Near-optimal Algorithms for Online Linear Programming

Zizhuo Wang

Department of Industrial and Systems Engineering
University of Minnesota

Joint work with Shipra Agrawal and Yinyu Ye

July 7th, 2014

Introduction

Introduction

In traditional optimization problems, the input data is given (either in deterministic form or stochastic form).

Introduction

In traditional optimization problems, the input data is given (either in deterministic form or stochastic form).

For example, in a linear program

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \pi^T \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{array}$$

one solves the decision variables all at once.

Introduction

In traditional optimization problems, the input data is given (either in deterministic form or stochastic form).

For example, in a linear program

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \pi^T \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{array}$$

one solves the decision variables all at once.

However, in many practical problems, the input information is not available at the start, but reveals sequentially. Decisions have to be made in an online fashion.

Introduction

In traditional optimization problems, the input data is given (either in deterministic form or stochastic form).

For example, in a linear program

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \pi^T \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{array}$$

one solves the decision variables all at once.

However, in many practical problems, the input information is not available at the start, but reveals sequentially. Decisions have to be made in an online fashion.

- Online linear programming problems

Introduction

Introduction

In this talk, we consider linear programs of the following format:

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

Introduction

In this talk, we consider linear programs of the following format:

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

In the online version of the problem: we only know B_i's at the start.

Introduction

In this talk, we consider linear programs of the following format:

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

In the online version of the problem: we only know B_i's at the start.

- The constraint matrix is revealed column by column sequentially along with the corresponding objective coefficient.

Introduction

In this talk, we consider linear programs of the following format:

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

In the online version of the problem: we only know B_i's at the start.

- The constraint matrix is revealed column by column sequentially along with the corresponding objective coefficient.
- An irrevocable decision must be made as soon as a column arrives without observing or knowing the future data.

Applications

Applications

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

Applications

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

This model is frequently used in resource allocation problems.

Applications

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

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer

Applications

$$
\begin{array}{lll}
\text{maximize}_{\mathbf{x}} & \sum_{t=1}^{n} \pi_t x_t & \\
\text{subject to} & \sum_{t=1}^{n} a_{it} x_t \leq B_i, & \forall i = 1, \ldots, m \\
 & 0 \leq x_t \leq 1, & \forall t = 1, \ldots, n
\end{array}
$$

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer
- B_i is the inventory for the ith good.

Applications

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

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer
- B_i is the inventory for the ith good.
- π_t is the price that the tth customer is willing to pay.

Applications

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

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer
- B_i is the inventory for the ith good.
- π_t is the price that the tth customer is willing to pay.
- x_t is the decision whether to accept or reject the tth customer

Applications

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

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer
- B_i is the inventory for the ith good.
- π_t is the price that the tth customer is willing to pay.
- x_t is the decision whether to accept or reject the tth customer

Customers arrive sequentially and an irrevocable decision must be made without observing future customer arrivals.

Applications

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

This model is frequently used in resource allocation problems.

- $\{a_{it}\}_{i=1}^{m}$ are the request of a bundle of goods by tth customer
- B_i is the inventory for the ith good.
- π_t is the price that the tth customer is willing to pay.
- x_t is the decision whether to accept or reject the tth customer

Customers arrive sequentially and an irrevocable decision must be made without observing future customer arrivals.

- Applications: revenue management, channel allocation in communication networks, charging allocation for electric vehicles, etc.

Our Objective

Our Objective

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

Our Objective

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

We call the above problem the *offline problem*. And we denote the optimal value of it by OPT.

Our Objective

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

We call the above problem the *offline problem*. And we denote the optimal value of it by OPT.

Our objective: to find a decision rule for the online problem such that it achieves *near-optimal* performance, i.e., an algorithm that can achieve values close to OPT.

