A Fast Distributed Stateless Algorithm for α-Fair Packing Problems*

Jelena Marašević, Clifford Stein, and Gil Zussman

Columbia University, New York, NY, USA
{jelena@ee, cliff@ieor, gil@ee}.columbia.edu

Abstract

We study weighted α-fair packing problems, that is, the problems of maximizing the objective functions (i) $\sum_j w_j x_j^{1-\alpha}/(1-\alpha)$ when $\alpha > 0$, $\alpha \neq 1$ and (ii) $\sum_j w_j \ln x_j$ when $\alpha = 1$, over linear constraints $Ax \leq b$, $x \geq 0$, where w_j are positive weights and A and b are non-negative. We consider the distributed computation model that was used for packing linear programs and network utility maximization problems. Under this model, *we provide a distributed algorithm for general* α that converges to an $\varepsilon-$approximate solution in time (number of distributed iterations) that has an inverse polynomial dependence on the approximation parameter ε and poly-logarithmic dependence on the problem size. *This is the first distributed algorithm for weighted $\alpha-$fair packing with poly-logarithmic convergence in the input size.* The algorithm uses simple local update rules and is stateless (namely, it allows asynchronous updates, is self-stabilizing, and allows incremental and local adjustments). We also obtain a number of structural results that characterize $\alpha-$fair allocations as the value of α is varied. These results deepen our understanding of fairness guarantees in $\alpha-$fair packing allocations, and also provide insight into the behavior of $\alpha-$fair allocations in the asymptotic cases $\alpha \to 0$, $\alpha \to 1$, and $\alpha \to \infty$.

1998 ACM Subject Classification F.2.1 [Theory of Computation]: Analysis of Algorithms and Problem Complexity – Numerical Algorithms and Problems; G.1.6 [Mathematics of Computing]: Numerical Analysis – Optimization, Convex programming, Gradient methods

Keywords and phrases Fairness, distributed and stateless algorithms, resource allocation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.XXX

1 Introduction

Over the past two decades, *fair resource allocation* problems have received considerable attention in many application areas, including Internet congestion control [32], rate control in software defined networks [36], scheduling in wireless networks [45], multi-resource allocation and scheduling in datacenters [12, 20, 21, 24], and a variety of applications in operations research, economics, and game theory [11, 23]. In most of these applications, positive linear (packing) constraints arise as a natural model of the allowable allocations.

In this paper, we focus on the problem of finding an α-*fair* vector on the set determined by packing constraints $Ax \leq \mathbb{1}, x \geq 0$ where all $A_{ij} \geq 0$.[1] We refer to this problem as $\alpha-$fair

* The full version of the paper can be found at http://arxiv.org/abs/1502.03372v3. This work was partially supported by the NSF grants CNS-14-23105, CCF-1349602, and CCF-1421161, and the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n°[PIIF-GA-2013-629740].11.

[1] Although in the network congestion control literature A is commonly assumed to be a 0-1 matrix [25, 26, 32, 38, 45], important applications (such as, e.g., multi-resource allocation in datacenters) are modeled by a more general constraint matrix A with arbitrary non-negative elements [12, 20, 21, 24].

packing. For a vector of positive weights w and $\alpha \geq 0$, an allocation vector x^* of size n is weighted α-fair , if it maximizes $p_\alpha(x) = \sum_j w_j f_\alpha(x_j)$ [38], where:

$$f_\alpha(x_j) = \begin{cases} \ln(x_j), & \text{if } \alpha = 1 \\ \frac{x_j^{1-\alpha}}{1-\alpha}, & \text{if } \alpha \neq 1 \end{cases}. \tag{1}$$

α-fairness provides a trade-off between efficiency (sum of allocated resources) and fairness (minimum allocated resource) as a function of α: the higher the α, the higher the fairness and the lower the efficiency [4,11,31]. Important special cases are max-min fairness ($\alpha \to \infty$) and proportional fairness ($\alpha = 1$). When $\alpha = 0$, we have the "unfair" case of linear optimization.

Distributed algorithms for $\alpha-$fair packing are of particular interest, as many applications are inherently distributed (e.g., network congestion control), while others require parallelization due to the large problem size (e.g., resource allocation in datacenters). We adopt the model of distributed computation commonly used in packing linear programming (LP) algorithms [3,7,8,29,33,41] and which generalizes the model from network congestion control [26]. In this model, an agent j controls the variable x_j and has information about: (i) the j^{th} column of the $m \times n$ constraint matrix A, (ii) the weight w_j, (iii) upper bounds on global problem parameters $m, n, w_{\max}$, and $A_{\max}$, where $w_{\max} = \max_j w_j$, and $A_{\max} = \max_{ij} A_{ij}$, and (iv) in each round, the relative slack of each constraint i in which x_j takes part.

Distributed algorithms for $\alpha-$fair resource allocations have been most widely studied in the network congestion control literature, using a control-theoretic approach [25,26,32,38,45]. Such an approach yields continuous-time algorithms that converge after "finite" time; however, the convergence time of these algorithms as a function of the input size is poorly understood. Some other distributed pseudo-polynomial-time approximation algorithms that can address α-fair packing are described in Table 1. These algorithms all have convergence times that are at least linear in the parameters describing the problem.

No previous work has given truly fast (poly-log iterations) distributed algorithms for the general case of α-fair packing. Only for the unfair $\alpha = 0$ case (packing LPs), are such algorithms known [3,7,8,29,33,46].

Our Results. *We provide the first efficient, distributed, and stateless algorithm for weighted α-fair packing*, namely, for the problem $\max\{p_\alpha(x) : Ax \leq \mathbb{1}, x \geq 0\}$, where distributed agents update the values of x_j's asynchronously and react only to the current state of the constraints. We assume that all non-zero entries A_{ij} of matrix A satisfy $A_{ij} \geq 1$. Considering such a normalized form of the problem is without loss of generality (see Appendix A in [35]).

The approximation provided by the algorithm, to which we refer as the ε-approximation, is (i) $(1+\varepsilon)$-multiplicative for $\alpha \neq 1$, and (ii) $W\varepsilon$-additive[2] for $\alpha = 1$, where $W = \sum_j w_j$. The main results are summarized in the following theorem, where, to unify the statement of the results, we treat α as a constant that is either equal to 1 or bounded away from 0 and 1, and we also loosen the bound in terms of $\varepsilon^{-1}, n, m, R_w = \max_{j,k} w_j/w_k$, and $A_{\max}$. For a more detailed statement, see Theorems 4.1 – 4.3.

▶ **Theorem 1.1.** *(Main Result) Given a weighted α-fair packing problem $\max\{\sum_j w_j f_\alpha(x_j) : Ax \leq \mathbb{1}, x \geq 0\}$, there exists a stateless and distributed algorithm (α-FairPSolver) that computes an ε-approximate solution in $O(\varepsilon^{-5} \ln^4(R_w n m A_{\max} \varepsilon^{-1}))$ rounds.*

[2] Note that W cannot be avoided here, as additive approximation is not invariant to the objective scaling.

