# Simple and Fast Iterative Algorithm for (Binary Integer) Online Linear Programming

Yinyu Ye

[1]Department of Management Science and Engineering
Institute of Computational and Mathematical Engineering
Stanford University, Stanford

July 14, 2020
(Joint work with Xiaocheng LI and Chunlin Sun)

# Table of Contents

# Motivation Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers

# Motivation Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods

# Motivation Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices

# Motivation Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices
- Decision: To sell or not to each individual customer?

# Motivation Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices
- Decision: To sell or not to each individual customer?
- Objective: Maximize the revenue.

# A Toy Example

Consider an auction problem:

| | Bid 1($t=1$) | Bid 2($t=2$) | ..... | Inventory($\mathbf{b}$) |
|---|---|---|---|---|
| Reward($r_t$) | \$100 | \$30 | ... | |
| Decision | $x_1$ | $x_2$ | ... | |
| Pants | 1 | 0 | ... | 100 |
| Shoes | 1 | 0 | ... | 50 |
| T-shirts | 0 | 1 | ... | 500 |
| Jackets | 0 | 0 | ... | 200 |
| Hats | 1 | 1 | ... | 1000 |

where the decision for each bid is "accept" ($x_t = 1$) or "reject" ($x_t = 0$)

# Offline Linear Programming Problem

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} r_t x_t \\ \text{subject to} & \sum_{t=1}^{n} \mathbf{a}_t x_t \leq \mathbf{b}, \\ & x_t \in \{0,1\}\ (0 \leq x_t \leq 1), \quad \forall t = 1, ..., n. \end{array}$$

# Offline Linear Programming Problem

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} r_t x_t \\ \text{subject to} & \sum_{t=1}^{n} \mathbf{a}_t x_t \leq \mathbf{b}, \\ & x_t \in \{0,1\}\ (0 \leq x_t \leq 1), \quad \forall t = 1, ..., n. \end{array}$$

$r_t$: reward/revenue offered by the $t$-th customer order

$\mathbf{a}_t \in R^m$: the bundle of resources requested by the $t$-th customer order

$x_t$: acceptance or rejection decision to the $t$-th customer order

$\mathbf{b} \in R^m$: initially available budget/resource inventories

The objective $\sum_{t=1}^{n} r_t x_t$: the total collected revenue.

# Offline Linear Programming Problem

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} r_t x_t \\ \text{subject to} & \sum_{t=1}^{n} \mathbf{a}_t x_t \leq \mathbf{b}, \\ & x_t \in \{0,1\}\ (0 \leq x_t \leq 1), \quad \forall t = 1, ..., n. \end{array}$$

$r_t$: reward/revenue offered by the $t$-th customer order

$\mathbf{a}_t \in R^m$: the bundle of resources requested by the $t$-th customer order

$x_t$: acceptance or rejection decision to the $t$-th customer order

$\mathbf{b} \in R^m$: initially available budget/resource inventories

The objective $\sum_{t=1}^{n} r_t x_t$: the total collected revenue.

- We only know $\mathbf{b}$ and $n$ at the start

# Offline Linear Programming Problem

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} r_t x_t \\ \text{subject to} & \sum_{t=1}^{n} \mathbf{a}_t x_t \leq \mathbf{b}, \\ & x_t \in \{0,1\}\ (0 \leq x_t \leq 1), \quad \forall t = 1, ..., n. \end{array}$$

$r_t$: reward/revenue offered by the $t$-th customer order

$\mathbf{a}_t \in R^m$: the bundle of resources requested by the $t$-th customer order

$x_t$: acceptance or rejection decision to the $t$-th customer order

$\mathbf{b} \in R^m$: initially available budget/resource inventories

The objective $\sum_{t=1}^{n} r_t x_t$: the total collected revenue.

- We only know $\mathbf{b}$ and $n$ at the start
- the bidder data information is revealed sequentially.

# Offline Linear Programming Problem

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} r_t x_t \\ \text{subject to} & \sum_{t=1}^{n} \mathbf{a}_t x_t \leq \mathbf{b}, \\ & x_t \in \{0,1\}\ (0 \leq x_t \leq 1), \quad \forall t = 1, ..., n. \end{array}$$

$r_t$: reward/revenue offered by the $t$-th customer order

$\mathbf{a}_t \in R^m$: the bundle of resources requested by the $t$-th customer order

$x_t$: acceptance or rejection decision to the $t$-th customer order

$\mathbf{b} \in R^m$: initially available budget/resource inventories

The objective $\sum_{t=1}^{n} r_t x_t$: the total collected revenue.

- We only know $\mathbf{b}$ and $n$ at the start
- the bidder data information is revealed sequentially.
- an irrevocable decision must be made as soon as an order arrives without observing or knowing the future data.

# Online Linear Programming (OLP) Problem

Precisely, at time $t = 1, ..., n$, decide variable $x_t \in [0, 1]$, given previous data and decisions $x_1, \cdots, x_{t-1}$ already made but without knowing the future data:

# Online Linear Programming (OLP) Problem

Precisely, at time $t = 1, ..., n$, decide variable $x_t \in [0, 1]$, given previous data and decisions $x_1, \cdots, x_{t-1}$ already made but without knowing the future data:

$$r_1 x_1 + \cdots + r_t x_t + \cdots ? \cdots$$

$$\begin{pmatrix} | & | & | & & & & \\ \mathbf{a}_1 & \cdots & \mathbf{a}_t & ? & \cdots & \cdots & ? \\ | & | & | & & & & \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \\ ? \\ \vdots \\ ? \end{pmatrix} \leq \mathbf{b}.$$

# Table of Contents

# Data Assumptions

**Stochastic Input (i.i.d) Model**:

(a) The column-coefficient pair $(r_t, \mathbf{a}_t)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$.

# Data Assumptions

**Stochastic Input (i.i.d) Model**:

(a) The column-coefficient pair $(r_t, \mathbf{a}_t)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$.

**Random Permutation (RP) Model**:

(a') The column-coefficient pair $(r_t, \mathbf{a}_t)$'s arrive in a random order.

# Data Assumptions

**Stochastic Input (i.i.d) Model**:

(a) The column-coefficient pair $(r_t, \mathbf{a}_t)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$.

**Random Permutation (RP) Model**:

(a') The column-coefficient pair $(r_t, \mathbf{a}_t)$'s arrive in a random order.

**Both** assume boundedness:

(b) There exist constants $\bar{r}$ and $\bar{a}$ such that $|r_t| \leq \bar{r}$ and $\|\mathbf{a}_t\|_\infty \leq \bar{a}$ for $t = 1, ..., n$.

(c) The right-hand-side $\mathbf{b} = n \cdot \mathbf{d}$ and there exists positive constant $\underline{d}$ and $\bar{d}$ such that $\underline{d} \leq d_t \leq \bar{d}$ for $i = 1, ..., m$.

# Data Assumptions

**Stochastic Input (i.i.d) Model**:

(a) The column-coefficient pair $(r_t, \mathbf{a}_t)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$.

**Random Permutation (RP) Model**:

(a') The column-coefficient pair $(r_t, \mathbf{a}_t)$'s arrive in a random order.

**Both** assume boundedness:

(b) There exist constants $\bar{r}$ and $\bar{a}$ such that $|r_t| \leq \bar{r}$ and $\|\mathbf{a}_t\|_\infty \leq \bar{a}$ for $t = 1, ..., n$.

(c) The right-hand-side $\mathbf{b} = n \cdot \mathbf{d}$ and there exists positive constant $\underline{d}$ and $\bar{d}$ such that $\underline{d} \leq d_t \leq \bar{d}$ for $i = 1, ..., m$.

$\Xi$ denotes the family of distributions that satisfy (a) and (b) where data are stationary

$\Xi_{\mathcal{D}}$ denote all the data sets $\mathcal{D} = \{(r_t, \mathbf{a}_t)\}_{t=1}^n$ that satisfy (a') and (b)

# Why Study the Random Permutation Model

- From the perspective of online LP, the random permutation model can be interpreted as an online algorithm that solves an online LP problem under data generation assumptions that are weaker than the i.i.d. assumptions. Hence, the stochastic input model can be viewed as a special case of the random permutation model.

# Why Study the Random Permutation Model

- From the perspective of online LP, the random permutation model can be interpreted as an online algorithm that solves an online LP problem under data generation assumptions that are weaker than the i.i.d. assumptions. Hence, the stochastic input model can be viewed as a special case of the random permutation model.
- Problem data are indeed heterogeneous and non-stationary in real world applications.

# Why Study the Random Permutation Model

- From the perspective of online LP, the random permutation model can be interpreted as an online algorithm that solves an online LP problem under data generation assumptions that are weaker than the i.i.d. assumptions. Hence, the stochastic input model can be viewed as a special case of the random permutation model.
- Problem data are indeed heterogeneous and non-stationary in real world applications.
- From the perspective of solving offline binary integer LPs, the algorithm creates the randomness (random-column-generation) for the problem when there is no inherent randomness with the original problem data.

# Why Study the Random Permutation Model

- From the perspective of online LP, the random permutation model can be interpreted as an online algorithm that solves an online LP problem under data generation assumptions that are weaker than the i.i.d. assumptions. Hence, the stochastic input model can be viewed as a special case of the random permutation model.
- Problem data are indeed heterogeneous and non-stationary in real world applications.
- From the perspective of solving offline binary integer LPs, the algorithm creates the randomness (random-column-generation) for the problem when there is no inherent randomness with the original problem data.
- What solution could you produce and say about its quality in a shortest possible time?

# Performance Metrics: Competitive Ratio

"Offline" Optimal solution $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ and an "Online" solution $\mathbf{x} = (x_1, ..., x_n)$. Let

$$R_n^* = \sum_{j=1}^{n} r_j x_j^* \quad \text{and} \quad R_n = \sum_{j=1}^{n} r_j x_j.$$

# Performance Metrics: Competitive Ratio

"Offline" Optimal solution $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ and an "Online" solution $\mathbf{x} = (x_1, ..., x_n)$. Let

$$R_n^* = \sum_{j=1}^{n} r_j x_j^* \quad \text{and} \quad R_n = \sum_{j=1}^{n} r_j x_j.$$

We call an online algorithm $\mathcal{A}$ to be $c$-competitive if and only if

$$\mathbb{E}[R_n] \geq c \cdot \mathbb{E}[R_n^*] \ \forall \text{ input problem data.}$$

Here the expectation is taken with respect to the distribution $\mathcal{P}$ that generates $(r_t, \mathbf{a}_t)$'s for the Stochastic Input, or is taken with respect to the Random Permutation ordering of $(r_t, \mathbf{a}_t)$'s.

