Using Predicted Weights for Ad Delivery*

Thomas Lavastida[1], Benjamin Moseley[1], R. Ravi[†1], and Chenyang Xu[‡2]

[1]Carnegie Mellon University, USA
{tlavasti, moseleyb, ravi}@andrew.cmu.edu
[2]Zhejiang University, China
xcy1995@zju.edu.cn

Abstract

We study the performance of a proportional weights algorithm for online capacitated bipartite matching modeling the delivery of impression ads. The algorithm uses predictions on the advertiser nodes to match arriving impression nodes fractionally in proportion to the weights of its neighbors. This paper gives a thorough empirical study of the performance of the algorithm on a data-set of ad impressions from Yahoo! and shows its superior performance compared to natural baselines such as a greedy water-filling algorithm and the ranking algorithm.

The proportional weights algorithm has recently received interest in the theoretical literature where it was shown to have strong guarantees beyond the worst-case model of algorithms augmented with predictions. We extend these results to the case where the advertisers' capacities are no longer stationary over time. Additionally, we show the algorithm has near optimal performance in the random-order arrival model when the number of impressions and the optimal matching are sufficiently large.

1 Introduction

There has been recent interest in augmenting online algorithms with machine-learned prediction. This line of work has lead to new models of algorithm analysis for going beyond worst-case analysis [23, 21, 17, 3]. The theoretical models considered in these works have led to the development of new algorithms which incorporate learned parameters (i.e. predictions) along with theoretical guarantees depending on the quality of the predictions. Typically, an algorithm's performance is parameterized by the error in the prediction. With a perfect prediction, an algorithm's performance should be stronger than the best worst-case algorithm. Additionally the algorithm's performance should be robust to moderate error in the predicted parameters.

An exciting question is how close the model is to practice and how to leverage it to develop improved practical algorithms. The goal of this paper is to show that the model is closely tied to practice for the online matching problem that arises in impression ad allocation and to demonstrate the empirical efficiency of a recently proposed algorithm based on proportional weights.

Capacitated Online Matching: Capacitated online matching is a fundamental problem faced in practice and is a special case of the Adwords problem [25]. In the problem, there is a set of known advertisers (offline) that each have a capacity (budget). Impressions arrive online that have edges to an arbitrary subset of advertisers. An impression can be matched (fractionally) to at most one advertiser. The goal is to match as many impressions as possible under the capacity constraints.

It is well-known that the best deterministic (greedy) algorithm is $\frac{1}{2}$-competitive. Allowing for randomization, the ranking algorithm is known to be $(1 - \frac{1}{e})$-competitive and this is the best possible competitive ratio for any online algorithm [19].

The best worst-case algorithm is far from optimal. Squeezing out the best performance in practice is fundamental in applications such as the Adwords problem [10].

An emerging line of work [21, 22, 5] has suggested that perhaps better matching algorithms exist if they use learned information about known real world match-

ing instances. That is, in practice there is often lots of data available on prior matching instances (e.g. the instance from yesterday). The idea is that an algorithm can use information from prior instances to perform better in future problem instances. Information learned from past data is referred to as a prediction. This work has the potential to offer new algorithmic ideas for solving matching instances in practice.

We consider the proportional weights algorithm, which has been suggested in this line of work.

Proportional Weights for Online Matching: This paper considers a recently proposed proportional weights algorithm for online matching. The algorithm was first proposed by Agrawal et al. [2] and further developed in [21, 22].

Let $G = (I, A, E)$ be a bipartite graph with capacities $C \in \mathbb{Z}_+^A$ on A. We refer to I as impressions and A as advertisers. We use $m = |I|$ and $n = |A|$ for the number of impressions and advertisers. Each advertiser (impression) has a subset N_a (N_i) of neighbors in the graph. In this paper we consider the fractional matching problem that is represented by the following linear program[1].

$$
\begin{aligned}
\text{(1.1)} \quad &\max & &\sum_{ia \in E} x_{ia} \\
&\text{s.t.} & &\sum_{a \in N_i} x_{ia} \leq 1 & &\forall i \in I \\
& & &\sum_{i \in N_a} x_{ia} \leq C_a & &\forall a \in A \\
& & &x_{ia} \geq 0 & &\forall ia \in E
\end{aligned}
$$

The proportional weights algorithm assigns each advertiser $a \in A$ a weight $\alpha_a > 0$. The vector of weights $\alpha \in \mathbb{R}_+^A$ on the advertisers encodes a fractional assignment of impressions to advertisers in the following way.

$$
\text{(1.2)} \quad x_{ia}(\alpha) = \frac{\alpha_a}{\sum_{a' \in N_i} \alpha_{a'}}
$$

That is, an impression is assigned to the advertisers in its neighborhood proportionally according to α. Notice that the allocation of each impression is independent of the others. Thus, if a set of weights is given a priori then the weights can be used to assign impressions online.

