European Journal of Operational Research 247 (2015) 379–388

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Continuous Optimization

A dynamic learning algorithm for online matching problems with concave returns

Xiao Alison Chen, Zizhuo Wang*

Department of Industrial and Systems Engineering, University of Minnesota, MN, USA

ARTICLE INFO

Article history:
Received 17 February 2015
Accepted 11 June 2015
Available online 20 June 2015

Keywords:
Online algorithms
Primal-dual
Dynamic price update
Random permutation model
Adwords problem

ABSTRACT

We consider an online matching problem with concave returns. This problem is a generalization of the traditional online matching problem and has vast applications in online advertising. In this work, we propose a dynamic learning algorithm that achieves near-optimal performance for this problem when the inputs arrive in a random order and satisfy certain conditions. The key idea of our algorithm is to learn the input data pattern dynamically: we solve a sequence of carefully chosen partial allocation problems and use their optimal solutions to assist with the future decisions. Our analysis belongs to the primal-dual paradigm; however, the absence of linearity of the objective function and the dynamic feature of the algorithm makes our analysis quite unique. We also show through numerical experiments that our algorithm performs well for test data.

1. Introduction

In traditional optimization models, inputs are usually assumed to be known and efficient algorithms are sought to find the optimal solutions. However, in many practical cases, data does not reveal itself at the beginning. Instead, it comes in an online fashion. For example, in many revenue management problems, customers arrive sequentially and each time a customer arrives, the decision maker has to make some irrevocable decisions (e.g., what product to sell, at what prices) for this customer without knowing any of the future inputs. Such a regime is often called *online optimization*. Online optimization has gained much attention in the research community in the past few decades due to its applicability in many practical problems, and much effort has been directed toward understanding the quality of solutions that can be obtained under such settings. For an overview of the online optimization literature and its recent developments, we refer the readers to Borodin and El-Yaniv (1998), Buchbinder and Naor (2009) and Devanur (2011).

In this paper, we consider a special type of online optimization problem - an online matching problem. Online matching problems are considered as fundamental problems in online optimization theory and have important applications in the online advertisement allocation problems. For a review of online matching problems, we refer the readers to Mehta (2012). In the problem we study, there is an underlying weighted bipartite graph $G = (I, J, E)$ with weights b_{ij} for each edge $(i, j) \in E$. The vertices in J arrive sequentially in some order, and whenever a vertex $j \in J$ arrives, the set of weights b_{ij} is revealed for all $i \in I$, $(i, j) \in E$. The decision maker then has to match j to one of its neighbors i, and a value of b_{ij} will be obtained from this matching. In our problem, the decision maker's gain from each vertex i is a function of the total matched value to this vertex, and his goal is to maximize the total gain from all vertices. Mathematically, the problem can be formulated as follows (assume $|I| = m$, $|J| = n$, and let $b_{ij} = 0$ for $(i, j) \notin E$):

$$\begin{aligned} \text{maximize}_{\boldsymbol{x}} \quad & \sum_{i=1}^{m} M_i\left(\sum_{j=1}^{n} b_{ij}x_{ij}\right) & \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} \le 1, & \forall j \\ & x_{ij} \ge 0, & \forall i, j, \end{aligned} \tag{1}$$

where x_{ij} denotes the fraction of vertex j that is matched to vertex i.[1] In (1), the coefficient $\boldsymbol{b}_j = \{b_{ij}\}_{i=1}^m$ is revealed only when vertex j arrives, and an irrevocable decision $\boldsymbol{x}_j = \{x_{ij}\}_{i=1}^m$ has to be made before observing the next input. For each i, $M_i(\cdot)$ is a nondecreasing concave function with $M_i(0) = 0$. In this paper, we assume that $M_i(\cdot)$s are continuously differentiable.

As mentioned earlier, online matching problems have a very important application in the online advertisement allocation problem,

* Corresponding author. Tel.: +1 612 624 6752.
E-mail addresses: chen2847@umn.edu (X.A. Chen), zwang@umn.edu (Z. Wang).

