Simpler and Better Algorithms for Minimum-Norm Load Balancing

Deeparnab Chakrabarty
Dept. of Computer Science, Dartmouth, 9 Maynard St, Hanover, NH 03755, USA
deeparnab@dartmouth.edu

Chaitanya Swamy
Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1, Canada
cswamy@uwaterloo.ca

Abstract

Recently, Chakrabarty and Swamy (STOC 2019) introduced the *minimum-norm load-balancing* problem on unrelated machines, wherein we are given a set J of jobs that need to be scheduled on a set of m unrelated machines, and a monotone, symmetric norm; We seek an assignment $\sigma : J \mapsto [m]$ that minimizes the norm of the resulting load vector $\vec{\mathsf{load}}_\sigma \in \mathbb{R}^m_+$, where $\vec{\mathsf{load}}_\sigma(i)$ is the load on machine i under the assignment σ. Besides capturing all ℓ_p norms, symmetric norms also capture other norms of interest including top-ℓ norms, and ordered norms. Chakrabarty and Swamy (STOC 2019) give a $(38+\varepsilon)$-approximation algorithm for this problem via a general framework they develop for minimum-norm optimization that proceeds by first carefully reducing this problem (in a series of steps) to a problem called min-max ordered load balancing, and then devising a so-called deterministic oblivious LP-rounding algorithm for ordered load balancing.

We give a direct, and simple $4+\varepsilon$-approximation algorithm for the minimum-norm load balancing based on rounding a (near-optimal) solution to a novel convex-programming relaxation for the problem. Whereas the natural convex program encoding minimum-norm load balancing problem has a large non-constant integrality gap, we show that this issue can be remedied by including a key constraint that bounds the "norm of the job-cost vector." Our techniques also yield a (essentially) 4-approximation for: (a) *multi-norm load balancing*, wherein we are given multiple monotone symmetric norms, and we seek an assignment respecting a given budget for each norm; (b) the best *simultaneous approximation factor* achievable for all symmetric norms for a given instance.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Convex optimization

Keywords and phrases Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.27

Funding *Deeparnab Chakrabarty*: Supported in part by NSF grant CCF-1813165.
Chaitanya Swamy: Supported in part by NSERC grant 327620-09 and an NSERC Discovery Accelerator Supplement Award.

1 Introduction

In the *minimum-norm load-balancing* (MinNormLB) problem, we are given a set J of n jobs, a set of m machines, and processing times $p_{ij} \geq 0$ for all $i \in [m]$ and $j \in J$. We use $[m]$ to denote $\{1, \ldots, m\}$. We are also given a monotone, symmetric norm $f : \mathbb{R}^m \to \mathbb{R}_+$. Recall that by definition of norm, this means that f satisfies: (i) $f(x) = 0$ iff $x = 0$; (ii) $f(x+y) \leq f(x) + f(y)$ for all $x, y \in \mathbb{R}^m$ (triangle inequality); and (iii) $f(\lambda x) = |\lambda| f(x)$ for all $x \in \mathbb{R}^m, \lambda \in \mathbb{R}$ (homogeneity). (Properties (ii) and (iii) imply that f is convex.) Monotonicity means that $f(x) \leq f(y)$ for all $x, y \in \mathbb{R}^m$ such that $x_i(y_i - x_i) \geq 0$ for all $i \in [m]$; symmetry means that permuting the coordinates of x does not affect its norm, i.e., $f(x) = f\big(\{x_{\pi(i)}\}_{i\in[m]}\big)$ for every $x \in \mathbb{R}^m$ and every permutation $\pi : [m] \mapsto [m]$.

The goal is to find an assignment $\sigma : J \to [m]$ that minimizes the norm (under f) of the induced load vector. More precisely, an assignment σ induces the m-dimensional load vector $\vec{\mathsf{load}}_\sigma \in \mathbb{R}^m_+$ where $\vec{\mathsf{load}}_\sigma(i) := \sum_{j:\sigma(j)=i} p_{ij}$. The objective is to find σ that minimizes $f(\vec{\mathsf{load}}_\sigma)$.

Besides ℓ_p-norms, monotone symmetric norms capture Top-ℓ norms – sum of ℓ largest coordinates in absolute value – and ordered norms (which are a nonnegative linear combination of Top-ℓ norms). The minimum-norm load-balancing problem was recently introduced by Chakrabarty and Swamy [8]. They develop a general framework for minimum-norm optimization problems based on reducing the problem to a special case called min-max ordered optimization, and devise a so-called deterministic oblivious rounding [8] to tackle the latter problem, which results in a $(38+\varepsilon)$-approximation algorithm for MinNormLB.

Our main result is a simpler $4(1+\varepsilon)$-approximation algorithm for MinNormLB that runs in time $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\varepsilon})\big)$.

▶ **Theorem 1.** *One can achieve a* $4(1+\varepsilon)$*-approximation for* MinNormLB *in time* $\mathrm{poly}\big(\textit{input size}, \log(\frac{1}{\varepsilon})\big)$*, assuming we have a value-oracle and subgradient-oracle for the norm* f*. More generally, if we have* ω*-approximate value- and subgradient- oracles for* f *(see Section 4), then one can compute a* $4(1+5\omega)(1+\varepsilon)$*-approximation to* MinNormLB *in time* $\mathrm{poly}\big(\textit{input size}, \log(\frac{1}{\varepsilon})\big)$*.*

This is a substantial improvement over the approximation factor of 38 obtained in [8]. Moreover, our algorithm is also simpler and more direct than the one in [8]. Notably, our approximation factor is close to the best-known approximation factor (of 2) known for the ℓ_∞ norm (wherein MinNormLB becomes the classical minimum-makespan problem). Our algorithm proceeds by rounding the solution to a novel convex-programming relaxation of the problem. The convex program can be solved (approximately) using an (approximate) first-order oracle for f that returns the function value, and its subgradient at a given point.

Our techniques also yield a $4(1+\varepsilon)$-approximation for (see Section 5): (a) *multi-norm load balancing*, wherein we are given multiple monotone, symmetric norms and budgets for each norm, and we seek an assignment (approximately) respecting these budgets; and (b) the best *simultaneous approximation factor* achievable for all symmetric norms for a given instance.

Motivation and perspective

One of the reasons for studying MinNormLB is that it generalizes various load-balancing problems considered in the literature, and its study therefore yields a unified methodology for dealing with monotone, symmetric norms.