# Performance Metrics: Regret Bounds

The worst-case absolute gap between the offline and online objective values

$$\sup_{\mathcal{P}\in\Xi} \mathbb{E}_{\mathcal{P}}\left[R_n^* - R_n\right] \quad \text{(Stochastic Input)}$$

$$\sup_{\mathcal{D}\in\Xi_{\mathcal{D}}} R_n^* - \mathbb{E}\left[R_n\right] \quad \text{(Random Permutation)}$$

# Performance Metrics: Regret Bounds

The worst-case absolute gap between the offline and online objective values

$$\sup_{\mathcal{P}\in\Xi} \mathbb{E}_{\mathcal{P}}\left[R_n^* - R_n\right] \quad \text{(Stochastic Input)}$$

$$\sup_{\mathcal{D}\in\Xi_{\mathcal{D}}} R_n^* - \mathbb{E}\left[R_n\right] \quad \text{(Random Permutation)}$$

Again, the expectation in the first line is taken with respect to the distribution $\mathcal{P}$ that generates $(r_t, \mathbf{a}_t)$'s, and the expectation in the second line is taken with respect to the random permutation ordering of $(r_t, \mathbf{a}_t)$'s.

# Performance Metrics: Budget Violation

Also, we measure the constraint violation of the online solution

$$v(\mathbf{x}) = \| (A\mathbf{x} - \mathbf{b})^+ \|_2$$

and establish a bound on the worst-case expected violation.

Remark: A bi-criteria performance measure

# Performance Metrics: Budget Violation

Also, we measure the constraint violation of the online solution

$$v(\mathbf{x}) = \| (A\mathbf{x} - \mathbf{b})^+ \|_2$$

and establish a bound on the worst-case expected violation.

Remark: A bi-criteria performance measure

Different from *Online Convex Optimization* (OCO):

- Presence of the constraints
- A stronger benchmark: the "offline" optimal $(x_1^*, ..., x_n^*)$ are allowed to take different values in online LP but are required to be the same (stationary) for OCO.

# Table of Contents

# Literature Review: Randomized/Fast LP Algorithms

- Early works on column generation algorithm: Ford Jr and Fulkerson (1958); Dantzig (1963)
- Random projection/Sampling based: De Farias and Van Roy (2004); Lakshminarayanan et al. (2017); Vu et al. (2018)
- ADMM: Yen et al. (2015); Wang and Shroff (2017); Lin et al. (2018)
- Many other first-order methods...

# Literature Review: Online Convex Optimization

- The problem of online LP can seemingly be viewed as a special form of online convex optimization with constraints (OCOwC).
- The key difference lies in defining the benchmark:
  - Online LP problem employs a stronger benchmark where the decision variables are allowed to take different values at each time period
  - OCOwC (Mahdavi et al., 2012; Yu et al., 2017; Yuan and Lamperski, 2018) and OCO problems usually considers a stationary benchmark where the the decision variables are required to be the same at each time period.
- Our focus here is to identify the connections between the primal and dual objectives and the constraint violation of the primal – we learn the dual while make online decisions for the primal problem.

# Necessary and Sufficient Conditions

## Theorem

*For the RP model and any fixed* $0 < \epsilon < 1$*, there is no online algorithm for solving the linear program with competitive ratio* $1 - \epsilon$ *if*

$$\min_i \{b_i\} < \frac{\log(m)}{\epsilon^2}.$$

# Necessary and Sufficient Conditions

## Theorem

*For the RP model and any fixed $0 < \epsilon < 1$, there is no online algorithm for solving the linear program with competitive ratio $1 - \epsilon$ if*

$$\min_i \{b_i\} < \frac{\log(m)}{\epsilon^2}.$$

## Theorem

*For the RP model and any fixed $0 < \epsilon < 1$, there is a $1 - \epsilon$ competitive online algorithm for solving the linear program if*

$$\min_i \{b_i\} \geq \Omega\left(\frac{m \log(n/\epsilon)}{\epsilon^2}\right).$$

# Necessary and Sufficient Conditions

## Theorem

*For the RP model and any fixed $0 < \epsilon < 1$, there is no online algorithm for solving the linear program with competitive ratio $1 - \epsilon$ if*

$$\min_i \{b_i\} < \frac{\log(m)}{\epsilon^2}.$$

## Theorem

*For the RP model and any fixed $0 < \epsilon < 1$, there is a $1 - \epsilon$ competitive online algorithm for solving the linear program if*

$$\min_i \{b_i\} \geq \Omega\left(\frac{m \log(n/\epsilon)}{\epsilon^2}\right).$$

Agrawal, Wang and Y [2010, 2014]

# Price Mechanism for OLP I

The problem would be easy if there are "ideal prices":

| | Bid 1($t=1$) | Bid 2($t=2$) | ..... | Inventory($\mathbf{b}$) | $\mathbf{p}^*$ |
|---|---|---|---|---|---|
| Bid($r_t$) | \$100 | \$30 | ... | | |
| Decision | $x_1$ | $x_2$ | ... | | |
| Pants | 1 | 0 | ... | 100 | \$45 |
| Shoes | 1 | 0 | ... | 50 | \$45 |
| T-shirts | 0 | 1 | ... | 500 | \$10 |
| Jackets | 0 | 0 | ... | 200 | \$55 |
| Hats | 1 | 1 | ... | 1000 | \$15 |

so that the online decision can be made by comparing the reward and "bundle cost" for each bid.

# Such Prices can be Constructed from the Dual

Primal:

$$\begin{array}{ll} \max & \mathbf{r}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{0} \le x \le \mathbf{e} \end{array}$$

Dual:

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \mathbf{e}^\top \mathbf{s} \\ \text{s.t.} & A^\top \mathbf{p} + \mathbf{s} \ge \mathbf{r} \\ & \mathbf{p} \ge \mathbf{0}, \mathbf{s} \ge \mathbf{0} \end{array}$$

where the decision variables are $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{p} \in \mathbf{R}^m$, $\mathbf{s} \in \mathbf{R}^n$

Denote the primal/dual optimal solution as $\mathbf{x}^* \in \mathbf{R}^n$, $\mathbf{p}_n^* \in \mathbf{R}^m$, $\mathbf{s}^* \in \mathbf{R}^n$

# Such Prices can be Constructed from the Dual

Primal:

$$\begin{array}{ll} \max & \mathbf{r}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{0} \le x \le \mathbf{e} \end{array}$$

Dual:

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \mathbf{e}^\top \mathbf{s} \\ \text{s.t.} & A^\top \mathbf{p} + \mathbf{s} \ge \mathbf{r} \\ & \mathbf{p} \ge \mathbf{0}, \mathbf{s} \ge \mathbf{0} \end{array}$$

where the decision variables are $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{p} \in \mathbf{R}^m$, $\mathbf{s} \in \mathbf{R}^n$

Denote the primal/dual optimal solution as $\mathbf{x}^* \in \mathbf{R}^n$, $\mathbf{p}_n^* \in \mathbf{R}^m$, $\mathbf{s}^* \in \mathbf{R}^n$

LP optimality condition tells that for $t = 1, ..., n$,

$$x_t^* = \begin{cases} 1, & r_t > \mathbf{a}_t^\top \mathbf{p}_n^* \\ 0, & r_t < \mathbf{a}_t^\top \mathbf{p}_n^* \end{cases}$$

$x_t^*$ may be fractional when $r_j = \mathbf{a}_t^\top \mathbf{p}_n^*$.

# Price Mechanism for OLP II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $r_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.

# Price Mechanism for OLP II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $r_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.
- Based on this observation, our online algorithm works by learning a threshold price vector $\hat{\mathbf{p}}$ and using $\hat{\mathbf{p}}$ to price the bids.

# Price Mechanism for OLP II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $r_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.
- Based on this observation, our online algorithm works by learning a threshold price vector $\hat{\mathbf{p}}$ and using $\hat{\mathbf{p}}$ to price the bids.
- One-time learning algorithm: learn the price vector once using the initial $\epsilon n$ input.

# Price Mechanism for OLP II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $r_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.
- Based on this observation, our online algorithm works by learning a threshold price vector $\hat{\mathbf{p}}$ and using $\hat{\mathbf{p}}$ to price the bids.
- One-time learning algorithm: learn the price vector once using the initial $\epsilon n$ input.
- Dynamic learning algorithm: dynamically update the prices at a carefully chosen pace.

# An One-Time Learning Algorithm

We illustrate a simple One-Time Learning Algorithm:

- Set $x_t = 0$ for all $1 \leq t \leq \epsilon n$;

# An One-Time Learning Algorithm

We illustrate a simple One-Time Learning Algorithm:

- Set $x_t = 0$ for all $1 \leq t \leq \epsilon n$;
- Solve the $\epsilon$ portion of the problem

$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\epsilon n} r_t x_t & \\ \text{subject to} & \sum_{t=1}^{\epsilon n} a_{it} x_t \leq (1-\epsilon)\epsilon b_i & i = 1, ..., m \\ & 0 \leq x_t \leq 1 & t = 1, ..., \epsilon n \end{array}$$

and get the optimal dual solution $\hat{\mathbf{p}}$;

# An One-Time Learning Algorithm

We illustrate a simple One-Time Learning Algorithm:

- Set $x_t = 0$ for all $1 \le t \le \epsilon n$;
- Solve the $\epsilon$ portion of the problem

$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\epsilon n} r_t x_t & \\ \text{subject to} & \sum_{t=1}^{\epsilon n} a_{it} x_t \le (1-\epsilon)\epsilon b_i & i = 1, ..., m \\ & 0 \le x_t \le 1 & t = 1, ..., \epsilon n \end{array}$$

  and get the optimal dual solution $\hat{\mathbf{p}}$;
- Determine the future allocation $x_t$ as:

$$x_t = \begin{cases} 0 & \text{if } r_t \le \hat{\mathbf{p}}^T \mathbf{a}_t \\ 1 & \text{if } r_t > \hat{\mathbf{p}}^T \mathbf{a}_t \end{cases}$$

  as long as $a_{it} x_t \le b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ for all $i$; otherwise, set $x_t = 0$.

