities from F_2 (that are approximately paid for by (α^1, β^1)), and then open facilities in a similar manner as before. The augmentation step will ensure that for every client j , there is some facility i that is opened with $h_{5\bar{t}}(c_{ij}) \leq 5\alpha_j$, and this leads to the 5-approximation guarantee.

- D1. **Augmenting F_1 .** Augment F_1 to a maximal set $F'_1 \supseteq F_1$ by adding facilities from F_2 while preserving the following property: for every $j \in \mathcal{D}$, there is at most one $i \in F'_1$ with $\beta_{ij}^1 > 0$. For every $j \in \mathcal{D}$, redefine $i_1(j)$ to be the facility in F'_1 that is closest (in terms of c_{ij}) to j .
- D2. Let $k'_1 = |F'_1|$, $k_2 = |F_2|$. For every $i \in F_2$, let $\sigma(i) \in F'_1$ denote the facility in F'_1 closest to i (which will be i if $i \in F'_1$). Let $\bar{F}_1 \subseteq F'_1$ be an arbitrary set such that $\sigma(F_2) \subseteq \bar{F}_1$ and $|\bar{F}_1| = k_2$.
- D3. **Opening facilities.** As before, we will open either all facilities in \bar{F}_1 or all facilities in F_2 , and we will also open $k - k_2$ facilities from $F'_1 \setminus \bar{F}_1$. To do this, we utilize an LP with the same variables and constraints as (R-P): variable θ to indicate if we open the facilities in \bar{F}_1 , and variables z_i for every $i \in F'_1 \setminus \bar{F}_1$ to indicate if we open facility i . But we use a different objective function. For each client j , we define an expression $A_j(\theta, z := \{z_i\}_{i \in F'_1 \setminus \bar{F}_1})$ that will serve as an upper bound on $h_{5\bar{t}}(\text{assignment cost of } j)$ when θ and z are integral, and our LP will seek to minimize $\sum_j A_j(\theta, z)$. Define

$$A_j(\theta, z) = \begin{cases} \theta h_{\bar{t}}(c_{i_1(j)j}) + (1 - \theta) h_{\bar{t}}(c_{i_2(j)j}) & i_1(j) \in \bar{F}_1, j \in P^1(F'_1) \cap P^2(F_2); \\ h_{\bar{t}}(c_{i_1(j)j}) + (1 - z_{i_1(j)}) \cdot 2h_{\bar{t}}(c_{i_2(j)j}) & i_1(j) \notin \bar{F}_1, j \in P^1(F'_1) \cap P^2(F_2); \\ (1 - \theta) h_{\bar{t}}(c_{i_2(j)j}) + \theta \cdot 5\alpha_j & j \in P^2(F_2) \setminus P^1(F'_1); \\ (1 - \theta) h_{3\bar{t}}(c_{i_2(j)j}) + \theta \cdot 5\alpha_j & j \notin P^1(F'_1) \cup P^2(F_2); \\ \theta \cdot h_{\bar{t}}(c_{i_1(j)j}) + (1 - \theta) \cdot 5\alpha_j & i_1(j) \in \bar{F}_1, j \in P^1(F'_1) \setminus P^2(F_2); \\ z_{i_1(j)} \cdot h_{\bar{t}}(c_{i_1(j)j}) + (1 - z_{i_1(j)}) \cdot 5\alpha_j & i_1(j) \notin \bar{F}_1, j \in P^1(F'_1) \setminus P^2(F_2); \end{cases}$$

We solve the following LP:

$$\min \sum_j A_j(\theta, z) \quad \text{s.t.} \quad \sum_{i \in F'_1 \setminus \bar{F}_1} z_i \leq k - k_2, \quad \theta \in [0, 1], \quad z_i \in [0, 1] \quad \forall i \in F'_1 \setminus \bar{F}_1. \quad (\text{O-P})$$

The above LP is integral, and we open the facilities specified by an integral optimal solution (as discussed above), and assign each client to the nearest open facility.