Load balancing under the ℓ_∞ norm, that is, minimizing the maximum load (also called the makespan) is a classical scheduling problem that has been extensively studied [18, 23, 10, 24, 6, 15] over the past three decades, both in its full generality for unrelated machines and for various special cases. The best known approximation factor for the unrelated-machines setting is still 2 [18], and it is *NP*-hard to obtain an approximation factor better than 3/2 [18]. For general ℓ_p-norms, Azar and Epstein [3] obtain a 2-approximation, and improved guarantees have been obtained for constant p [3, 16, 19]. More recently, the load-balancing problem has also been considered for other monotone, symmetric norms. Top-ℓ- and ordered-norms have been proposed in the location-theory literature (see "Other related work") as a means of interpolating between the ℓ_1 and ℓ_∞ norms (and an alternative to using ℓ_p norms), and motivated by this, Chakrabarty and Swamy [8] studied the Top-ℓ load-balancing

problem – minimize the total load on the ℓ most loaded machines – and the ordered load-balancing problem. They give a $(2+\varepsilon)$-approximation algorithm in both settings, and also (as noted earlier) devise a $(38+\varepsilon)$-approximation algorithm for an arbitrary monotone, symmetric norm.

For load balancing, there has been considerable interest in *simultaneous optimization*. Given an instance, the objective is to find an assignment that *simultaneously* approximates a large suite of objective functions. Building upon previous works [2, 4], Goel and Meyerson [11] describe a 2-approximation for the problem of simultaneously approximating all monotone symmetric norms in the *restricted assignment* setting. However, it is known that such an $O(1)$-factor is *impossible* in the unrelated-machines setting [4, 11]. As a byproduct of their MinNormLB algorithm, in the unrelated-machines setting, Chakrabarty and Swamy [8] give an *instance-wise* $(38+\varepsilon)$-approximation to the best simultaneous approximation-factor possible for the instance. To elaborate, let $\alpha^*_{\mathcal{I}}$ denote the smallest factor for instance $\mathcal{I}$ such that there exists a schedule that achieves an $\alpha^*_{\mathcal{I}}$-approximation for all monotone, symmetric norms; [8] returns a schedule for $\mathcal{I}$ that achieves a $38(1+\varepsilon)\alpha^*_{\mathcal{I}}$-approximation for all monotone, symmetric norms. As mentioned above, we devise an algorithm that for every instance $\mathcal{I}$ returns a schedule that simultaneously achieves a $\big(4+O(\varepsilon)\big)\alpha^*_{\mathcal{I}}$-approximation for all monotone, symmetric norms (see Theorem 13).

Our techniques

Since a norm is a convex function, a natural convex-programming relaxation for MinNormLB is to minimize the norm of the fractional load vector $\vec{L} = L(\vec{x}) := \big\{\sum_j p_{ij}x_{ij}\big\}_{i\in[m]}$, where the x_{ij}s are the usual variables denoting if job j is assigned to machine i, and we have the usual job-assignment constraints encoding that every job is assigned to some machine. This convex program, however, has a large integrality gap, even when f is the ℓ_∞-norm due to the issue that the convex program could split a large job across multiple machines.

In the case of the ℓ_∞ norm (the makespan minimization problem), the typical way of circumventing the above issue is to "guess" the optimal value, say T, and add constraints to encode that no single job contributes more than T to the objective. The usual way of capturing this is to explicitly set $x_{ij}=0$ if $p_{ij}>T$. A less common, and weaker, way of encoding this is to enforce that $\sum_i p_{ij}x_{ij} \le T$ for all j, that is, the total processing time contribution of any job j across the machines cannot exceed T.

For an arbitrary (monotone, symmetric) norm, it is unclear how to extend either of the above approaches, since the contribution of a job to the objective is a now a somewhat vague notion. One way to generalize things would be to encode (either explicitly or in the alternate weaker sense above) that the "norm" of the job-cost vector is at most T, where the job-cost vector is indexed by jobs and the cost for job j (under x) is $P_j := \sum_i p_{ij}x_{ij}$. But the norm f is defined over $\mathbb{R}^m$, whereas the job-cost vector lies in $\mathbb{R}^n$. For certain specific (families of) norms – e.g., ℓ_p-norms, top-ℓ norm, ordered norm – there is a natural version of the norm over $\mathbb{R}^n$,[1] but what does such a constraint mean in general, and how can one encode this?

Our key insight, which leads to our convex program, is that one can capture the above consideration by examining the vector $\vec{P} \in \mathbb{R}^m$ comprising the *costs of the m most-costly jobs* and enforcing the constraint $f(\vec{P}) \le T$; since f is monotone, this can be equivalently encoded as $f\big(\{P_j\}_{j\in S}\big) \le T$ for all $S \subseteq J$ with $|S| = m$. It is not apparent that such a

[1] For ℓ_p-norms, a variant of this that considers the ℓ_p^p expression does work, but this crucially exploits the separability of ℓ_p^p [3].

constraint is valid, but we derive some insights about symmetric norms and show that this is indeed the case (see Theorem 3). This yields our convex program (CP), which can be solved efficiently (within ε additive error, for any $\varepsilon > 0$, in time $\text{poly}\big(\text{input size}, \ln(1/\varepsilon)\big)$) using the ellipsoid method provided we have a value oracle and subgradient oracle for f.

Rounding a solution x to the convex program is now quite easy. Let $\vec{L} \in \mathbb{R}^m$ denote the load-vector arising from x. We use a filtering step to ensure that each job j is only assigned to machines i for which $p_{ij} \leq 2P_j$. This causes a factor-2 blowup in the machine loads. Now we use the rounding algorithm of Shmoys and Tardos [23] for the generalized assignment problem (GAP). The resulting assignment σ has load-vector at most $2\vec{L} + \vec{Z}$, where $\vec{Z} \in \mathbb{R}^m$ and $Z_i = \max_{j:\sigma(j)=i} p_{ij}$; the filtering step and our constraints ensure that $f(\vec{Z}) \leq 2T$, so $f(2\vec{L} + \vec{Z}) \leq 4T$. Our algorithm is much more direct than the one in [8]: it avoids the sequence of steps (and the associated approximation-factor losses) used in [8], wherein MinNormLB is reduced to a special case, called min-max ordered load balancing, which is then tackled by a deterministic oblivious rounding procedure.

