A Dynamic Near-Optimal Learning Algorithm for Online Linear Programming[1]

Cole Sturza Alex Book

Computer Science
University of Colorado Boulder

April 18, 2022

[1] [AWY14]

Operations Research
Vol. 62, No. 4, July–August 2014, pp. 876–890
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

informs

http://dx.doi.org/10.1287/opre.2014.1289

A Dynamic Near-Optimal Algorithm for Online Linear Programming

Shipra Agrawal
Microsoft Research India, Bangalore, India, shipra@microsoft.com

Zizhuo Wang
Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, Minnesota 55455, zwang@umn.edu

Yinyu Ye
Department of Management Science and Engineering, Stanford University, Stanford, California 94305, yinyu-ye@stanford.edu

A natural optimization model that formulates many online resource allocation problems is the online linear programming (LP) problem in which the constraint matrix is revealed column by column along with the corresponding objective coefficient. In such a model, a decision variable has to be set each time a column is revealed without observing the future inputs, and the goal is to maximize the overall objective function. In this paper, we propose a near-optimal algorithm for this general class of online problems under the assumptions of random order of arrival and some mild conditions on the size of the LP right-hand-side input. Specifically, our learning-based algorithm works by dynamically updating a threshold price vector at geometric time intervals, where the dual prices learned from the revealed columns in the previous period are used to determine the sequential decisions in the current period. Through dynamic learning, the competitiveness of our algorithm improves over the past study of the same problem. We also present a worst case example showing that the performance of our algorithm is near optimal.

Subject classifications: online algorithms; linear programming; primal-dual; dynamic price update.
Area of review: Optimization
History: Received August 2013; revision received December 2013; accepted April 2014. Published online in *Articles in Advance* June 13, 2014.

Online vs. Offline

- An *offline* algorithm has access to the whole problem data from the beginning
- An *online* algorithm takes a sequence of inputs or pieces of information and, at each round, must make a decision or produce some output[2]
- In Operations Research, the area in which online algorithms are developed is called *Online Optimization*

[2] [Wag20]

Competitive Analysis[3]

- To evaluate online algorithms, we compare the algorithm's output to the optimal choices (as if we had known the whole problem data from the beginning)
- For minimization problems, we say an algorithm is C-**competitive** if it guarantees $\text{ALG} \leq C \cdot \text{OPT}$ for all instances of the problem
- For maximization problems, we say an algorithm is α-**competitive** if it guarantees $\text{ALG} \geq \alpha \cdot \text{OPT}$ for all instances of the problem

[3] [Wag20]

(Offline) Linear Program (LP)

We consider the following (offline) linear program:

$$\begin{array}{rll} \text{maximize} & \sum_{j=1}^{n} \pi_j x_j & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i = 1, \ldots, m \\ & 0 \leq x_j \leq 1, & j = 1, \ldots, n \ , \end{array}$$

where for all j,

- $\pi_j \geq 0$,
- $\mathbf{a}_j = \{a_{ij}\}_{i=1}^{m} \in [0,1]^m$, and
- $\mathbf{b} = \{b_i\}_{i=1}^{m} \in \mathbb{R}^m$

Online Linear Program (OLP)

In the corresponding online LP problem, at each time t,

- the coefficients $(\pi_t, \mathbf{a}_t)$ are revealed, and the decision variable x_t has to be chosen
- Given the previous $t-1$ decisions $x_1, \ldots, x_{t-1}$ and inputs $\{\pi_j, \mathbf{a}_j\}_{j=1}^{t}$ until time t, the t^{th} decision variable x_t has to satisfy

$$\sum_{j=1}^{t} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \text{ and } 0 \leq x_t \leq 1$$

The goal is to chose x_t's such that the objective $\sum_{t=1}^{n} \pi_t x_t$ is maximized.

Assumptions

Assumption 1

The columns $\mathbf{a}_j$ (with the objective coefficient π_j) arrive in random order. The set of columns $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$ can be adversarily picked at the start. However, the arrival order of $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$ is uniformly distributed over all the permutations.

Assumption 2

We know the total number of columns n a priori.

Assumptions (Continued)

Assumption 2

We know the total number of columns n a priori.

