Fast Algorithms for Packing Proportional Fairness and its Dual

David Martínez-Rubio
joint work with Francisco Criado and Sebastian Pokutta

Zuse Institute Berlin

Math+ Spotlight · July 13, 2022

Berlin Mathematics Research Center

α-Fairness and Proportional Fairness

α-fairness: a family of fair objectives

Maximize the $(1-\alpha)$-mean of coordinates of a point in a convex set.

- $\alpha = 0 \Rightarrow$ arithmetic mean, maximize utility, no fairness.
- $\alpha = 1 \Rightarrow$ geometric mean, proportional fairness.
- $\alpha \to \infty \Rightarrow$ max-min fairness.

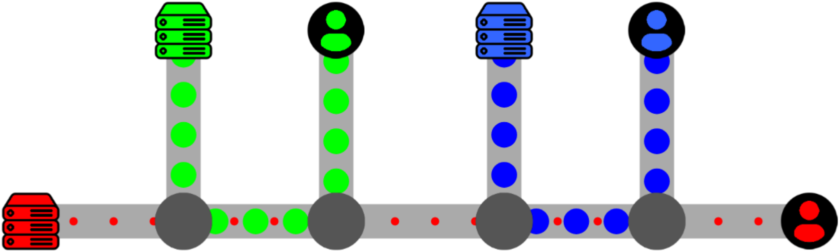
In this work: Proportional fairness.

- Studied in economics in Nash bargaining solutions, in game theory, **multi-resource allocation in compute clusters, rate control in networks**.
- The applications in bold have packing constraints: $Ax \leq b$, where $A_{ij} \geq 0, b_i > 0, x \in \mathbb{R}^n_{\geq 0}$, which is what we focus on.

ZIB ZUSE INSTITUTE BERLIN

Application Example: Fair Multicommodity Flow

- Pairs server-user in a shared network with limited link capacities.
- How much flow should each pair receive?



Packing Proportional Fairness and its Dual

After a simple reformulation wlog our problem is, for $A \in \mathcal{M}_n(\mathbb{R}_{\geq 0})$:

$$\max_{x \in \mathbb{R}^n_{\geq 0}} \left\{ f(x) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \log x_i : Ax \leq \mathbb{1}_m \right\}.$$

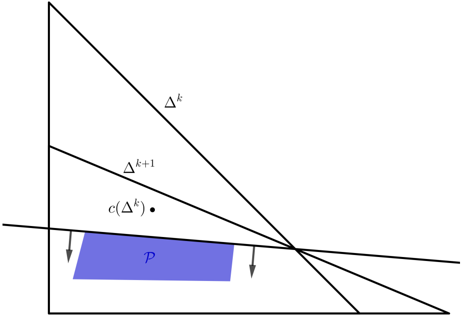
And its Lagrange dual is:

$$\min_{\lambda \in \Delta^m} \left\{ g(\lambda) \stackrel{\text{def}}{=} -\sum_{i=1}^{n} \log(A^T\lambda)_i - n \log n \right\},$$

- Approximate primal solution $\stackrel{\text{fast}}{\not\to}$ approximate dual solution.
- We design two very different algorithms for each problem.
- We find an application of the dual solution to the *simplexoid* algorithm of (Yamnitsky and Levin, 1982) for linear programming.

Results and Comparison

Paper	Problem	Iterations	Width-dependence?
(Beck et al., 2014)	Primal	$O(\rho^2 mn/\varepsilon)$	Yes
(Maraševič et al., 2016)	Primal	$\widetilde{O}(n^5/\varepsilon^5)$	nearly No (polylog)
(Diakonikolas et al., 2020)	Primal	$\widetilde{O}(n^2/\varepsilon^2)$	nearly No (polylog)
CMP (Theorem 5)	Primal	$\widetilde{O}(n/\varepsilon)$	No
(Beck et al., 2014)	Dual	$O(\rho\sqrt{mn/\varepsilon})$	Yes
CMP (Theorem 9)	Dual	$\widetilde{O}(n^2/\varepsilon)$	No



Online learning

Online convex optimization: A sequential game

You play $x_t \in \mathcal{C}$, an adversary picks a convex loss ℓ_t and you pay $\ell_t(x_t)$. How good can the regret be?

$$\textbf{Regret} \stackrel{\text{def}}{=} \sum_{t=1}^{T} \ell_t(x_t) - \min_{u \in \mathcal{C}} \sum_{t=1}^{T} \ell_t(u).$$

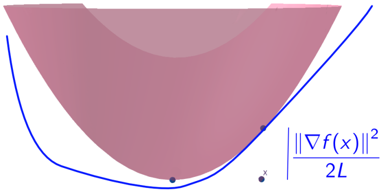
If you use $\ell_t(\cdot) = \langle \nabla f(x_t), \cdot \rangle$, you can reduce convex optimization to online convex optimization:

