Vol.2019-AL-174 No.3
2019/9/17

# Improved Algorithms for Online Load Balancing

Yaxiong Liu[1,3,a)] Kohei Hatano[2,3,b)] Eiji Takimoto[1,c)]

概要：We construct algorithms for online load balancing and its extension in the framework of online learning. On each round a player predicts a distribution over $K$-machines. Then the player receives the true load of each machine. The cost incurred by the player is the $p$-norm (if $p = \infty$, the makespan) of the cumulative load vector. Our algorithms achieve the best known bound for $p = \infty$ and an improved bound for $p > 2$. In particular, our algorithm for the online load balancing involves linear programming and second order cone programming, which are solved in polynomial time.

キーワード：online learning, regret bounds, approachability, online convex optimization, online load balancing, makespan, second order cone programming

## 1. Introduction

We consider the online learning problem for global cost functions posed by Even-Dar et al. [5]. A motivating example of the problem is an online load balancing problem defined as follows. There are $K$-parallel machines and the protocol is defined as a game between the player and the environment. On each round $t = 1, \ldots, T$, (i) the player selects a distribution $\boldsymbol{\alpha}_t$ over $K$-machines, which can be viewed as an allocation of a task, (ii) then the environment assigns loss $l_{t,i}$ for each machine $i$ and the load of machine $i$ is given as $\alpha_{t,i} l_{t,i}$. The goal of the player to minimize the makespan of cumulative loads of all machines after $T$ rounds, i.e., $\max_{i=1,\ldots,K} \sum_{t=1}^{T} \alpha_{t,i} l_{t,i}$, compared relatively to the makespan obtained by the optimal static allocation $\boldsymbol{\alpha}^*$ in hindsight. More precisely, the goal is to minimize the regret, the difference between the player's makespan and the static optimal makespan. The makespan cost can be viewed as a $\infty$-norm of the vector of cumulative loss of each machine and can be generalized to the $p$-norm ($p > 2$). We refer the online problem for global cost functions as the problem where the cost is given as the $p$-norm of cumulative losses (we will give a formal definition of the problem in the next section).

The problem of online learning for global functions is different from the standard problem of online learning or online convex optimization (e.g., [4], [6], [11]) in that the cost of online learning/online convex optimization is given as a sum of convex functions among rounds, while the cost in this problem is not. So, apparently, techniques of online learning cannot be applied to this problem directly. The problem of online learning for global functions also differs from the standard online load balancing problem in the online algorithm literature (e.g., [2]). A main difference is that the online algorithm literature considers the competitive ratio, the total cost of the player divided by that of the best sequence of allocations. Therefore, the competitor in the competitive analysis is stronger, while the regret analysis of online learning considers the regret w.r.t. the static optimal solution as a performance measure.

Even-Dar et al.[5] gives an algorithm based on the regret minimum framework by involving an extra concept, the Blackwell approachability [3], to a target set, which is defined in the following section. This algorithm can give an upper bound to the regret of the online load balancing problem in $O(\sqrt{KT})$. In the same paper, the authors give another algorithm, DIFF, for the makespan problem with an upper bound of the regret as $O((\ln K)\sqrt{T})$. The reason why the first algorithm achieves a relative bad regret bound is that we can not give a good enough convergence rate to the target set from the $L_2$-norm to $L_\infty$-norm. With other words, we can not give a small enough upper bound for the $L_\infty$-norm distance between the average payoff vector to the target set, if we can only apply $L_2$-norm distance to measure the convergence rate to our target set. Since the

1 Department of Informatics, Kyushu University
2 Faculty of Arts and Science, Kyushu University
3 RIKEN AIP
a) yaxiong.liu@inf.kyushu-u.ac.jp
b) hatano@inf.kyushu-u.ac.jp
c) eiji@inf.kyushu-u.ac.jp

running procedure of DIFF is not easy, naturally we prefer to improve the algorithm following Blackwell approachability.

Rahklin et al. [10] give a theoretical result for the online load balancing problem, that the upper bound to regret can achieve $O(\sqrt{(\ln K)T})$, rather than $O((\ln K)\sqrt{T})$. However there is no efficient algorithm given in this paper to obtain the regret. The above papers inspire us that we can product an efficient algorithm to achieve $O(\sqrt{(\ln K)T})$ regret, if we can find a good enough convergence rate to our target set, i.e., if we can give a tighter bound for the $L_\infty$-norm distance from the average payoff vector to target set.

In following years, there are some new explorations about the equivalence between the Blackwell approachability and online linear optimization [1], in addition and online convex optimization by involving a support function [12]. These results show that choosing an appropriate online convex optimization algorithm we may give a better bound to the convergence rate not only for $L_2$-norm but also for $L_\infty$-norm. More specifically speaking, the convergence rate of the target set can be bounded by the regret of a chosen online convex optimization algorithm, i.e., the distance from the average payoff vector to the target set is bounded by the regret, of course this distance can be in $L_\infty$-norm.