This algorithm does not consider if an advertiser has been saturated and may assign extra impressions to an advertiser above its capacity. These impressions are effectively not allocated. In the online setting, we consider an improved version that uses the weights to assign the impression proportionally, but only among the neighborhood of advertisers that have remaining capacity. This is the natural adaptation to the case where an advertiser becomes saturated.

Agrawal et al. [2] showed that for any $\epsilon > 0$, there exists a set of weights $\alpha \in \mathbb{R}_+^A$ such that the allocation given by (1.2) is a $(1-\epsilon)$-approximate solution to (1.1): the running time to arrive at such weights is inversely proportional to ϵ^2. This establishes that there exists a set of weights giving a high quality matching; notice that it is not obvious that such weights exist in the first place.

The work of Agrawal et al. [2] was interested in this proportional weights algorithm because they give a static assignment of impressions (order independent) as well as allowing for each impression to be assigned only knowing the neighborhood of the impressions, which is useful for distributed algorithms.

Later these weights were considered in the algorithms augmented with predictions model [21, 22]. Lavastida et al. [22] showed that predicting these weights can be used to go beyond the worst-case for online matching. This prediction could come from computing the weights from prior instances of matching. The work of Agrawal et al. [2] imply the weights give near optimal performance if predicted perfectly. [22] showed that the weights are instance-robust. Informally, this guarantees that if the weights give good performance on one instance, then they have strong performance on similar instances. Moreover, [22] showed that if the matching instance is drawn from an unknown product distribution then weights that give a near optimal solution are learnable in the PAC learning model (learnability). This suggests that weights can be learned in practice from prior matching instances and used on future instances to get strong online performance.

This line of work begs the question, does the proportional weights algorithm perform well empirically? In particular, if weights are computed from prior instances of online matching can they be used to give strong performance on future instances of the matching problem in practice? For instance, can learning the weights on yesterdays data be used on today's online instance? Understanding these questions has the potential to influence matching algorithms in numerous applications.

Results: This paper's goal is to demonstrate the empirical effectiveness of the proportional weights algorithm. To do so, we consider a data set obtained from Yahoo! [29] on the Adwords problem [25]. This data set gives instances of advertisers and impressions over multiple days. We propose two algorithms utilizing pre-

---

[1]The more general AdWords problem has an objective coefficient for each allocated impression representing different values of an impression for different advertisers.

dictions, one is the standard proportional weights algorithm while the other is an improved version.

We use the following strong benchmarks. One is the water-filling algorithm [18], which fractionally allocates the current impression so that it maximizes the minimum occupied proportion of its capacity among its neighbours. The other is the randomized ranking algorithm, which uses a single random ordering of the advertisers and assigns each impression to the available advertiser in its neighborhood with highest priority. In the following results we consider several methods for setting the advertisers' capacities and the impression arrival orders.

- Our improved proportional weights algorithm significantly outperforms the standard version.

- Fix a single day. Consider weights that are computed from a random sample of the day's impressions and use them for the remaining impressions online. The improved weights algorithm is consistently ahead of the baselines, often giving a near optimal matching. This shows the impressions can be learned from a sample of a day's impressions and used on the remaining impressions effectively. This empirically demonstrates that the weights are learnable.

- Next we consider learning weights on prior days and using them on future days. The impression distribution varies drastically from day to day. Despite this high variance, our improved weights algorithm has stronger performance than the baselines on every day tested, demonstrating robustness.

To complement our results, we give two theoretical results. First, we consider the weights in the random order arrival model. We show that the weights give a $(1 - \epsilon)$-approximate online matching in the random order model for any constant $\epsilon > 0$. This algorithm uses the first σ fraction of the arriving impressions to compute the weights (i.e. learn the weights) and then applies them to the remaining instance. In the result below, m refers to the number of impressions arriving online while n is the number of (offline) advertisers. This gives a similar result to prior work in the random order model [10], but uses proportional weights instead of a primal-dual scheme.