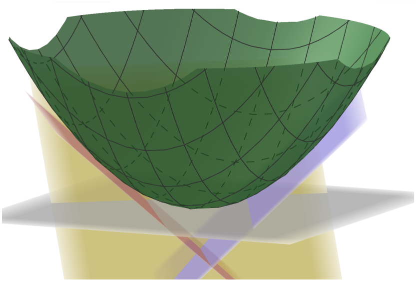
$$f(\frac{1}{T}\sum_{i=1}^{T} x_i) - f(x^*) \leq \frac{1}{T}\sum_{i=1}^{T} \langle \nabla f(x_i), x_i - x^* \rangle \leq \frac{1}{T}\,\textbf{Regret}$$

But you can use other losses: we will use a truncated gradient $\overline{\nabla f_r}(x) \stackrel{\text{def}}{=} (\min\{\nabla_1 f_r(x), 1\}, \ldots, \min\{\nabla_n f_r(x), 1\})$ for the primal problem.

Acceleration

Usually fast algorithms are obtained by combining a gradient descent algorithm with an online learning algorithm: Progress of the former compensates instantaneous regret of the latter.





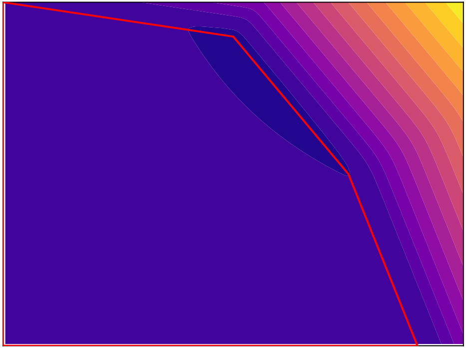
We use non-standard versions of an algorithm that makes primal progress and of an online learning algorithm.

Primal problem

Reparametrize $x \to \exp(y)$ and remove constraints by adding a fast growing barrier (Diakonikolas et al, 2020):

$$f_r(y) \stackrel{\text{def}}{=} -\sum_{i=1}^{n} y_i + \frac{\beta}{1+\beta}\sum_{i=1}^{m}(A\exp(y))_i^{\frac{1+\beta}{\beta}}, \text{ where } \beta \approx \frac{\varepsilon}{n\log(mn^2/\varepsilon)}.$$

Proposition: If y^ε is an ε-minimizer of f_r, then $\frac{1}{1+\varepsilon/n}y^\varepsilon$ is feasible and is an $O(\varepsilon)$-maximizer of f.



Primal problem

1. Smoothness and Lipschitz constants are bad but the objective has structure:
   - $\nabla_j f_r(x) \in [-1, \infty)$ for $j \in [n]$.
   - A small gradient step decreases the function value significantly:

     $$\langle \nabla f_r(x), \Delta \rangle \geq f_r(x) - f_r(x - \Delta) \geq \frac{1}{2} \langle \nabla f_r(x), \Delta \rangle \geq 0,$$

     for $\Delta \in \mathbb{R}^n$ satisfying the following:

     $$\Delta_j \stackrel{\text{def}}{=} \frac{c_j \beta}{4(1+\beta)} \min\{\nabla_j f_r(x), 1\}, \quad \forall c_j \in [0, 1], \forall j \in [n].$$

Primal problem

1. Smoothness and Lipschitz constants are bad but the objective has structure:
   - $\nabla_j f_r(x) \in [-1, \infty)$ for $j \in [n]$.
   - A small gradient step decreases the function value significantly:

     $$\langle \nabla f_r(x), \Delta \rangle \geq f_r(x) - f_r(x - \Delta) \geq \frac{1}{2} \langle \nabla f_r(x), \Delta \rangle \geq 0,$$

     for $\Delta \in \mathbb{R}^n$ satisfying the following:

     $$\Delta_j \stackrel{\text{def}}{=} \frac{c_j \beta}{4(1+\beta)} \min\{\nabla_j f_r(x), 1\}, \quad \forall c_j \in [0,1], \forall j \in [n].$$

2. We run Mirror Descent on truncated losses.

$$f_r\left(\frac{1}{T}\sum_{i=1}^{T} x_i\right) - f_r(x^*) \leq \frac{1}{T}\sum_{i=1}^{T} \langle \nabla f_r(x_i) - \overline{\nabla f_r}(x_i), x_i - x^* \rangle + \underbrace{\langle \overline{\nabla f_r}(x_i), x_i - x^* \rangle}_{\text{Regret}_i}$$

3. The gradient step compensates the MD regret and the regret we ignored due to truncation.

Accelerated, Distr., Deterministic and Width-Indep.

- We carefully choose several parameters (depending on known quantities): learning rates η_k, coupling parameter τ, number of iterations T, etc. Then, the algorithm has a simple form below.
- Our algorithm is distributed, deterministic and we prove deterministic guarantees.