Other related work

The algorithmic problem of finding minimum-norm solutions has also been investigated in the context of *k-clustering*, wherein the goal is to open k "facilities" in a metric space to serve a set of clients, and the cost vector induced by a solution is the vector of distances of clients to their nearest open facility. The setting of ℓ_p-norms, especially when $p \in \{1, 2, \infty\}$ (where the problem is called the k-{median, means, center} problem) has been extensively studied, and $O(1)$-approximations are known in these settings [13, 9, 14, 1]. Top-ℓ and ordered norms have been proposed in the context of k-clustering in the Operations Research literature (see, e.g., [21, 17]), but constant-factor approximations for these norms were obtained quite recently [5, 7, 8]. Furthermore, Chakrabarty and Swamy [8] utilize their general framework to obtain an $O(1)$-approximation for the k-clustering problem under any monotone, symmetric norm. We do not know of any alternate approach that works in the k-clustering setting.

2 A convex-programming relaxation

By possibly adding dummy jobs with zero processing times, we may assume without loss of generality that $n \geq m$. A natural convex program for MinNormLB has non-negative variables x_{ij} denoting if job j is assigned to machine i (or the extent of j assigned to i) with the constraint (1) encoding that every job is assigned to a machine. These x-variables define a load vector $\vec{L} = \big(L_i = L_i(x)\big)_{i\in[m]}$ where $L_i(x) = \sum_{j\in J} p_{ij}x_{ij}$. The objective seeks to minimize $T := f\big(\vec{L}\big)$. As noted earlier, this convex program has a large integrality gap (even when f is the ℓ_∞ norm). We *strengthen* the convex program as follows.

Given the x-assignment, define $P_j = P_j(x) := \sum_i p_{ij}x_{ij}$, which is the load incurred by the fractional solution for scheduling job j. Fix any subset $S \subseteq J$ with $|S| = m$. Note that this is well-defined since we have assumed $n \geq m$. This defines the m-dimensional vector $\vec{P}_S := \{P_j\}_{j\in S}$. We add the constraints (6) enforcing that $f(\vec{P}_S) \leq T$ for each such subset S. Throughout, we use i to index the machines in $[m]$, and j to index the jobs in J.

$$\begin{aligned}
&\min \quad T && && \text{(CP)}\\
&\text{s.t.} \quad \sum_i x_{ij} \geq 1 && \forall j \in J && (1)\\
&\qquad x \geq 0 && && (2)\\
&\qquad L_i = \sum_{j\in J} p_{ij}x_{ij} && \forall i \in [m] && (3)\\
&\qquad P_j = \sum_{i\in[m]} p_{ij}x_{ij} && \forall j \in J && (4)\\
&\qquad f(\vec{L}) \leq T && && (5)\\
&\qquad f(\vec{P}_S) \leq T && \forall S \subseteq J, |S| = m && (6)
\end{aligned}$$

Let $\mathsf{OPT} := \mathsf{OPT}_{\mathrm{CP}}$ denote the optimal value of (CP), and let O^* be the optimal value of the minimum-norm load-balancing problem. Since the x_{ij}-variables completely determine a solution to (CP), we will sometimes abuse notation and say that x is a feasible solution to (CP). We argue that (CP) is a valid relaxation. The proof uses the following simple observation about symmetric convex functions.

▷ Claim 2. Let $h : \mathbb{R}^m \to \mathbb{R}$ be a symmetric convex function. Let $v \in \mathbb{R}^m_+$, and $i, j \in [m]$. Let $w \in \mathbb{R}^m_+$ be the vector where $w_i = v_i + v_j$, $w_j = 0$, and $w_k = v_k$ otherwise. Then, $h(v) \leq h(w)$.

Proof. Consider the vector w' constructed in a symmetric fashion to w: set $w'_j = v_i + v_j$, $w'_i = 0$, and $w'_k = v_k$ otherwise. Observe that v is a convex combination of w and w' (we have $v = \frac{v_i}{v_i+v_j} \cdot w + \frac{v_j}{v_i+v_j} \cdot w'$), and $h(w) = h(w')$ since h is symmetric. By convexity and symmetry, $h(v) \leq \max\{h(w), h(w')\} = h(w)$. ◁

▶ **Theorem 3.** *Constraints* (6) *are valid, and so for any instance of* MinNormLB*, we have* $\mathsf{OPT} \leq O^*$.

Proof. Let $\sigma^* : J \to [m]$ be an optimal assignment, so $f(\vec{\mathsf{load}}_{\sigma^*}) = O^*$. We now describe a feasible solution to (CP) with $T = O^*$. Set $x_{ij} = 1$ if $\sigma^*(j) = i$, and 0 otherwise. Clearly, constraints (1) hold. Note, $L_i = \mathsf{load}_{\sigma^*}(i)$ for all i, and $P_j = p_{\sigma^*(j)j}$ for all j. Therefore, (5) holds with equality.

The interesting bit is to show that (6) holds. To that end, fix a subset $S \subseteq J$ of m jobs. Consider the load vector induced by jobs in S. That is, define $L'_i := \sum_{j \in S : \sigma^*(j) = i} p_{ij}$. Note that $\vec{L}$ coordinate wise dominates $\vec{L'}$, so by monotonicity of f, we have $f(\vec{L'}) \leq f(\vec{L}) = T$.

We argue that $f(\vec{P}_S) \leq f(\vec{L'})$, which will complete the proof. To see this, first note that if σ^* assigns the jobs in S to distinct machines, then $\vec{P}_S$ is simply a permutation of $\vec{L'}$, so $f(\vec{P}_S) = f(\vec{L'})$. Otherwise, observe that $\vec{L'}$ can be obtained from $\vec{P}_S$ by applying the operation in Claim 2 to pairs of jobs in S assigned to the same machine; therefore, we have $f(\vec{P}_S) \leq f(\vec{L'})$. ◀

The proof above relied only on convexity, monotonicity, and symmetry of the function f. In Section 3 (see Theorem 4) we describe a rounding procedure which takes a feasible solution for (CP) and returns an assignment with a factor-4 blow-up in the objective. This will utilize the homogeneity of the norm f. In Section 4, we show how to (approximately) solve (CP) given an (approximate) first-order oracle for the underlying norm (see Theorem 9). Combining these two results yields Theorem 1.

3 The rounding algorithm

We now describe and analyze our simple rounding algorithm, which yields the following guarantee.

▶ **Theorem 4.** *Given a feasible fractional solution* $\left(x = \{x_{ij}\}_{i,j}, \vec{L}, \vec{P}, T\right)$ *to* (CP)*, there is a polynomial time algorithm to obtain a schedule* σ *with* $f(\vec{\mathsf{load}}_\sigma) \leq 4T$.