THEOREM 1.1. *There exists an algorithm which is $(1 - \epsilon)$-competitive with probability $1 - \delta$ for online matching in the random order model whenever $m = \Omega(\frac{n^2}{\sigma \epsilon^2} \log(\frac{n}{\delta}))$ and $\mathrm{OPT} \geq \epsilon m$ for any $\epsilon, \delta, \sigma \in (0, 1)$.*

Next, we give a theorem that demonstrates the robustness of the weights. This is an extension of the robustness result in [22]. Intuitively, we show that predicted weights are robust to modest changes in the input, including changes in advertiser capacity.

Consider a problem instance where advertiser a has capacity C_a and there is a set of impression types, where each type is defined by a subset of advertisers. Each impression of the same type has the same set of advertisers as neighbors. Let C_i be the number of impressions of type i.

In Section 5 (Theorem 5.2), we show that if a fixed set of weights can match at least $(1 - \epsilon)\mathrm{OPT}$ impressions on a given instance defined by C, then the same weights have value at least $(1 - \epsilon)\mathrm{OPT} - 2\eta$ on any instance C' where $\eta = \sum_i |C_i - C_i'| + \sum_a |C_a' - C_a|$. Here η measures the difference in the two instances. This gives a theoretical explanation for the strong empirical performance of the weights even when advertiser capacities change and the impression volumes vary.

2 Related Work

Algorithms with Predictions: In this paper we do a practical evaluation of online matching algorithms using predictions learned from past data. There has been significant recent interest in analyzing online algorithms in the presence of erroneous predictions [23, 28, 16, 4, 27, 13, 3, 21, 8, 31, 32, 5]

Antoniadis et al. [5] looks at online weighted bipartite matching problems with predictions in the random order model. In this setting each offline node can be matched at most once and the edges are weighted. This differs from our setting where we consider the unweighted problem with capacities on the offline nodes.

Data Driven Algorithm Design: Using past data to learn the weights for our algorithm can be seen as a case of data driven algorithm design [6, 15, 7]. This line of work is concerned with using past problem instances to learn an algorithm from some class of algorithms which will perform well on future instances drawn from the same population as the past instances. In particular, the sample complexity (i.e. the number of past instances used by the learning algorithm) is of particular importance.

Practical Algorithms for Online Matching: In addition to the deep theoretical understanding of online matching algorithms, there has been effort to develop algorithms that work well on real data sets. Zhou et al. [30] develop a robust online weighted matching algorithm based on primal-dual schemes for the random order model which account for changes in the underlying distribution and evaluate it on a display ad data set. Ma et al. [24] develop an algorithm for online assortment

optimization and evaluate it on data from a hotel chain, but their emphasis is on revenue maximization rather than capacity allotment. Chen et al. [9] develop a real-time bidding algorithm for display ad allocation and evaluate it on a proprietary display ad data set; their methods closely mirror the water-filling algorithm we study.

Random Order Model: There has been a line of work in analyzing online algorithms in the random order model for matching problems and more general packing integer linear programs [10, 12, 26, 1, 11, 20, 14]. These algorithms are usually based off of some sort of primal-dual approach. We give an alternative approach for online capacitated matching in the random order model based on using proportional weights.

3   Preliminaries

Recall the capacitated online matching problem represented by the linear program (1.1). In the online version of the problem, only the advertisers and their capacities are initially known to the algorithm. The impressions I (along with their neighbors) are revealed to the algorithm one at a time and it must commit to an assignment $\{x_{ia}\}_{a \in N_i}$ satisfying the constraints in (1.1). We consider the case when the set of impressions I are decided by an adversary, but revealed to the algorithm in random order.

Proportional Weights: We consider fractional solutions to (1.1) parameterized by weights $\alpha \in \mathbb{R}_+^A$, using the proportional allocation scheme in (1.2).