To the best of our knowledge, for any constant approximation parameter ε, our algorithm is the first distributed algorithm for weighted α-fair packing problems with a poly-logarithmic convergence time.[3] The algorithm is *stateless* according to the definition given by Awerbuch and Khandekar [6, 7]: it starts from any initial state, the agents update the variables x_j in a cooperative but uncoordinated manner, reacting only to the current state of the constraints that they observe, and without access to a global clock. Statelessness implies various desirable properties of a distributed algorithm, such as: asynchronous updates, self-stabilization, and incremental and local adjustments [6, 7].

We also obtain the following structural results that characterize $\alpha-$fair packing allocations as a function of the value of α:

- We derive a lower bound on the minimum coordinate of the $\alpha-$fair packing allocation as a function of α and the problem parameters (Lemma 4.12). This bound deepens our understanding of how the fairness (a minimum allocated value) changes with α.
- We prove that for $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, $\alpha-$fair packing can be $O(\varepsilon)-$approximated by any $\varepsilon-$approximation packing LP solver (Lemma 4.13).
- We show that for $|\alpha-1| = O(\varepsilon^2/\ln^2(\varepsilon^{-1}R_w mnA_{\max}))$, $\alpha-$fair allocation is $\varepsilon-$approximated by a $1-$fair allocation returned by our algorithm (Lemmas 4.14 and 4.15).
- We show that for $\alpha \geq \ln(R_w nA_{\max})/\varepsilon$, the $\alpha-$fair packing allocation x^* and the max-min fair allocation z^* are ε-close to each other: $(1-\varepsilon)z^* \leq x^* \leq (1+\varepsilon)z^*$ element-wise. This result is especially interesting as (i) max-min fair packing is not a convex problem, but rather a multi-objective problem (see, e.g., [27, 43]) and (ii) the result yields the first convex relaxation of max-min fair allocation problems with a $1 \pm \varepsilon$ gap.

We now overview some of the main technical details of α-FairPSolver. In doing so, we point out connections to the two main bodies of previous work, from packing LPs [7] and network congestion control [25]. We also outline the new algorithmic ideas and proofs.

The algorithm and KKT conditions. The algorithm maintains primal and dual feasible solutions and updates each primal variable x_j whenever a Karush-Kuhn-Tucker (KKT) condition ${x_j}^{\alpha}\sum_i y_i A_{ij} = w_j$ is not *approximately* satisfied. In previous work, relevant update rules include: [25] (for $\alpha = 1$), where the update of each variable x_j is proportional to the difference $w_j - x_j\sum_i y_i A_{ij}$, and [7] (for $\alpha = 0$), where each x_j is updated by a multiplicative factor $1 \pm \beta$, whenever $\sum_i y_i A_{ij} = w_j$ is not approximately satisfied. For our techniques (addressing a general α) such rules do not suffice and we introduce the following modifications: (i) in the $\alpha < 1$ case we use multiplicative updates by factors $(1+\beta_1)$ and $(1-\beta_2)$, where $\beta_1 \neq \beta_2$ and (ii) we use additional threshold values δ_j to make sure that x_j's do not become too small. These thresholds guarantee that we maintain a feasible solution, but they significantly complicate (compared to the linear case) the argument that each step makes a significant progress.

Dual Variables. In α-FairPSolver, a dual variable y_i is an exponential function of the i^{th} constraint's relative slack: $y_i(x) = C \cdot e^{\kappa(\sum_j A_{ij}x_j - 1)}$, where C and κ are functions of global input parameters $\alpha, w_{\max}, n, m$, and $A_{\max}$. Packing LP algorithms [3, 7, 8, 17, 18, 28, 42] use similar dual variables with $C = 1$. Our work requires choosing C to be a function of $\alpha, w_{\max}, n, m, A_{\max}$ rather than a constant.

[3] The total amount of work per (distributed) round is linear in the number of non-zero entries of the constraint matrix A, which matches the best bound achieved in previous work on distributed packing LPs [3, 7, 8, 29, 33, 46] and distributed network utility maximization [9, 39].

Paper	Number of Distributed Iterations[4]	Statelessness	Notes
[15]	$\Omega(\varepsilon^{-1}nA_{\max})$	Semi-stateless[5]	Only for $\alpha = 1$
[9]	$\Omega(\varepsilon^{-1}mn{A_{\max}}^2)$	Not stateless	
[39]	$\text{poly}(\varepsilon^{-1}, m, n, A_{\max})$	Semi-stateless	
[this work]	$O(\varepsilon^{-5}\ln^4(R_w mnA_{\max}/\varepsilon))$	Stateless	

Table 1 Comparison among distributed algorithms for $\alpha-$fair packing.

Convergence Argument. The convergence analysis of α-FairPSolver relies on the appropriately chosen concave potential function that is bounded below and above for $x_j \in [\delta_j, 1]$, $\forall j$, and that increases with every primal update. The algorithm can also be interpreted as a gradient ascent on a regularized objective function (the potential function), using a generalized entropy regularizer (see [1,3]). A similar potential function was used in many works on packing and covering linear programs, such as, e.g., in [7] and (implicitly) in [46]. The Lyapunov function from [25] is also equivalent to this potential function when $y_i(x) = C \cdot e^{\kappa(\sum_j A_{ij}x_j - 1)}$, $\forall i$. As in these works, the main idea in the analysis is to show that whenever a solution x is not "close" to the optimal one, the potential function increases substantially. However, our work requires several new ideas in the convergence proofs, the most notable being *stationary rounds*. A stationary round is roughly a time when the variables x_j do not change much and are close to the optimum. Poly-logarithmic convergence time is then obtained by showing that: (i) there is at most a poly-logarithmic number of non-stationary rounds where the potential function increases additively and the increase is "large enough", and (ii) in all the remaining non-stationary rounds, the potential function increases multiplicatively. Our use of stationary rounds is new, as is the use of Lagrangian duality and all the arguments that follow (see [35] for a detailed discussion).

Relationship to Previous Work. Very little progress has been made in the design of efficient distributed algorithms for the general class of α-fair objectives. Classical work on distributed rate control algorithms in the networking literature uses a control-theoretic approach to optimize α-fair objectives. While such an approach has been extensively studied [25, 26, 32, 38, 45], it has never been proven to lead to a polynomial convergence time.

Since α-fair objectives are concave, their optimization over a region determined by linear constraints is solvable in polynomial time in a centralized setting through convex programming (see, e.g., [13, 40]). Distributed gradient methods for network utility maximization problems, such as e.g., [9, 39] summarized in Table 1, can be applied to α-fair packing. However, the convergence times of these algorithms depend on the dual gradient's Lipschitz constant to produce good approximations. While [9, 39] provide a better dependence on the accuracy ε than our work, the dependence on the dual gradient's Lipschitz constant, in general, leads to at least linear convergence time as a function of n, m, and $A_{\max}$.