Analysis. The road map of the analysis is as follows. Recall that $\alpha_j = \max\{\alpha_j^1, \alpha_j^2\}$ and $\beta_{ij} = \max\{\beta_{ij}^1, \beta_{ij}^2\}$. We first show that by combining [Lemma E.1](#) and [Theorem 9.17](#), we can infer two things (see [Lemma E.2](#)):

- (1) for both the F'_1 and F_2 solutions, $\sum_j 3\alpha_j$ can be used to pay for the λ_2 -cost of all open facilities and

$\sum_j h_{3t}(\text{assignment cost of } j)$; (2) for every client j , due to our augmentation step C1, we have facilities $i \in F_2, i' \in F'_1$ such that i is close to j , and i' is close to i .

Next, we show that the optimal value of (O-P) is (roughly) at most $5OPT$ (Lemma E.3). Finally, we show that if we have an integral solution $(\tilde{\theta}, \tilde{z}^{\text{int}})$ to (O-P), then this yields a solution $\sum_j h_{5t}(\text{assignment cost of } j)$ is (roughly) bounded by $\sum_j A_j(\tilde{\theta}, \tilde{z}^{\text{int}})$ (Lemma E.4). Here, we use property (2) above to argue that for every client j , there is some facility i opened in our final solution with $h_{5t}(c_{ij})$ bounded by (roughly) $5\alpha_j$. Combining Lemmas E.3 and E.4 yields Theorem 9.18.

Lemma E.2. *The following hold.*

- (i) $3\lambda_2|F'_1| + \sum_{j \in P^1(F'_1)} 3h_t(c_{i_1(j)j}) + \sum_{j \notin P^1(F'_1)} h_{3t}(c_{i_1(j)j}) \leq 3 \sum_j \alpha_j + 3\varepsilon \text{lb}$.
- (ii) $3\lambda_2|F_2| + \sum_{j \in P^2(F_2)} 3h_t(c_{i_2(j)j}) + \sum_{j \notin P^2(F_2)} h_{3t}(c_{i_2(j)j}) \leq 3 \sum_j \alpha_j$.
- (iii) *For any $j \in \mathcal{D}$, there are facilities $i \in F_2$ and $i' \in F'_1$ such that $h_{3t}(c_{ij}) \leq 3\alpha_j$, and $h_{2t}(c_{ii'}) \leq 2\alpha_j + \frac{2\varepsilon \text{lb}}{n^2}$.*

Proof. Part (ii) follows immediately from part (i) of Theorem 9.17.

Consider part (i). Since $F'_1 \subseteq F_1 \cup F_2$, by Lemma E.1, for every $i \in F'_1$, we have that $\sum_j \beta_{ij}^1 \geq \lambda_2 - \frac{\varepsilon \text{lb}}{n}$. When adding facilities to F_1 in step C1, we ensure that the sets $\{P^1(i)\}_{i \in F'_1}$ remain pairwise disjoint. For every client j , we know that if $\beta_{ij}^1 > 0$ for some $i \in F'_1$, then $i_1(j) = i$; we also know from part (ii) of Theorem 9.17 that $h_{3t}(c_{i_1(j)j}) \leq 3\alpha_j^1$. So

$$\begin{aligned} \sum_j 3\alpha_j &\geq \sum_j 3\alpha_j^1 \geq \sum_{i \in F'_1} \sum_{j \in P^1(i)} 3(\beta_{ij}^1 + h_t(c_{ij})) + \sum_{j \notin P^1(F'_1)} h_{3t}(c_{i_1(j)j}) \\ &\quad 3\lambda_2|F'_1| - \frac{3|F'_1|\varepsilon \text{lb}}{n} + \sum_{j \in P^1(F'_1)} 3h_t(c_{i_1(j)j}) + \sum_{j \notin P^1(F'_1)} h_{3t}(c_{i_1(j)j}). \end{aligned}$$