Observe that given a fixed set of weights α, the allocation $x_{ia}(\alpha)$ is order independent and thus is suitable for the online setting. Next, note that $x_{ia}(\alpha)$ always satisfies the first constraint of (1.1) with equality, but it may not satisfy the second constraint. Any reasonable way of decreasing the allocation to satisfy the second constraint suffices, so we define $R_a(\alpha) = \min\{\sum_{i \in I} x_{ia}(\alpha), C_a\}$ to be a's contribution to the size of the fractional matching. We utilize the following theorem due to Agrawal et al. [2].

DEFINITION 3.1. *For* $T \in \mathbb{Z}_+$ *and* $\epsilon > 0$, *define* $\mathcal{A}(T, \epsilon) = \{\alpha \in \mathbb{R}_+^A \mid \alpha_a = (1 + \epsilon)^k, k \in [T]\}$.

THEOREM 3.1. *([2]) For any bipartite graph* $G = (I, A, E)$, *capacities* $C \in \mathbb{Z}_+^A$, *and* $\epsilon > 0$ *there exists* $T \in \mathbb{Z}_+$ *and* $\alpha \in \mathcal{A}(T, \epsilon)$ *such that*

$$\sum_{a \in A} R_a(\alpha) \geq (1 - \epsilon)\text{OPT}.$$

In particular, we can take $T = O(\frac{1}{\epsilon^2} \log(\frac{n}{\epsilon}))$. *Moreover, there is a polynomial time algorithm which computes* α.

4   Proportional Weights under Random Orders

This section considers the capacitated online matching problem in the random order model and shows that the proportional weights are learnable in this model. Under some mild assumptions, the weights computed by the first small portion of impressions can obtain a good performance for the whole instance.

We first state the definition of the random order model. Suppose that $I = \{i_1, i_2, \ldots, i_m\}$ and let $\gamma : [m] \to [m]$ be a permutation. Denote $I(\gamma)$ to be the sequence $(i_{\gamma(1)}, i_{\gamma(2)}, \ldots, i_{\gamma(m)})$, i.e. consider revealing the impressions I in the order induced by γ. If γ is a permutation drawn uniformly at random and $\delta \in (0, 1)$, then we say that an algorithm is c-competitive with high probability if for all impression sets I:

$$(4.3) \qquad \Pr[\text{ALG}(I(\gamma)) \geq c\text{OPT}(I)] \geq 1 - \delta.$$

where $\text{ALG}(I(\gamma))$ is the size of the matching when $I(\gamma)$ is given as input to the online algorithm and $\text{OPT}(I)$ is the optimal value of (1.1). Note that $\text{OPT}(I)$ does not depend on the ordering γ. We will use OPT instead of $\text{OPT}(I)$ when the context is clear.

Our algorithm takes the first σm impressions for some $\sigma \in (0, 1)$ and reduces the capacity of each advertiser by a σ factor, then computes proportional weights α using the algorithm of Theorem 3.1.

Denote the first σm impressions by $S \subseteq I$. Let $R_a(\alpha, S) = \min\{\sum_{i \in S} x_{ia}(\alpha), \frac{|S|}{m} C_a\}$. Similarly, let $\text{OPT}(S)$ be the size of a maximum cardinality matching on the graph $G' = (S, A, E)$ with capacities $C_a' = \frac{|S|}{m} C_a$. We think of $\sum_a R_a(\alpha, S)$ as the value obtained by weights α on an instance restricted to the impressions in S and the capacities scaled down appropriately.

Algorithm 1 Proportional Weights in the Random Order Model
___

Input: $G = (I, A, E), C, \gamma, \sigma, \epsilon$.
Let S be the first σm impressions in $I(\gamma)$
Set $x_{ia} = 0$ for each $i \in S, a \in N_i$
Set $T = \Theta(\frac{1}{\epsilon^2} \log(\frac{n}{\epsilon}))$
Compute weights $\alpha \in \mathcal{A}(T, \epsilon)$ on $G' = (S, A, E)$ with capacities $C_a' = \sigma C_a \; \forall a \in A$
for each remaining impression i **do**
    For each $a \in N_i$, let $x_{ia}(\alpha) = \frac{\alpha_a}{\sum_{a' \in N_i} \alpha_{a'}}$
end for

We show that this algorithm performs well when the number of impressions m is large relative to the number of advertisers.