[1] We allow fractional allocations in our model. However, our proposed algorithms output integer solutions. Thus all our results hold if one confines to integer solutions.

http://dx.doi.org/10.1016/j.ejor.2015.06.029

which we will later refer to as the Adwords problem. In the Adwords problem, there are m advertisers (which we also call the bidders). A sequence of n keywords are searched during a fixed time horizon. Based on the relevance of the keyword, the ith bidder would bid a certain amount b_{ij} to show his advertisement on the result page of the jth keyword. The search engine's decision is to allocate each keyword to one of the m bidders (we only consider a single allocation in this paper). Note that each allocation decision can only depend on the information earlier in the arrival sequence but not on any future data. As pointed out in Devanur and Jain (2012), there are several practical motivations for considering a concave function of the matched bids in the Adwords problem. Among them are convex penalty costs for under-delivery in search engine-advertiser contracts, the concavity of the click-through rate in the number of allocated bids observed in empirical data and fairness considerations. In each of the situations mentioned above, one can write the objective as a concave function. We refer the readers to Devanur and Jain (2012) for a more thorough review of the motivations for this problem. It is worth noting that there is a special case of this problem where $M_i(x) = \min\{x, B_i\}$. In this case, one can view that the bidder has a budget B_i and the revenue from each bidder is bounded by B_i.

One important question when studying online algorithms is the assumptions on the input data. In this work, we adopt a *random permutation model*. More precisely, we assume:

1. The total number of arrivals $n = |J|$ is known a priori.
2. The weights $\{b_{ij}\}$ can be adversarially chosen. However, the order that j arrives is uniformly distributed over all the permutations.

The random permutation model has been adopted in much recent literature in the study of online matching problems, see, e.g., Agrawal, Wang, and Ye (2014); Devanur and Hayes (2009); Feldman, Henzinger, Korula, Mirrokni, and Stein (2010), etc. It is equivalent to saying that a set of $\mathcal{B} = \{\tilde{\boldsymbol{b}}_1, \tilde{\boldsymbol{b}}_2, \ldots, \tilde{\boldsymbol{b}}_n\}$ is arbitrarily chosen beforehand (unknown to the decision maker). Then the arrivals $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$ are drawn randomly without replacement from $\mathcal{B}$. The random permutation model is an intermediate path between using a worst-case analysis and assuming each input data is drawn independently and identically distributed (i.i.d.) from a certain distribution. On one hand, compared to the worst-case analysis (see, e.g., Buchbinder, Jain, & Naor, 2007; Devanur & Jain, 2012; Feldman, Korula, Mirrokni, Muthukrishnan, & Pal, 2009; Mehta, Saberi, Vazirani, & Vazirani, 2005), the random permutation model is practically reasonable yet much less conservative. On the other hand, the random permutation model is much less restrictive than assuming the inputs are drawn i.i.d. from a certain distribution (Devanur, 2011). Also, the assumption of the knowledge of n is necessary for any online algorithm to achieve near-optimal performance (see Devanur & Hayes, 2009). Therefore, for large problems with relatively stationary inputs, the random permutation model is a good approximation and the study of such models is of practical interest. Next we define the performance measure of an algorithm under the random permutation model:

Definition 1 (c-competitiveness). *Let OPT* be the optimal value for the offline problem (1). An online algorithm A is called c-competitive in the random permutation model if the expected value of the online solutions by using A is at least c times the optimal value of (1), that is

$$\mathbb{E}_\sigma\left[\sum_{i=1}^{m} M_i\left(\sum_{j=1}^{n} b_{ij}x_{ij}(\sigma, A)\right)\right] \geq cOPT,$$

where the expectation is taken over uniformly random permutations σ of $1, \ldots, n$, and $x_{ij}(\sigma, A)$ is the ijth decision made by algorithm A when the inputs arrive in order σ.