To prove part (iii), consider any client j . By Theorem 9.17 (ii), we know that there is some $i \in F_2$ such that $h_{3t}(c_{ij}) \leq 3\alpha_j^2 \leq 3\alpha_j$, and $\alpha_j^2 \geq \alpha_k^2$ for all $k \in P^2(i)$. If $i \in F'_1$, then taking $i' = i$ finishes the proof. Otherwise, since i was not added to F'_1 in step C1, it must be that there is some client k and some facility $i' \in F'_1$ such that $\beta_{ik}^1, \beta_{i'k}^1 > 0$. So we have

$$h_{2t}(c_{ii'}) \leq h_t(c_{ik}) + h_t(c_{i'k}) \leq 2\alpha_k^1 \leq 2\alpha_k^2 + \frac{2\varepsilon \text{lb}}{n^2} \leq 2\alpha_j^2 + \frac{2\varepsilon \text{lb}}{n^2} \leq 2\alpha_j + \frac{2\varepsilon \text{lb}}{n^2}. \quad \square$$

Lemma E.3. *The optimal value of (O-P) is at most $5OPT + 5\varepsilon \text{lb}(1 + \frac{1}{n^2})$.*

Proof. Let $a, b \geq 0$ be such that $ak'_1 + bk_2 = k$ and $a + b = 1$. Define $\text{charge}_j = a \cdot 5\beta_{i_1(j)j}^1 + b \cdot 5\beta_{i_2(j)j}^2$. Then, we have

$$\sum_j \text{charge}_j = a \sum_{j \in P^1(F'_1)} 5\beta_{i_1(j)j}^1 + b \sum_{j \in P^2(F_2)} 5\beta_{i_2(j)j}^2 \geq a(5\lambda_2 k'_1 - 5\varepsilon \text{lb}) + b \cdot 5\lambda_2 k_2 = 5k\lambda_2 - 5\varepsilon \text{lb}$$

where the inequality follows from Lemma E.1. Set $\theta = a$ and $z_i = a$ for all $i \in F'_1 \setminus \overline{F}_1$. We show that $\text{charge}_j + A_j(\theta, z := \{z_i\}_{i \in F'_1 \setminus \overline{F}_1}) \leq 5\alpha_j$ for every client j . This will complete the proof since this implies that

$$5k\lambda_2 + \sum_j A_j(\theta, z) \leq 5\alpha_j + 5\varepsilon \text{lb} \leq 5\alpha_j^2 + 5\varepsilon \text{lb} \left(1 + \frac{1}{n^2}\right),$$

and $\sum_j \alpha_j^2 - k\lambda_2 \leq OPT$ since $(\alpha^2, \beta^2, \lambda_2)$ is a feasible solution to (D_ρ) .

To prove the claim, consider any client j . Recall that $a \geq 0.5$. Observe that:

- if $j \in P^1(F'_1)$, then $h_t(c_{i_1(j)j}) + \beta_{i_1(j)j}^1 = \alpha_j^1$;
- if $j \in P^2(F_2)$, then $h_t(c_{i_2(j)j}) + \beta_{i_2(j)j}^2 = \alpha_j^2$, and otherwise, we have $h_{3t}(c_{i_2(j)j}) \leq 3\alpha_j^2$.

By considering each case in the definition of A_j , and plugging in the above bounds, we obtain the claimed bound on $\text{charge}_j + A_j(\theta, z)$. \square

Lemma E.4. *Let $(\tilde{\theta}, \tilde{z})$ be an integral solution to (O-P). Let X_j denote its assignment cost under the resulting solution. We have $\sum_j h_{5t}(X_j) \leq \sum_j A_j(\tilde{\theta}, \tilde{z}) + \frac{2\varepsilon \text{lb}}{n}$.*

Proof. Consider any client j . We abbreviate $A_j(\tilde{\theta}, \tilde{z})$ to A_j . We show that $h_{5t}(X_j) \leq A_j + \frac{2\varepsilon \text{lb}}{n^2}$, which will prove the lemma. We first note the following. By Lemma E.2 (iii), there are facilities $i \in F_2, i' \in F_1$ such that $h_{3t}(c_{ij}) \leq 3\alpha_j$ and $h_{2t}(c_{ii'}) \leq 2\alpha_j + \frac{2\varepsilon \text{lb}}{n^2}$. If $\tilde{\theta} = 1$, then we know that $\sigma(i)$ is open. Hence,

$$h_{5t}(X_j) \leq h_{5t}(c_{\sigma(i)j}) \leq h_{3t}(c_{ij}) + h_{2t}(c_{i\sigma(i)}) \leq h_{3t}(c_{ij}) + h_{2t}(c_{ii'}) \leq 5\alpha_j + \frac{2\varepsilon \text{lb}}{n^2}.$$

Consider each case in the definition of A_j .