THEOREM 4.1. *Algorithm 1 is* $(1 - \epsilon)$-*competitive with probability* $1 - \delta$ *for online matching in the random order*

Algorithm 2 Proportional Weights (PW)

Input: $G = (I, A, E)$ where I and E arrive online, $\{C_a\}_{a \in A}$, predicted weights $\{\hat{\alpha}\}$
while an impression i comes **do**
    For each $a \in N_i$, let $x_{ia} = \frac{\hat{\alpha}_a}{\sum_{a' \in N_i} \hat{\alpha}_{a'}}$.
end while

Algorithm 3 Improved Proportional Weights (IPW)

Input: $G = (I, A, E)$ where I and E arrive online, $\{C_a\}_{a \in A}$, predicted weights $\{\hat{\alpha}\}$
while an impression i comes **do**
    Let $N^* \subseteq N_i$ be the unfull advertisers in its neighbourhood (A full advertiser is one whose capacity is equal to its current total allocation).
    For each $a \in N^*$, let $x_{ia} = \frac{\hat{\alpha}_a}{\sum_{a' \in N^*} \hat{\alpha}_{a'}}$.
end while

---

model whenever $m = \Omega(\frac{n^2}{\sigma \epsilon^2} \log(\frac{n}{\delta}))$ and OPT $\geq \epsilon m$ for any $\epsilon, \delta, \sigma \in (0, 1)$.

Our analysis applies two probabilistic arguments. First, we show that $\sum_a R_a(\alpha, S) \approx \sigma \sum_a R_a(\alpha)$ for the computed weights α with high probability. This involves a union bound over all possible weights in $\mathcal{A}(R, \epsilon)$. Second, we show that OPT$(S) \approx \sigma$OPT with high probability. This involves a union bound over cuts in the bipartite graph G. Formal versions of these two statements imply the theorem. See Appendix B for complete arguments.

5 Robustness of Proportional Weights on Similar Instances

A learning-augmented algorithm can be given directly if we can predict the proportional weights. See Algorithm 2 for the description. For this algorithm, Lavastida et al. [22] give a theoretical result of its competitive ratio under an assumption about advertiser capacities. This section extends this result by relaxing that assumption.

Define an impression vector for an instance to be the m-dimensional vector $(\ldots, C_i, \ldots)$, where each component corresponds to the supply of each impression i. In [22], Lavastida et al. prove that when the capacity of each advertiser is fixed, for any constant $\epsilon > 0$, the $(1 - \epsilon)$-approximated weights $\hat{\alpha}$ for instance $\hat{\mathcal{I}}$ has a competitive ratio at least $1 - \epsilon - 2\eta/$OPT on instance $\mathcal{I}$, where OPT is the optimal value of instance $\mathcal{I}$ and η is ℓ_1 norm between the impression vectors of these two instances. In this paper, we relax the condition that the capacity needs to be fixed, and obtain a similar theorem. Define an advertiser vector, similarly, to be the n-dimensional vector $(\ldots, C_a, \ldots)$, where each component corresponds to the capacity of each advertiser a.

THEOREM 5.1. *For any constant $\epsilon > 0$, with the $(1-\epsilon)$-approximated weights $\hat{\alpha}$ for instance $\hat{\mathcal{I}}$, algorithm PW has a competitive ratio at least*

$$1 - \epsilon - \frac{2\eta}{\text{OPT}}$$

on instance $\mathcal{I}$, where OPT is the optimal value of instance $\mathcal{I}$ and η is ℓ_1 norm between the impression

vectors of these two instances plus the ℓ_1 norm between two advertiser vectors.

The basic idea of this proof is first showing the difference between the performances of weights $\hat{\alpha}$ in the two instances is at most η and then proving the difference of these two instances' optimal values is also at most η with the help of a vertex cut. The details are deferred to Appendix C.

This simple algorithm is consistent, but not robust, which means that it will achieve good results if the prediction is accurate, but will perform badly if the prediction error is large. Several ideas are introduced in [22] to obtain robust algorithms with proportional weights. In this paper, instead of using the complicated algorithms in [22], we propose a very simple algorithm called *Improved Proportional Weights* (IPW) that can achieve a very competitive and robust performance in the experiments. The description is given in Alg. 3.

Clearly, algorithm IPW always matches more (fractional) impressions than algorithm PW and obtains a maximal matching, whose competitive ratio is at least $1/2$. Thus, we have the following theorem.