As mentioned before, some special cases have been addressed, particularly max-min fairness ($\alpha \to \infty$) and packing LPs ($\alpha = 0$). Relevant work on max-min fairness includes

[4] The convergence times in [9, 15, 39] are not stated only in terms of the input parameters, but also in terms of intermediary parameters that depend on the problem structure. Stated here are our lowest estimates of the worst-case convergence times.

[5] A distributed algorithm is semi-stateless, if all the updates depend only on the current state of the constraints, the updates are performed in a cooperative but non-coordinated manner, and *the updates need to be synchronous* [3].

[10, 14, 22, 27, 30, 34, 37], but none of these works have poly-logarithmic convergence time. There is a long history of interesting work on packing LPs in both centralized and distributed settings, e.g., [1, 3, 7, 8, 18, 19, 28, 29, 33, 42, 46]. Only a few of these works are stateless, including the packing LP algorithm of Awerbuch and Khandekar [7], flow control algorithm of Garg and Young [19], and the algorithm of Awerbuch, Azar, and Khandekar [5] for the special case of load balancing in bipartite graphs. Additionally, the packing LP algorithm of Allen-Zhu and Orecchia [3] is "semi-stateless"; the lacking property to make it stateless is that it requires synchronous updates. The $\alpha = 1$ case of α-fair packing problems is equivalent to the problem of finding an equilibrium allocation in Eisenberg-Gale markets with Leontief utilities [15]. Similar to the aforementioned algorithms, the algorithm from [15] converges in time linear in ε^{-1} but also (at least) linear in the input size (see Table 1).

2 Preliminaries

Weighted α-Fair Packing. Consider the following optimization problem with positive linear (packing) constraints: $(Q_\alpha) = \max\{p_\alpha(x) \equiv \sum_{j=1}^{n} w_j f_\alpha(x_j) : Ax \leq b, x \geq 0\}$, where $f_\alpha(x_j)$ is given by (1), $x = (x_1, ..., x_n)$ is the vector of variables, A is an $m \times n$ matrix with non-negative elements, and $b = (b_1, ..., b_m)$ is a vector with strictly positive[6] elements. We refer to (Q_α) as the weighted α-fair packing. The following definition and lemma introduced by Mo and Walrand [38] characterize weighted α-fair allocations. In the rest of the paper, we will use the terms weighted α-fair and α-fair interchangeably.

▶ **Definition 2.1.** [38] Let w be a vector with positive entries and $\alpha > 0$. A vector x is weighted α-fair, if it is feasible and for any other feasible vector x: $\sum_{j=1}^{n} w_j \frac{x_j - x_j^*}{x_j^{*\alpha}} \leq 0$.

▶ **Lemma 2.2.** [38] *A vector x^* solves (Q_α) if and only if it is weighted α-fair.*

Notice in (Q_α) that since $b_i > 0$, $\forall i$, and the partial derivative of the objective with respect to any of the variables x_j goes to ∞ as $x_j \to 0$, the optimal solution must lie in the positive orthant. Moreover, since the objective is strictly concave and maximized over a convex region, the optimal solution is unique and (Q_α) satisfies strong duality (see, e.g., [13]). The same observations are true for the scaled version of the problem denoted by (P_α) and introduced in the following subsection.

Normalized Form. We consider weighted α-fair packing in the normalized form:

$$(P_\alpha) = \max\{p_\alpha(x) : Ax \leq \mathbb{1}, x \geq 0\},$$

where $p_\alpha(x) = \sum_{j=1}^{n} w_j f_\alpha(x_j)$, f_α is defined by (1), $w = (w_1, ..., w_n)$ is a vector of positive weights, $x = (x_1, ..., x_n)$ is the vector of variables, A is an $m \times n$ matrix with non-negative entries, and $\mathbb{1}$ is a size-m vector of 1's. We let $A_{\max}$ denote the maximum element of the constraint matrix A, and assume that every entry A_{ij} of A is non-negative, and moreover, that $A_{ij} \geq 1$ whenever $A_{ij} \neq 0$. The maximum weight is denoted by $w_{\max}$ and the minimum weight is denoted by $w_{\min}$. The sum of the weights is denoted by W and the ratio $\frac{w_{\max}}{w_{\min}}$ by R_w. We remark that considering (Q_α) in the normalized form (P_α) is without loss of generality: any problem (Q_α) can be scaled to this form by (i) dividing both sides of each inequality i by b_i and (ii) working with scaled variables $c \cdot x_j$, where $c = \min\{1, \min_{\{i,j:A_{ij}\neq 0\}} \frac{A_{ij}}{b_i}\}$. Moreover, such scaling preserves the approximation (see [35]).

[6] If, for some i, $b_i = 0$, then trivially $x_j = 0$, for all j such that $A_{ij} \neq 0$.

Model of Distributed Computation We adopt the same model of distributed computation as [3, 7, 8, 29, 33, 41], described as follows. We assume that for each $j \in \{1, ..., n\}$, there is an agent controlling the variable x_j. Agent j is assumed to have information about the following problem parameters: (i) the j^{th} column of A, (ii) the weight w_j, and (iii) (an upper bound on) $m, n, w_{\max}$, and $A_{\max}$. In each round, agent j collects the relative slack[7] $1 - \sum_{j=1}^{n} A_{ij}x_j$ of all constraints i for which $A_{ij} \neq 0$.

We remark that this model of distributed computation is a generalization of the model considered in network congestion control problems [26] where a variable x_j corresponds to the rate of node j, A is a 0-1 routing matrix, such that $A_{ij} = 1$ if and only if a node j sends flow over link i, and b is the vector of link capacities. Under this model, the knowledge about the relative slack of each constraint corresponds to each node collecting (a function of) congestion on each link that it utilizes. Such a model was used in network utility maximization problems with α-fair objectives [25] and general strongly-concave objectives [9].

KKT Conditions and Duality Gap We will denote the Lagrange multipliers for (P_α) as $y = (y_1, ..., y_m)$ and refer to them as "dual variables". The KKT conditions for (P_α) are:

$$\sum_{j=1}^{n} A_{ij}x_j \leq 1,\ \forall i \in \{1, ..., m\};\ x_j \geq 0,\ \forall j \in \{1, ..., n\} \text{ (primal feasibility)} \tag{K1}$$

$$y_i \geq 0,\ \forall i \in \{1, ..., m\} \text{ (dual feasibility)} \tag{K2}$$

$$y_i \cdot \Big(\sum_{j=1}^{m} A_{ij}x_j - 1 \Big) = 0,\ \forall i \in \{1, ..., m\} \text{ (complementary slackness)} \tag{K3}$$

$${x_j}^{\alpha} \sum_{i=1}^{m} y_i A_{ij} = w_j,\ \forall j \in \{1, ..., m\} \text{ (gradient conditions)} \tag{K4}$$