In conclusion we can construct an algorithm for online load balancing problem with a potential better upper bound to regret, based on a faster convergence speed to the target set. This faster convergence speed is guaranteed by an online convex optimization algorithm. Our main contributions in this paper are in following:

- 1. We give a reduction from online learning with global cost function with respect to $p$-norm to online linear optimization.
- 2. We construct an algorithm for online load balancing problem with the foundation of an online convex optimization algorithm, EG plus minus [7], [8]. This algorithm can be considered as a combination of an online convex optimization algorithm and linear programming. And we show that this new algorithm achieves an upper bound to regret as $O(\sqrt{(\ln K)T})$ for running $T$ rounds.
- 3. We give some details for implementation of this algorithm. In our algorithm we do not apply Follow the Leader or Gradient descent these ordinary online learning algorithms directly, but on each round Linear Programming and Second Order Cone Programming(SOCP) [9].

This paper is composed in following form. In section 2 we introduce the basic definitions in this paper like online load balancing problem, Blackwell approachability and target set. Next in section 3 we give the algorithm for online global cost function with respect to $p$-norm and online load balancing problem. Then in section 4 we give some details in implementation of this algorithm.

## 2. Preliminaries

First we give some notations in this paper. We utilize $\|\cdot\|$ to denote the norm of a vector. Especially, for $\boldsymbol{x} \in \mathbb{R}^d$, $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^d (x_i)^d\right)^{1/d}$. $\|\boldsymbol{x}\|_\infty = \max_i x_i$. Moreover we denote $\|\boldsymbol{x}\|_*$ as dual norm of $\|\boldsymbol{x}\|$, and $\|\boldsymbol{x}\|_* = \sup\{\boldsymbol{x}|\langle \boldsymbol{x}, \boldsymbol{z}\rangle, \|\boldsymbol{z}\| \le 1\}$.

The formal definition of online global cost function problem is as follows: At round $t = 1, \cdots, T$, firstly a learner chooses a distribution $\boldsymbol{\alpha}_t \in \Delta(K)$, then this learner receives the vector of load of $K$ alternatives or machines as $\boldsymbol{l}_t \in [0,1]^K$. We define that $L_T(i) = \sum_{t=1}^T \boldsymbol{l}_t(i)$, $(\boldsymbol{\alpha} \odot \boldsymbol{l}) = (\alpha(1)l(1), \cdots, \alpha(K)l(K))$. A function $C(\cdot) : \mathbb{R}^K \to \mathbb{R}$ is defined as global cost function. We denote that $C_p(a) = \|a\|_p$. For a certain algorithm $\mathcal{A}$ we define the cumulative load as follows:

$$C_p(L_T^{\mathcal{A}}) = C_p\left(\sum_{t=1}^T \boldsymbol{\alpha}_t \odot \boldsymbol{l}_t\right), \tag{1}$$

and the competitor is defined as follows:

$$C_p^*(L_T) = \min_{\boldsymbol{\alpha}\in\Delta(K)} C_p\left(\sum_{t=1}^T \boldsymbol{\alpha} \odot \boldsymbol{l}_t\right). \tag{2}$$

Hence the regret of the online global cost function with $p$-norm is in following:

$$\mathrm{Regret}_{T,p} = C_p(L_T^{\mathcal{A}}) - C_p^*(L_T). \tag{3}$$

Note that in this paper we consider the case that $p > 2$. Of course in online load balancing problem, we set that $p = \infty$, and thus we have the regret for online load balancing problem as

$$\mathrm{Regret}_{T,\infty} = C_\infty(L_T^{\mathcal{A}}) - C_\infty^*(L_T). \tag{4}$$

Now we have a review to Blackwell approachability theorem. For two convex and compact set $A \subset \mathbb{R}^a$ and $B \subset \mathbb{R}^b$, we suppose a vector valued function $r : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$, as payoff function, and an arbitrary convex and closed set $S \in \mathbb{R}^c$ as target set.

Firstly we introduce a Blackwell approaching game, which is a game between a player and environment. At each round $t = 1, \cdots, T$,

- 1. this player picks up a vector $\boldsymbol{a}_t \in A$.
- 2. environment chooses a vector $\boldsymbol{b}_t \in B$.

The target of the player is to let the average payoff vector into the target set after repeating this game $T$ rounds with the environment. To ensure that this player can achieve his goal, we may involve the following definitions.

**Definition1** [3] We say a set $S$ is approachable with convergence rate $\gamma(t)$ if there exists an algorithm for choosing a sequence of vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_t \in A$ such that for every sequence $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_t \in B$ and all integers $t \geq 1$ :

- 1. the value of $\boldsymbol{a}_t$ depends only on the value of $r(\boldsymbol{a}_s, \boldsymbol{b}_s)$ for $s < t$.
- 2. the average payoff $\bar{r}_t = \frac{1}{t}\sum_{s=1}^{t} r(\boldsymbol{a}_s, \boldsymbol{b}_s)$ satisfies $\mathrm{dist}(\bar{r}_t, S) \leq \gamma(t)$.

Note that in this definition the distance of average payoff to target set $\mathrm{dist}(\bar{r}_t, S)$ is defined according to $L_p$-norm, as $\mathrm{dist}(\bar{r}_t, S) = \inf_{\boldsymbol{s}\in S} \|\bar{r}_T - \boldsymbol{s}\|_p$. In online load balancing problem $p = \infty$, specially.

**Definition2** We say that $S$ satisfies the Blackwell criterion if $S$ is convex and closed and for every $\boldsymbol{b} \in B$ there exists $\boldsymbol{a} \in A$ such that $r(\boldsymbol{a}, \boldsymbol{b}) \in S$.