# The One-Time Learning Algorithm Result

## Theorem

*For any fixed $\epsilon > 0$, the one-time learning algorithm is $(1-\epsilon)$ competitive for solving the linear program when*

$$B \geq \Omega\left(\frac{m \log(n/\epsilon)}{\epsilon^3}\right)$$

This is one $\epsilon$ worse than the optimal lower bound, which can be overcome by a dynamic-learning algorithm.

# A Dynamic Learning Algorithm

In the dynamic price learning algorithm, we update the price at time $\epsilon n$, $2\epsilon n$, $4\epsilon n$, ..., till $2^k \epsilon \geq 1$.

# A Dynamic Learning Algorithm

In the dynamic price learning algorithm, we update the price at time $\epsilon n$, $2\epsilon n$, $4\epsilon n$, ..., till $2^k \epsilon \geq 1$.
At time $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price vector is the optimal dual solution to the following linear program:

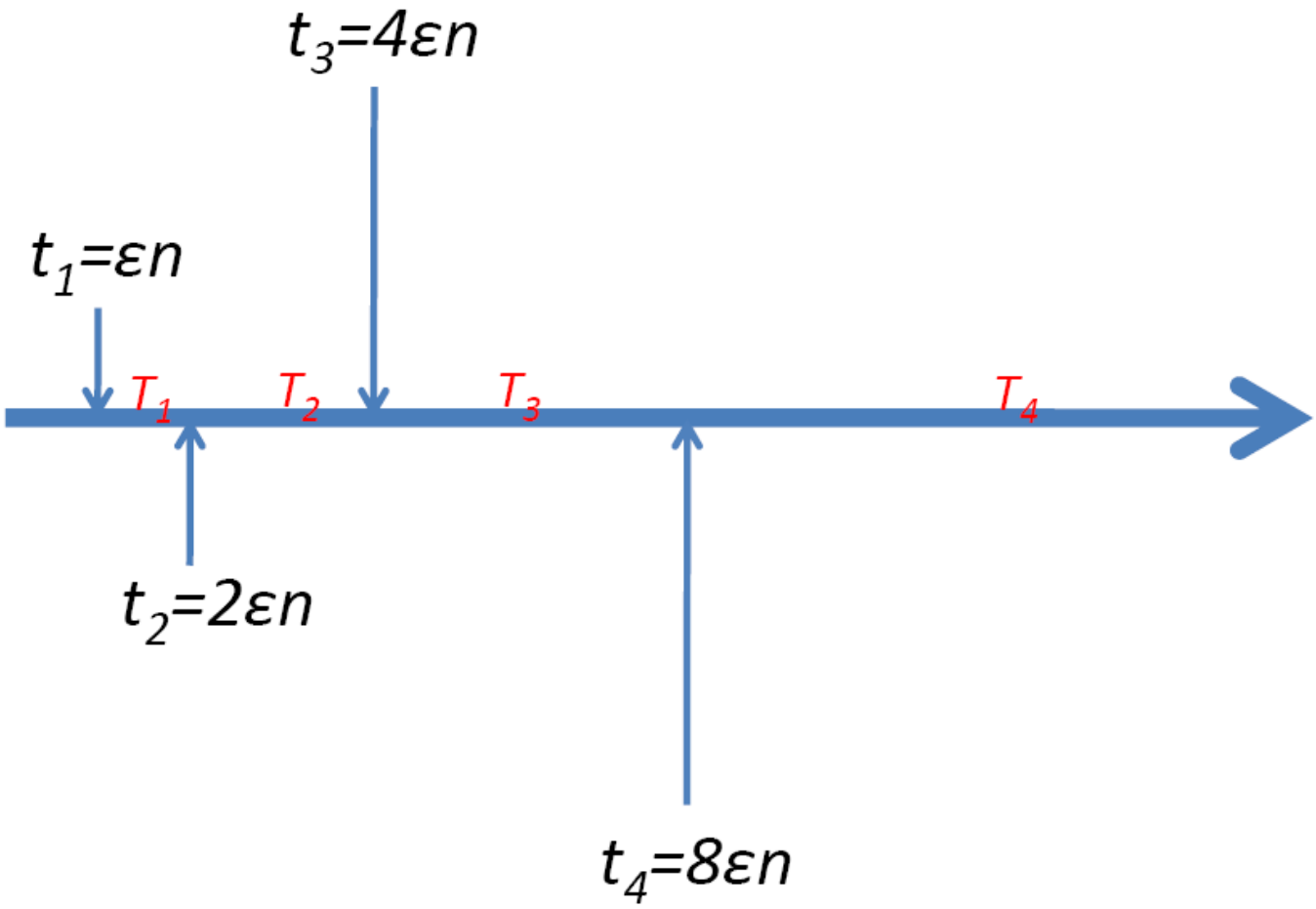
$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} r_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1 - h_\ell)\frac{\ell}{n} b_i & i = 1, ..., m \\ & 0 \leq x_t \leq 1 & t = 1, ..., \ell \end{array}$$

where

$$h_\ell = \epsilon \sqrt{\frac{n}{\ell}};$$

and this price vector is used to determine the allocation for the next immediate period.

# Geometric Pace/Grid of Price Updating



# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |

# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |
| Devanur et al [2009] | $OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$ | One-time |

# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |
| Devanur et al [2009] | $OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$ | One-time |
| Feldman et al [2010] | $b_{\min} \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$ | One-time |

# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |
| Devanur et al [2009] | $OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$ | One-time |
| Feldman et al [2010] | $b_{\min} \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$ | One-time |
| Agrawal et al [2010,14] | $b_{\min} \geq \frac{m \log n}{\epsilon^2}$ **or** $OPT \geq \frac{m^2 \log n}{\epsilon^2}$ | Dynamic |

# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |
| Devanur et al [2009] | $OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$ | One-time |
| Feldman et al [2010] | $b_{\min} \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$ | One-time |
| Agrawal et al [2010,14] | $b_{\min} \geq \frac{m \log n}{\epsilon^2}$ **or** $OPT \geq \frac{m^2 \log n}{\epsilon^2}$ | Dynamic |
| Molinaro/Ravi [2013] | $b_{\min} \geq \frac{m^2 \log m}{\epsilon^2}$ | Dynamic |

# Competitive-Ratio Result Summary on OLP

| | Sufficient Condition | Learning |
|---|---|---|
| Kleinberg [2005] | $b_{\min} \geq \frac{1}{\epsilon^2}$, for $m = 1$ | Dynamic |
| Devanur et al [2009] | $OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$ | One-time |
| Feldman et al [2010] | $b_{\min} \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$ | One-time |
| Agrawal et al [2010,14] | $b_{\min} \geq \frac{m \log n}{\epsilon^2}$ **or** $OPT \geq \frac{m^2 \log n}{\epsilon^2}$ | Dynamic |
| Molinaro/Ravi [2013] | $b_{\min} \geq \frac{m^2 \log m}{\epsilon^2}$ | Dynamic |
| **Kesselheim et al** [2014] | $b_{\min} \geq \frac{\log m}{\epsilon^2}$ | Dynamic* |
| **Gupta/Molinaro** [2014] | $b_{\min} \geq \frac{\log m}{\epsilon^2}$ | Dynamic* |
| **Agrawal/Devanur** [2014] | $b_{\min} \geq \frac{\log m}{\epsilon^2}$ | Dynamic* |

Table: Competitive-Ratio Progresses on OLP

Dynamic*: resolve LP or update prices at each time point.

# Dual Convergence: Equivalent Form of the Dual I

Recall the dual problem

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^{n} s_j \\ \text{s.t.} & s_j \geq r_j - \mathbf{a}_j^\top \mathbf{p}, \quad j = 1, ..., n \\ & \mathbf{p}, \mathbf{s} \geq 0 \end{array}$$

# Dual Convergence: Equivalent Form of the Dual I

Recall the dual problem

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^n s_j \\ \text{s.t.} & s_j \geq r_j - \mathbf{a}_j^\top \mathbf{p}, \quad j = 1, ..., n \\ & \mathbf{p}, \mathbf{s} \geq 0 \end{array}$$

Equivalently, by removing $s_j$'s,

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^n \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+ \\ \text{s.t.} & \mathbf{p} \geq 0 \end{array}$$

$(\cdot)^+$ is the positive-part function.

# Equivalent Form of the Dual II

Normalize the objective (note that $\mathbf{b} = n\mathbf{d}$)

$$\min_{\mathbf{p}\geq\mathbf{0}} \; f_n(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{n}\sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

# Equivalent Form of the Dual II

Normalize the objective (note that $\mathbf{b} = n\mathbf{d}$)

$$\min_{\mathbf{p}\geq\mathbf{0}} \; f_n(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{n}\sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

Implication for online LP:

- At time $t$, one can solve $f_t(\mathbf{p})$ (based on the observed samples) to obtain an estimate of $\mathbf{p}_n^*$ (based on all samples)

$$\min_{\mathbf{p}\geq\mathbf{0}} \; f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t}\sum_{j=1}^{t} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

# Equivalent Form of the Dual II

Normalize the objective (note that $\mathbf{b} = n\mathbf{d}$)

$$\min_{\mathbf{p}\geq\mathbf{0}} \; f_n(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{n}\sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

Implication for online LP:

- At time $t$, one can solve $f_t(\mathbf{p})$ (based on the observed samples) to obtain an estimate of $\mathbf{p}_n^*$ (based on all samples)

$$\min_{\mathbf{p}\geq\mathbf{0}} \; f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t}\sum_{j=1}^{t} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

Implication for efficiently solving offline LP:

- Perform projected sub-gradient descent for $f_n(\mathbf{p})$

# Understanding $f_n(\mathbf{p})$

Under the stochastic input model, when the column-coefficient pair $(r_j, \mathbf{a}_j)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$, define the stochastic program Define the stochastic program

$$\begin{array}{lll} \min & f(\mathbf{p}) := & \mathbf{d}^\top \mathbf{p} + \mathbb{E}_{(r,\mathbf{a})\sim\mathcal{P}} \left[ (\mathbf{r} - \mathbf{a}^\top \mathbf{p})^+ \right] \\ \text{s.t.} & & \mathbf{p} \geq \mathbf{0}, \end{array}$$

# Understanding $f_n(\mathbf{p})$

Under the stochastic input model, when the column-coefficient pair $(r_j, \mathbf{a}_j)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$, define the stochastic program Define the stochastic program

$$\begin{array}{lll} \min & f(\mathbf{p}) := & \mathbf{d}^\top \mathbf{p} + \mathbb{E}_{(r,\mathbf{a})\sim\mathcal{P}} \left[(\mathbf{r} - \mathbf{a}^\top \mathbf{p})^+\right] \\ \text{s.t.} & & \mathbf{p} \geq \mathbf{0}, \end{array}$$

Observation: $f_t(\mathbf{p})$ or $f_n(\mathbf{p})$ is a *sample average approximation* for $f(\mathbf{p})$.