Model Assumptions

Model Assumptions

Main Assumptions

Model Assumptions

Main Assumptions

- We know the total number of columns n a priori

Model Assumptions

Main Assumptions

- We know the total number of columns n a priori
- The columns $\mathbf{a}_t$ arrive in a random order, i.e., the set of columns together with their objective coefficients π_t can be adversarially picked at the start. However, their arriving order is uniformly distributed over all the permutations

Model Assumptions

Main Assumptions

- We know the total number of columns n a priori
- The columns $\mathbf{a}_t$ arrive in a random order, i.e., the set of columns together with their objective coefficients π_t can be adversarially picked at the start. However, their arriving order is uniformly distributed over all the permutations

The algorithm is evaluated on the expected performance over all the permutations comparing to the offline optimal solution, i.e., an algorithm $\mathcal{A}$ is c-competitive if and only if

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

Comment on the Assumptions

Comment on the Assumptions

Random permutation of input:

Comment on the Assumptions

Random permutation of input:

- An intermediate path between worst-case and i.i.d model.

Comment on the Assumptions

Random permutation of input:

- An intermediate path between worst-case and i.i.d model.
- Worst case model: Too conservative, no algorithm can achieve better than $O(1/n)$ approximation of the optimal offline solution [Babaioff et al 2008].

Comment on the Assumptions

Random permutation of input:

- An intermediate path between worst-case and i.i.d model.
- Worst case model: Too conservative, no algorithm can achieve better than $O(1/n)$ approximation of the optimal offline solution [Babaioff et al 2008].
- i.i.d. model: Need distribution information. Performance might suffer if the actual input distribution is not as assumed.

Comment on the Assumptions

Random permutation of input:

- An intermediate path between worst-case and i.i.d model.
- Worst case model: Too conservative, no algorithm can achieve better than $O(1/n)$ approximation of the optimal offline solution [Babaioff et al 2008].
- i.i.d. model: Need distribution information. Performance might suffer if the actual input distribution is not as assumed.
- Our assumption is strictly weaker than the i.i.d. model

Main Results

Main Results

Theorem

We propose an algorithm such that for any fixed $\epsilon > 0$*, our online algorithm is* $1 - O(\epsilon)$ *competitive for online linear programming on all inputs when*

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

Main Results

Theorem

We propose an algorithm such that for any fixed $\epsilon > 0$, our online algorithm is $1 - O(\epsilon)$ competitive for online linear programming on all inputs when

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

Theorem

For any online algorithm for the online linear program in random permutation model, there exists an instance such that the competitive ratio is less than $1 - O(\epsilon)$ if

$$B = \min_i B_i \leq \frac{\log(m)}{\epsilon^2}$$

Compare to Previous Results

Compare to Previous Results

k-secretary problem [Kleinberg 2005]: special case when $m = 1$, and $a_t = 1$ for all t. His algorithm is $1 - O(\epsilon)$-competitive for $B \geq \frac{1}{\epsilon^2}$

Compare to Previous Results

k-secretary problem [Kleinberg 2005]: special case when $m = 1$, and $a_t = 1$ for all t. His algorithm is $1 - O(\epsilon)$-competitive for $B \geq \frac{1}{\epsilon^2}$

- We extend the result to multi-products case.

Compare to Previous Results

k-secretary problem [Kleinberg 2005]: special case when $m = 1$, and $a_t = 1$ for all t. His algorithm is $1 - O(\epsilon)$-competitive for $B \geq \frac{1}{\epsilon^2}$

- We extend the result to multi-products case.
- The ϵ part of the bound is still unchanged, thus must be optimal

Compare to Previous Results

k-secretary problem [Kleinberg 2005]: special case when $m = 1$, and $a_t = 1$ for all t. His algorithm is $1 - O(\epsilon)$-competitive for $B \geq \frac{1}{\epsilon^2}$

- We extend the result to multi-products case.
- The ϵ part of the bound is still unchanged, thus must be optimal
- We show, for the first time, that the dimension of the problem m indeed adds to its difficulty (at least $\log m$).

Compare to Previous Results

k-secretary problem [Kleinberg 2005]: special case when $m = 1$, and $a_t = 1$ for all t. His algorithm is $1 - O(\epsilon)$-competitive for $B \geq \frac{1}{\epsilon^2}$

- We extend the result to multi-products case.
- The ϵ part of the bound is still unchanged, thus must be optimal
- We show, for the first time, that the dimension of the problem m indeed adds to its difficulty (at least $\log m$).