THEOREM 5.2. *For any constant $\epsilon > 0$, with the $(1-\epsilon)$-approximated weights $\hat{\alpha}$ for instance $\hat{\mathcal{I}}$, algorithm IPW has a competitive ratio at least*

$$\max\left\{1 - \epsilon - \frac{2\eta}{\text{OPT}}, \frac{1}{2}\right\}$$

on instance $\mathcal{I}$, where OPT is the optimal value of instance $\mathcal{I}$ and η is ℓ_1 norm between the impression vectors of these two instances plus the ℓ_1 norm between two advertiser vectors.

6 Experimental Results

In this section, we validate the performance of PW and IPW on realistic data empirically. We investigate two main aspects of applying predicted weights in practice:

- Learnability - we sample a small fraction of the data we assembled as training data (motivated by the

results in Section 4) and observe that this provides enough information to set weights that are superior in performance to the benchmarks.

- Robustness - we use data obtained over several days and examine the performance of our weights from previous day(s) on the current day's set of impression arrivals, and show consistent improved performance over the benchmarks across time.

First, we describe our data source and how we set up instances, such as the definition of impressions, the bipartite graph between impressions and advertisers, and how the capacities of the advertisers are set based on the supply of impressions.

6.1 Experimental Setup We used the Yahoo! Search Marketing Advertiser Bid-Impression-Click data, provided as part of their Advertising and Market data [29][2]. The data contains nearly 78 million records each containing an allocated advertising impression. Each record provides a day d, an anonymized account id (for the advertiser) a, a rank r, a set of anonymized keyphrases P, average bid b, the number of impressions C_P of the keyphrase P allocated to this advertiser, and clicks k out of these impressions. A row (d, a, r, P, b, C_p, k) indicates that on day d, advertiser a was allocated C_P copies of impression P at (average monetary) cost b and obtained k clicks. Since the impressions are presumably made available in a ranked order of preference of placement, an advertiser's placement for each of these impressions is also accompanied by its rank.

Our maximum cardinality matching instances are built from this data set, one for each day. The average bid and click data are ignored. We first define the vertex set, then the edge set and finally the supply and capacity functions.

Vertex Sets: The advertiser set A is the set of advertiser account ids. Keyphrases in the data are defined as a combination of a set of elementary (anonymized) keywords. We use this to construct the impression set I as follows. The keyphrase base set S is obtained by taking the union of all elementary keyphrases in P. We count the number of occurrences of each elementary keyphrase $p \in S$ and select the top 20 most frequent keyphrases (based on the supply of the impressions they are part of). Let S^* denote the set of these most popular keyphrases and define the impression set I to be the power set of S^* excluding the empty set, i.e., $I := 2^{S^*} \setminus \emptyset$. Each vertex $i \in I$ can thus be viewed as an impression type. Different keyphrases that have the

same intersection with S^* are this 'contracted' into the same impression vertex in this formulation.

Edge Set: Next we construct the edges. Define a mapping f from the given keyphrase sets $\{P\}$ to the impression set I by $f(P) = P \cap S^*$. For each row, add edge $(f(P), a)$ into the edge set. Additionally, for any subset $i \in I$ of $f(P)$, the edge (i, a) is also added. Although these keyphrases might not be assigned to advertiser a in this table, we assume that they are relevant and hence can be assigned to advertiser a[3].

Impression supply: Intuitively, on each day, the total impression sizes corresponding to different ranks should be the same. However, this is not the case in the provided data set, potentially due to sampling effects. Thus, for each day and for each keyphrase, we only retain the rows with the rank that contains the *maximum* total impression size and remove the rows with other ranks [4]. For each vertex $i \in I$, its impression supply C_i is the total size of the keyphrase sets that belongs to type i, i.e., $C_i = \sum_{f(P)=i} C_P$.

Advertiser Capacities: The appropriate capacity function of the advertiser set for creating challenging instances is not readily apparent. Thus, we consider three ways to set advertiser capacities.

1. **Random Quota:** For each impression vertex i, split its size C_i among its neighbourhood randomly. The capacity of each advertiser is set to be its final allocation obtained in this way.

2. **Max-min Quota:** Sort all impressions lexicographically and process each impression one by one: for each impression, allocate its supply to its neighbourhood such that the minimum allocation in the neighborhood is maximized. The capacity of each advertiser is set to be its final allocation at the end of this process.

3. **Least-degree Quota:** For each impression, assign its size equally to the advertisers with the least degree[5] in its neighbourhood.