Denote (stationary) $\mathbf{p}^*$ as the optimal solution of $f(\mathbf{p})$

# Convergence Result of $\mathbf{p}_n$

## Theorem (Dual Convergence, Li & Y 2019)

*Under moderate conditions that guarantees a local strong convexity of $f(\mathbf{p})$ around $\mathbf{p}^*$, there exists a constant $C$ such that*

$$\mathbb{E}\|\mathbf{p}_n^* - \mathbf{p}^*\|_2^2 \leq \frac{Cm \log\log n}{n}$$

*holds for all $n > m$ and $\mathcal{P} \in \Xi$.*

$L_2$ convergence (stronger convergence) for the dual optimal solution

Heuristically,

$$\mathbf{p}_n^* \approx \mathbf{p}^* + \frac{1}{\sqrt{n}} \cdot \textbf{Noise}$$

# Table of Contents

# Algorithmic Summaries for OLP

- "Offline" optimal $\mathbf{p}_n^*$ is determined by

$$\min_{\mathbf{p}\geq 0} \quad f_n(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{n}\sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

- At time $t$, OLP algorithms in literature essentially compute $\mathbf{p}_t^*$

$$\min_{\mathbf{p}\geq 0} \quad f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t}\sum_{j=1}^{t} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

- Dual convergence implies both $\mathbf{p}_t^*$ and $\mathbf{p}_n^*$ are close to $\mathbf{p}^*$
- At time $t$, we can decide the value of $x_t$ by setting $\mathbf{p} = \mathbf{p}_t^*$ in the decision rule

$$x_t = \begin{cases} 1, & r_t > \mathbf{a}_t^\top \mathbf{p} \\ 0, & r_t < \mathbf{a}_t^\top \mathbf{p} \end{cases}$$

# Sub-Gradient of the Dual

Instead of finding the optimal solution of $f_t(\mathbf{p})$, we perform sub-gradient descent

$$\min_{\mathbf{p}\geq 0} \quad f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t}\sum_{j=1}^{t} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

At time $t$, the sub-gradient constructed from the new observation is

$$\partial_{\mathbf{p}} \left(\mathbf{d}^\top \mathbf{p} + \left(r_t - \mathbf{a}_t^\top \mathbf{p}\right)^+\right)\Big|_{\mathbf{p}=\mathbf{p}_t} = \mathbf{d} - \mathbf{a}_t I(r_t > \mathbf{a}_t^\top \mathbf{p})\Big|_{\mathbf{p}=\mathbf{p}_t}$$
$$= \mathbf{d} - \mathbf{a}_t x_t$$

where $\mathbf{p}_t$ is the current dual price at time $t$

# Simple Online Algorithm for Solving Binary OLP

1: Input: $\mathbf{d}(= \mathbf{b}/n)$
2: Initialize $\mathbf{p}_1 = \mathbf{0}$
3: For $t = 1, 2, ..., n$
4: Set

$$x_t = \begin{cases} 1, & \text{if } r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{if } r_t \leq \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$$

5: Compute

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \gamma_t (\mathbf{a}_t x_t - \mathbf{d}) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$

6: Output $\mathbf{x} = (x_1, ..., x_n)$

# Remarks on the Simple Online (SO) Algorithm

- The algorithm is a first-order online algorithm and it does not involve any matrix inversion and it performs $O(mn)$ or $O(NNZ)$ (number of non-zero entries of the input data) operations
- $\gamma_t$ is the step size and it is chosen to be $\frac{1}{\sqrt{n}}$ (or $\frac{1}{\sqrt{t}}$) in the following analyses
- The algorithm does not require any prior knowledge on the boundedness constants $\bar{a}$ and $\bar{r}$

# Remarks on the Simple Online (SO) Algorithm

- The algorithm is a first-order online algorithm and it does not involve any matrix inversion and it performs $O(mn)$ or $O(NNZ)$ (number of non-zero entries of the input data) operations
- $\gamma_t$ is the step size and it is chosen to be $\frac{1}{\sqrt{n}}$ (or $\frac{1}{\sqrt{t}}$) in the following analyses
- The algorithm does not require any prior knowledge on the boundedness constants $\bar{a}$ and $\bar{r}$

The algorithm works for both the stochastic input model and the random permutation model. In the following, we will analyze the algorithm performance under these two models separately.

# Objective Regret and Constraint Violation

Recall

Worst-case regret/optimality gap:

$$\sup_{\mathcal{P}} \mathbb{E}_{\mathcal{P}}[R_n^* - R_n]$$

Worst-case constraint violation

$$\sup_{\mathcal{P}} \mathbb{E}_{\mathcal{P}}[v(\mathbf{x})] = \sup_{\mathcal{P}} \mathbb{E}_{\mathcal{P}} \left[ \| (A\mathbf{x} - \mathbf{b})^+ \|_2 \right]$$

where $\mathbf{x}$ is the output of the simple online (SO) algorithm

# Boundedness of $\mathbf{p}_t$ under SO Algorithm

## Lemma (Li, Sun, & Y (2020))

*If the step size $\gamma_t \leq 1$ for $t = 1, ..., n$ in the SO algorithm, then*

$$\|\mathbf{p}_t\|_2 \leq \frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})$$

*with probability 1 for $t = 1, ..., n$, where $\mathbf{p}_t$'s are specified by the SO algorithm.*

# Boundedness of $\mathbf{p}_t$ under SO Algorithm

## Lemma (Li, Sun, & Y (2020))

*If the step size $\gamma_t \leq 1$ for $t = 1, ..., n$ in the SO algorithm, then*

$$\|\mathbf{p}_t\|_2 \leq \frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})$$

*with probability 1 for $t = 1, ..., n$, where $\mathbf{p}_t$'s are specified by the SO algorithm.*

Remarks:

- The boundedness is crucial for the algorithm analysis
- It is automatically guaranteed by the algorithm. In other words, no explicit projection (which could be computationally costly) to certain bounded region is required for the algorithm.

# Performance Analysis for Stochastic Input Model

## Theorem (Li, Sun, & Y (2020))

*With step size* $\gamma_t = 1/\sqrt{n}$, *the regret and expected constraint violation of the algorithm satisfy*

$$\mathbb{E}[R_n^* - R_n] \leq m(\bar{a} + \bar{d})^2\sqrt{n}$$

$$\mathbb{E}\left[v(\mathbf{x})\right] \leq \left(\frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})\right)\sqrt{n}.$$

*hold for all* $m, n \in \mathbb{N}^+$ *and distribution* $\mathcal{P} \in \Xi$.

# Performance Analysis for Stochastic Input Model

## Theorem (Li, Sun, & Y (2020))

*With step size* $\gamma_t = 1/\sqrt{n}$, *the regret and expected constraint violation of the algorithm satisfy*

$$\mathbb{E}[R_n^* - R_n] \leq m(\bar{a} + \bar{d})^2\sqrt{n}$$

$$\mathbb{E}\left[v(\mathbf{x})\right] \leq \left(\frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})\right)\sqrt{n}.$$

*hold for all* $m, n \in \mathbb{N}^+$ *and distribution* $\mathcal{P} \in \Xi$.

Remark:

- Both are on the order of $m\sqrt{n}$

# Proof Sketch

For the proof of the **regret**, it utilizes the structure of the LP and largely mimics the analyses of the stochastic/online gradient descent algorithm

For the proof of the **constraint violation**, the updating equation tells

$$\mathbf{p}_{t+1} \geq \mathbf{p}_t + \frac{1}{\sqrt{n}}(\mathbf{a}_t x_t - \mathbf{d})$$

where the inequality is element-wise. Therefore,

$$\begin{aligned} \sum_{t=1}^{n} \mathbf{a}_t x_t &\leq n\mathbf{d} + \sum_{t=1}^{n} \sqrt{n}(\mathbf{p}_{t+1} - \mathbf{p}_t) \\ &\leq \mathbf{b} + \sqrt{n}\mathbf{p}_{n+1} \end{aligned}$$

The proof is completed by plugging in the boundedness of $\mathbf{p}_t$'s

# Challenge of the Random Permutation Model

- Under the stochastic input model, $(r_j, \mathbf{a}_j)$'s are i.i.d. drawn from $\mathcal{P}$. In the online context, it is easier to infer the future samples based on the past observations.

- Under the random permutation model, it is more challenging to infer the future samples based on the past observations. Because the values of the entries $(r_j, \mathbf{a}_j)$ can be adversarially chosen at the first place.

# Good News (I) – Boundedness of $\mathbf{p}_t$

## Lemma (Same as the Stochastic Input Model)

*If the step size $\gamma_t \leq 1$ for $t = 1, ..., n$ in SO Algorithm, then*

$$\|\mathbf{p}_t\|_2 \leq \frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})$$

*with probability 1 for $t = 1, ..., n$, where $\mathbf{p}_t$'s are specified by SO Algorithm.*

# Good News (I) – Boundedness of $\mathbf{p}_t$

## Lemma (Same as the Stochastic Input Model)

*If the step size $\gamma_t \leq 1$ for $t = 1, ..., n$ in SO Algorithm, then*

$$\|\mathbf{p}_t\|_2 \leq \frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})$$

*with probability 1 for $t = 1, ..., n$, where $\mathbf{p}_t$'s are specified by SO Algorithm.*

The boundedness of $\mathbf{p}_t$ still holds under the random permutation model. Its proof only relies on the boundedness of $(r_j, \mathbf{a}_j)$'s but not how they are generated

# Good News (II) – Optimal Objective Value Bound

Denote the optimal objective value as $R_t^*$ for the scaled LP:

$$\max \quad \sum_{j=1}^{t} r_j x_j \qquad (t\text{-Scaled-LP})$$

$$\text{s.t.} \quad \sum_{j=1}^{t} \mathbf{a}_j x_j \le t \cdot \mathbf{d}, \ \mathbf{0} \le \mathbf{x} \le \mathbf{e}.$$

## Proposition (Li, Sun, & Y (2020))

*Under the random permutation model, for* $t > \max\{16\bar{a}^2, e^{16\bar{a}^2}, e\}$, *the following inequality holds*

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] \ge \frac{1}{n}R_n^* - \frac{m\bar{r}}{n} - \frac{\bar{r}\log t}{\underline{d}\sqrt{t}} - \frac{m\bar{r}}{t}.$$

*for all* $n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_D$.