In $p$-norm global cost function problem, where $p > 2$, we let $A$ as $\Delta(K)$ and $B$ as $[0, 1]^K$. And payoff function $r_t$ is defined as $\boldsymbol{r}_t(\boldsymbol{\alpha}_t, \boldsymbol{l}_t) = (\boldsymbol{\alpha}_t \odot \boldsymbol{l}_t, \boldsymbol{l}_t)$, where $(\boldsymbol{\alpha}_t \odot \boldsymbol{l}_t, \boldsymbol{l}_t)$ is a $2K$-dimensional vector, moreover we define that $\tilde{\boldsymbol{r}}_T = \sum_{t=1}^{T} \boldsymbol{r}_t = (L_T^{\mathcal{A}}, L_T)$. Finally, we define the target set in online learning as

$$S = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^k \times \mathbb{R}^k : x_i, y_i \in [0, 1]; C_p(\boldsymbol{x}) \leq C_p^{\star}(\boldsymbol{y})\}. \quad (5)$$

**Lemma1** ([5]) For any $p \geq 2$, $C_p(\cdot)$ is convex function and $C_p^*(\cdot)$ is concave function. And $S$ defined above is a convex set.

**Definition3** The support function $h_S : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ of $S$ is defined as

$$h_S(\boldsymbol{w}) = \sup_{\boldsymbol{s}\in S} \langle \boldsymbol{s}, \boldsymbol{w} \rangle, \boldsymbol{w} \in \mathbb{R}^d. \quad (6)$$

**Proposition1** $\boldsymbol{s}^* = \arg\min_{\boldsymbol{s}\in S} \langle \boldsymbol{s}, \boldsymbol{w} \rangle$ is a sub-gradient of $h_S(\boldsymbol{w})$ at $\boldsymbol{w}$.

*Proof* For any $\boldsymbol{w}, \boldsymbol{u} \in S$, we define that $\boldsymbol{s}^* = \arg\max_{\boldsymbol{s}\in S} \langle \boldsymbol{s}, \boldsymbol{w} \rangle$, and $\boldsymbol{s}^{\boldsymbol{u}} = \arg\max_{\boldsymbol{s}\in S} \langle \boldsymbol{s}, \boldsymbol{u} \rangle$. Therefore we have

$$\begin{aligned} h_S(\boldsymbol{w}) - h_S(\boldsymbol{u}) &= \sup_{s\in S} \langle \boldsymbol{s}, \boldsymbol{w} \rangle - \sup_{s\in S} \langle \boldsymbol{s}, \boldsymbol{u} \rangle = \langle \boldsymbol{s}^*, \boldsymbol{w} \rangle - \langle \boldsymbol{s}^{\boldsymbol{u}}, \boldsymbol{u} \rangle \\ &\leq \langle \boldsymbol{s}^*, \boldsymbol{w} - \boldsymbol{u} \rangle, \end{aligned}$$

where the inequality is from the definition of $h_S(\cdot)$. So we get our proposition. □

**Lemma2** ([12]) Let $S$ be a closed convex set with support function $h_S$ and let $d(\boldsymbol{z}, S) = \min_{\boldsymbol{s}\in S} \|\boldsymbol{z} - \boldsymbol{s}\|$ denote the point to set distance with respect to a norm. Then, for any $\boldsymbol{z} \in \mathbb{R}^d$,

$$d(\boldsymbol{z}, S) = \max_{\|\boldsymbol{w}\|_* \leq 1} \{\langle \boldsymbol{w}, \boldsymbol{z} \rangle - h_S(\boldsymbol{w})\}. \quad (7)$$

Now we need consider our concrete case that

$$S = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^k \times \mathbb{R}^k : x_i, y_i \in [0, 1]; C_p(\boldsymbol{x}) \leq C_p^{\star}(\boldsymbol{y})\},$$

where $C_p(\boldsymbol{x}) = \|\boldsymbol{x}\|_p$, $C_p^*(\boldsymbol{y}) = \min_{\boldsymbol{\alpha}\in\Delta(K)} \|\boldsymbol{\alpha} \odot \boldsymbol{y}\|_p$ and for any $w \in B_q(2K)$, where $1/p + 1/q = 1$, we have $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2)$, where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are two $K$-dimensional vectors. $B_q$ is the unit ball in $\mathbb{R}^{2K}$ space with respect to $\|\cdot\|_q$. $S$ is a convex set by Lemma 1.

**Lemma3** $S$ is given as above, for any $\boldsymbol{w} \in B_q(2K)$ we have following equation:

$$\min_{\boldsymbol{\alpha}\in\Delta(K)} \max_{\boldsymbol{l}\in[0,1]^K} \langle \boldsymbol{w}, r(\boldsymbol{\alpha}, \boldsymbol{l}) \rangle \leq h_S(\boldsymbol{w}). \quad (8)$$

Before we prove our Lemma, we need introduce a theorem.