The duality gap for $\alpha \neq 1$ is (see Appendix B in [35]):

$$G_\alpha(x, y) = \sum_{j=1}^{n} w_j \frac{{x_j}^{1-\alpha}}{1-\alpha} \big({\xi_j}^{\frac{\alpha-1}{\alpha}} - 1 \big) + \sum_{i=1}^{m} y_i - \sum_{j=1}^{n} w_j x_j^{1-\alpha} \cdot {\xi_j}^{\frac{\alpha-1}{\alpha}}, \tag{2}$$

where $\xi_j = \frac{{x_j}^{\alpha} \sum_{i=1}^{m} y_i A_{ij}}{w_j}$, while for $\alpha = 1$:

$$G_1(x, y) = -\sum_{j=1}^{n} w_j \ln \Big(\frac{x_j \sum_{i=1}^{m} y_i A_{ij}}{w_j} \Big) + \sum_{i=1}^{m} y_i - W. \tag{3}$$

3 Algorithm

The pseudocode for the α-FairPSolver algorithm run at each node j is provided in Fig 1. The basic intuition is that the algorithm keeps KKT conditions (K1) and (K2) satisfied and works towards (approximately) satisfying the remaining two KKT conditions (K3) and (K4) to minimize the duality gap. The algorithm can run in the distributed setting described in Section 2. In each round, an agent j updates the value of x_j based on the relative slack of all the constraints in which j takes part, as long as the KKT condition (K4) of agent j is not approximately satisfied. The updates need not be synchronous: we will require that all agents make updates at the same speed, but without access to a global clock.

[7] The slack is "relative" because in a non-scaled version of the problem where one could have $b_i \neq 1$, agent j would need to have information about $\frac{b_i - \sum_{j=1}^{n} A_{ij}x_j}{b_i}$.

α-FairPSolver(ε)

(Parameters $\delta_j, C, \kappa, \gamma, \beta_1$, and β_2 are set as described in the text below the algorithm.)

In each round of the algorithm:

1: $x_j \leftarrow \max\{x_j, \delta_j\}$, $x_j = \min\{x_j, 1\}$

2: Update the dual variables: $y_i = C \cdot e^{\kappa\left(\sum_{j=1}^{n} A_{ij}x_j - 1\right)}$ $\forall i \in \{1, ..., m\}$

3: **if** $\frac{{x_j}^{\alpha} \cdot \sum_{i=1}^{m} y_i A_{ij}}{w_j} \leq (1-\gamma)$ **then**

4: $\quad x_j \leftarrow x_j \cdot (1+\beta_1)$

5: **else**

6: $\quad$ **if** $\frac{{x_j}^{\alpha} \cdot \sum_{i=1}^{m} y_i A_{ij}}{w_j} \geq (1+\gamma)$ **then**

7: $\quad\quad x_j \leftarrow \max\{x_j \cdot (1-\beta_2), \delta_j\}$

Figure 1 Pseudocode of α-FairPSolver algorithm.

To allow for self-stabilization and dynamic changes, the algorithm runs forever at all the agents, which is a standard requirement for self-stabilizing algorithms (see, e.g., [16]). The convergence of the algorithm is measured as the number of rounds between the round in which the algorithm starts from some initial solution and the round in which it reaches an $\varepsilon-$approximate solution, assuming that there are no hard reset events or node/constraint insertions/deletions in between.

Without loss of generality, we assume that the input parameter ε that determines the approximation quality satisfies $\varepsilon \leq \min\{\frac{1}{6}, \frac{9}{10\alpha}\}$ for any α, and $\varepsilon \leq \frac{1-\alpha}{\alpha}$ for $\alpha < 1$. The parameters $\delta_j, C, \kappa, \gamma$, β_1, and β_2 are set as follows. For technical reasons (mainly due to reinforcing dominant multiplicative updates of the variables x_j), we set the values of the lower thresholds δ_j below the actual lower bound of the optimal solution that we derive in Lemma 4.12:

$$\delta_j = \left(\frac{1}{2} \cdot \frac{w_j}{w_{\max}}\right)^{1/\alpha} \cdot \begin{cases} \left(\frac{1}{m \cdot n^2 \cdot A_{\max}}\right)^{1/\alpha}, & \text{if } 0 < \alpha \leq 1 \\ \frac{1}{m \cdot n^2 {A_{\max}}^{2-1/\alpha}}, & \text{if } \alpha > 1 \end{cases}.$$

We denote $\delta_{\max} \equiv \max_j \delta_j$, $\delta_{\min} \equiv \min_j \delta_j$. The constant C that multiplies the exponent in the dual variables y_i is chosen as $C = \frac{W}{\sum_{j=1}^{n} {\delta_j}^{\alpha}}$. Because δ_j only depends on w_j and on global parameters, we also have $C = \frac{w_j}{{\delta_j}^{\alpha}}$, $\forall j$. The parameter κ that appears in the exponent of the y_i's is chosen as $\kappa = \frac{1}{\varepsilon} \ln\left(\frac{CmA_{\max}}{\varepsilon w_{\min}}\right)$. The "absolute error" of (K4) γ is set to $\varepsilon/4$. For $\alpha \geq 1$, we set $\beta_1 = \beta_2 = \beta$, where the choice of β is described below. For $\alpha < 1$, we set $\beta_1 = \beta$, $\beta_2 = \beta^2(\ln(\frac{1}{\delta_{\min}}))^{-1}$.

Similar to [7], we choose the value of β so that if we set $\beta_1 = \beta_2 = \beta$, in any round the value of each $\frac{{x_j}^{\alpha} \sum_{i=1}^{m} y_i(x) A_{ij}}{w_j}$ changes by a multiplicative factor of at most $(1 \pm \gamma/4)$. Since the maximum increase over any x_j in each iteration is by a factor $1+\beta$, and x is feasible in each round (see Lemma 4.4), we have that $\sum_{j=1}^{n} A_{ij}x_j \leq 1$, and therefore, the maximum increase in each y_i is by a factor of $e^{\kappa\beta}$. A similar argument holds for the maximum decrease. Hence, we choose β so that:

$$(1+\beta)^{\alpha} e^{\kappa\beta} \leq 1 + \gamma/4 \quad \text{and} \quad (1-\beta)^{\alpha} e^{-\kappa\beta} \geq 1 - \gamma/4,$$

and it suffices to set:

$$\beta = \begin{cases} \frac{\gamma}{5(\kappa+1)}, & \text{if } \alpha \leq 1 \\ \frac{\gamma}{5(\kappa+\alpha)}, & \text{if } \alpha > 1 \end{cases}.$$

Remark: In the $\alpha < 1$ cases, since $\beta_2 = \beta^2(\ln(1/\delta_{\min}))^{-1}$, the maximum decrease in $\frac{{x_j}^{\alpha}\sum_i y_i(x)A_{ij}}{w_j}$ is by a factor $(1-(\gamma/4)\cdot\beta(\ln(1/\delta_{\min}))^{-1})$, $\forall j$.