# Good News (II) – Optimal Objective Value Bound

Denote the optimal objective value as $R_t^*$ for the scaled LP:

$$\max \quad \sum_{j=1}^{t} r_j x_j \qquad (t\text{-Scaled-LP})$$

$$\text{s.t.} \quad \sum_{j=1}^{t} \mathbf{a}_j x_j \le t \cdot \mathbf{d}, \ \mathbf{0} \le \mathbf{x} \le \mathbf{e}.$$

## Proposition (Li, Sun, & Y (2020))

*Under the random permutation model, for* $t > \max\{16\bar{a}^2, e^{16\bar{a}^2}, e\}$, *the following inequality holds*

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] \ge \frac{1}{n}R_n^* - \frac{m\bar{r}}{n} - \frac{\bar{r}\log t}{d\sqrt{t}} - \frac{m\bar{r}}{t}.$$

*for all* $n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_D$.

Remark: The proposition relates $R_t^*$ with the offline optimal $R_n^*$.

# Differences of the Two Input Models

Under the stochastic input model,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] = \frac{1}{n}\mathbb{E}\left[R_n^*\right]$$

Under the random permutation model, heuristically,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] \geq \frac{1}{n}R_n^* - O\left(\frac{\log t}{t}\right)$$

# Differences of the Two Input Models

Under the stochastic input model,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] = \frac{1}{n}\mathbb{E}\left[R_n^*\right]$$

Under the random permutation model, heuristically,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] \geq \frac{1}{n}R_n^* - O\left(\frac{\log t}{t}\right)$$

The quantity $O\left(\frac{\log t}{t}\right)$ can be interpreted as an information toll for the relaxation of the data generation mechanism

Here expectation is not taken for the offline optimal $R_n^*$ under the random permutation model because it is a deterministic value given the data set $\mathcal{D} = \{(r_j, \mathbf{a}_j)\}_{j=1}^n$

# Performance Analysis under the RP Model

## Theorem (Li, Sun, & Y (2020))

*With the step size* $\gamma_t = \frac{1}{\sqrt{n}}$ *for* $t = 1, ..., n$, *the regret and expected constraint violation of the SO algorithm satisfy*

$$\begin{aligned} R_n^* - \mathbb{E}[R_n] &\leq O\left((m + \log n)\sqrt{n}\right) \\ \mathbb{E}\left[v(\mathbf{x})\right] &\leq O(m\sqrt{n}). \end{aligned}$$

*for all* $m, n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_D$.

# Performance Analysis under the RP Model

## Theorem (Li, Sun, & Y (2020))

*With the step size* $\gamma_t = \frac{1}{\sqrt{n}}$ *for* $t = 1, ..., n$, *the regret and expected constraint violation of the SO algorithm satisfy*

$$\begin{aligned} R_n^* - \mathbb{E}[R_n] &\leq O\left((m + \log n)\sqrt{n}\right) \\ \mathbb{E}\left[v(\mathbf{x})\right] &\leq O(m\sqrt{n}). \end{aligned}$$

*for all* $m, n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_D$.

Remark:

- Extra term $O(\sqrt{n}\log n)$ compared to the stochastic input case

# Proof Sketch

1. Relate the term $(\mathbf{d} - \mathbf{a}_t x_t)^\top \mathbf{p}_t$ with $R^*_{n-t}$ (the optimal objective value of the scaled LP for the remaining $n - t$ time periods)
2. Utilize the relationship between $R^*_{n-t}$ and $R^*_n$ (via the last proposition)
3. The constraint violation part is identical to the previous analysis under the stochastic input model.

# Regret Analyses with Permutational Rademacher Complexity

We have established relation between $R_t^*$ and $R_n^*$ under the random permutation model. Is there a more general and systematic way to relate the past and future data?

Specifically, many OLP algorithms utilize past observations to find a dual price $\mathbf{p}_t$ at each time $t$ and make decision $x_t$ based on $\mathbf{p}_t$.

Question: If $\mathbf{p}_t$ performs well in the past data $\{(r_j, \mathbf{a}_j)\}_{j=1}^t$, will it also perform well in the future data $\{(r_j, \mathbf{a}_j)\}_{j=t+1}^n$, given $\{(r_j, \mathbf{a}_j)\}_{j=1}^n$ formed by a random permutation?

# Machine Learning Perspective

- Informally, we can view the decision of $x_t$ as a binary classification task.
- There is an optimal (or true) classifier determined by the offline dual optimal solution $\mathbf{p}_n^*$
- At time $t$, we randomly split the data set in two groups, one with $t$ samples and the other with $n - t$ samples
- The performance of $\mathbf{p}_t$ on the first $t$ samples should be close to its performance on the remaining $n - t$ samples
- This motivates the notion of *Permutational Rademacher Complexity* (Tolstikhin et al. 2015) originally used for *transductive learning*

# Past is "Similar" to Future

## Proposition (Li, Sun, & Y (2020))

*If* $\{(r_j, \mathbf{a}_j)\}_{j=1}^n$ *is a random permutation of dataset* $\mathcal{D}$*, then*

$$\mathbb{E}\left[\sup_{\mathbf{p}\geq 0}\left|\frac{1}{n-t}\sum_{j=t+1}^{n} r_j I(r_j > \mathbf{a}_j^\top \mathbf{p}) - \frac{1}{t}\sum_{j=1}^{t} r_j I(r_j > \mathbf{a}_j^\top \mathbf{p})\right|\right] \leq \frac{2\sqrt{2\bar{r}^2 m \log n}}{\sqrt{\min\{t, n-t\}}}$$

$$\mathbb{E}\left[\sup_{\mathbf{p}\geq 0}\left|\frac{1}{n-t}\sum_{j=t+1}^{n} a_{ij} I(r_j > \mathbf{a}_j^\top \mathbf{p}) - \frac{1}{t}\sum_{j=1}^{t} a_{ij} I(r_j > \mathbf{a}_j^\top \mathbf{p})\right|\right] \leq \frac{2\sqrt{2\bar{a}^2 m \log n}}{\sqrt{\min\{t, n-t\}}}$$

*for any* $1 \leq i \leq m$, $1 \leq t \leq n-1$.

The difference between objective values and constraint consumption of the past and the future (under the same dual price $\mathbf{p}$) is no greater than

$$O\left(\frac{\sqrt{m \log n}}{\sqrt{\min\{t, n-t\}}}\right)$$

# Technical Remarks

The proof is largely adapted from the analysis of Permutational Rademacher Complexity in (Tolstikhin et al. 2015)

This provides a powerful tool to analyze the algorithm performance under the random permutation model.

For two earlier OLP algorithms, we establish

- Sublinear regret bound than competitiveness ratio
- With no requirement on the non-negativeness of $r_j$'s and $a_{ij}$'s

# Algorithm 1 – Agrawal et al. 2010, 14

1: Input: $\mathbf{d}$
2: Let $\mathbf{p}_1 = \mathbf{0}$
3: For $t = 1, 2, ..., n$
4: Set $x_t = \begin{cases} 1, & \text{if } r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{if } r_t \leq \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$
5: Solve for the optimal dual solution $\mathbf{p}_{t+1}$ of the scaled primal LP:

$$\begin{aligned} \max \quad & \sum_{j=1}^{t} r_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{t} \mathbf{a}_j x_j \leq t \cdot \mathbf{d}, \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}. \end{aligned}$$

(The algorithm is a refined version (Li & Ye 2019) of the original algorithm)

# Algorithm 2 – Kesselheim et al. 2014

1: Input: $\mathbf{d}$
2: Let $\mathbf{p}_1 = \mathbf{0}$
3: For $t = 1, 2, ..., n$
4: Solve for the optimal dual solution $\tilde{\mathbf{x}}^{(t)} = (\tilde{x}_1^{(t)}, ..., \tilde{x}_t^{(t)})$ from

$$\begin{aligned} \max \quad & \sum_{j=1}^{t} r_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{t} \mathbf{a}_j x_j \leq t \cdot \mathbf{d}, \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}. \end{aligned}$$

5: Set

$$x_t = \begin{cases} 1, & \text{with probability } \tilde{x}_t^{(t)} \\ 0, & \text{with probability } 1 - \tilde{x}_t^{(t)} \end{cases}$$

# Remarks on Algorithm 1 and Algorithm 2

- Algorithm 2 (primal-beats-dual) can be viewed as a primal version of Algorithm 1 (dynamic learning algorithm)
- Both algorithms solve scaled version of LP frequently so they are more computationally costly than our SO algorithm
- Both of the previous algorithm analyses require non-negativeness for the input of the LP

# Performance Analyses

## Theorem (Li, Sun, & Y (2020))

*Under the random permutation model, the regret and expected constraint violation of the two algorithms (Agrawal et al. 2014, Kesselheim et al. 2014) both satisfy*

$$\begin{aligned} R_n^* - \mathbb{E}[R_n] &\leq O\left(\sqrt{mn}\right) \\ \mathbb{E}[v(\mathbf{x})] &\leq O(\sqrt{mn}\log n) \end{aligned}$$

*for all* $m, n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_{\mathcal{D}}$.

# Performance Analyses

## Theorem (Li, Sun, & Y (2020))

*Under the random permutation model, the regret and expected constraint violation of the two algorithms (Agrawal et al. 2014, Kesselheim et al. 2014) both satisfy*

$$\begin{aligned} R_n^* - \mathbb{E}[R_n] &\leq O\left(\sqrt{mn}\right) \\ \mathbb{E}[v(\mathbf{x})] &\leq O(\sqrt{mn}\log n) \end{aligned}$$

*for all* $m, n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_{\mathcal{D}}$.

Remark: Compared with the SO algorithm, the upper bound on the regret and constraint violation are reduced by a factor of $\sqrt{m}$ with the price of more computation cost.

