LP Research Topics

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/~yyye

Online LP: An Example

	Bid 1($t = 1$)	Bid 2($t = 2$)	.....	Inventory($\mathbf{b}$)
Price(π_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

Online Linear Programming Model

The classical **offline** version of the above program can be formulated as a linear (integer) program as all information data would have arrived: compute x_t, $t = 1, ..., n$ and

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

Now we consider the **online or streamline** and **data-driven** version of this problem:

- We only know $\mathbf{b}$ and n at the start
- the bidder information is revealed sequentially along with the corresponding objective coefficient.
- an **irrevocable decision** must be made as soon as an order arrives.

Model Assumptions

Main Assumptions

- $0 \leq a_{it} \leq 1$, for all (i, t);
- $\pi_t \geq 0$ for all t
- The data $(\mathbf{a}_t, \pi_t)$ arrive in a **random order**.

Denote the offline LP **maximal value** by $OPT(A, \pi)$. We call an online algorithm $\mathcal{A}$ to be c-**competitive** if and only if

$$E_\sigma \left[\sum_{t=1}^{n} \pi_t x_t(\sigma, \mathcal{A}) \right] \geq c \cdot OPT(A, \pi) \; \forall (A, \pi),$$

where σ is the **permutation** of arriving orders.

In what follows, we let

$$B = \min_i \{b_i\} (> 0).$$

Main Results: Necessary and Sufficient Conditions

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

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

Theorem 2 *For any fixed* $0 < \epsilon < 1$*, there is a* $1 - \epsilon$ *competitive online algorithm for solving the linear program if*

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

Agrawal, Wang and Y [Operations Research 2014]

Key Ideas: A Worst-Case Distribution Example

The proof of the negative result is based on a **distribution** of instances (the number of each types of columns is chosen according to certain distribution) with $m = 2^k$, and then show that no allocation rule can achieve $(1-\epsilon)$-optimality in expectation under randomized permutation.

Key Ideas: A Learning Algorithm is Needed

The proof of the positive result is **constructive** and based on a learning policy.

- There is no distribution known so that any type of **stochastic optimization** models is not applicable.
- Unlike dynamic programming, the decision maker does not have full information/data so that a **backward recursion** can not be carried out to find an optimal sequential decision policy.
- Thus, the online algorithm needs to be **learning-based**, in particular, **learning-while-doing**.

Price Observation of Online Learning I

The problem would be easy if there is an **"ideal price"** vector:

	Bid 1($t = 1$)	Bid 2($t = 2$)	.....	Inventory($\mathbf{b}$)	$\mathbf{p}^*$
Bid(π_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

Price Observation of Online Learning 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 $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ 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 ϵn input.
- **Dynamic learning algorithm**: dynamically update the price vector at a carefully chosen pace.

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 ϵ portion of the problem

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

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

- While inventory remains, determine x_t as:

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

One-Time Learning Algorithm Result

Theorem 3 *For any fixed $\epsilon > 0$, the one-time learning algorithm is $(1-\epsilon)$ competitive for solving the linear program when*

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

Outline of the Proof

- With high probability, we clear the market;
- With high probability, the revenue is near-optimal if we include the initial ϵ portion revenue;
- With high probability, the first ϵ portion revenue, a learning cost, doesn't contribute too much.

Then, we prove that the one-time learning algorithm is $(1-\epsilon)$ competitive under condition $B \geq \frac{6m\log(n/\epsilon)}{\epsilon^3}$.

But this is one ϵ factor higher than the **lower bound**...

Dynamic Learning Algorithm

In the dynamic price learning algorithm, we update the price at time ϵn, $2\epsilon n$, $4\epsilon n$, ..., 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:

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

where

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

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

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we **update** the prices $\log_2(1/\epsilon)$ times during the entire time horizon.
- The numbers h_ℓ play an important role in improving the condition on B in the main theorem. It basically **balances** the probability that the inventory ever gets violated and the lost of revenue due to the factor $1 - h_\ell$.
- Choosing large h_ℓ (more conservative) at the beginning periods and smaller h_ℓ (more aggressive) at the later periods, one can now control the loss of revenue by an ϵ order while the required size of B can be **weakened** by an ϵ factor.

Related Work on Random-Permutation

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

Table 1: Comparison of several existing results

Open Questions on Online LP

- Is a **Price Mechanism** similar to Prediction Market to achieve optimal learning?
- **Buy-and-sell** or double market?
- **Price-posting** market?