4 Convergence Analysis

In this section, we analyze the convergence time of α-FairPSolver. We first state our main theorems and provide some general results that hold for all $\alpha > 0$. We show that starting from an arbitrary solution, the algorithm reaches a feasible solution within poly-logarithmic (in the input size) number of rounds, and maintains a feasible solution forever after. Similar to [7, 25, 46], we use a concave potential function that, for feasible x, is bounded below and above and increases with any algorithm update. Then, we sketch the proof of Theorem 4.3 ($\alpha > 1$), while we defer the full proofs of the three theorems to the full paper [35]. The main proof idea in all the cases is as follows. With an appropriate definition of a *stationary round* for each of the three cases $\alpha < 1$, $\alpha = 1$, and $\alpha > 1$, we show that in every stationary round, x approximates "well" the optimal solution by bounding the duality gap. On the other hand, for any non-stationary round, we show that the potential increases substantially. This large increase in the potential leads to the conclusion that there cannot be too many non-stationary rounds, thus bounding the overall convergence time.

We make a few remarks here. First, we require that α be bounded away from zero. This requirement is without loss of generality because we show that when $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, any $\varepsilon-$approximation LP provides a $3\varepsilon-$approximate solution to (P_α) (Lemma 4.13). Thus, when $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$ we can switch to the algorithm of [7], and when $\alpha > \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, the convergence time remains poly-logarithmic in the input size and polynomial in ε^{-1}. Second, the assumption that $\varepsilon \leq \frac{1-\alpha}{\alpha}$ in the $\alpha < 1$ case is also without loss of generality, because we show that when α is close to 1 (roughly, $1 - O(\varepsilon^2/\ln^2(R_w mnA_{\max}/\varepsilon))$), we can approximate (P_α) by switching to the $\alpha = 1$ case of the algorithm (Lemma 4.14). Finally, when $\alpha > 1$, the algorithm achieves an $\varepsilon-$approximation in time $O(\alpha^4\varepsilon^{-4}\ln^2(R_w nmA_{\max}\varepsilon^{-1}))$. We believe that a polynomial dependence on α is difficult to avoid in this setting, because by increasing α, the gradient of the α-fair utilities f_α blows up on the interval $(0,1)$: as α increases, $f_\alpha(x)$ quickly starts approaching a step function that is equal to $-\infty$ on the interval $(0,1]$ and equal to 0 on the interval $(1,\infty]$. To characterize the behavior of $\alpha-$fair allocations as α becomes large, we show that when $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, all the coordinates of the $\alpha-$fair vector are within a $1 \pm \varepsilon$ multiplicative factor of the corresponding coordinates of the max-min fair vector (Lemma 4.17).

Main Results. Our main results are summarized in the following three theorems. The objective is denoted by $p_\alpha(x)$, x^t denotes the solution at the beginning of round t, and x^* denotes the optimal solution.

▶ **Theorem 4.1.** *(Convergence for $\alpha < 1$) α-FairPSolver solves (P_α) approximately for $\alpha < 1$ in time that is polynomial in $\frac{\ln(nmA_{\max})}{\alpha\varepsilon}$. In particular, after at most*

$$O\left(\alpha^{-2}\varepsilon^{-5}\ln^2\left(R_w mnA_{\max}\right)\ln^2\left(\varepsilon^{-1}R_w mnA_{\max}\right)\right) \tag{4}$$

rounds, there exists at least one round t such that $p_\alpha(x^*) - p_\alpha(x^t) \leq \varepsilon p_\alpha(x^t)$. Moreover, the total number of rounds s in which $p_\alpha(x^*) - p_\alpha(x^s) > \varepsilon p_\alpha(x^s)$ is also bounded by (4).

▶ **Theorem 4.2.** *(Convergence for $\alpha = 1$) α-FairPSolver solves (P_1) approximately in time that is polynomial in $\varepsilon^{-1}\ln(R_w nmA_{\max})$. In particular, after at most*

$$O\left(\varepsilon^{-5}\ln^2\left(R_w nmA_{\max}\right)\ln^2\left(\varepsilon^{-1}R_w nmA_{\max}\right)\right) \tag{5}$$

rounds, there exists at least one round t such that $p(x^*) - p(x^t) \leq \varepsilon W$. Moreover, the total number of rounds s in which $p(x^*) - p(x^s) > \varepsilon W$ is also bounded by (5).

▶ **Theorem 4.3.** *(Convergence for $\alpha > 1$) α-FairPSolver solves (P_α) approximately for $\alpha > 1$ in time that is polynomial in $\varepsilon^{-1} \ln(nmA_{\max})$. In particular, after at most:*

$$O\left(\alpha^4 \varepsilon^{-4} \ln\left(R_w nmA_{\max}\right) \ln\left(\varepsilon^{-1} R_w nmA_{\max}\right)\right) \tag{6}$$

rounds, there exists at least one round t such that $p_\alpha(x^*) - p_\alpha(x^t) \leq \varepsilon(-p_\alpha(x^t))$. Moreover, the total number of rounds s in which $p_\alpha(x^*) - p_\alpha(x^s) > \varepsilon(-p_\alpha(x^s))$ is also bounded by (6).

Proofs of Theorem 4.1 and Theorem 4.2 are provided in the full paper [35]. We sketch the proof of Theorem 4.3 in Section 4.1.

Feasibility and Approximate Complementary Slackness. The following three lemmas are preliminaries for the convergence time analysis. Lemma 4.4 shows that starting from a feasible solution, the algorithm always maintains a feasible solution. Lemma 4.5 shows that any violated constraint becomes feasible within poly-logarithmic number of rounds, and remains feasible forever after. Combined with Lemma 4.4, Lemma 4.5 allows us to focus only on the rounds with feasible solutions x. Lemma 4.6 shows that after a poly-logarithmic number of rounds, approximate complementary slackness (KKT condition (K3)) holds in an aggregate sense: $\sum_{i=1}^m y_i(x)\left(\sum_{j=1}^n A_{ij}x_j - 1\right) \approx 0$. Proofs are provided in [35].

▶ **Lemma 4.4.** *If the algorithm starts from a feasible solution, then the algorithm maintains a feasible solution x: $x_j \geq 0$, $\forall j$ and $\sum_{j=1}^n A_{ij}x_j \leq 1$, $\forall i$, in each round.*

▶ **Lemma 4.5.** *If for any i: $\sum_{j=1}^n A_{ij}x_j > 1$, then after at most $\tau_1 = O(\frac{1}{\beta_2}\ln(nA_{\max}))$ rounds, it is always true that $\sum_{j=1}^n A_{ij}x_j \leq 1$.*

▶ **Lemma 4.6.** *If the algorithm starts from a feasible solution, then after at most $\tau_0 = \frac{1}{\beta}\ln\left(\frac{1}{\delta_{\min}}\right)$ rounds, it is always true that:*

1. *At least one constraint is approximately tight: $\max_i\left\{\sum_{j=1}^n A_{ij}x_j\right\} \geq 1 - (1 + 1/\kappa)\varepsilon$,*
2. *$\sum_{i=1}^m y_i \leq (1 + 3\varepsilon)\sum_{j=1}^n x_j \sum_{i=1}^m y_i A_{ij}$, and*
3. *$(1 - 3\varepsilon)\sum_{i=1}^m y_i \leq \sum_{j=1}^n x_j \sum_{i=1}^m y_i A_{ij} \leq \sum_{i=1}^m y_i$.*

Lemmas analogous to 4.4 and 4.6 also appear in [7]. However, the proofs of Lemmas 4.4 and 4.6 require new ideas compared to the proofs of the corresponding lemmas in [7]. We need to be much more careful in our choice of lower thresholds δ_j and constant C in the dual variables, particularly by choosing C as a function of several variables, rather than as a constant. The choice of δ_j's is also sensitive as smaller δ_j's would make the potential function range too large, while larger δ_j's would cause more frequent decrease of "small" variables. In either case, the convergence time would increase.