Proof. First, we filter x. For every i, j, we set $\hat{x}_{ij} = 2x_{ij}$ if $p_{ij} \leq 2P_j$, and 0 otherwise. A standard Markov-inequality style argument shows that $\hat{x}$ satisfies (1). Now we apply the Shmoys-Tardos GAP-rounding algorithm [23] to $\hat{x}$. This yields an assignment $\sigma : J \to [m]$ such that: for every job j, we have $\sigma(j) = i$ only if $\hat{x}_{ij} > 0$, and for every machine i, we have $\mathsf{load}_\sigma(i) \leq \sum_{j \in J} p_{ij}\hat{x}_{ij} + Z_i \leq 2L_i + Z_i$, where $Z_i = \max_{j:\sigma(j)=i} p_{ij}$. Thus, $\vec{\mathsf{load}}_\sigma \leq 2\vec{L} + \vec{Z}$.

Let j_i be a maximum-length job assigned to machine i in σ, i.e., $\sigma(j_i) = i$ and $Z_i = p_{ij_i}$. By our filtering step, we know that $Z_i \leq 2P_{j_i}$. Let $S = \{j_i : i \in [m]\}$. Then $\vec{Z} := (Z_i)_{i\in[m]} \leq 2\vec{P}_S$. By monotonicity, the triangle inequality, and homogeneity of f, we then obtain that

$$f(\vec{\mathsf{load}}_\sigma) \leq 2f(\vec{L}) + f(\vec{Z}) \leq 2T + 2f(\vec{P}_S) \leq 4T. \qquad \blacktriangleleft$$

Interestingly, and notably, observe that the rounding procedure above is *oblivious to the norm* f: given a fractional solution x, the same rounding procedure works for all monotone, symmetric norms. This will be useful in Section 5, where we seek an assignment that is simultaneously good for multiple norms.

4 Solving the convex program

We now discuss how to solve the convex program (CP). To maintain the flow of reading, proofs of certain technical claims are deferred to Section 6. It is well known [20, 12] that we can efficiently solve a convex program $\min_{x\in S} h(x)$ (where $S \subseteq \mathbb{R}^n$ is convex) to within any additive error $\varepsilon > 0$ using the ellipsoid method provided that (we state things more precisely below): (i) S has non-zero volume and is contained in some ball; (ii) we have a separation oracle for S; (iii) we have a *first-order* oracle for h that given input $x \in S$, returns $h(x)$, and a subgradient of h at x. More generally, we show that by utilizing the machinery of Shmoys and Swamy [22], even an approximate value and subgradient oracle suffices (see Theorem 9). This is particularly relevant since the norm and/or components of the subgradient vector may involve irrational numbers.

By scaling we may assume that all p_{ij}s are integers. Let O^* denote the optimal value for the MinNormLB instance. We can easily detect if $O^* = 0$, since this implies an assignment with 0 load on every machine. Therefore, we assume $O^* \geq 1$. It will be convenient to reformulate (CP) as follows. Let $\mathcal{P} := \left\{x \in \mathbb{R}^{[m]\times J} : \sum_i x_{ij} \geq 1 \ \ \forall j \in J, \ \ 0 \leq x_{ij} \leq 1 \ \ \forall i \in [m], j \in J\right\}$ denote the feasible region for the assignment variables.

$$\min \quad g(x) \ := \ \max\Big\{f\big(\vec{L}(x)\big), \ \max_{S\subseteq J:|S|=m} f\big(\vec{P}(x)_S\big)\Big\} \qquad \text{s.t.} \qquad x \in \mathcal{P}. \tag{CP'}$$

Note that the x_{ij}s are the only variables above. Recall that OPT is the optimal value of (CP) (and (CP')).

We recall a few standard concepts from optimization. Let $h : \mathbb{R}^k \mapsto \mathbb{R}$ and let $\|u\|$ denote the ℓ_2 norm of u.

- We say that h has *Lipschitz constant* (at most) K if $|h(v) - h(u)| \leq K\|v - u\|$ for all $u, v \in \mathbb{R}^k$.
- We say that $d \in \mathbb{R}^k$ is a *subgradient* of h at $u \in \mathbb{R}^k$ if we have $h(v) - h(u) \geq d \cdot (v - u)$ for all $v \in \mathbb{R}^k$. We say that $\widehat{d}$ is an ω*-subgradient* of h at $u \in \mathbb{R}^k$ if for every $v \in \mathbb{R}^k$, we have $h(v) - h(u) \geq \widehat{d} \cdot (v - u) - \omega h(u)$; we call this the approximate-subgradient inequality.
- An ω*-first-order oracle* for h is an algorithm that at any point $u \in \mathbb{R}^k$, returns an estimate est such that $h(u) \leq \mathsf{est} \leq (1 + \omega)h(u)$, and an ω-subgradient of h at u.
  (In the optimization literature, the notions of approximate first-order oracle and approximate subgradient typically involve additive errors; since our problems are scale-invariant, multiplicative approximations, where the error at u is measured relative to $h(u)$, are more apt here.)

We remark that since f is a norm, an ω-subgradient $\widehat{d}$ of f at u also yields an estimate of $f(u)$ as follows: taking $v = \vec{0}$ and $v = 2u$ respectively in the approximate-subgradient inequality, we obtain the bounds $\widehat{d} \cdot u \geq (1 - \omega)f(u)$ and $\widehat{d} \cdot u \leq (1 + \omega)f(u)$. (Thus, an ω-first-order oracle for f boils down to an ω-subgradient oracle for f.)

By input size, we mean the total encoding length of the p_{ij}s. It is easy to separate over $\mathcal{P}$, and easy to find radii R, and $0 < V \leq 1$ such that $\mathcal{P} \subseteq B(0,R) := \{x : \|x\| \leq R\}$, $\mathcal{P}$ contains a ball of radius V, and $\log\left(\frac{R}{V}\right) = \text{poly}(m,n)$. In particular, $R = \sqrt{mn}$ suffices, and $\mathcal{P}$ contains a ball of radius $V = \frac{0.5}{m}$ around the point x with $x_{ij} = \frac{1.5}{m}$ for all i,j. (We may assume $m \geq 2$ as otherwise the problem is trivial.) Throughout, we use K_f to denote an efficiently-computable upper bound on the Lipschitz constant of f; Lemma 8 shows how to obtain this. Given a bound on the Lipschitz constant of f, one can compute an upper bound on the Lipschitz constant of g.

▷ Claim 5. The Lipschitz constant of g is at most $K = \sqrt{mn} \cdot \max_{i,j} p_{ij} \cdot K_f$.