**Theorem1 ([4])** Let $f(x, y)$ denote a bounded real-valued function defined on $X \times Y$, where $X$ and $Y$ are convex sets and $X$ is compact. Suppose that $f(\cdot, y)$ is convex and continuous for each fixed $y \in Y$ and $f(x, \cdot)$ is concave for each fixed $x \in X$. Then

$$\inf_{x\in X} \sup_{y\in Y} f(x, y) = \sup_{y\in Y} \inf_{x\in X} f(x, y).$$

By this theorem we know that

$$\min_{\boldsymbol{\alpha}\in\Delta(K)} \max_{\boldsymbol{l}\in[0,1]^K} \langle \boldsymbol{w}, r(\boldsymbol{\alpha}, \boldsymbol{l}) \rangle = \max_{\boldsymbol{l}\in[0,1]^K} \min_{\boldsymbol{\alpha}\in\Delta(K)} \langle \boldsymbol{w}, r(\boldsymbol{\alpha}, \boldsymbol{l}) \rangle,$$

while by the definition of $r$ we have

$$f(\boldsymbol{\alpha}, \boldsymbol{l}) = \langle \boldsymbol{w}, r(\boldsymbol{\alpha}, \boldsymbol{l}) \rangle = \sum_{i=1}^{K} \alpha_i \cdot l_i \cdot w_{1,i} + \sum_{i=1}^{K} l_i \cdot w_{2,i}.$$

Now we give the proof of Lemma 3.

*Proof* $\forall \boldsymbol{\alpha} \in \Delta(K)$ and $\boldsymbol{l} \in [0, 1]^K$ we have

$$f(\boldsymbol{\alpha}, \boldsymbol{l}) \leq \max_{\boldsymbol{l}\in[0,1]^K} f(\boldsymbol{\alpha}, \boldsymbol{l}) = f(\boldsymbol{\alpha}, \boldsymbol{l}^*),$$

where $\boldsymbol{l}^* = \arg\max_{\boldsymbol{l}} f(\boldsymbol{\alpha}, \boldsymbol{l})$. So that we have

$$\min_{\boldsymbol{\alpha}\in\Delta(K)} f(\boldsymbol{\alpha}, \boldsymbol{l}) \leq \min_{\boldsymbol{\alpha}\in\Delta(K)} \max_{\boldsymbol{l}} f(\boldsymbol{\alpha}, \boldsymbol{l}) = \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{l}^*) = f(\boldsymbol{\alpha}^*, \boldsymbol{l}^*),$$

where for $\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{l}^*)$.

For $\boldsymbol{l}^*$ by the definition of $S$ we can find an $\bar{\boldsymbol{\alpha}}^*$ such that

$$\|\bar{\boldsymbol{\alpha}}^* \odot \boldsymbol{l}^*\|_p = \min_{\boldsymbol{\alpha}\in\Delta(K)} \|\boldsymbol{\alpha} \odot \boldsymbol{l}^*\|_p. \quad (9)$$

Hence we have that $(\bar{\boldsymbol{\alpha}}^* \odot \boldsymbol{l}^*, \boldsymbol{l}^*) \in S$.

By Theorem 1 we have

$$\min_{\boldsymbol{\alpha}} \max_{\boldsymbol{l}} f(\boldsymbol{\alpha}, \boldsymbol{l}) = \max_{\boldsymbol{l}} \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{l}) = f(\boldsymbol{\alpha}^*, \boldsymbol{l}^*).$$

Hence we get

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \max_{\boldsymbol{l}} f(\boldsymbol{\alpha}, \boldsymbol{l}) &= \max_{\boldsymbol{l}} \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{l}) = \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{l}^*) \\ &\leq f(\bar{\boldsymbol{\alpha}}^*, \boldsymbol{l}^*) \\ &= \sum_{i=1}^{K} \bar{\alpha}_i^* \cdot l_i^* \cdot w_{1,i} + \sum_{i=1}^{K} l_i^* \cdot w_{2,i} \\ &= \langle \boldsymbol{w}, ((\bar{\boldsymbol{\alpha}}^* \odot \boldsymbol{l}^*), \boldsymbol{l}^*) \rangle \\ &\leq h_S(\boldsymbol{w}). \end{aligned}$$

□

## 3. Main result

In this section, we propose algorithms for online learning with global cost functions.

Note that in Algorithm 1 we denote $s_t = \arg\max_{s \in S} \langle s, w_t \rangle$, and we denote SOCP as the Second Order Conic Programming to calculate $s \in S$.

Therefore we can give the algorithm as follows:

**Algorithm 1** Algorithm for global cost function

**Initialization:** An online linear optimization algorithm OLO,

$$S = \{(x, y) \in \mathbb{R}^k \times \mathbb{R}^k : x_i, y_i \in [0,1]; C_p(x) \le C_p^\star(y)\}. \quad (10)$$

**for** $t = 1, \cdots, T$ **do**

1. Let $w_t$ be the output of OLO.
2. Compute

$$\alpha_t = \arg\min_{\alpha \in \Delta(K)} \max_{l \in [0,1]^K} \langle w_t, (\alpha \odot l, l) \rangle, \quad (11)$$

3. Compute a subgradient $s_t \in \partial h_S(w_t)$ and feed $g_t = -(\alpha_t \odot l_t, l_t) + s_t$ to OLO.

**end for**

Moreover we have the following theorem.