Decrease of Small Variables. The following lemma is also needed for the convergence analysis. It shows that if some variable x_j decreases by less than a multiplicative factor $(1 - \beta_2)$, i.e., $x_j < \frac{\delta_j}{1-\beta_2}$ and x_j decreases, then x_j must be part of at least one approximately tight constraint. This lemma will be used later to show that in any round the increase in the potential due to the decrease of "small" variables is dominated by the decrease of "large" variables (i.e., the variables that decrease by a multiplicative factor $(1 - \beta_2)$). The proof of Lemma 4.7 is provided in [35].

▶ **Lemma 4.7.** *Consider the rounds that happen after the initial $\tau_1 = O(\frac{1}{\beta_2}\ln(nA_{\max}))$ rounds. If in some round there is a variable $x_j < \frac{\delta_j}{1-\beta_2}$ that decreases, then in the same round for some i with $A_{ij} \neq 0$ it holds that: $y_i(x) \geq \frac{\sum_{l=1}^m A_{lj}y_l(x)}{mA_{\max}}$ and $\sum_{k=1}^n A_{ik}x_k > 1 - \frac{\varepsilon}{2}$.*

Potential. We use the following potential function to analyze the convergence time:

$$\Phi(x) = p_\alpha(x) - \frac{1}{\kappa}\sum_{i=1}^{m} y_i(x),$$

where $p_\alpha(x) = \sum_{j=1}^{n} w_j f_\alpha(x_j)$ and f_α is defined by (1). The potential function is strictly concave and its partial derivative with respect to any variable x_j is:

$$\frac{\partial \Phi(x)}{\partial x_j} = \frac{w_j}{{x_j}^\alpha} - \sum_{i=1}^{m} y_i(x)A_{ij} = \frac{w_j}{{x_j}^\alpha}\left(1 - \frac{{x_j}^\alpha \sum_{i=1}^{m} y_i(x)A_{ij}}{w_j}\right). \tag{7}$$

The following fact (given in a similar form in [7]), which follows directly from the Taylor series representation of concave functions, will be useful for the potential increase analysis:

▶ **Fact 4.8.** For a differentiable concave function $f : \mathbb{R}^n \to \mathbb{R}$ and any two points $x^0, x^1 \in \mathbb{R}^n$:

$$\sum_{j=1}^{n} \frac{\partial f(x^0)}{\partial x_j}(x_j^1 - x_j^0) \geq f(x^1) - f(x^0) \geq \sum_{j=1}^{n} \frac{\partial f(x^1)}{\partial x_j}(x_j^1 - x_j^0).$$

Using Fact 4.8 and (7), we show the following lemma:

▶ **Lemma 4.9.** *Starting with a feasible solution and throughout the course of the algorithm, the potential function $\Phi(x)$ never decreases. Letting x^0 and x^1 denote the values of x before and after a round update, respectively, the potential function increase is lower-bounded as:*

$$\Phi(x^1) - \Phi(x^0) \geq \sum_{j=1}^{n} w_j \frac{\left|x_j^1 - x_j^0\right|}{(x_j^1)^\alpha}\left|1 - \frac{(x_j^1)^\alpha \sum_{i=1}^{m} y_i(x^1)A_{ij}}{w_j}\right|.$$

4.1 Proof Sketch of Theorem 4.3

In this section, we outline the main ideas of the proof of Theorem 4.3, while the technical details are omitted and are instead provided in [35]. First, we show that in any round of the algorithm the variables that decrease by a multiplicative factor $(1 - \beta_2)$ dominate the potential increase due to *all the variables* that decrease (see Lemma 4.21 in [35]). This result is then used in Lemma 4.10 to show the following lower bound on the potential increase:

▶ **Lemma 4.10.** *Let x^0 and x^1 denote the values of x before and after any fixed round, respectively, and let $S^+ = \{j : x_j^1 > x_j^0\}$, $S^- = \{j : x_j^1 < x_j^0\}$. The potential increase in the round is lower bounded as:*

1. $\Phi(x^1) - \Phi(x^0) \geq \Omega(\beta\gamma)\sum_{j\in\{S^+\cup S^-\}} x_j^0 \sum_{i=1}^{m} y_i(x^0)A_{ij}$;
2. $\Phi(x^1) - \Phi(x^0) \geq \Omega\left(\frac{\beta}{(1-\beta)^\alpha}\right)\left(\sum_{j=1}^{n} x_j^0 \sum_{i=1}^{m} y_i(x^0) - (1+\gamma)\sum_{j=1}^{n} w_j (x_j^0)^{1-\alpha}\right)$;
3. $\Phi(x^1) - \Phi(x^0) \geq \Omega\left(\frac{\beta}{(1+\beta)^\alpha}\right)\left((1-\gamma)\sum_{j=1}^{n} w_j (x_j^0)^{1-\alpha} - \sum_{j=1}^{n} x_j^0 \sum_{i=1}^{m} y_i(x^0)\right)$.

Observe that for $\alpha > 1$ the objective function $p_\alpha(x)$, and, consequently, the potential function $\Phi(x)$, is negative for any feasible x. To yield a poly-logarithmic convergence time in R_w, m, n, and $A_{\max}$, the idea is to show that the negative potential $-\Phi(x)$ decreases by some multiplicative factor whenever x is not a "good" approximation to x^* – the optimal solution to (P_α). This idea, combined with the fact that the potential never decreases (and therefore $-\Phi(x)$ never increases) and with upper and lower bounds on the potential then leads to the desired convergence time. Consider the following definition of a stationary round:

▶ **Definition 4.11.** (Stationary round.) A round is stationary, if both:
1. $\sum_{j\in\{S^+\cup S^-\}} x_j^0 \sum_{i=1}^m y_i(x)A_{ij} < \gamma \sum_{j=1}^n w_j (x_j^0)^{1-\alpha}$, and
2. $(1-2\gamma)\sum_{j=1}^n w_j (x_j^0)^{1-\alpha} \le \sum_{j=1}^n x_j^0 \sum_{i=1}^m y_i(x^0)A_{ij}$,

where $S^+ = \{j : x_j^1 > x_j^0\}$, $S^- = \{j : x_j^1 < x_j^0\}$. Otherwise, the round is non-stationary.