▶ **Theorem 6** (Follows from [20]; see also [12])**.** *Let* alg *be a first-order oracle for f. Then, for any $\eta > 0$, we can compute $x^* \in \mathcal{P}$ such that $g(x^*) \leq \mathsf{OPT} + \eta$ in* $\text{poly}\left(\textit{input size}, \log(\frac{K_f R}{\eta V})\right)$ *time and using* $\text{poly}\left(\textit{input size}, \log(\frac{K_f R}{\eta V})\right)$ *calls to* alg.

Theorem 6 follows from the ellipsoid method for convex optimization, due to the bound on the Lipschitz constant of g obtained from Claim 5, and since one can use alg to obtain a first-order oracle for g. We next use [22] to obtain a stronger result that utilizes only an approximate first-order oracle for f.

▶ **Theorem 7** (Lemma 4.5 in [22] paraphrased)**.** *Consider a convex optimization problem: $\min_{x \in \mathcal{P}} h(x)$. Let K_h be a known bound on the Lipschitz constant of h. Let $\omega < 1$ and $\eta > 0$. In* $\text{poly}\left(m, n, \log(\frac{K_h R}{V\eta})\right)$ *time and using* $\text{poly}\left(m, n, \log(\frac{K_h R}{V\eta})\right)$ *calls to an ω-first-order oracle for h, one can compute a solution $x^* \in \mathcal{P}$ such that $h(x^*) \leq \frac{1+\omega}{1-\omega} \cdot \left(\min_{x \in \mathcal{P}} h(x) + \eta\right)$.*

To utilize Theorem 7 to solve (CP), we show how to obtain an approximate first-order oracle for g given one for f. Also, in order to convert the additive error in Theorem 7 (and Theorem 6) into a multiplicative guarantee, we show how to obtain a lower bound lb on O^* such that K_f/lb is small.

▶ **Lemma 8.** *Let* alg *be an ω-first-order oracle for f (where $\omega < 1$).*

- *We can obtain a 2ω-first-order oracle for g using $O(1)$ calls to* alg.
- *Using* alg*, we can efficiently compute $\mathsf{lb} \leq O^*$, and an upper bound K_f on the Lipschitz constant of f such that $\frac{K_f}{\mathsf{lb}} \leq 2\sqrt{m}$.*

▶ **Theorem 9.** *Let* alg *be an ω-first-order oracle for f with $\omega \leq \frac{1}{10}$. Given a* MinNormLB *instance with optimum value O^*, there is an algorithm that, for any $\varepsilon > 0$, computes a feasible solution x^* to* (CP) *of objective value $g(x^*) \leq (1+5\omega)(1+\varepsilon)O^*$. The algorithm runs in* $\text{poly}\left(\textit{input size}, \log(\frac{1}{\varepsilon})\right)$ *time and makes* $\text{poly}\left(\textit{input size}, \log(\frac{1}{\varepsilon})\right)$ *calls to* alg.

Proof. This follows by combining Theorem 7 and Lemma 8. Recall that we are assuming that $O^* \geq 1$. By part 8 of Lemma 8, we can compute a 2ω-first-order oracle for g. We use part 8 of Lemma 8 to obtain lb and K_f. Now we apply Theorem 7 to the problem $\min_{x \in \mathcal{P}} g(x)$, taking $\eta = \varepsilon\mathsf{lb}$. The point x^* returned satisfies $g(x^*) \leq \frac{1+2\omega}{1-2\omega} \cdot (\mathsf{OPT} + \varepsilon\mathsf{lb}) \leq (1+5\omega)(1+\varepsilon)O^*$.

Recall that $\log(R/V) = \text{poly}(m,n)$. Since we have an upper bound K on the Lipschitz constant of g, where $\log K = \text{poly(input size)} \cdot \log K_f$ (Claim 5), the running time and number of calls to the first-order oracle for g (and hence alg) is at most $\text{poly}\left(\text{input size}, \log(\frac{1}{\varepsilon})\right)$. ◀

5 Extensions: multi-norm load balancing and simultaneous approximation

5.1 Multi-norm load balancing

In the *multi-norm load-balancing problem*, we are given a load-balancing instance $\big(J, m, \{p_{ij}\}_{i\in[m],j\in J}\big)$, multiple monotone, symmetric norms $f_1, \ldots, f_k$, and budgets $T_1, \ldots, T_k$ for these norms respectively. The goal is to find an assignment $\sigma : J \to [m]$ such that $f_r(\vec{\mathsf{load}}_\sigma) \le T_r$ for all $r \in [k]$. Our approximation guarantee extends easily to this problem.

▶ **Theorem 10.** *Let $\big(J, m, \{p_{ij}\}_{i\in[m],j\in J}\big)$ be a load-balancing instance. Let $f_1, \ldots, f_k$ be k monotone, symmetric norms, with associated budgets $T_1, \ldots, T_k$. Given an ω-first-order oracle for each norm, for any $\varepsilon > 0$, in $\mathrm{poly}(\text{input size}, k, \log(1/\varepsilon))$ time, one can either determine that there is no feasible solution to the multi-norm load-balancing problem, or return an assignment $\sigma : J \to [m]$ such that $f_r(\vec{\mathsf{load}}_\sigma) \le 4(1+7\omega)(1+\varepsilon)T_r$ for all $r \in [k]$.*

The convex-programming relaxation for this problem is a variant of (CP) where there is no objective function, and constraints (5), (6) are replaced with

$$f_r(\vec{L}) \le T_r, \qquad f_r(\vec{P}_S) \le T_r \quad \forall S \subseteq J : |S| = m, \qquad \forall r = 1, \ldots, k \tag{7}$$

Let (Multi-CP) denote the resulting feasibility problem: find $(x, \vec{L}, \vec{P})$ satisfying (1)–(4), and (7). As noted earlier, the rounding procedure in Section 3 is *oblivious* to the underlying norm, and so our task boils down to finding an (approximately) feasible solution to (Multi-CP).

In order to solve (Multi-CP), as with (CP), it will be convenient to move the nonlinear constraints to the objective and consider the following reformulation:

$$\min \quad q(x) \;:=\; \max\Big\{ \max_{r\in[k]} \tfrac{f_r(\overrightarrow{L(x)})}{T_r},\; \max_{r\in[k]} \max_{S\subseteq J:|S|=m} \tfrac{f(\overrightarrow{P(x)_S})}{T_r} \Big\} \qquad \text{s.t.} \qquad (1), (2). \tag{MNCP}$$

Observe that finding a feasible solution to (Multi-CP) is equivalent to finding a feasible solution to (MNCP) with objective value at at most 1. As before, we may assume that the p_{ij}s are integers, and can determine if there is an assignment σ such that $\vec{\mathsf{load}}_\sigma = \vec{0}$ (which clearly satisfies (7)). So assume otherwise. We prove the following.