**Theorem2** Given an algorithm OLO for online linear optimization with its regret $\mathrm{Regret}_{\mathrm{OLO}}$, algorithm 1 achieves

$$\mathrm{Regret}_{T,p} \le 2^{\frac{p-1}{p}} \mathrm{Regret}_{\mathrm{OLO}}.$$

Before we prove this Theorem, we give a Lemma for subgradient of $f_t(w)$.

**Lemma4** For our target set $S$, and the loss function $f_t(w) = \langle -r_t, w \rangle + h_S(w)$, so we have that for any $z \in \partial f_t(w)$:

$$\|z\|_p \le (2K)^{1/p}.$$

*Proof* For any $w, u$ in unit ball with respect to $L_q$-norm. And we define that $s^w = \arg\max_{s \in S} \langle s, w \rangle$, and $s^u = \arg\max_{s \in S} \langle s, u \rangle$. we have

$$\begin{aligned} f_t(w) - f_t(u) &= \langle -r_t, w - u \rangle + \langle s^w, w \rangle - \langle s^u, u \rangle \\ &\le \langle -r_t, w - u \rangle + \langle s^w, w \rangle - \langle s^w, u \rangle \\ &= \langle -r_t, w - u \rangle + \langle s^w, w - u \rangle \\ &\le \| - r + s^w \|_p \|w - u\|_q \\ &\le (2K)^{1/p} \|w - u\|_q, \end{aligned}$$

the first inequality is from the definition of support function and the second is from Cauchy-Schwartz inequality. Therefore we showed that $f_t$ is Lipschitz with respect to $\|\cdot\|_q$ with Lipschitz constant $(2K)^{1/p}$. By Lemma 2.6 [11] we obtain $z \in \partial f_t(w_t)$, $\|z\|_p \le (2K)^{1/p}$. □

Now we give the proof of Theorem 2.

*Proof* By above Lemma 3 we see that if we choose $\alpha_t$ according to our algorithm, we see that each $\alpha_t$ satisfies that

$$\langle w_t, (\alpha_t \odot l_t, l_t) \rangle \le h_S(w_t),$$

for any $w_t \in B_q$, and $1/p + 1/q = 1$.

Now for $\bar{r}_T$ there exists $s = (x, y) \in S$ such that

$$\begin{aligned} C_p(L_T^A) - C_p^\star(L_T) &= [C_p(x) - C_p^\star(y)] + [C_p(L_T^A) - C_p(x)] \\ &\quad + [C_p^\star(y) - C_p^\star(L_T)] \\ &\le 1(\|L_T^A - x\|_p + \|L_T - y\|_p) \\ &\le 2^{(p-1)/(p)} \|((L_T^A, L_T) - (x, y))\|_p \\ &= 2^{(p-1)/p} \|\tilde{r}_T - (x, y)\|_p, \end{aligned}$$

where the first inequality is from the definition of $S$, and triangle inequality for $p$-norm. The second inequality is from the following inequality

$$(x^p + y^p) \le 2^{p-1}(x + y)^p.$$

By the definition of support function, $h_S(w_t)$ is a convex function, hence we know that $f_t(w_t) = -\langle r_t, w_t \rangle + h_S(w_t)$ is a convex function, too. Thus we have that

$$f_t(w_t) - f_t(w^*) \le \langle g_t, (w_t - w^*) \rangle, \quad (12)$$

where $g_t \in \partial f_t(w_t)$.

Therefore we obtain that

$$\begin{aligned} \mathrm{dist}(\tilde{r}_T, S) = \|\tilde{r}_T - (x, y)\|_p &= \max_{\|w\|_q \le 1} \{\langle w, \tilde{r}_T \rangle - h_S(w)\} \\ &= -\min_{\|w\|_q \le 1} \{-\langle w, \tilde{r}_T \rangle + h_S(w)\} \\ &\le -\min_{\|w\|_q \le 1} \{-\langle w, \tilde{r}_T \rangle + h_S(w)\} + \sum_{t=1}^{T} \{-\langle w_t, r_t \rangle + h_S(w_t)\} \\ &= -\min_{\|w\|_q \le 1} \sum_{t=1}^{T} f_t(w) + \sum_{t=1}^{T} f_t(w_t) \\ &\le \sum_{t=1}^{T} \langle g_t, w_t \rangle - \min_{\|w\|_q \le 1} \sum_{t=1}^{T} \langle g_t, w \rangle \\ &\le \mathrm{Regret}_T(OLO), \end{aligned}$$

the first inequality holds by $-\langle w_t, r_t \rangle + h_S(w_t)$ is always positive, and the second inequality is by the convexity of our loss function $f_t$. □

For online load balancing problem we have a specific algorithm. Principally, in this algorithm we utilize EG plus-minus in [7] as an oracle OLO algorithm. First we give an introduction of EG plus-minus algorithm. The EG plus-minus algorithm begins with two $2K$-dimensional initialization weight vectors $w_1^+ = w_1^- = (1/4K, \cdots, 1/4K)$. We denote $g_t(i)$ for the $i$-th element in vector $g_t$, at each round $t$, we update

$$w_{t+1,i}^+ = \frac{w_{t,i}^+ e^{-\eta \cdot g_t(i)}}{\sum_{j=1}^{2K} w_{t,j}^+ e^{-\eta \cdot g_t(j)} + w_{t,j}^- e^{\eta g_t(j)}},$$

$$w^-_{t+1,i} = \frac{w^-_{t,i} e^{\eta \cdot g_t(i)}}{\sum_{j=1}^{2K} w^+_{t,j} e^{-\eta \cdot g_t(j)} + w^-_{t,j} e^{\eta g_t(j)}},$$

and $\boldsymbol{w}_t = \boldsymbol{w}^+_t - \boldsymbol{w}^-_t$. This algorithm predicts $\boldsymbol{w}_t \in \{\boldsymbol{w} : \|\boldsymbol{w}\|_1 \leq 1\}$. The regret of this algorithm is given in following:

**Theorem3** ([8]) Assuming that $\|\boldsymbol{g}_t\|_\infty \leq G$ for all rounds $t$. Then the regret of EG plus-minus satisfies

$$\sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{w}_t \rangle - \min_{\|\boldsymbol{w}\|_1 \leq 1} \sum_{t=1}^{T} \langle \boldsymbol{g}_t, \boldsymbol{w} \rangle \leq G\sqrt{2T \ln(4K)}, \qquad (13)$$

setting $\eta = \sqrt{\frac{2 \ln 4K}{TG^2}}$.

According to above Theorem we get the regret for online load balancing problem in following corollary.

**Corollary1** Consider the online load balancing problem with $K$-machines. If we run algorithm 2 for $T$ times. The regret of online load balancing problem is as follows:

$$\mathrm{Regret}_{T,\infty} \leq 2\sqrt{2T \ln 4K}.$$

*Proof*

$$\begin{aligned} C_\infty(L^A_T) - C^\star_\infty(L_T) &= [C_\infty(\boldsymbol{x}) - C^\star_\infty(\boldsymbol{y})] + [C_\infty(L^A_T) - C_\infty(\boldsymbol{x})] \\ &\quad + [C^\star_\infty(\boldsymbol{y}) - C^\star_\infty(L_T)] \\ &\leq 1(\|L^A_T - \boldsymbol{x}\|_\infty + \|L_T - \boldsymbol{y}\|_\infty) \\ &\leq 1\|2((L^A_T, L_T) - (\boldsymbol{x}, \boldsymbol{y}))\|_\infty \\ &= 2\|\tilde{\boldsymbol{r}}_T - (\boldsymbol{x}, \boldsymbol{y})\|_\infty, \end{aligned}$$

The rest part of the proof is similar to the proof of Theorem 2. So we obtain that

$$\mathrm{dist}(\tilde{\boldsymbol{r}}_T, S) \leq \mathrm{Regret}_T(OLO).$$

In online load balancing problem we have known that the sub-gradient is $\boldsymbol{g}_t = -(\alpha_t \odot l_t) + \boldsymbol{s}_t$, so by Lemma 4 $\|\boldsymbol{g}_t\|_\infty \leq 1$. Therefore if we utilize EG plus-minus we have

$$C_\infty(L^A_T) - C^\star_\infty(L_T) \leq 2\sqrt{2T \ln 4K}.$$

□

Note that our regret is better than the regret achieved by algorithm DIFF in [5], since the parameter here is $\frac{2\sqrt{2\ln 4K}}{\sqrt{T}}$ rather than $\frac{\ln K}{\sqrt{T}}$.

## 4. Algorithmic details

In this section we give details to proceed Algorithm 1. In particular, for the makespan problem, i.e., $p = \infty$, we give a polynomial time algorithm.

### 4.1 Computing $\alpha_t$

There are two procedures in Algorithm 1. Specifically, on the round $t$, we need to choose $\alpha_t$, which is the optimal solution of the problem (11). That is,

$$\min_{\boldsymbol{\alpha} \in \Delta(K)} \max_{\boldsymbol{l} \in [0,1]^K} \langle w_{t,1}, (\boldsymbol{\alpha} \odot \boldsymbol{l}) \rangle + \langle w_{t,2}, \boldsymbol{l} \rangle, \qquad (14)$$

where we set that $w_t = (w_{t,1}, w_{t,2})$ and $w_{t,1}$ is a $K$-dimensional vector. We see that the optimization of this objective function is defined by $l_i = 0$ if $w_{t,1}(i) \cdot \alpha(i) + w_{t,2}(i) \leq 0$, otherwise we let $l(i) = 1$. Hence we can convert our problem to choose $\alpha$ as

$$\min_{\boldsymbol{\alpha} \in \Delta(K)} \max_{\boldsymbol{l} \in [0,1]^K} \langle w_{t,1}, (\boldsymbol{\alpha} \odot \boldsymbol{l}) \rangle + \langle w_{t,2}, \boldsymbol{l} \rangle = \min_{\boldsymbol{\alpha} \in \Delta(K)} \sum_{i=1}^{K} \max\{0, \alpha(i) w_{t,1}(i) + w_{t,2}(i)\},$$

which is equivalent to

$$\begin{aligned} &\min \sum_{i=1}^{K} \beta(i) \\ &\mathrm{s.t.} \beta(i) \geq w_{t,1}(i)\alpha(i) + w_{t,2}(i) \quad \forall i \\ &\sum_{i=1}^{K} \alpha_t(i) = 1 \\ &\beta(i), \alpha_t(i) \geq 0 \quad \forall i. \end{aligned}$$

The above problem is a linear program with $O(K)$ variables and $O(K)$ linear constraints.