- $i_1(j) \in \overline{F}_1, j \in P^1(F'_1) \cap P^2(F_2)$. If $\tilde{\theta} = 1$, then $i_1(j)$ is open, and if $\tilde{\theta} = 0$, then $i_2(j)$ is open, so $h_t(X_j) \leq A_j$.
- $i_1(j) \notin \overline{F}_1, j \in P^1(F'_1) \cap P^2(F_2)$. If $\tilde{z}_{i_1(j)}^{\text{int}} = 1$, then the bound clearly holds. Otherwise, either $i_2(j)$ is open, or $i := \sigma(i_2(j))$ is open. We have $c_{ii_2(j)} \leq c_{i_1(j)i_2(j)} \leq c_{i_1(j)j} + c_{i_2(j)j}$, and so $X_j \leq 2c_{i_2(j)j} + c_{i_1(j)j}$ holds in both cases. So $h_{3t}(X_j) \leq h_{2t}(2c_{i_2(j)j}) + h_t(c_{i_1(j)j}) = A_j$.
- $j \in P^2(F_2) \setminus P^1(F'_1)$. If $\tilde{\theta} = 0$, clearly $h_t(X_j) \leq A_j$. Otherwise, as shown above, we have $h_{5t}(X_j) \leq 5\alpha_j + \frac{2\varepsilon \text{lb}}{n^2} = A_j + \frac{2\varepsilon \text{lb}}{n^2}$.
- $j \notin P^1(F'_1) \cup P^2(F'_2)$. If $\tilde{\theta} = 0$, then $i_2(j)$ is open and $h_{3t}(X_j) \leq A_j$. Otherwise, as above, we have $h_{5t}(X_j) \leq A_j + \frac{2\varepsilon \text{lb}}{n^2}$.
- $i_1(j) \in \overline{F}_1, j \in P^1(F'_1) \setminus P^2(F'_2)$. If $\tilde{\theta} = 1$, then clearly $h_t(X_j) \leq A_j$. Otherwise, $i_2(j)$ is open, and $h_{3t}(X_j) \leq 3\alpha_j \leq A_j$.
- $i_1(j) \notin \overline{F}_1, j \in P^1(F'_1) \setminus P^2(F'_2)$. If $\tilde{z}_{i_1(j)}^{\text{int}} = 1$, then clearly $h_t(X_j) \leq A_j$. Otherwise, if $\tilde{\theta} = 0$, then $i_2(j)$ is open, and $h_{3t}(X_j) \leq h_{3t}(c_{i_2(j)j}) \leq 3\alpha_j \leq A_j$. If $\tilde{\theta} = 1$, then as shown at the beginning, we have $h_{5t}(X_j) \leq 5\alpha_j + \frac{2\varepsilon \text{lb}}{n^2} = A_j + \frac{2\varepsilon \text{lb}}{n^2}$. \square

Proof of Finishing up the proof of Theorem 9.18. Let X_j be the assignment cost of client j in the solution returned. Combining Lemmas E.3 and E.4, we obtain that $\sum_j h_{5t}(X_j) \leq 5\text{OPT} + \varepsilon \text{lb}(5 + \frac{2}{n} + \frac{5}{n^2})$. Since $\text{OPT} \leq \sum_j h_t(\tilde{o}_j^\downarrow)$, combining this with Lemma 6.10 shows that $\text{cost}(w; \cdot)$ -cost of the solution returned is at most $5(1 + \varepsilon)(1 + 2\varepsilon)\text{opt} + (1 + \varepsilon)\varepsilon \text{lb}(5 + \frac{2}{n} + \frac{5}{n^2})$. \square