# Proof Sketch

1. Apply the "past-similar-to-future" proposition, we have that the performance of $\mathbf{p}_t$ at time $t+1$ is similar to the performance of $\mathbf{p}_t$ over first $t$ samples.
2. Apply the proposition on the relation between $R_t^*$ and $R_n^*$, we have that the optimal value obtained by $\mathbf{p}_t^*$ over first $t$ samples is similar to the offline optimal value.
3. Combine the first two parts at each time point $t$, we can upper bound the difference between the performance of $\mathbf{p}_t$ at time $t+1$ and the offline optimal value. Then, we can establish the regret bound.
4. The constraint violation analysis is trickier than the SO algorithm. It involves some martingale-based argument.

# Extension to Multi-dimensional Online LP

Consider a general OLP with following form:

$$\text{Multi-ILP} \quad \max \quad \sum_{j=1}^{n} \mathbf{r}_j^\top \mathbf{x}_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} A_j \mathbf{x}_j \le \mathbf{b}$$

$$\mathbf{e}^\top \mathbf{x}_j \le 1, \quad \mathbf{x}_j \in \{0,1\}^k, \quad j = 1, \ldots, n$$

where $\mathbf{r}_j = \{r_{jl}\}_{l=1}^k \in \mathbf{R}^k$, $A_j = \{\mathbf{a}_{jl}\}_{l=1}^k \in \mathbf{R}^{m \times k}$, $\mathbf{a}_{jl} = \{a_{ijl}\}_{i=1}^m \in \mathbf{R}^m$, and $\mathbf{b} = \{b_i\}_{i=1}^m \in \mathbf{R}^m$.

- $(\mathbf{r}_j, A_j)$ are revealed sequentially
- Decide $\mathbf{x}_j$'s sequentially (each time a batch of decision variables)
- Conform to constraints

# SO Algorithm for Multi-dimensional ILP

- Similar to the one-dimensional OLP setting, the primal optimal solution is largely determined by the dual optimal $\mathbf{p}_n^*$ of the general OLP.
- Idea: At each time $t$, we use the dual price $\mathbf{p}_t$ to assign a value for each dimension of $\mathbf{x}_t$. Unless all the dimensions have negative assigned values, we select the dimension with largest assigned value and set the corresponding decision variable to be 1; otherwise, $\mathbf{x}_t = \mathbf{0}$.

# SO Algorithm for Multi-dimensional ILP

1: Input: $\mathbf{d}$

2: Initialize $\mathbf{p}_1 = \mathbf{0}$

3: For $t = 1, ..., n$

4: Set $v_t = \max_{l=1,...,k} r_{tl} - \mathbf{a}_{tl}^\top \mathbf{p}_t$

5: If $v_t > 0$, pick an index $l_t$ randomly from the non-empty set $\left\{ l : r_{tl} - \mathbf{a}_{tl}^\top \mathbf{p}_t = v_t \right\}$ set $x_{tl} = \begin{cases} 1, & l = l_t \\ 0, & \text{otherwise} \end{cases}$; else set $\mathbf{x}_t = \mathbf{0}$

6: Compute

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \frac{1}{\sqrt{n}} (A_t \mathbf{x}_t - \mathbf{d}) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$

7: Output: $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$

# Performance Analysis

## Theorem (Li, Sun, & Ye (2020))

*Under the stochastic input model and the random permutation model, the regret and constraint violation of multi-dimensional SO Algorithm are the same as the one-dimensional SO algorithm, respectively.*

Remarks:

- The problem covers a wider range of applications
- The regret and constraint violation are not dependent on the dimension $k$ of $\mathbf{x}_t$'s

# Table of Contents

# Algorithm 3 – Simple Feasible Online Algorithm

The first variant is aimed to output feasible online solutions. Compared with SO Algorithm, this algorithm sets $x_t = 1$ only when the constraints permit.

1: ... (Same as in SO)
2: If any budget constraint is violated, reset $x_t = 0$.
3: Output: $\mathbf{x} = (x_1, ..., x_n)$

- Empirically, the algorithm performs well (as we will see shortly)
- Without additional assumptions on $(r_j, \mathbf{a}_j)$'s, the analysis of this feasible version of the SO Algorithm could be challenging
- An alternative way of converting infeasible solution to feasible solution for offline LP is using rounding, which is provable that the feasible solution $\tilde{\mathbf{x}}$ achieves a regret $O(m\sqrt{n})$

# Algorithm 4 – Simple Adaptive Algorithm I

Another variant takes into account the constraint consumption while doing the subgradient descent. The intuition is similar to the (non-stationary) action-history-dependent algorithm in (Li & Y, 2019).

1: Input: $\mathbf{d}$
2: Initialize $\mathbf{p}_1 = \mathbf{0}$, $\mathbf{b}_0 = \mathbf{b}$
3: For $t = 1, ..., n$
4: Set $x_t = \begin{cases} 1, & r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{otherwise} \end{cases}$
5: Update $\mathbf{b}_t = \mathbf{b}_{t-1} - \mathbf{a}_t x_t$
6: Compute

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \gamma_t \left( \mathbf{a}_t x_t - \frac{\mathbf{b}_t}{n-t} \right) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$

7: Output: $\mathbf{x} = (x_1, ..., x_n)$

# Algorithm 4 – Simple Adaptive Algorithm II

- In SO Algorithm, the updating step is

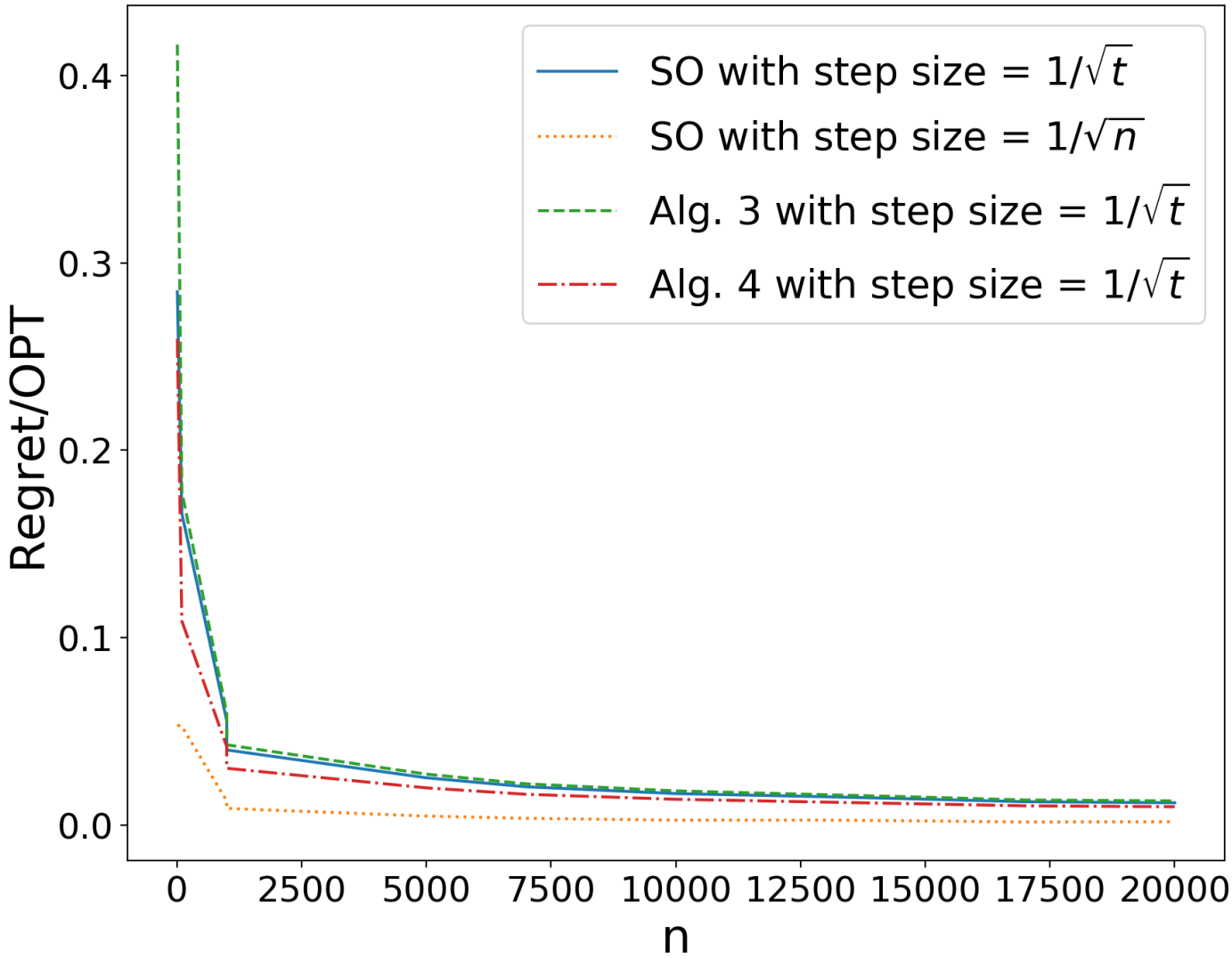
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \gamma_t \left( \mathbf{a}_t x_t - \mathbf{d} \right).$$