### 4.2 Computing subgradients $\boldsymbol{g}_t$ for $p = \infty$

The second component of Algorithm 1 is to compute subgradients $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{w}_t)$. By Proposition 1, $\boldsymbol{g}_t$ is given as $\boldsymbol{g}_t = -(\boldsymbol{\alpha}_t \odot \boldsymbol{l}_t, \boldsymbol{l}_t) + \boldsymbol{s}_t$ and thus the main part is to compute $\boldsymbol{s}_t = \arg\min_{\boldsymbol{s} \in S} \langle \boldsymbol{s}, \boldsymbol{w}_t \rangle$. Recall that $S = \{(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^K \times [0,1]^K \mid C_\infty(\boldsymbol{x}) \leq C^*_\infty(\boldsymbol{y})\}$ for $p = \infty$.

First of all, $C^*_\infty(\boldsymbol{y})$ is given as follows:

**Proposition2 (Even-Dar et al.[5])** For $\boldsymbol{y} \in [0,1]^K$,

$$C^*_\infty(\boldsymbol{y}) = \frac{1}{\sum_{i=1}^{K} \frac{1}{y_i}}.$$

In particular, the condition that $C_\infty(\boldsymbol{x}) \leq C^*_\infty(\boldsymbol{y})$ can be represented as

$$\max_i x_i \leq \min_{\boldsymbol{\alpha} \in \Delta(K)} C_\infty(\boldsymbol{\alpha} \odot \boldsymbol{y}) \iff x_i \leq \frac{1}{\sum_{j=1}^{K} \frac{1}{y_j}}, \forall i.$$

Therefore, the computation of the subgradient $\boldsymbol{s}_t$ is formulated as

$$\begin{aligned} &\max_{\boldsymbol{x}, \boldsymbol{y}} w_1 \cdot \boldsymbol{x} + w_2 \cdot \boldsymbol{y} \\ &\mathrm{s.t.} x_i \leq \frac{1}{\sum_j \frac{1}{y_j}} \quad \forall i; \qquad (15) \\ &x_i, y_i \in [0,1] \quad \forall i. \end{aligned}$$

Now we show that there exists an equivalent second order cone programming(SOCP) formulation (e.g., [9]) for this problem.

First we give the definition of second order cone programming, and then we give a proposition, which states that our optimization problem is equivalent to second order cone programming.

**Definition4** The standard form for the Second Order Conic Programming(SOCP) model is as follows:

$$\min_{\boldsymbol{x}} \boldsymbol{c}^\top \boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \|C_i \boldsymbol{x} + \boldsymbol{d}_i\|_2 \le \boldsymbol{e}_i^\top \boldsymbol{x} + \boldsymbol{f}_i \quad i = 1, \cdots, m,$$

where the problem parameters are $\boldsymbol{c} \in \mathbb{R}^n$, $C_i \in \mathbb{R}^{n_i \times n}$, $\boldsymbol{d}_i \in \mathbb{R}^{n_i}$, $\boldsymbol{e} \in \mathbb{R}^n$, $\boldsymbol{f}_i \in \mathbb{R}$, $A \in \mathbb{R}^{p \times n}$, and $\boldsymbol{b} \in \mathbb{R}^p$. $\boldsymbol{x} \in \mathbb{R}^n$ is the optimization variable.

Then we obtain the following proposition.

**Proposition3** $\sum_{i=1}^k \frac{x^2}{y_i} \le x$, $x \ge 0$ and $y_i \ge 0$ is equivalent to $x^2 \le y_i z_i$, where $y_i, z_i \ge 0$ and $\sum_{i=1}^k z_i = x$.

*Proof* On the direction " $\Longrightarrow$ "

From $\sum_{i=1}^k \frac{x^2}{y_i} \le x$ we obtain that $\sum_{i=1}^k \frac{x}{y_i} \le 1$. By setting

$$z_i = x \cdot \frac{\frac{1}{y_i}}{\sum_i \frac{1}{y_i}},$$

we can have that $x^2 \le y_i z_i$, and $\sum_{i=1}^k z_i = x$.

On the other direction " $\Longleftarrow$ " Due to $x^2 \le y_i z_i$, we have $\frac{x^2}{y_i} \le z_i$. So we have that

$$\sum_{i=1}^k \frac{x^2}{y_i} \le \sum_{i=1}^k z_i = x.$$

□

Again in our case we need find the optimal vector $\boldsymbol{s} \in S$, which satisfies that $\boldsymbol{s}_t = \arg\max_{\boldsymbol{s} \in S} \langle \boldsymbol{w}_t, \boldsymbol{s} \rangle$. Note that by Lemma 1 $S$ is a cone and it is a convex set. Therefore we can reduce our problem in following theorem.

**Theorem4** The optimization problem represented in Equation (15) can be solved by second order cone programming.

*Proof* To prove this theorem we only need to represent the original problem in equation (15) as a standard form of SOCP problem.

Note that we only consider the case that $y_i \neq 0$. If $y_i = 0$ the problem is very trivial. If it exists $y_i = 0$, by the definition of $S$ we know that for all $i$, $x_i = 0$.