Proof of Lemma E.1. We mimic the proof in [17]. We use x_- to denote a quantity infinitesimally smaller than x . Let $\delta = \lambda_2 - \lambda_1$. Sort the clients in increasing order of their $\alpha_j^0 := \min\{\alpha_j^1, \alpha_j^2\}$ value. So $\alpha_1^0 \leq \dots \leq \alpha_n^0$. We prove that $|\alpha_j^1 - \alpha_j^2| \leq 2^{j-1}\delta$ for all $j = 1, \dots, n$, which implies the lemma.

We proceed by induction on j . Consider running the dual-ascent phase of the primal-dual algorithm for $\lambda = \lambda_1$ and $\lambda = \lambda_2$ in parallel. For the base case, suppose that $\alpha_1^0 = \alpha_1^r$, where $r \in \{1, 2\}$. Consider the time point $\tau = \alpha_1^0$ in the two executions. By definition, at time τ_- , all clients are active in the two executions. So at time τ , we have $\alpha_j^1 = \alpha_j^2 = t$ for all j , and so $\beta_{ij}^1 = \beta_{ij}^2$ for all i, j . Client 1 froze in execution r at time t , because at that time it can reach some facility f for which constraint (25) became tight at time τ ; we say that f got paid for at time t (in the execution r). Let $\bar{r} = 2 - r$. We have $\sum_j \beta_{fj}^{\bar{r}} = \sum_j \beta_{fj}^r$

at time τ , so $\sum_j \beta_{fj}^{\bar{\tau}}$ can increase by at most δ beyond time t . Hence, $\alpha_1^{\bar{\tau}}$ can increase by at most δ beyond time τ (since any increase in $\alpha_1^{\bar{\tau}}$ translates to the same increase in $\beta_{f1}^{\bar{\tau}}$ as $\alpha_1^{\bar{\tau}} \geq h_t(c_{f1})$ at time τ).

Suppose we have shown that $|\alpha_j^1 - \alpha_j^2| \leq 2^{j-1}\delta$ for all $j = 1, \dots, \ell - 1$ (where $\ell \geq 2$). Now consider client ℓ . The induction step follows from a similar argument. Consider time point $\tau = \alpha_\ell^0$ in both executions. By definition, all clients $j \geq \ell$ are active at time τ_- in the two executions. So at time τ , we have $\alpha_j^1 = \alpha_j^2 = \tau$ for all $j \geq \ell$. Suppose $\alpha_\ell^0 = \alpha_\ell^r$, where $r \in \{1, 2\}$, and let $\bar{r} = 2 - r$. In execution r , client ℓ froze at time t due to some facility f , where either: (1) f was paid for by time τ , and ℓ reached f at time τ ; or (2) f got paid for at time τ , and ℓ reached f at or before time τ . At time τ , we have $\beta_{fj}^{\bar{r}} \geq \beta_{fj}^r - 2^{j-1}\delta$ for all $j < \ell$ by the induction hypothesis, and $\beta_{fj}^1 = \beta_{fj}^2$ for all $j \geq \ell$. Therefore, the contribution $\sum_j \beta_{fj}^{\bar{r}}$ from clients to the LHS of (25) at time t is at least $\lambda_1 - \sum_{j=1}^{\ell-1} 2^{j-1}\delta$. So this contribution can increase by at most $\delta + \sum_{j=1}^{\ell-1} 2^{j-1}\delta = 2^{\ell-1}\delta$ beyond time τ in execution \bar{r} . So since $\alpha_\ell^{\bar{r}} = \alpha_\ell^r \geq h_t(c_{f\ell})$ at time τ , it follows that $\alpha_\ell^{\bar{r}}$ can increase by at most $2^{\ell-1}\delta$ beyond time τ . \square