Our algorithm is quite different from theirs due to the higher dimension

Compare to Previous Results

Compare to Previous Results

Online adwords problem [Devanur and Hayes 2009]: special case of the online linear programming problem. They have an algorithm that achieves $1 - \epsilon$-competitiveness if

$$OPT \geq \Omega\left(\frac{m^2 \log(mn)}{\epsilon^3}\right)$$

Compare to Previous Results

Online adwords problem [Devanur and Hayes 2009]: special case of the online linear programming problem. They have an algorithm that achieves $1-\epsilon$-competitiveness if

$$OPT \geq \Omega\left(\frac{m^2\log(mn)}{\epsilon^3}\right)$$

- We consider a much more general problem

Compare to Previous Results

Online adwords problem [Devanur and Hayes 2009]: special case of the online linear programming problem. They have an algorithm that achieves $1-\epsilon$-competitiveness if

$$OPT \geq \Omega\left(\frac{m^2 \log(mn)}{\epsilon^3}\right)$$

- We consider a much more general problem
- Their results is weaker by a factor or ϵ and also depend on OPT which is not known until the problem is solved. (Our condition can be verified before solving the problem)

Key Ideas of Our Algorithm

Key Ideas of Our Algorithm

Dual Optimal Price

- For the offline linear program, there exists a dual price vector $\mathbf{p}^*$ for each goods such that, $x_t^* = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, is near optimal

Key Ideas of Our Algorithm

Dual Optimal Price

- For the offline linear program, there exists a dual price vector $\mathbf{p}^*$ for each goods such that, $x_t^* = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, is near optimal

Learning the price:

- Our online algorithm works by learning a price vector $\hat{\mathbf{p}}$. The price vector is determined by solving the dual problem using existing arrival data.

Key Ideas of Our Algorithm

Dual Optimal Price

- For the offline linear program, there exists a dual price vector $\mathbf{p}^*$ for each goods such that, $x_t^* = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, is near optimal

Learning the price:

- Our online algorithm works by learning a price vector $\hat{\mathbf{p}}$. The price vector is determined by solving the dual problem using existing arrival data.

We first show a one-time learning algorithm to illustrate this idea. Then we show that we can improve the algorithm by updating the price vector more frequently.

One-Time Learning Algorithm

One-Time Learning Algorithm

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

One-Time Learning Algorithm

1. Set $x_t = 0$ for all $t \leq \epsilon n$
2. Solve the ϵ part of the problem

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

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

One-Time Learning Algorithm

1. Set $x_t = 0$ for all $t \le \epsilon n$
2. Solve the ϵ part of the problem

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

   Get the optimal dual solution $\hat{\mathbf{p}}$;
3. Determine the future allocation $x_t(\hat{\mathbf{p}})$ as:

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

   If $a_{it} x_t(\hat{\mathbf{p}}) \le B_i - \sum_{j=1}^{t-1} a_{ij} x_j$, set $x_t = x_t(\hat{\mathbf{p}})$; otherwise, set $x_t = 0$.

Analysis Idea

Analysis Idea

By the complementarity conditions of LP, if we can show that $\sum_{t=1}^{n} a_{it} x_t(\hat{\mathbf{p}}) = B_i$ for each i, then $x_t(\hat{\mathbf{p}})$ is the optimal solution to the offline problem.

Analysis Idea

By the complementarity conditions of LP, if we can show that $\sum_{t=1}^{n} a_{it} x_t(\hat{\mathbf{p}}) = B_i$ for each i, then $x_t(\hat{\mathbf{p}})$ is the optimal solution to the offline problem.

Lemma

With probability $1 - \epsilon$,

$$(1 - 3\epsilon) B_i \leq \sum_{t=1}^{n} a_{it} x_t(\hat{\mathbf{p}}) \leq B_i, \forall i = 1, \ldots, m$$

given $B \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$.

Analysis Idea

By the complementarity conditions of LP, if we can show that $\sum_{t=1}^{n} a_{it} x_t(\hat{\mathbf{p}}) = B_i$ for each i, then $x_t(\hat{\mathbf{p}})$ is the optimal solution to the offline problem.

Lemma

With probability $1 - \epsilon$,

$$(1 - 3\epsilon) B_i \leq \sum_{t=1}^{n} a_{it} x_t(\hat{\mathbf{p}}) \leq B_i, \forall i = 1, \ldots, m$$

given $B \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$.