▶ **Theorem 11.** *Let alg_r be an ω-first-order oracle for f_r for all $r \in [k]$, where $\omega \le \frac{1}{18}$. For any $\varepsilon > 0$, in $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\varepsilon})\big)$ time and using $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\varepsilon})\big)$ calls to each alg_r oracle, we can determine that either (Multi-CP) is infeasible, or compute $x^* \in \mathcal{P}$ such that $q(x^*) \le (1+7\omega)(1+\varepsilon)$.*

Using Theorem 11, for any $\varepsilon > 0$, we can determine in time $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\varepsilon})\big)$ that (Multi-CP) is infeasible, or return a fractional assignment x^* satisfying

$$f_r(L(\vec{x^*})) \le \kappa T_r, \qquad f_r(P(\vec{x^*})_S) \le \kappa T_r \quad \forall S \subseteq J : |S| = m, \qquad \forall r = 1, \ldots, k$$

where $\kappa = (1+7\omega)(1+\varepsilon)$. As noted earlier, the rounding procedure in Section 3 is *oblivious* to the underlying norm, and so by utilizing this to round x^*, we obtain an assignment σ such that $f_r(\vec{\mathsf{load}}_\sigma) \le 4\kappa T_r$ for all $r \in [k]$. This yields Theorem 10.

In the rest of this section, we discuss the proof of Theorem 11. If the multi-norm problem is feasible, we must have $T_r \ge f_r(e_1)$ for all $r \in [k]$. We assume in the sequel that T_r is at least the estimate of $f_r(e_1)$ returned by alg_r scaled by $(1+\omega)$, for all $r \in [k]$; if this

does not hold, then we declare infeasibility. Given this, the proof of Lemma 8 8 shows that $K_r = (1+\omega)\sqrt{m}\cdot T_r$ is an upper bound on the Lipschitz constant of f_r, for all $r \in [k]$. We assume this bound in the sequel. Similar to Claim 5 and Lemma 8, we show that the Lipschitz constant of q can be bounded in terms of the K_rs, and we can obtain a 2ω-first-order oracle for q using the alg_r oracles.

▶ **Lemma 12.** *(i) The Lipschitz constant of q is bounded by $K = \mathrm{poly}(m, n, \max_{i,j} p_{ij})$. (ii) We can obtain a 2ω-first order oracle for q by making $O(1)$ calls to alg_r for each $r \in [k]$.*

Proof of Theorem 11. We utilize Lemma 12 in conjunction with Theorem 7. Part (ii) of Lemma 12 shows how to obtain a 2ω-first-order oracle, alg, for q. So invoking Theorem 7 with $\eta = \varepsilon$, and the bound K on the Lipschitz constant of q obtained from part (i) of Lemma 12, we obtain $\overline{x} \in \mathcal{P}$ such that

$$q(\overline{x}) \leq \frac{1+2\omega}{1-2\omega}\Big(\min_{x\in\mathcal{P}} q(x) + \eta\Big). \tag{8}$$

The running time is $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\varepsilon})\big)$ (since $\log(R/V)$, $\log K = \mathrm{poly}(\text{input size})$), and this is also a bound on the number of calls to the alg_r oracles. Using alg, we obtain an estimate est such that $q(\overline{x}) \leq \mathsf{est} \leq (1+2\omega)q(\overline{x})$. If $\mathsf{est} > \frac{(1+2\omega)^2}{1-2\omega}\cdot(1+\eta)$, then (8) implies that $\big(\min_{x\in\mathcal{P}} q(x)\big) > 1$, and so (Multi-CP) is infeasible. Otherwise, taking $x^* = \overline{x}$, we obtain that $q(x^*) \leq \mathsf{est} \leq \frac{(1+2\omega)^2}{1-2\omega}\cdot(1+\varepsilon) \leq (1+7\omega)(1+\varepsilon)$ since $\omega \leq \frac{1}{18}$. ◀

5.2 Simultaneous approximation

Given a load-balancing instance $\mathcal{I} = \big(J, m, \{p_{ij}\}_{i\in[m], j\in J}\big)$, let $\alpha^*_{\mathcal{I}}$ be the smallest α such that there *exists* an assignment σ^* satisfying $f(\vec{\mathsf{load}}_{\sigma^*}) \leq \alpha\big(\min_{\sigma:J\mapsto[m]} f(\vec{\mathsf{load}}_\sigma)\big)$ for *every* monotone, symmetric norm. That is, $\alpha^*_{\mathcal{I}}$ is the best *simultaneous approximation factor* achievable on instance $\mathcal{I}$. Instead of seeking absolute bounds on $\alpha^*_{\mathcal{I}}$ over a class of instances [2, 4, 11], as discussed in [8], another pertinent problem is to seek *instance-wise* guarantees: given an instance $\mathcal{I}$, we want to find a polytime-computable assignment $\overline{\sigma}$ such that, for some factor $\gamma \geq 1$, we have $f(\vec{\mathsf{load}}_{\overline{\sigma}}) \leq \gamma\alpha^*_{\mathcal{I}}\big(\min_{\sigma:J\mapsto[m]} f(\vec{\mathsf{load}}_\sigma)\big)$ for every monotone, symmetric norm; i.e., the simultaneous approximation factor of $\overline{\sigma}$ at most γ times the best simultaneous approximation factor achievable for $\mathcal{I}$.