On the one hand, for $x_i \le \frac{1}{\sum_j \frac{1}{y_j}}$, we can multiply $x_i$ on both sides and rearrange the inequality then we have following:

$$\sum_{j=1}^k \frac{x_i^2}{y_j} \le x_i,$$

which implies the conic representation as

$$y_j z_j \ge x_i^2, \quad y_j, z_j \ge 0, \quad \sum_{j=1}^k z_j = x_i.$$

By the paper [9] we may rewrite it as follow: For each $i$,

$$x_i^2 \le y_j z_j; \quad y_j, z_j \ge 0 \Longleftrightarrow \|(2x_i, y_j - z_j)\|_2 \le y_j + z_j \quad \forall j \in [k]. \tag{16}$$

The above equivalence is trivial.

On the other hand, since $x_i \le \frac{1}{\sum_j \frac{1}{y_j}}$, and $y_i \in [0, 1]$, naturally we have $x_i \in [0, 1]$. So we need only constrain that $y_i \in [0, 1]$. We can apply the face that if $y_i$ is positive so $|y_i| = y_i$, and if $y_i \le 1$, so $|y_i| \le 1$.

Therefore we may give a $(k^2 + 2k) \times (k^2 + 2k)$-matrix $C_i$ in SOCP, and the variable vector is composed as follow:

$$\tilde{\boldsymbol{x}} = (x_1, \cdots, x_k, y_1, \cdots, y_k, z_{1,1}, \cdots, z_{1,k}, \cdots, z_{k,1} \cdots, z_{k,k}), \tag{17}$$

where for $z_{i,j}$, $i$ is corresponding to $x_i$.

Now we may give the second order cone programming of our target problem as follows:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{x}}} \langle -(w_1, w_2, 0, \cdots, 0), \tilde{\boldsymbol{x}} \rangle \\ &\text{s.t.} \|C_i \tilde{\boldsymbol{x}}\|_2 \le \boldsymbol{e}_i^\top \tilde{\boldsymbol{x}} + \boldsymbol{d}_i \quad \forall i = 1, \cdots, k^2 + 2k, \\ &A\tilde{\boldsymbol{x}} = \boldsymbol{b}. \end{aligned} \tag{18}$$

where $C_i$, $\boldsymbol{e}_i$, $A$ and $\boldsymbol{b}$ are defined as follows:

Firstly for hyperbolic constraints:

for a fixed $s \in [k]$, where $[k] = 1, \cdots, k$, in matrix $C_i$, where $i \in [(s-1)k, sk]$ we let $(C_i)_{1,s} = 2$, $(C_i)_{k+i,k+i} = 1$, $(C_i)_{2k+(s-1)k+i, 2k+(s-1)k+i} = -1$, and others are 0. $\boldsymbol{e}_i$ is defined as $(\boldsymbol{e}_i)_{k+i} = 1$ and $(\boldsymbol{e}_i)_{2k+(s-1)k+i} = 1$, others are 0.

Next we need to constrain that $y_i$ is less than 1 :

For $i \in [k^2, k^2 + k]$ we let that $(C_i)_{k+i,k+i} = 1$ and others are 0. And we let that $\boldsymbol{e}_i$ is a zero vector and $\boldsymbol{d}_i = 1$. It means that $\|y_i\| \le 1$. For $i \in [k^2 + k, k^2 + 2k]$, we set $(C_i)_{k+i,k+i} = 1$ $\boldsymbol{e}_{k+i} = 1$, and $\boldsymbol{d}_i = 0$

At last we need to constrain that $\sum_{j=1}^k z_j = x_i$ in equation 16:

Let $A \in \mathbb{R}^{k \times (3k+k^2)}$ for each row vector $A_j$, where $j \in [k]$, we have that $(A_j)_j = 1$ and $(A_j)_{2k+(j-1)j+m} = -1$, for all $m = 1, \cdots, k$. And matrix $A$ is composed by the row vectors $A_j$. and $\boldsymbol{b}$ is a zero vector.

□

## 5. conclusion

In this paper we give a framework for online global cost function with respect to $p$-norm, where $p > 2$. Especially, for online load balancing problem we have an efficient algorithm by involving EG plus-minus. On each update step of our algorithm we can directly predict the allocation of loads by a linear programming. Simultaneously, we show that we can update the EG plus-minus by SOCP in polynomial time for online load balancing problem. Since the efficiency of these two programmings, we construct our algorithm, and show that the upper bound of regret is better than other algorithms to the best we known.

## 参考文献


[1] Jacob Abernethy, Peter L Bartlett, and Elad Hazan. Blackwell approachability and no-regret learning are equivalent. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 27–46, 2011.

[2] Yossi Azar. On-line load balancing. In *Online Algorithms*, pages 178–195. Springer, 1998.

[3] David Blackwell et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[4] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[5] Eyal Even-Dar, Robert Kleinberg, Shie Mannor, and Yishay Mansour. Online learning for global cost functions. In *COLT*, 2009.

[6] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[7] Dirk Hoeven, Tim Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In *Conference On Learning Theory*, pages 2067–2092, 2018.

[8] Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *information and computation*, 132(1):1–63, 1997.

[9] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998.

[10] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Beyond regret. 2011.

[11] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[12] Nahum Shimkin. An online convex optimization approach to blackwell's approachability. *The Journal of Machine Learning Research*, 17(1):4434–4456, 2016.