The proof uses intensively concentration inequalities (Hoeffding-Bernstein Inequalities) and union bound arguments

Analysis Idea Continued

Analysis Idea Continued

However, by the complementarity conditions, $x(\hat{\mathbf{p}})$ is optimal to

Analysis Idea Continued

However, by the complementarity conditions, $x(\hat{\mathbf{p}})$ is optimal to

$$\begin{array}{lll} \text{maximize}_x & \sum_t \pi_t x_t & \\ \text{subject to} & \sum_t a_{it} x_t \leq \sum_t a_{it} x_t(\hat{\mathbf{p}}) & i = 1, ..., m \\ & 0 \leq x_t \leq 1 & t = 1, ..., n \end{array}$$

Analysis Idea Continued

However, by the complementarity conditions, $x(\hat{\mathbf{p}})$ is optimal to

$$\begin{array}{lll} \text{maximize}_x & \sum_t \pi_t x_t & \\ \text{subject to} & \sum_t a_{it} x_t \le \sum_t a_{it} x_t(\hat{\mathbf{p}}) & i = 1, \ldots, m \\ & 0 \le x_t \le 1 & t = 1, \ldots, n \end{array}$$

Then by the above lemmas, it is easy to show that with high probability $\sum_{t=1}^{n} \pi_t x_t(\hat{\mathbf{p}}) \ge (1 - 3\epsilon) OPT$.

Analysis Idea Continued

Analysis Idea Continued

Another thing we need to take care of is that the algorithm does not make allocation in the first ϵn period.

Analysis Idea Continued

Another thing we need to take care of is that the algorithm does not make allocation in the first ϵn period.

Lemma

Let $OPT(S)$ denote the optimal value of the linear program

$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t \in S} \pi_t x_t & \\ \text{subject to} & \sum_{t \in S} a_{it} x_t \leq \epsilon B_i, & i = 1, ..., m \\ & 0 \leq x_t \leq 1, & t \in S. \end{array}$$

over random sample $S \subset N$ where $|S| = \epsilon |N|$, and $OPT(N)$ denote the optimal value of the original offline linear program. Then,

$$E[OPT(S)] \leq \epsilon OPT(N)$$

Analysis Idea Continued

Analysis Idea Continued

Summarizing all the lemmas above:

Analysis Idea Continued

Summarizing all the lemmas above:

- With high probability, we never violate the inventory constraint

Analysis Idea Continued

Summarizing all the lemmas above:

- With high probability, we never violate the inventory constraint
- With high probability, the objective value is near-optimal if we include the initial ϵ portion

Analysis Idea Continued

Summarizing all the lemmas above:

- With high probability, we never violate the inventory constraint
- With high probability, the objective value is near-optimal if we include the initial ϵ portion
- With high probability, the first ϵ portion of the objective value, a learning cost, doesn't contribute too much.

Analysis Idea Continued

Summarizing all the lemmas above:

- With high probability, we never violate the inventory constraint
- With high probability, the objective value is near-optimal if we include the initial ϵ portion
- With high probability, the first ϵ portion of the objective value, a learning cost, doesn't contribute too much.

Therefore, we proved that our one-time learning algorithm is $1 - O(\epsilon)$-competitive if $B \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$.

Dynamic Price Updating Algorithm

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

- ϵ^3 versus ϵ^2

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

- ϵ^3 versus ϵ^2

The one-time learning algorithm only computes the price once.

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

- ϵ^3 versus ϵ^2

The one-time learning algorithm only computes the price once.

- Potential improvement might be made by updating the price dynamically during the process.

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

- ϵ^3 versus ϵ^2

The one-time learning algorithm only computes the price once.

- Potential improvement might be made by updating the price dynamically during the process.

Question

Dynamic Price Updating Algorithm

The one-time learning algorithm is simple, but the condition required on the size of B is stronger than the main theorem claims

- ϵ^3 versus ϵ^2

The one-time learning algorithm only computes the price once.

- Potential improvement might be made by updating the price dynamically during the process.

Question

- How often should we update the price?