- Assumption 2 is required since we need to use the quantity n to decide the length of history for learning the threshold.
- Assumption 2 can be relaxed to an approximate knowledge of n (within at most $1 \pm \epsilon$ multiplicative error), without affecting performance

Competitive Analysis in the Random Permutation Model

$$\mathbb{E}_\sigma\left[\sum_{t=1}^{n} \pi_t x_t\right] \geq \alpha \cdot \mathsf{OPT},$$

where the expectation is taken over uniformly random permutations σ of $1, \ldots, n$, and x_t is the tth decision made by the algorithm when inputs arrive in order σ.

Online Knapsack/Secretary Problem

- Hiring workers
- Scheduling jobs
- Bidding in sponsored search auctions

	Bid 1 ($t = 1$)	Bid 2 ($t = 2$)	...	Inventory($\mathbf{b}$)
Bid(π_t)	\$100	\$50	...	
Decision(x_t)	1	0	...	
Item 1	0	1	...	100
Item 2	1	1	...	50
Item 3	0	1	...	45
Item 4	1	0	...	18
Item 5	0	0	...	25

Online Routing Problem

- A network with m edges
- Each edge i has a corresponding bounded capacity b_i
- There are a large number of requests arriving online, each asking for certain capacities $\mathbf{a}_t \in \mathbb{R}^m_{\geq 0}$
- Each request has a utility or price $\pi_t \in \mathbb{R}_{\geq 0}$

$$\begin{array}{rll} \text{maximize} & \sum_{j=1}^n \pi_j x_j & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \leq b_i, & i = 1, \ldots, m \\ & x_j \in \{0, 1\}, & j = 1, \ldots, n \end{array}$$

Multidimensional Decisions

- Given a sequence of n non-negative vectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n \in \mathbb{R}^k$,
- mn non-negative vectors $\mathbf{g}_{i1}, \mathbf{g}_{i2}, \ldots, \mathbf{g}_{in} \in [0,1]^k$ for $i = 1, \ldots, m$, and
- $K = \{\mathbf{x} \in \mathbb{R}^k \mid \mathbf{x}^T\mathbf{e} = 1, \mathbf{x} \succeq 0\}$[4]

$$
\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{j=1}^n \mathbf{f}_j^T \mathbf{x}_j \\
\text{subject to} \quad & \textstyle\sum_{j=1}^n \mathbf{g}_{ij}^T \mathbf{x}_j \le b_i, \quad i = 1, \ldots, m \\
& \mathbf{x}_j \in K, \qquad\qquad j = 1, \ldots, n
\end{aligned}
$$

[4] $\mathbf{e}$ is a column vector of all ones.

Online Adwords Problem

- There are n search queries arriving online and m bidders (advertisers) each with a daily budget b_i
- Based on the relevance of each search keyword, the ith bidder will bid a certain amount π_{ij} on query j to display their advertisement along with the search result
- For the jth query the decision maker has to choose an m-dimensional vector $x_j = \{x_{ij}\}_{i=1}^m$, where $x_{ij} \in \{0, 1\}$ indicates whether the jth query is allocated to the ith bidder

$$
\begin{array}{rll}
\text{maximize} & \sum_{j=1}^n \boldsymbol{\pi}_j^T \mathbf{x}_j & \\
\text{subject to} & \sum_{j=1}^n \pi_{ij} x_{ij} \leq b_i, & i = 1, \ldots, m \\
& \mathbf{x}_j^T \mathbf{e} = 1, & j = 1, \ldots, n \\
& \mathbf{x}_j \in \{0, 1\}^m, & j = 1, \ldots, n
\end{array}
$$

Motivation

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

	Bid 1 ($t = 1$)	Bid 2 ($t = 2$)	...	Inventory($\mathbf{b}$)	$\mathbf{p}^\star$
Bid(π_t)	\$100	\$50	...		
Decision(x_t)	1	0	...		
Item 1	0	1	...	100	\$45
Item 2	1	1	...	50	\$35
Item 3	0	1	...	45	\$100
Item 4	1	0	...	18	\$50
Item 5	0	0	...	25	\$15

Motivation (Continued)