Our techniques coupled with insights from [11, 8] yields *a $4\big(1+O(\varepsilon)\big)$-approximation to the best simultaneous approximation factor*, in time $\mathrm{poly}\big(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)}\big)$. To obtain this guarantee, following [11, 8], incurring a $(1+\varepsilon)$-factor loss, it suffices to obtain a 4-approximation to the best simultaneous-approximation achievable for Top-ℓ-norms – Top-$\ell(x) := \max_{S\subseteq[m]:|S|=\ell}\sum_{i\in S}|x_i|$ – for the $O(\log m)$ indices ℓ in $\mathsf{POS} := \big\{\min\{\lceil(1+\varepsilon)^s\rceil, m\} : s \geq 0\big\}$. If we knew the optimal value opt_ℓ for each such Top-ℓ norm, then we can set set a budget $T_\ell = \alpha opt_\ell$ for each $\ell \in \mathsf{POS}$, and utilize our result for multi-norm load balancing to do a binary search for α. Importantly, notice that the resulting feasibility problem (Multi-CP) can now be cast as an *linear-programming* feasibility problem, since a budget constraint of the form Top-$\ell(\vec{v}) \leq T_\ell$ can be modeled using exponentially many linear constraints that one can separate over. Thus, this would yield a $4(1+\varepsilon)$-approximation. To make this idea work, we enumerate all choices for the opt_ℓ values in powers of $(1+\varepsilon)$. As argued in [8], there are at most $\mathrm{poly}\big(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)}\big)$ candidates to enumerate over, and this yields the stated guarantee.

▶ **Theorem 13.** *Given a load-balancing instance $\mathcal{I} = \big(J, m, \{p_{ij}\}_{i\in[m],j\in J}\big)$, let α_I^* be the smallest α such that there is an assignment σ^* satisfying $f(\vec{\mathsf{load}}_{\sigma^*}) \le \alpha\big(\min_{\sigma:J\mapsto[m]} f(\vec{\mathsf{load}}_\sigma)\big)$ for every monotone, symmetric norm f. In $\mathrm{poly}\big(\text{input size}, (\frac{m}{\varepsilon})^{O(1/\varepsilon)}\big)$ time, we can find an assignment $\widehat{\sigma}$ such that we have $f(\vec{\mathsf{load}}_{\widehat{\sigma}}) \le \big(4+O(\varepsilon)\big)\alpha_{\mathcal{I}}^*\big(\min_{\sigma:J\mapsto[m]} f(\vec{\mathsf{load}}_\sigma)\big)$ for every monotone, symmetric norm f.*

6 Proofs from Sections 4 and 5

Proof of Claim 5. The bound follows easily from the definition of g. Let $x, y \in \mathbb{R}^{[m]\times J}$. Let $\vec{L}, \vec{L'} \in \mathbb{R}^m$ be the load vectors induced by x, y respectively; let $\vec{P}_S$, $\vec{P'}_S$, be the job-cost vectors for the jobs in S induced by x, y respectively. Then, $g(y) - g(x) \le \max\big\{f(L') - f(L), \max_{S\subseteq J:|S|=m} f(P'_S) - f(P_S)\big\}$. So $g(y) - g(x) \le K_f\|L' - L\|_2$ or $g(y) - g(x) \le K_f\|P'_S - P_S\|$ for some $S \subseteq J$ with $|S| = m$. Let $p_{\max} := \max_{i,j} p_{ij}$. In the former case, we have $g(y) - g(x) \le K_f p_{\max} \sum_{i,j} |y_{ij} - x_{ij}| \le \sqrt{mn} \cdot K_f p_{\max}\|y - x\|_2$; the same bound also applies in the latter case. This shows shows that $K = \sqrt{mn} \cdot K_f p_{\max}$ is a bound on the Lipschitz constant of g. ◀

The following claim will be useful in proving part 8 of Lemma 8, as also part (ii) of Lemma 12.

▷ Claim 14. Let $h : \mathbb{R}^N \mapsto \mathbb{R}$ be defined by $h(x) := \max_{r\in[k]} h_r(x)$, where $h_r : \mathbb{R}^N \mapsto \mathbb{R}$ is convex for all $r \in [k]$. Let alg_r be an ω-first order oracle for h_r for all $r \in [k]$ (where $\omega < 1$).

- One can obtain a 2ω-first order oracle for h using $O(1)$ calls to $\mathsf{alg}_1, \ldots \mathsf{alg}_k$.
- More generally, suppose that given $x \in \mathbb{R}^n$, one can identify $I(x) \subseteq [k]$ such that $h(x) = \max_{r\in I(x)} h_r(x)$. Then, one can compute a 2ω-first-order oracle for h that, on input $x \in \mathbb{R}^n$, makes $O(1)$ calls to alg_r for all $r \in I(x)$.

Proof. We focus on proving part (i); part (ii) follows from a very similar argument. Fix $x \in \mathbb{R}^N$. For every $r \in [k]$, we call alg_r to obtain an estimate est^r of $h_r(x)$. We set the estimate for $h(x)$ to be $\mathsf{est} := \max_{r\in[k]} \mathsf{est}^r$. From the properties of est^r, it is easy to see that $h(x) \le \mathsf{est} \le (1+\omega)h(x)$.

Let d^r be the ω-subgradient of f_r at x returned by alg_r. Let $s \in [k]$ be such that $\mathsf{est} = \mathsf{est}^s$. We set $\mu = d^s$. We now argue that μ is a 2ω-subgradient of h at x. Consider any $y \in \mathbb{R}^N$. We have

$$\begin{aligned}\mu^T(y-x) = (y-x)^T d^s &\le h_s(y) - h_s(x) + \omega h_s(x) \le h(y) - \tfrac{1-\omega}{1+\omega}\cdot \mathsf{est}^s = h(y) - \tfrac{1-\omega}{1+\omega}\cdot \mathsf{est}\\ &\le h(y) - \tfrac{1-\omega}{1+\omega}\cdot h(x) \le h(y) - (1-2\omega)h(x).\end{aligned}$$

The first two inequalities follow due to the fact that (est^s, d^s) was returned by the ω-first order oracle for h_s; the next equality follows from the definition of index s; and the penultimate inequality follows since $\mathsf{est} \ge h(x)$ as established earlier.

The proof of the more general statement in (ii) is essentially identical: on input x, we now run alg^r for all $r \in I(x)$; we set $\mathsf{est} = \max_{r\in I(x)} \mathsf{est}^r$, and $d = d^s$, where $s \in I(x)$ is an index such that $\mathsf{est} = \mathsf{est}^s$. ◁

Proof of Lemma 8. For part 8, fix $x \in \mathbb{R}^{[m]\times J}$. Recall that $P_j = P_j(x) := \sum_i p_{ij}x_{ij}$, Let S^* be the set of m jobs with the highest P_j values. Let $\vec{L} = L(\vec{x})$ and $\vec{P}_{S^*} = P(\vec{x})_{S^*}$. Then, $g(x) = \max\big\{f(\vec{L}), f(\vec{P}_{S^*})\big\}$. Observe that alg can be used to obtain an ω-first-order oracle for both $f\big(L(\vec{x})\big)$ and $f\big(P(\vec{x})_{S^*}\big)$. Thus, by using Claim 14 (ii), we obtain a 2ω-first-order oracle for g using $O(1)$ calls to alg.