Recall the expression for the negative potential: $-\Phi(x) = \frac{1}{\alpha-1}\sum_j w_j {x_j}^{1-\alpha} + \frac{1}{\kappa}\sum_i y_i(x)$. Then, using Lemma 4.10, it suffices to show that in a non-stationary round the decrease in the negative potential $-\Phi(x)$ is a multiplicative factor of the larger of the two terms $\frac{1}{\alpha-1}\sum_j w_j {x_j}^{1-\alpha}$ and $\frac{1}{\kappa}\sum_i y_i(x)$. The last part of the proof shows that the solution x that corresponds to any stationary round is close to the optimal solution. This part is done by appropriately upper-bounding the duality gap. Denoting by $S^+ \cup S^-$ the set of coordinates j for which x_j either increases or decreases in the observed stationary round and using Definition 4.11, we show that the terms $j \in \{S^+ \cup S^-\}$ contribute to the duality gap by no more than $O(\varepsilon\alpha)\cdot(-p_\alpha(x))$. The terms corresponding to $j \notin \{S^+ \cup S^-\}$ are bounded recalling (from α-FairPSolver) that for such terms $\frac{x_j^\alpha \sum_{i=1}^m y_i(x)A_{ij}}{w_j} \in (1-\gamma, 1+\gamma)$.

4.2 Structural Properties

Lower Bound on the Minimum Allocated Value. Recall (from Section 2) that the optimal solution x^* to (P_α) must lie in the positive orthant. We show in Lemma 4.12 that not only does x^* lie in the positive orthant, but the minimum element of x^* can be bounded below as a function of the problem parameters. This lemma motivates the choice of parameters δ_j in α-FairPSolver (Section 3). The proof is provided in [35].

▶ **Lemma 4.12.** *Let $x^* = (x_1^*, ..., x_n^*)$ be the optimal solution to (P_α). Then $\forall j \in \{1, ..., n\}$:*
- $x_j^* \ge \left(\frac{w_j}{w_{\max} M}\min_{i:A_{ij}\neq 0}\frac{1}{n_i A_{ij}}\right)^{1/\alpha}$, *if* $0 < \alpha \le 1$,
- $x_j^* \ge {A_{\max}}^{(1-\alpha)/\alpha}\left(\frac{w_j}{w_{\max} M}\right)^{1/\alpha}\min_{i:A_{ij}\neq 0}\frac{1}{n_i A_{ij}}$, *if* $\alpha > 1$,

where $n_i = \sum_{j=1}^n \mathbb{1}_{\{A_{ij}\neq 0\}}$[8] is the number of non-zero elements in the i^{th} row of the constraint matrix A, and $M = \min\{m, n\}$.

Asymptotics of $\alpha-$Fair Allocations The following lemma states that for sufficiently small (but not too small) α, the values of the linear and the $\alpha-$fair objectives at their respective optimal solutions are approximately the same. This statement will then lead to a conclusion that to $\varepsilon-$approximately solve an $\alpha-$fair packing problem for a very small α, one can always use an $\varepsilon-$approximation packing LP algorithm.

▶ **Lemma 4.13.** *Let (P_α) be an $\alpha-$fair packing problem with optimal solution x^*, and (P_0) be the LP with the same constraints and the same weights w as (P_α) and an optimal solution z^*. Then if $\alpha \le \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, we have that $\sum_j w_j z_j^* \ge (1-3\varepsilon)\sum_j \frac{(x_j^*)^{1-\alpha}}{1-\alpha}$, where $\varepsilon \in (0, 1/6]$.*

Observing that for any $\alpha \in (0,1)$, $\frac{(z_j^*)^{1-\alpha}}{1-\alpha} \ge z_j^*$ (since, due to the scaling, $z_j^* \in [0,1]$), a simple corollary of Lemma 4.13 is that an $\varepsilon-$approximation z to (P_0) $(\sum_j w_j z_j \ge (1-\varepsilon)\sum_j w_j z_j^*)$ is also an $O(\varepsilon)-$approximation to (P_α), for $\alpha \le \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$. Thus, to find an $\varepsilon-$approximate solution for $\alpha \le \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, the packing LP algorithm of [7] can be run, which means that

[8] With the abuse of notation, $\mathbb{1}_{\{e\}}$ is the indicator function of the expression e, i.e., 1 if e holds, and 0 otherwise.

there is a stateless distributed algorithm that converges in $\text{poly}(\ln(\varepsilon^{-1}R_w mnA_{\max})/\varepsilon)$ time for α arbitrarily close to zero.

The following two lemmas show that when α is sufficiently close to 1, (P_α) can be $\varepsilon-$approximated by $\varepsilon-$approximately solving (P_1) with the same constraints and weights.

▶ **Lemma 4.14.** *Let x be an $\varepsilon-$approximate solution to a 1-fair packing problem (P_1) returned by α-FairPSolver. Then, for any $\alpha \in [1 - 1/\tau_0, 1)$, where $\tau_0 = \frac{1}{\beta}\ln(\frac{1}{\delta_{\min}})$, x is also a $2\varepsilon-$approximate solution to (P_α), where the only difference between (P_1) and (P_α) is in the value of α in the objective.*

▶ **Lemma 4.15.** *Let x be an $\varepsilon-$approximate solution to a 1-fair packing problem (P_1) returned by α-FairPSolver. Then, for any $\alpha \in (1, 1 + 1/\tau_0]$, where $\tau_0 = \frac{1}{\beta}\ln(\frac{1}{\delta_{\min}})$, x is also a $2\varepsilon-$approximate solution to (P_α), where the only difference between (P_1) and (P_α) is in the value of α in the objective.*

Finally, we consider the asymptotics of $\alpha-$fair allocations, as α becomes large. This result complements the result from [38] that states that $\alpha-$fair allocations approach the max-min fair one as $\alpha \to \infty$ by showing how fast the max-min fair allocation is reached as a function of α, R_w, n, and $A_{\max}$. First, for completeness, we provide the definition of max-min fairness.

▶ **Definition 4.16.** (Max-min fairness [10].) Let $\mathcal{R} \subset \mathbb{R}^n_+$ be a compact and convex set. A vector $x \in \mathcal{R}$ is max-min fair on $\mathcal{R}$ if for any vector $z \in \mathcal{R}$ it holds that: if for some $j \in \{1, ..., n\}$ $z_j > x_j$, then there exists $k \in \{1, ..., n\}$ such that $z_k < x_k$ and $x_k \leq x_j$.

On a compact and convex set $\mathcal{R} \subset \mathbb{R}^n$, the max-min fair vector is unique [43, 44]. The following lemma shows that for $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, the $\alpha-$fair vector and the max-min fair vector are $\varepsilon-$close to each other. Notice that because of a very large gradient of $p_\alpha(x)$ as α becomes large, the max-min fair solution gives only an $O(\varepsilon\alpha)-$approximation to (P_α).