  In Simple Nonstationary Algorithm, the step is

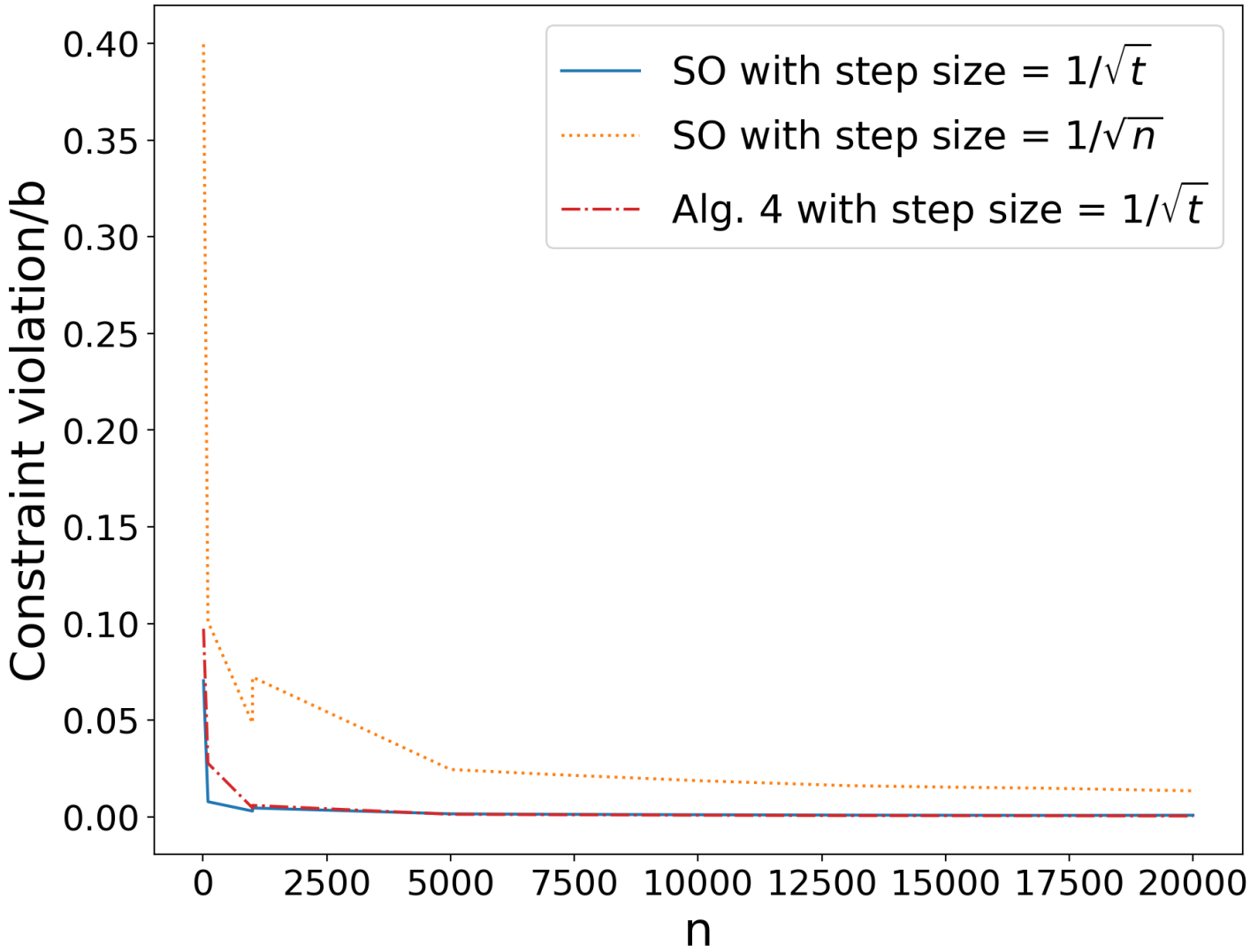
$$\mathbf{p}_{t+1} = \mathbf{p}_t + \gamma_t \left( \mathbf{a}_t x_t - \frac{\mathbf{b}_t}{n-t} \right)$$

- $\mathbf{b}_t/(n-t)$ is the average available budgets for the remaining $(n-t)$ decision variables
- Empirically, it performs better. Theoretical analysis remains an open question

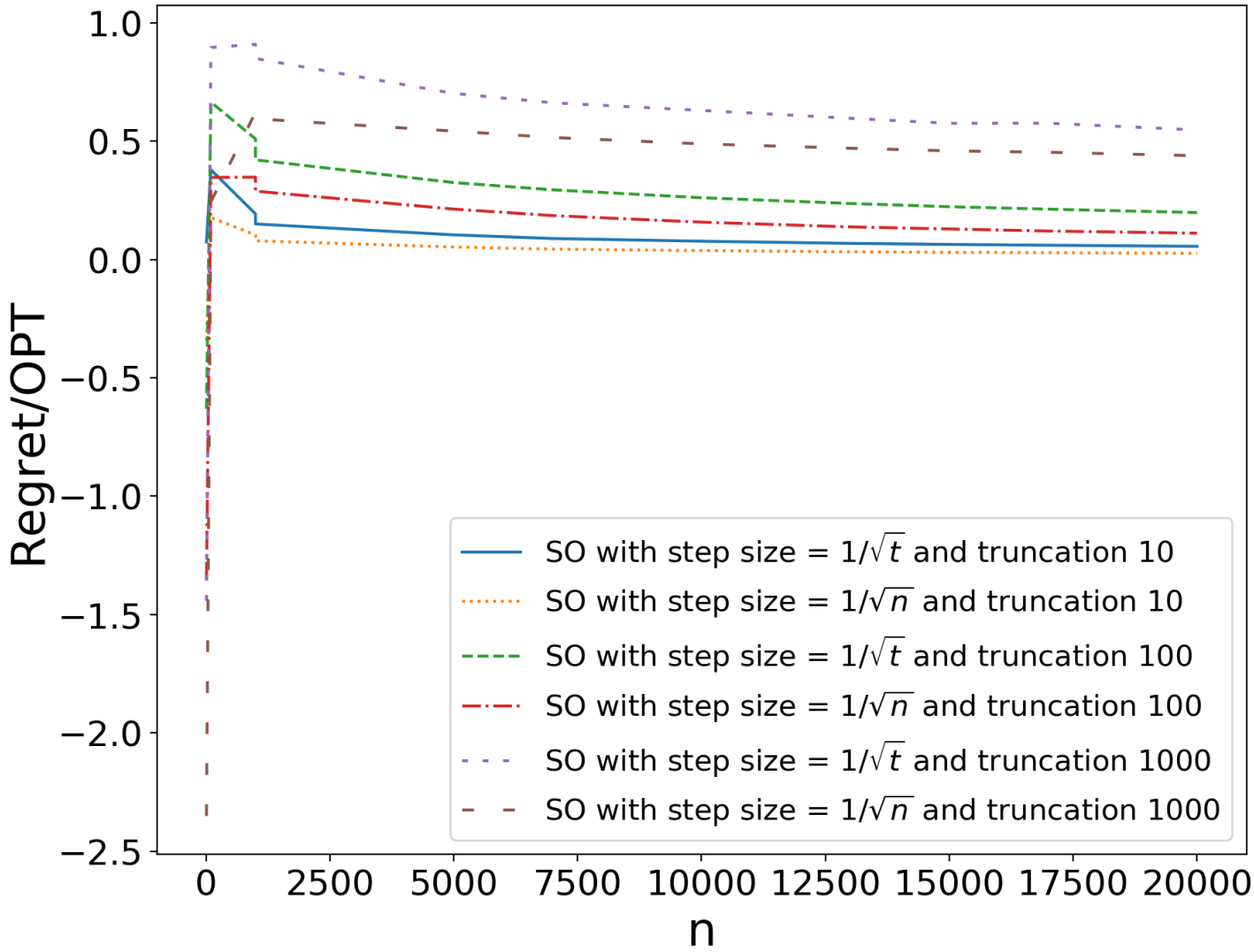
# Uniform i.i.d. Input with $m = 10$: Regret



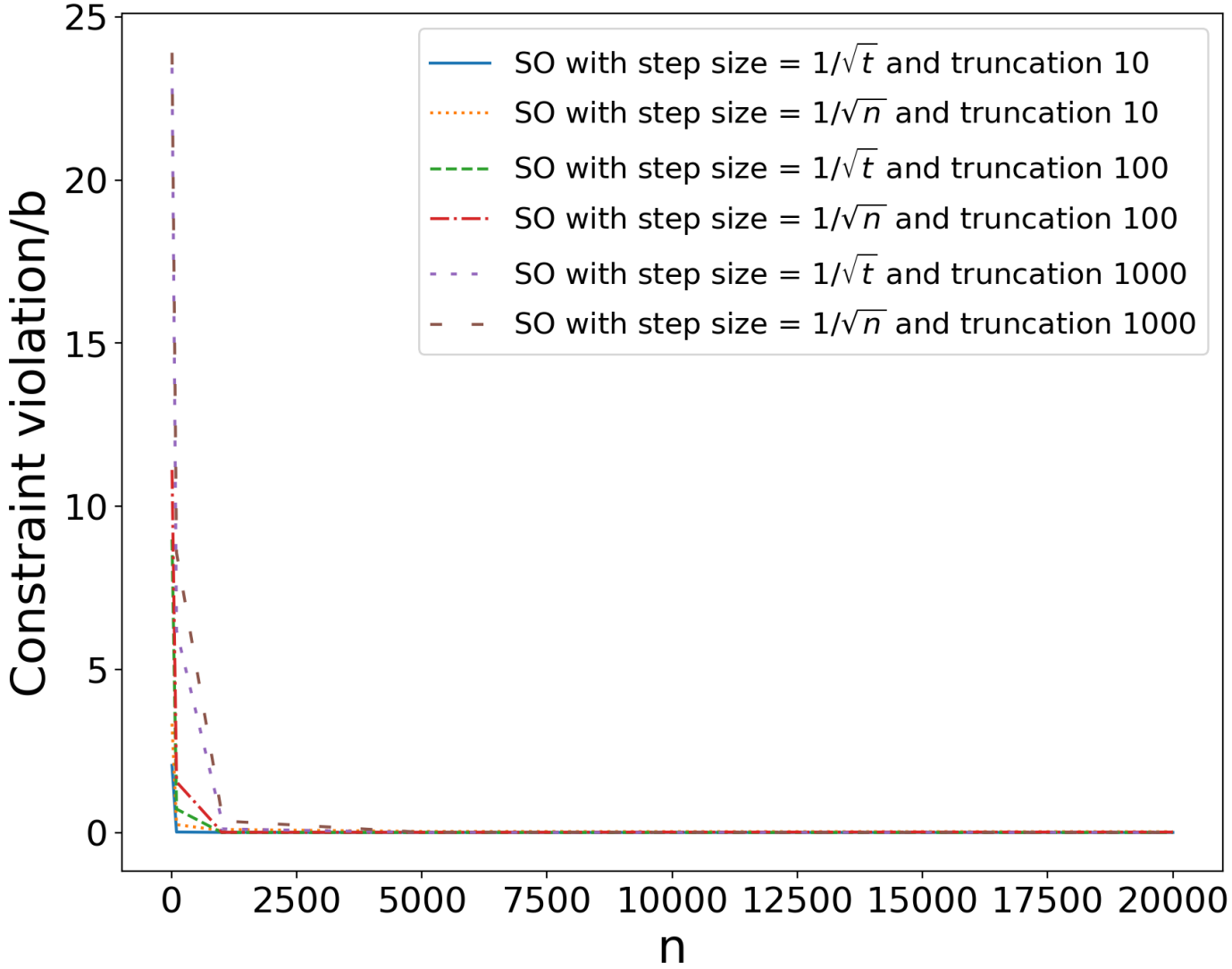
# Uniform i.i.d. Input: Budget Violation



# Cauchy i.i.d. input: Regret



# Cauchy i.i.d. input: Budget Violation



# Offline LP Experiment Setup

- Two groups of experiments are presented:
  - Worst-case example in (Agrawal, Wang, & Y 2014)
  - Multi-knapsack benchmark problem instances
- Gurobi is used as a benchmark for solving binary LPs
- Our SO algorithm is implemented on Matlab without much code optimization. (The for-loop is slow!)
- The competitiveness ratio is computed against the optimal objective value of the relaxed LP. For each experiment, we run 100 simulation trials and report the average results.
- For all the algorithms, we terminate and set the remaining $x_t$ to be 0 when any of the resources is exhausted. In this way, all the online algorithms output feasible solutions. More analysis on feasibility is referred to our paper.