Algorithm 1 Accelerated descent method for 1-Fair Packing

Input: Normalized matrix $A \in \mathcal{M}_{m\times n}(\mathbb{R}_{\geq 0})$ and accuracy ε.

```
x^(0) ← y^(0) ← z^(0) ← −ω𝟙_n
for k = 1 to T do
    x^(k) ← τ z^(k−1) + (1 − τ) y^(k−1)
    z^(k) ← argmin_{z∈B} { (1/(2ω)) ‖z − z^(k−1)‖_2^2 + ⟨η_k ∇̄f(x^(k)), z⟩ }
    y^(k) ← x^(k) + (1/(η_k L)) (z^(k) − z^(k−1))        ◇ Gradient descent step
end for
return x̂ ≝ exp(y^(T)) / (1 + ε/n)
```

$$x^{(0)} \leftarrow y^{(0)} \leftarrow z^{(0)} \leftarrow -\omega \mathbb{1}_n$$

for $k = 1$ **to** T **do**

$$x^{(k)} \leftarrow \tau z^{(k-1)} + (1-\tau) y^{(k-1)}$$

$$z^{(k)} \leftarrow \mathrm{argmin}_{z\in B} \left\{ \frac{1}{2\omega} \left\| z - z^{(k-1)} \right\|_2^2 + \langle \eta_k \overline{\nabla f}(x^{(k)}), z \rangle \right\}$$

$$y^{(k)} \leftarrow x^{(k)} + \frac{1}{\eta_k L}(z^{(k)} - z^{(k-1)})$$ ◇ Gradient descent step

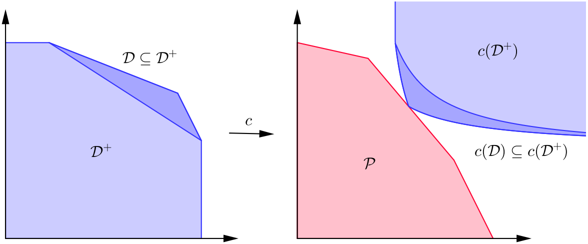
end for

return $\widehat{x} \stackrel{\mathrm{def}}{=} \exp(y^{(T)})/(1+\varepsilon/n)$

Dual Problem: The Centroid Map and a Reduction

$$\mathcal{P} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n_{\geq 0} : Ax \leq \mathbb{1}_m\}, \qquad c(h) = \left(\frac{1}{nh_1}, \ldots, \frac{1}{nh_n}\right),$$

$$\mathcal{D} = \operatorname{conv}\{A_i : i \in [m]\} \qquad \mathcal{D}^+ = (\operatorname{conv}\{A_i : i \in [m]\} + (-\infty, 0]^n) \cap \mathbb{R}^n_{\geq 0}$$



$$\min_{p \in c(\mathcal{D}^+)} \left\{ \hat{g}(p) \stackrel{\text{def}}{=} \max_{i \in [m]} \langle A_i, p \rangle \right\}.$$

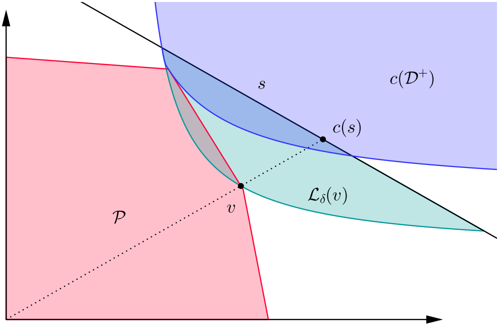
Proposition: If $p = c(A^T\lambda)$ and p is an (ε/n)-minimizer of $\hat{g}$, then λ is an ε-minimizer of the dual problem g.

Dual Problem: The PST Framework

Optimizing $\hat{g}$ is an (approximate) linear feasibility problem: Find $x \in c(D^+)$ such that $Ax \leq (1+\varepsilon)\mathbb{1}_m$.

PST Framework

- Generate a covering constraint as $h = A^T\lambda$, for weights $\lambda \in \Delta^m$.
- Use an oracle to satisfy h: Find $x \in c(D^+)$ s.t. $\langle h, x \rangle \leq 1$
- Increase the weight λ_i the more, the greater $\langle A_i, x \rangle - 1 \in [-\tau, \sigma]$ is, i.e., the more x does not satisfy A_i (MWs algorithm).
- Guarantees convergence in $O(\sigma\tau/\varepsilon^2)$.



Improving over PST: Adaptive Oracle

The closer we are to a solution the smaller the lens L_δ is.
$\Rightarrow$ the smaller τ and σ are.

Improved strategy

- Implement an oracle that yields smaller τ_δ and σ_δ if δ is lower.
- Start with a δ-minimizer of $\hat{g}$.
- Find a $\delta/2$-minimizer using the adaptive oracle and PST: It takes $O(\tau_\delta \sigma_\delta / (\delta/2)^2)$.
- Repeat until $\delta < \varepsilon/n$. Total complexity is $O(n^2/\varepsilon)$.