▶ **Lemma 4.17.** *Let x^* be the optimal solution to $(P_\alpha) = \max\{p_\alpha(x) : Ax \leq 1, x \geq 0\}$, z^* be the max-min fair solution for the convex and compact set determined by the constraints from (P_α). Then if $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, we have that:*
1. *$p_\alpha(x^*) \leq (1 - \varepsilon(\alpha - 1))p_\alpha(z^*)$, i.e., z^* is an $\varepsilon(\alpha - 1)-$approximate solution to (P_α), and*
2. *$(1 - \varepsilon)z^*_j \leq x^*_j \leq (1 + \varepsilon)z^*_j$, for all $j \in \{1, ..., n\}$.*

5 Conclusion

We presented an efficient stateless distributed algorithm for the class of α-fair packing problems. To the best of our knowledge, this is the first algorithm with poly-logarithmic convergence time in the input size. Additionally, we obtained results that characterize the fairness and asymptotic behavior of allocations in weighted $\alpha-$fair packing problems that may be of independent interest. An interesting open problem is to determine the class of objective functions for which the presented techniques yield fast and stateless distributed algorithms, together with a unified convergence analysis. This problem is especially important in light of the fact that α-fair objectives are not Lipschitz continuous, do not have a Lipschitz gradient, and their dual gradient's Lipschitz constant scales at least linearly with n and $A_{\max}$. Therefore, the properties typically used in fast first-order methods are lacking [2, 40]. Finally, for applications of α-fair packing that do not require stateless updates, it seems plausible that the dependence on ε^{-1} in the convergence bound can be improved from ε^{-5} to ε^{-3} by relaxing the requirement for asynchronous updates, similarly as was done in [3] over [7].

References

1 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proc. ACM STOC'15*, 2015.

2 Zeyuan Allen-Zhu and Lorenzo Orecchia. A novel, simple interpretation of Nesterov's accelerated method as a combination of gradient and mirror descent, Jan. 2015. arXiv preprint, `http://arxiv.org/abs/1407.1537`.

3 Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proc. ACM-SIAM SODA'15*, 2015.

4 Anthony B Atkinson. On the measurement of inequality. *J. Econ. Theory*, 2(3):244–263, 1970.

5 Baruch Awerbuch, Yossi Azar, and Rohit Khandekar. Fast load balancing via bounded best response. In *Proc. ACM-SIAM SODA'08*, 2008.

6 Baruch Awerbuch and Rohit Khandekar. Greedy distributed optimization of multi-commodity flows. In *Proc. ACM PODC'07*, 2007.

7 Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. *SIAM J. Comput.*, 38(6):2468–2486, 2009.

8 Yair Bartal, John Byers, and Danny Raz. Global optimization using local information with applications to flow control. In *Proc. IEEE FOCS'97*, 1997.

9 Amir Beck, Angelia Nedić, Asuman Ozdaglar, and Marc Teboulle. An $O(1/k)$ gradient method for network resource allocation problems. *IEEE Trans. Control Netw. Syst.*, 1(1):64–73, 2014.

10 Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 1992.

11 Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. On the efficiency-fairness trade-off. *Manag. Sci*, 58(12):2234–2250, 2012.

12 Thomas Bonald and James Roberts. Multi-resource fairness: Objectives, algorithms and performance. In *Proc. ACM SIGMETRICS'15*, 2015.

13 Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

14 Anna Charny, David D Clark, and Raj Jain. Congestion control with explicit rate indication. In *Proc. IEEE ICC'95*, 1995.

15 Yun Kuen Cheung, Richard Cole, and Nikhil Devanur. Tatonnement beyond gross substitutes?: Gradient descent to the rescue. In *Proc. ACM STOC'13*, 2013.

16 Shlomi Dolev. *Self-stabilization*. MIT press, 2000.

17 Lisa Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

18 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

19 Naveen Garg and Neal Young. On-line end-to-end congestion control. In *Proc. IEEE FOCS'02*, 2002.

20 Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. USENIX NSDI'11*, 2011.

21 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *Proc. ACM STOC'14*, 2014.

22 Jeffrey Jaffe. Bottleneck flow control. *IEEE Trans. Commun.*, 29(7):954–962, 1981.

23 Kamal Jain and Vijay Vazirani. Eisenberg-gale markets: Algorithms and structural properties. In *Proc. ACM STOC'07*, 2007.

24 Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Trans. Netw.*, 21(6):1785–1798, 2013.

25 Frank Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *J. Oper. Res. Soc.*, 49(3):237–252, 1998.

26 Frank Kelly and Elena Yudovina. *Stochastic networks*, volume 2. Cambridge University Press, 2014.

27 Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. In *Proc. IEEE FOCS'99*, 1999.

28 Christos Koufogiannakis and Neal Young. Beating simplex for fractional packing and covering linear programs. In *Proc. IEEE FOCS'07*, 2007.

29 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. ACM-SIAM SODA'06*, 2006.

30 Amit Kumar and Jon Kleinberg. Fairness measures for resource allocation. In *Proc. IEEE FOCS'00*, 2000.

31 Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. In *Proc. IEEE INFOCOM'10*, 2010.

32 Steven Low, Fernando Paganini, and John Doyle. Internet congestion control. *IEEE Control Systems*, 22(1):28–43, 2002.

33 Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. ACM STOC'93*, 1993.

34 Jelena Marašević, Clifford Stein, and Gil Zussman. Max-min fair rate allocation and routing in energy harvesting networks: Algorithmic analysis. In *Proc. ACM MobiHoc'14*, 2014.

35 Jelena Marašević, Cliff Stein, and Gil Zussman. A fast distributed stateless algorithm for α-fair packing problems, Feb. 2016. arXiv preprint, `http://arxiv.org/abs/1502.03372v3`.

36 Bill McCormick, Frank Kelly, Patrice Plante, Paul Gunning, and Peter Ashwood-Smith. Real time alpha-fairness based traffic engineering. In *Proc. ACM HotSDN'14*, 2014.

37 Nimrod Megiddo. Optimal flows in networks with multiple sources and sinks. *Math. Program.*, 7(1):97–107, 1974.

38 Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.*, 8(5):556–567, October 2000.

39 Damon Mosk-Aoyama, Tim Roughgarden, and Devavrat Shah. Fully distributed algorithms for convex optimization problems. In *Proc. DISC'07*, 2007.

40 Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

41 Christos Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proc. ACM STOC'93*, 1993.

42 Serge Plotkin, David Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

43 Božidar Radunović and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Trans. Netw.*, 15(5):1073–1083, 2007.

44 Saswati Sarkar and Leandros Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *Proc. IEEE INFOCOM'00*, 2000.

45 Yung Yi and Mung Chiang. Stochastic network utility maximisation — a tribute to Kelly's paper published in this journal a decade ago. *Eur. Trans. Telecommun.*, 19(4):421–442, 2008.

46 Neal Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. IEEE FOCS'01*, 2001.