The Dual of the offline LP:

$$\begin{aligned} \text{maximize } & \textstyle\sum_{j=1}^{m} b_i p_j + \sum_{i=1}^{n} y_i \\ \text{subject to } & \textstyle\sum_{i=1}^{m} a_{ij} p_i + y_j \le \pi_j, \; j = 1, \ldots, n \\ & \mathbf{p}, \mathbf{y} \succeq 0 \end{aligned}$$

- Pricing the bid: The optimal dual price vector $\mathbf{p}^\star$ of the offline LP problem can play the role of an "ideal price" vector.
  - $x_t^\star = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^\star$ and $x_t^\star = 0$ otherwise will yield a near optimal solution
- From this observation we can create an online algorithm that creates an approximation of threshold price $\hat{\mathbf{p}}$

One-Time Learning Algorithm

- Learn the price vector once using the initial $s = \epsilon n$ input, then use this vector at every later time step to decide on and execute the current allocation (pending no constraint infractions)

One-Time Learning Algorithm (Continued)

Primal:

$$\begin{array}{rll} \text{maximize} & \sum_{t=1}^{s} \pi_t x_t & \\ \text{subject to} & \sum_{j=1}^{n} a_{it} x_t \leq (1-\epsilon)\frac{s}{n} b_i, & i = 1, \dots, m \\ & 0 \leq x_t \leq 1, & t = 1, \dots, n \end{array}$$

Dual:

$$\begin{array}{rll} \text{minimize} & \sum_{i=1}^{m} (1-\epsilon)\frac{s}{n} b_i p_i + \sum_{t=1}^{s} y_t & \\ \text{subject to} & \sum_{i=1}^{m} a_{it} p_i + y_t \geq \pi_t, & t = 1, \dots, s \\ & p_i, y_t \geq 0, & i = 1, \dots, m, t = 1, \dots, s \end{array}$$

One-Time Learning Algorithm (Continued)

Let $(\hat{\mathbf{p}}, \hat{\mathbf{y}})$ be the optimal dual solution. For any given price vector $\mathbf{p}$ the allocation rule is:

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

One-Time Learning Algorithm

1. Initialize $x_t = 0$, for all $t \le s$. $\hat{\mathbf{p}}$ is defined as above.
2. For $t = s+1, s+2, \ldots n$, if $a_{it} x_t(\hat{\mathbf{p}}) \le b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ for all i, set $x_t = x_t(\hat{\mathbf{p}})$; otherwise, set $x_t = 0$. Output x_t.

One-Time Learning Algorithm (Continued)

Theorem

For any $\epsilon > 0$, the One-Time Learning Algorithm is $1 - 6\epsilon$ competitive for the OLP in the random permutation model, for all inputs such that

$$B = \min_i b_i \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$$

Dynamic Learning Algorithm

- Dynamically update the price vector every so often, keeping a more up-to-date price vector to use for allocation decisions
- Each time the history doubles, we learn a new price vector ($\epsilon n, 2\epsilon n, 4\epsilon n, \ldots$, giving us $\lceil \log_2(\frac{1}{\epsilon}) \rceil$ total updates)

Dynamic Learning Algorithm (Continued)

Let

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

Primal:

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

Dual:

$$\begin{array}{rll} \text{minimize} & \sum_{i=1}^{m} (1-h_\ell)\frac{\ell}{n} b_i p_i + \sum_{t=1}^{s} y_t & \\ \text{subject to} & \sum_{i=1}^{m} a_{it} p_i + y_t \geq \pi_t, & t = 1, \ldots, \ell \\ & p_i, y_t \geq 0, & i = 1, \ldots, m, t = 1, \ldots, \ell \end{array}$$

Dynamic Learning Algorithm (Continued)

Let $(\hat{\mathbf{p}}^\ell, \mathbf{y}^\ell)$ denote the optimal dual solution to the dual on the inputs until time ℓ, and let $x_t(\mathbf{p})$ be defined as before.

Dynamic Learning Algorithm

1. Initialize $t_0 = \epsilon n$. Set $x_t = 0$, for all $t \leq t_0$. $\hat{\mathbf{p}}^\ell$ is defined as above.
2. Set $\hat{x}_t = x_t(\hat{\mathbf{p}}^\ell)$. Here $\ell = 2^r \epsilon n$ where r is the largest integer s.t. $\ell < t$.
3. If $a_{it}\hat{x}_t \leq b_i - \sum_{j=1}^{t-1} a_{ij}x_j$ for all i, set $x_t = x_t(\hat{\mathbf{p}})$; otherwise, set $x_t = 0$. Output x_t.