References

- [1] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k -means and Euclidean k -median by primal-dual algorithms. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017.
- [2] Soroush Alamdari and David B. Shmoys. A bicriteria approximation algorithm for the k -center and k -median problems. In *Proceedings, Workshop on Approximation and Online Algorithms (WAOA)*, 2017.
- [3] Noga Alon, Yossi Azar, Gerhard J Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.
- [4] Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2017.
- [5] Ali Aouad and Danny Segev. The ordered k -median problem: surrogate models and approximation algorithms. *Math. Programming*, pages 1–29, 2018.
- [6] Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey S. Vitter. Load balancing in the L_p norm. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 383–391, 1995.
- [7] Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2005.
- [8] Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.
- [9] Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM Journal on Computing (SICOMP)*, 39(4):1413–1431, 2009.
- [10] Rajendra Bhatia. *Matrix analysis*, volume 169. Springer, 2013.
- [11] Jarosław Błasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming symmetric norms via measure concentration. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2017.

- [12] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k -median, and positive correlation in budgeted optimization. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.
- [13] Jarosław Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k -median. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2018.
- [14] Robert Carr and Santosh Vempala. Randomized metarounding. *Random Structures Algorithms*, 20(3):343–352, 2002.
- [15] Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. On $(1, \varepsilon)$ -restricted assignment makespan minimization. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015.
- [16] Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between k -median and k -center: Approximation algorithms for ordered k -median. In *Proceedings, International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.
- [17] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k -median problems. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.
- [18] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k -median problem. *J. Comput. System Sci.*, 65(1):129–149, 2002.
- [19] Moses Charikar and Shi Li. A dependent LP-rounding approach for the k -median problem. In *Proceedings, International Colloquium on Automata, Languages and Programming (ICALP)*, 2012.
- [20] Tomáš Ebenlendr, Marek Krčál, and Jiří Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014.
- [21] Ashish Goel and Adam Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica*, 44(4):301–323, 2006.
- [22] Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-norms and all- ℓ_p -norms approximation algorithms. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science. (FSTTCS)*, 2008.
- [23] Teofilo F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38:293 – 306, 1985.
- [24] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554*, 2008.
- [25] Godfrey H. Hardy, John E. Littlewood, and George Pólya. *Inequalities*. Cambridge Univ Press, 1934.
- [26] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k -center problem. *Math. Oper. Res.*, 10(2):180–184, 1985.
- [27] Kamal Jain and Vijay V. Vazirani. 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):274–296, 2001.
- [28] Klaus Jansen and Lars Rohwedder. On the configuration-lp of the restricted assignment problem. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.

- [29] Igor Kabiljo, Brian Karrer, Mayank Pundir, Sergey Pupyrev, and Alon Shalita. Social hash partitioner: a scalable distributed hypergraph partitioner. *Proceedings, Very Large Databases (VLDB) Endowment*, 10(11):1418–1429, 2017.
- [30] Amit Kumar and Jon Kleinberg. Fairness measures for resource allocation. *SIAM Journal on Computing (SICOMP)*, 36(3):657–680, 2006.
- [31] V. S. Kumar, Madhav V Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. A unified approach to scheduling on unrelated parallel machines. *Journal of the ACM (JACM)*, 56(5):28, 2009.
- [32] G. Laporte, S. Nickel, and F. S. da Gama. *Location Science*. Springer, 2015.
- [33] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.
- [34] Jan Karel Lenstra, David B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46(1-3):259–271, 1990.
- [35] Shi Li and Ola Svensson. Approximating k -median via pseudo-approximation. *SIAM Journal on Computing (SICOMP)*, 45(2):530–547, 2016.
- [36] László Lovász and Michael D. Plummer. Matching theory. *Annals of Discrete Mathematics*, 29, 1986.
- [37] Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.
- [38] S. Nickel and J. Puerto. *Location Theory: A Unified Approach*. Springer Science & Business Media, 2005.
- [39] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [40] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1-3):461–474, 1993.
- [41] Ola Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012.
- [42] Arie Tamir. The k -centrum multi-facility location problem. *Discrete Applied Mathematics*, 109(3):293–307, 2001.