We now justify the observation. A $(1+\omega)$-approximate value oracle is obtained by simply calling alg to obtain estimates of $f(\vec{L})$ and $f(\vec{P}_{S^*})$. Let $d^L = (d^L_i)_{i\in[m]}$, and $d^P = (d^P_j)_{j\in S^*}$ be the ω-subgradients of f at $\vec{L}$ at $\vec{P}_{S^*}$ respectively returned by alg.

$$\text{For all } i\in[m], j\in J, \text{ define} \qquad \beta_{ij} = p_{ij}d^L_i, \qquad \gamma_{ij} = \begin{cases} p_{ij}d^P_j & \text{if } j\in S^*; \\ 0 & \text{otherwise.}\end{cases}$$

Then, for any $y\in\mathbb{R}^{[m]\times J}$, we have $\beta^T(y-x) = \sum_{i,j} d^L_i p_{ij}(y_{ij}-x_{ij}) = (L(\vec{y}) - L(\vec{x}))^T d^L$ showing that β is an ω-subgradient of $f\big(L(\vec{\cdot})\big)$ at x. Similarly, $\gamma^T(y-x) = \big(P(\vec{y})_{S^*} - P(\vec{x})_{S^*}\big)^T d^P$ showing that γ is an ω-subgradient of $f\big(P(\vec{\cdot})\big)$ at x.

For part 8, Let σ^* be an optimal assignment. Since we are assuming that $O^*\geq 1$, we have $\mathsf{load}_{\sigma^*}(i)\geq 1$ for some $i\in[m]$. Let $e_i\in\mathbb{R}^m$ be the vector with 1 in coordinate i and 0s everywhere else. Then, $O^*\geq f(e_1)$. Let lb be the estimate of $f(e_1)$ obtained by alg scaled down by $(1+\omega)$. So we have $f(e_1)/(1+\omega)\leq \mathsf{lb}\leq O^*$. Consider any $x,y\in\mathbb{R}^m$. We have $y = x + \sum_{i=1}^m (y_i-x_i)e_i$, so by the triangle inequality and symmetry, we have $|f(y)-f(x)|\leq \sum_{i=1}^m |y_i-x_i| f(e_i)$ Therefore, $|f(y)-f(x)|\leq (1+\omega)\mathsf{lb}\sum_{i=1}^m |y_i-x_i| \leq (1+\omega)\sqrt{m}\cdot\mathsf{lb}\cdot\|y-x\|$. So we can set $K_f = (1+\omega)\sqrt{m}\cdot\mathsf{lb}$. ◀

Proof of Lemma 12. Part (i) follows by applying Claim 5 to each norm f_r, and since the Lipschitz constant of the maximum of a collection of functions is bounded by the maximum of the Lipschitz constants of the functions in the collection. Let $p_{\max} = \max_{i,j} p_{ij}$. By Claim 5, for each $r\in[k]$, and $S\subseteq J$ with $|S|=m$, both $f_r(L(\vec{x}))/T_r$ and $f_r(P(\vec{x})_S)/T_r$ have Lipschitz constant at most $\sqrt{mn}\cdot p_{\max}\cdot K_r/T_r \leq (1+\omega)m\sqrt{n}p_{\max}$. Hence, the Lipschitz constant of q is at most $K = (1+\omega)m\sqrt{n}p_{\max}$.

For part (ii), we mimic the proof of part 8 of Lemma 8. Fix $x\in\mathbb{R}^{[m]\times J}$. Let S^* be the set of m jobs with the highest $P_j(x)$ values, where $P_j(x) := \sum_i p_{ij}x_{ij}$. Let $\vec{L} = L(\vec{x})$ and $\vec{P}_{S^*} = P(\vec{x})_{S^*}$. Then,

$$q(x) = \max\Big\{\max_{r\in[k]} f_r(\vec{L})/T_r,\ \ \max_{r\in[k]} f_r(\vec{P}_{S^*})/T_r\Big\}.$$

As in the proof of Lemma 8 8, for each $r\in[k]$, we can use alg_r to obtain an ω-first-order oracle for $f_r\big(L(\vec{x})\big)/T_r$ and $f_r\big(P(\vec{x})_{S^*}\big)/T_r$. Thus, by using Claim 14 (ii), we obtain a 2ω-first-order oracle for q using $O(1)$ calls to alg_r, for each $r\in[k]$. ◀

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, FOCS*, pages 61–72, 2017.
2. Noga Alon, Yossi Azar, Gerhard Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.
3. Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *Proceedings, STOC*, pages 331–337, 2005.
4. Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.
5. Jarosław Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k-median. In *Proceedings, STOC*, pages 620–631, 2018.
6. Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. On $(1,\varepsilon)$-restricted assignment makespan minimization. In *Proceedings, SODA*, pages 1087–1101, 2015.

7 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between k-median and k-center: Approximation Algorithms for Ordered k-median. In *Proceedings, ICALP*, pages 29:1–29:14, 2018.

8 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings, STOC*, pages 126–137, 2019.

9 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.

10 Tomáš Ebenlendr, Marek Krčál, and Jiří Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014.

11 Ashish Goel and Adam Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica*, 44(4):301–323, 2006.

12 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

13 Dorit S. Hochbaum and David B. Shmoys. A Best Possible Heuristic for the k-Center Problem. *Math. Oper. Res.*, 10(2):180–184, 1985.

14 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.

15 Klaus Jansen and Lars Rohwedder. On the configuration-LP of the restricted assignment problem. In *Proceedings, SODA*, pages 2670–2678, 2017.

16 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.

17 G. Laporte, S. Nickel, and F. S. da Gama. *Location Science*. Springer, 2015.

18 Jan K. Lenstra, David B. Shmoys, and Éva. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46(1-3):259–271, 1990.

19 Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. In *Proceedings, FOCS*, pages 571–580, 2014.

20 A. Nemirovski and Yudin D. *Problem complexity and method efficiency in optimization*. John Wiley and Sons, 1983.

21 S. Nickel and J. Puerto. *Location Theory: A Unified Approach*. Springer Science & Business Media, 2005.

22 David B. Shmoys and Chaitanya Swamy. An Approximation Scheme for Stochastic Linear Programming and Its Application to Stochastic Integer Programs. *Journal of the ACM*, 53(6):978–1012, 2006.

23 David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1-3):461–474, 1993.

24 Ola Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318–1341, 2012.