Dynamic Learning Algorithm (Continued)

Theorem

For any $\epsilon > 0$*, the Dynamic Learning Algorithm is* $1 - \mathcal{O}(\epsilon)$ *competitive for the OLP in the random permutation model, for all inputs such that*

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

Dynamic Learning Algorithm (Continued)

Theorem

For any $\epsilon > 0$, the Dynamic Learning Algorithm is $1 - \mathcal{O}(\epsilon)$ competitive for the multidimensional decision OLP in the random permutation model, for all inputs such that

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

Relative Loss (RL)

$$\text{Relative Loss} = 1 - \frac{\text{Expected Value of ALG}}{\text{OPT}}$$

- **Note:** RL is simply 1 minus the competitive ratio
- We will use RL to evaluate the algorithms

Empirical Results (Continued)

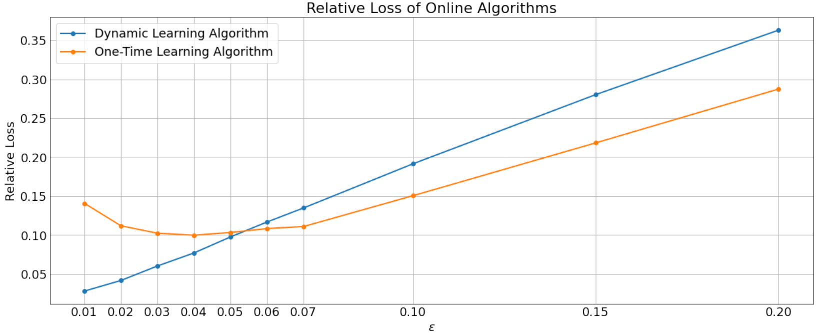

Figure: For each ϵ, 500 iterations of the algorithm were run on different permutations of the incoming data; the relative losses were averaged over all iterations. $B = 1,000$, $n = 10,000$, and $m = 5$.

Empirical Results (Continued)

B	RL of DLA	RL of OLA
100	0.0183	0.0415
500	0.0214	0.0446
1,000	0.0228	0.0467
2,000	0.0101	0.0448
5,000	0.0100	0.0430
10,000	0.0100	0.0414
50,000	0.0100	0.0400
100,000	0.0100	0.0387
500,000	0.0100	0.0375
690,776	0.0101	0.0364

Table: Epsilon was held constant at $\epsilon = 0.01$, $m = 5$, $n = 10,000$, while B varied. For each B, 500 iterations of the algorithm were run on a different permutation of the incoming data; the relative losses were averaged over all iterations.

Empirical Results (Continued)

m	RL of DLA	RL of OLA
1	0.0131	0.0284
2	0.0160	0.0459
3	0.0180	0.0586
4	0.0201	0.0717
5	0.0216	0.0813
6	0.0227	0.0873
7	0.0247	0.0987
8	0.0261	0.1050
9	0.0273	0.1118
10	0.0279	0.1128
15	0.0317	0.1257
20	0.0344	0.1289

Table: Epsilon was held constant at $\epsilon = 0.01$, $B = 1,000$, $n = 10,000$, while m varied. For each m, 500 iterations of the algorithm were run on a different permutation of the incoming data; the relative losses were averaged over all iterations.

https://github.com/colesturza/CSCI5654-Final-Project

References I

Shipra Agrawal, Zizhuo Wang, and Yinyu Ye.
A dynamic near-optimal algorithm for online linear programming.
Operations Research, 62(4):876–890, 2014.

Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg.
A knapsack secretary problem with applications.
In *APPROX-RANDOM'07*, pages 16–28, January 2007.

Bo Waggoner.
Online algorithms.
https://bowaggoner.com/courses/gradalg/notes/lect07-online.pdf, 2020.

References II

Zizhuo Wang.
Dynamic Learning Mechanisms in Revenue Management Problems.
Stanford University, 2012.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose.
Budget constrained bidding in keyword auctions and online knapsack problems.
In *International Workshop on Internet and Network Economics*, pages 566–576. Springer, 2008.