# Worst-Case Example in (Agrawal et al. 2014)

| | | Gurobi | Fast Alg. | Alg. 1 | Alg. 2 |
|---|---|---|---|---|---|
| $m = 8, n = 10^3$ | time | 0.082 | 0.015 | 126.684 | 126.696 |
| | Cmpt. Ratio | 100.0% | 99.7% | 89.8% | 97.9% |
| $m = 128, n = 10^4$ | time | 0.408 | 0.138 | 2338.149 | 2338.285 |
| | Cmpt. Ratio | 100.0% | 98.8% | 94.8% | 97.7% |
| $m = 1024, n = 10^5$ | time | 52.496 | 3.270 | $> 3000$ | $> 3000$ |
| | Cmpt. Ratio | 100.0% | 98.7% | | |
| $m = 4096, n = 10^5$ | time | 114.96 | 32.020 | $> 3000$ | $> 3000$ |
| | Cmpt. Ratio | 99.6% | 83.3% | | |
| $m = 4096, n = 2 \times 10^5$ | CPU time | 254.243 | 53.887 | $> 3000$ | $> 3000$ |
| | Cmpt. Ratio | 99.7% | 89.1% | | |

Alg. 1 (Dynamic Learning Algorithm): Agrawal et al. (2010, 14)

Alg. 2 (Primal-beats-dual): Kesselheim et al. (2014)

# Multi-knapsack Problem Instances

Chu and Beasley (1998), Drake et al. (2016)

| | | Gurobi | SO Alg. | Alg. 1 | Alg. 2 |
|---|---|---|---|---|---|
| $m = 5, n = 500$ | Time | 0.116 | 0.006 | 70 | 70 |
| | Cmpt. Ratio | 99.6% | 92.3% | 91.8% | 91.5% |
| $m = 10, n = 500$ | Time | 0.136 | 0.006 | 132 | 132 |
| | Cmpt. Ratio | 99.6% | 91.8% | 91.6% | 90.7% |
| $m = 30, n = 500$ | Time | 95.2 | 0.005 | 134 | 133 |
| | Cmpt. Ratio | 99.4% | 91.5% | 89.1% | 90.4% |
| $m = 10^3, n = 10^5$ | Time | 857 | 2.711 | $> 3000$ | $> 3000$ |
| | Cmpt. Ratio | 99.8% | 94.9% | | |
| $m = 3 \times 10^3, n = 2 \times 10^5$ | Time | 2799 | 40.21 | $> 3000$ | $> 3000$ |
| | Cmpt. Ratio | 93.8% | 84.7% | | |

# Results Interpretation

- The computation time of our algorithm is (unsurprisingly) much less than the other algorithms
- SO Alg. performs better than Alg. 1 and Alg. 2, though its theoretical upper bound has an extra $O(\sqrt{m})$ term. It remains an open question whether its current regret bound is tight in $m$.
- The performance of our algorithm deteriorates when $m$ increases

# Table of Contents

# Limitation of the Previous Algorithm/Analyses

- A crucial assumption is that the right-hand-side $\mathbf{b} = n\mathbf{d}$ scales linearly with $n$
- This assumption makes sense in many but not all application contexts. For some LP problems, one may have $\mathbf{b} = o(n)$
- Is there a remedy for this case? Also, we do not want to compromise the computational efficiency of SO algorithm

# A Variable-Replicating Algorithm (I)

For an LP problem,

$$\begin{array}{ll} \max & \sum_{j=1}^{n} r_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} \mathbf{a}_j x_j \leq \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}, \end{array}$$

Consider a new LP

$$\begin{array}{ll} \max & \sum_{j=1}^{n} \sum_{h=1}^{k} r_j x_{jh} \\ \text{s.t.} & \sum_{j=1}^{n} \sum_{h=1}^{k} \mathbf{a}_j x_{jh} \leq k\mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq 1. \end{array}$$

Algorithm: Solve the new LP with SO Algorithm and use $x_j = \frac{1}{k}(x_{j1} + ... + x_{jk})$ as the solution to the original LP.

Sample with $k$ replacements, similar to the "Boosting" idea...

# A Variable-Replicating Algorithm (II)

Remarks:

- By replacing $x_j = \frac{1}{k}(x_{j1} + ... + x_{jk})$, we scale up the right-hand-side with a factor of $k$. Also, the number of variables go up with a factor of $k$
- The solution may no longer be integer-valued but it could be helpful to solve offline LP problems
- Even for online setting, it presents a multi-rounds ($k$-rounds) resource allocation decision making process where each round allocates up to $1/k$ fraction-level for each bid.

# Performance Analyses of Variable-Replicating

Define $b_{\min}$ as the minimum amount of budget, i.e. $b_{\min} = \min_{i=1,\ldots,m} b_i$.

## Proposition

*Under the random permutation model, the variable-replicating algorithm finds a solution for the original LP that achieves a competitiveness ratio of* $(1 - \mathcal{O}(\varepsilon))$ *if* $\sqrt{k} b_{\min}^2 \geq \frac{n^{3/2} \log kn}{\varepsilon}$ *and* $\sqrt{k} b_{\min} \geq \frac{mn \log kn}{\varepsilon}$ *for any* $\varepsilon > 0$.

The proof comes from a direct application of performance analyses of SO algorithm under the random permutation model.

```
% A K-round Matlab Code on max c'x s.t. Ax\le b,0\le x \le 1:
function [x,y] = fastLP(A,c,b,K);
 [m,n] = size(A); d=b/n; step=1/sqrt(K*n); % set parameters
 x = zeros(n,1); y = zeros(m,1);  br= K*b; % set solutions
% start out loop
 for k=1:K;
  p=randperm(n); % randomly permute the variable order
% start inner loop
  for i=1:n,
   ii=p(i); aa=A(:,ii);
   xk = (sign(c(ii)-aa'*y)+1)/2; % set the primal increment
   y=max(0,y+step*(xk*aa-d)); % update the dual soluton
   if min(br-xk*aa) >= 0,
       br=br-xk*aa;
       x(ii)=x(ii)+xk;
   end;  % update the remaining inventory and primal sol
  end;
 end;
 x=x/K; % output final primal solution
```

# Numerical Experiment of Variable-Replicating

In this experiment, the right-hand-side $\mathbf{b} = O(\sqrt{n})$. The Gurobi solves the relaxed LP. The competitiveness ratio is reported against the optimal objective value of the relaxed LP.

| | | Gurobi | New Alg. |
|---|---|---|---|
| $m = 10^2, n = 10^4, k = 10$ | Time | 2.08 | 0.16 |
| | Ratio | 100% | 90.2% |
| $m = 10^2, n = 10^4, k = 10^2$ | Time | 1.92 | 1.39 |
| | Ratio | 100% | 95.1% |
| $m = 10^3, n = 10^5, k = 10$ | Time | 200.89 | 7.82 |
| | Ratio | 100% | 95.5% |
| $m = 10^3, n = 10^5, k = 10^2$ | Time | 175.78 | 79.40 |
| | Ratio | 100% | 98.0% |
| $m = 3 \times 10^3, n = 10^5, k = 10$ | Time | 1209.5 | 26.89 |
| | Ratio | 100% | 96.2% |

# Table of Contents

# Bounds in terms of $n$

- SO algorithm achieves $O(m\sqrt{n})$ regret under the stochastic input model and $O((m+\log n)\sqrt{n})$ regret under the random permutation model.

- Arlotto & Gurvich (2019) shows that the lower bound of the regret is $O(\sqrt{n})$ based on Kleinberg's (2017) multi-secretary example

- An open question is that with more assumptions on the model input $(r_j, \mathbf{a}_j)$'s, is it possible for a first-order algorithm to achieve better regret than $O(\sqrt{n})$

    - Li & Ye 2019 shows that $O(\log n)$ regret is achievable with stronger conditions on $(r_j, \mathbf{a}_j)$'s but with a computationally costly algorithm

# Bounds in terms of $m$

- SO algorithm achieves $O(m\sqrt{n})$ regret under the stochastic input model and $O((m+\log n)\sqrt{n})$ regret under the random permutation model.
- LP-solver based algorithms reduce a factor $\sqrt{m}$
- For packing LPs (coefficients all nonnegative), the lower bound is $\log(m)$ (Agrawal et al. 2014)
- What the lower bound would be on $m$ for general data?

# Other Forms of the Dual

For the LP in standard form, consider the dual with a barrier

$$\max_{\mathbf{p}} \quad \mathbf{b}^\top \mathbf{p} + \frac{1}{n} \sum_{j=1}^{n} \log\left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)$$

Start from an interior-point solution $\mathbf{p}^1$ and apply the simple algorithm – set at time $t$:

$$x_t = \frac{1}{n(r_t - \mathbf{a}_t^\top \mathbf{p}_t)}?$$

with a randomly permuted order?

Apply the process $k$ rounds using the $\mathbf{p}$ of the last round as the initial solution for the next round?

# More Future Research Directions

- Implementation in C+?
- Project $\mathbf{p}$ into an cube $\mathbf{0} \leq \mathbf{p} \leq \kappa$ where $\kappa$ is a known upper bound on the optimal dual prices?
- Analysis of the adaptive algorithm (Algorithm 4)?
- Variance Reduction?
- Sub-gradient with a batch size?
- More research on approximately solve large-scale offline binary LPs with fast algorithms?