In Devanur and Jain (2012), the authors propose an algorithm for the online matching problem with concave returns that has a constant competitive ratio under the *worst-case model* (the constant depends on the forms of each $M_i(\cdot)$). They also show that a constant competitive ratio is the *best* possible result under that model. In this paper, we propose an algorithm under the random permutation model, which achieves *near-optimal performance* under some conditions on the input.

Our main result is stated as follows:

Theorem 1. *Fix $\epsilon \in (0, 1/2)$. There exists an algorithm (Algorithm DLA) that is $1-\epsilon$ competitive for the online matching problem with concave returns $M_i(\cdot)$s under the random permutation model if*

$$n \geq \Omega\left(\max\left\{\frac{\log(m/\epsilon)}{\epsilon\bar{b}^2}, \frac{m^2\log(m^2n/\epsilon)F(M,\eta)}{\epsilon^3\bar{b}}\right\}\right), \tag{2}$$

where $\bar{b} = \frac{1}{n}\min_i\{\sum_{j=1}^{n} b_{ij}\}$, $\eta = \frac{\min_{i,j}\{b_{ij}|b_{ij}>0\}}{\max_{i,j} b_{ij}}$, and $F(M,\eta)$ is a constant that only depends on each $M_i(\cdot)$ and η.

In condition (2), $\bar{b}$ can be viewed as the average bid value of a bidder over time. Given that each bidder is at least interested in some fractions of the keywords, this average will go to a certain constant as n becomes large. Also, η can be viewed as the ratio between the value of the smallest non-zero bid and the highest bid. In practice, this is often bounded below by a constant by enforcing a reserve price and a maximum price for any single bid. The exact functional form of $F(M,\eta)$ is somewhat complicated, and is given in Proposition 1. Just to give an example, if we choose $M_i(x) = x^p$ $(0 < p < 1)$, then $F(M,\eta) = \frac{2}{\eta^{(2-p)/(1-p)}}$. Therefore, condition (2) can be viewed as simply requiring the total number of inputs is large, which is often the case in practice. For example, in the Adwords problem, n is the number of keyword searches in a certain period, and for instance, Google receives more than 5 billion searches per day. Even if we focus on a specific category, the number can still be in the millions. Thus, this condition is reasonable. We note that most learning algorithms in the literature make similar requirements, see Agrawal et al. (2014); Devanur and Hayes (2009), and Molinaro and Ravi (2014). Furthermore, as we will show in our numerical tests, our algorithm performs well even for problems with sizes that are significantly smaller than the condition requires, which validates the potential usefulness of our algorithm.

To propose an algorithm that achieves near-optimal performance, the main idea is to utilize the observed data in the allocation process. In particular, since the input data arrives in a random order, using the past input data and projecting it into the future should present a good approximation for the problem. To mathematically capture this idea, we use a primal-dual approach. We obtain the dual optimal solutions to suitably constructed optimization problems and use them to assist with future allocations. We first propose a one-time learning algorithm (OLA, see Section 2) that only solves an optimization problem once at time ϵn. By carefully examining this algorithm, we prove that it achieves near-optimal performance when the inputs satisfy certain conditions. However, the conditions are stronger than those stated in Theorem 1. To improve our algorithm, we further propose a dynamic learning algorithm (DLA, see Section 3). The dynamic learning algorithm makes better use of the observed data and updates the dual solution at a geometric pace, that is, at time ϵn, $2\epsilon n$, $4\epsilon n$ and so on. We show that these resolvings can lift the performance of the algorithm and thus prove Theorem 1. As one will see in the proof of the DLA, the choice of the resolving points perfectly balances the trade-off between *exploration* and *exploitation*, which are the main trade-offs in such types of learning algorithms.

It is worth mentioning that a similar kind of dynamic learning algorithm has been proposed in Agrawal et al. (2014) and further studied in Wang (2012) and Molinaro and Ravi (2014). However, those works only focus on linear objectives. In our analysis, the nonlinearity of the objective function presents a non-trivial hurdle since one can no longer simply analyze the revenue generated in each time