The ADMM for LP Primal I

$$(LP) \quad \text{minimize} \quad \mathbf{c} \bullet \mathbf{x}$$
$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geq \mathbf{0},$$

We consider an equivalent problem:

$$(LP) \quad \text{minimize} \quad \mathbf{c} \bullet \mathbf{x}_1$$
$$\text{subject to} \quad A\mathbf{x}_1 = \mathbf{b}, \ \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}, \ \mathbf{x}_2 \geq \mathbf{0},$$

The ADMM for LP Primal II

Consider its Augmented Lagrangian for the primal

$$L^p(\mathbf{x}_1,\mathbf{x}_2,\mathbf{y},\mathbf{s}) = \mathbf{c}^T\mathbf{x}_1 - \mathbf{y}^T(A\mathbf{x}_1-\mathbf{b}) - \mathbf{s}^T(\mathbf{x}_1-\mathbf{x}_2) + \frac{\beta}{2}\|A\mathbf{x}_1-\mathbf{b}\|^2 + \frac{\beta}{2}\|\mathbf{x}_1-\mathbf{x}_2\|^2.$$

Then, for any given $(\mathbf{x}_1^k, \mathbf{x}_2^k \in K, \mathbf{y}^k, \mathbf{s}^k)$, we compute a new iterate pair

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1} L^p(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{y}^k, \mathbf{s}^k), \quad \mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2 \geq \mathbf{0}} L^p(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{y}^k, \mathbf{s}^k)$$

and

$$\mathbf{y}^{k+1} = \mathbf{y}^k - \beta(A\mathbf{x}_1^{k+1} - \mathbf{b}), \quad \mathbf{s}^{k+1} = \mathbf{s}^k - \beta(\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1}).$$

The minimization over $\mathbf{x}_1$ is a unconstrained optimization, and the minimization over $\mathbf{x}_2$ is easy!

The ADMM for LP Dual

Consider the dual problem and its Augmented Lagrangian

$$\begin{array}{lll} (LD) & \text{maximize} & \mathbf{b}^T\mathbf{y} \\ & \text{subject to} & A^T\mathbf{y}+\mathbf{s}=\mathbf{c},\ \mathbf{s}\geq\mathbf{0}, \end{array}$$

$$L^d(\mathbf{y},\mathbf{s},\mathbf{x}) = -\mathbf{b}^T\mathbf{y} - \mathbf{x}^T(A^T\mathbf{y}+\mathbf{s}-\mathbf{c}) + \frac{\beta}{2}\|A^T\mathbf{y}+\mathbf{s}-\mathbf{c}\|^2.$$

Then, starting from $\mathbf{y}^0$, $\mathbf{s}^0 \geq \mathbf{0}$, and multiplier $\mathbf{x}^0$, do the iterative update:

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} L^d(\mathbf{y},\mathbf{s}^k,\mathbf{x}^k), \quad \mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\geq\mathbf{0}} L^d(\mathbf{y}^{k+1},\mathbf{s},\mathbf{x}^k),$$
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \beta(A^T\mathbf{y}^{k+1}+\mathbf{s}^{k+1}-\mathbf{c}).$$

The minimization over $\mathbf{y}$ is a least-squares problem with constant matrix, and the update of $\mathbf{s}$ has a simple close form. ($\mathbf{x}$ would be non-positive since we changed maximization to minimization of the dual.)

The Interior-Point ADMM for CLP Dual I

Now consider solving CLP with the logarithmic barrier function $B(\mathbf{s}) = \sum_j \ln(s_j)$ and fixed positive constant μ:

$$
\begin{array}{lll}
(BLD) & \text{maximize} & \mathbf{b}^T\mathbf{y} + \mu B(\mathbf{s}) \\
 & \text{subject to} & A^T\mathbf{y} + \mathbf{s} = \mathbf{c},\ \mathbf{s} > \mathbf{0},
\end{array}
$$

and its Augmented Lagrangian

$$
L_\mu^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b}^T\mathbf{y} - \mu B(\mathbf{s}) - \mathbf{x}^T(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}) + \frac{\beta}{2}\|A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\|^2.
$$

When applying ADMM to (BCLD), again minimization over $\mathbf{y}$ is a least-squares problem with constant matrix, and the update of $\mathbf{s} > \mathbf{0}$ has a simple close form.

The Interior-Point ADMM for CLP Dual II

Now, we gradually reduced μ as an outer iteration. That is, we start some $\mu = \mu^0$ and apply the ADMM to compute an approximate optimizer, with its multiplier, for barrier-dual (BCLD). Now set $\mu = \mu^1 = \gamma\mu^0$ where $0 < \gamma < 1$. Then we use the approximate optimizer and multiplier as the initial point to start ADMM barrier-dual (BCLD) with the newly reduced μ.

Does it work?