Dynamic Price Updating Algorithm

Dynamic Price Updating Algorithm

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

Dynamic Price Updating Algorithm

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

At time $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price is the optimal dual solution to the following linear program:

Dynamic Price Updating Algorithm

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

At time $\ell \in \{\epsilon n, 2\epsilon n, \ldots\}$, the price is the optimal dual solution to the following linear program:

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

where

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

Dynamic Price Updating Algorithm

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

At time $\ell \in \{\epsilon n, 2\epsilon n, \ldots\}$, the price is the optimal dual solution to the following linear program:

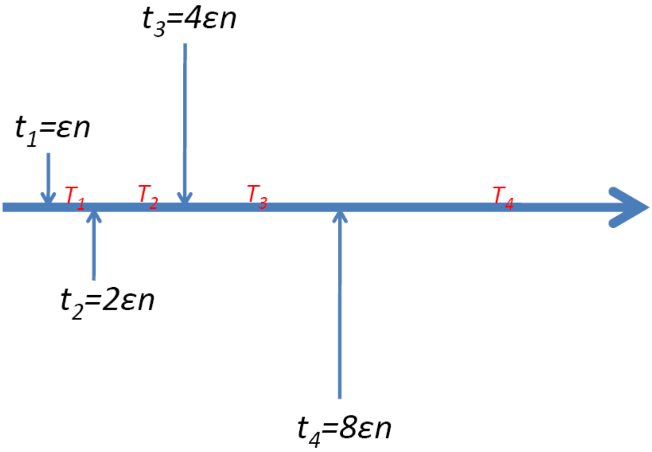
$$
\begin{array}{lll}
\text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\
\text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1 - h_\ell) \frac{\ell}{n} b_i & i = 1, \ldots, m \\
 & 0 \leq x_t \leq 1 & t = 1, \ldots, \ell
\end{array}
$$

where

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

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

Geometric Pace of Price Updating



Dynamic Price Updating Algorithm

Dynamic Price Updating Algorithm

- In this algorithm, we update the price $\log_2(1/\epsilon)$ times during the entire time horizon.

Dynamic Price Updating Algorithm

- In this algorithm, we update the price $\log_2(1/\epsilon)$ times during the entire time horizon.
- The numbers h_ℓ play an important role in improving the condition on B in our main theorem. It balances the probability that the inventory constraint ever gets violated and the lost of objective value due to the factor $1 - h_\ell$.

Dynamic Price Updating Algorithm

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

Proof Outline of the Dynamic Price Updating Algorithm

The proof is similar to the proof of the one-time learning algorithm. We first show the following lemma:

Lemma

For any $\epsilon > 0$, with probability $1 - \epsilon$:

$$\sum_{t=\ell+1}^{2\ell} a_{it} x_t(\hat{\mathbf{p}}^\ell) \leq \frac{\ell}{n} b_i, \quad \textit{for all } i \in \{1, \ldots, m\}, \ \ell \in \{\epsilon n, 2\epsilon n, \ldots\}$$

given $B = \min_i b_i \geq \frac{10 m \log(n/\epsilon)}{\epsilon^2}$.

This lemma states that for each time period, with high probability, the inventory consumed by $x(\hat{\mathbf{p}})$ is less than the proportional total inventory.

Proof Outline of the Dynamic Price Updating Algorithm

Next we need to show that by introducing a factor h_ℓ, the loss of objective value is small.

It is easy to see that at period ℓ, the lost of objective value is about $h_\ell \cdot \frac{\ell}{n} OPT$. Then the total loss of objective value is about

$$\begin{array}{rl} & \sum_{\ell \in \{\epsilon n, 2\epsilon n, \ldots\}} h_\ell \frac{\ell}{n} OPT \\ = & \epsilon \sum_{\ell \in \{\epsilon n, 2\epsilon n, \ldots\}} \sqrt{\frac{\ell}{n}} OPT \\ \leq & \epsilon OPT \sum \sqrt{\frac{1}{2^i}} \\ \leq & O(\epsilon OPT) \end{array}$$

Therefore, the loss of objective value is very small. And we can conclude that this algorithm gives a near-optimal solution.

Summary

We propose a dynamic near-optimal algorithm for a class of online linear programming problems under the random permutation model

- It solves linear programs for dual price based on revealed data, and uses these prices to make future allocations
- The algorithm has the feature of "learning-while-doing", and the pace the price is updated is neither too fast nor too slow
- The application includes various online resource allocation and revenue management problems