The random method above tries to de-correlate an impression's capacity and its neighborhood's eventual demands. The second and the third are also designed

---

[3]While some advertisers may only seek out very specific combinations of keyphrases in their bidding, we assume that most advertisers are interested in impressions broadly matching their keyphrases of interest in our formulation, and ignore the existence of the former type of advertisers.

[4]The experimental results do not vary significantly if selecting other ranks in this process of defining the impression supply.

[5]In the graph constructed from the data set, the proportion of the least-degree neighbours is roughly half its total neighborhood for each impression.
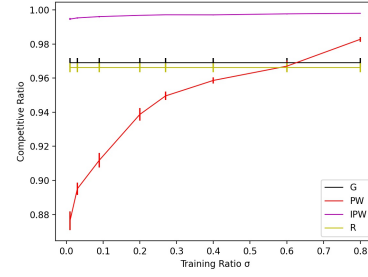
to deliberately avoid correlations between the neighborhood's demands.

For the daily instance constructed in this way, there are roughly 4500 advertisers and 85 impressions with non-zero sizes, with roughly 8000 edges between them that can be used to allocate a total supply of about 1.8 million copies of these impressions.
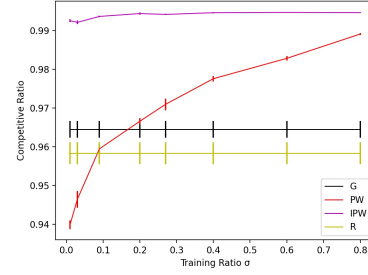
Computational Setup and Methods. We conducted the experiments[6] on a machine running Ubuntu 18.04 with 12 i7-7800X CPUs and 48 GB memory. In the experiments, our algorithms (PW and IPW) are compared to the competitive greedy/water-filling algorithm (G) [18] and ranking algorithm (R) [19]. The greedy 'water-filling' algorithm (G) fractionally allocates the current impression so that the proportion of capacities of all its neighbors capacities that are filled are as equal as possible (Imagining these filled proportions to be water levels, the allocation fills the lowest levels until they all rise to include another in this set, and so on). The ranking algorithm (R) uses a single random permutation of the advertisers to set a priority order among the neighborhoods of any arriving impressions and allocates the impression in this order. All algorithms are implemented in Python 3.6.11. All results are averaged over 4 runs.

6.2 Learnability To test the learnability of our algorithms, for each daily instance we sample a σ proportion of impressions (for a sampling parameter $\sigma \in [0, 1]$) to construct the training instance. The graph connectivity is the same as in the original data set while the advertiser capacities are constructed using these impressions and one of the three rules above. We compute proportional weights on this training instance and use them in Algorithms 2 and 3 for that day's whole instance.
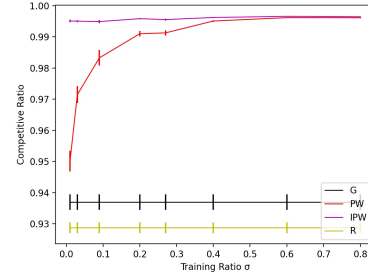
We investigate the performance when impressions arrive in a random order or in an adversarial order. We measure performance by the traditional measure of competitive ratio. Since the instances were engineered so that all impressions are allocable, this is simply the fraction of all impressions that were assigned by each of the methods. Since it is hard to find the most adversarial order for each algorithm for the given impressions, we set up five different arrival orders and use the worst performance of an algorithm among these orders to approximate its performance in an adversarial order. Finally, we illustrate the performance of the algorithms across the daily order, which we think is closest to the order which occurs in practice. The description of the daily order is given later.

[6]Code is available at `https://github.com/Chenyang-1995/PredictiveWeights`

(a) Random Quota
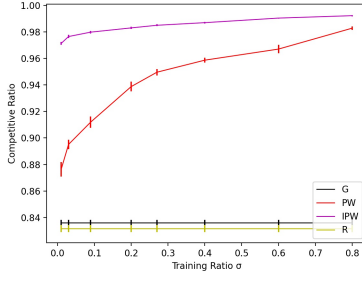

(b) Max-min Quota

(c) Least-degree Quota

Figure 1: The performance of each algorithm on the test data when impressions arrive in a random order, plotted as a function of the training ratio.
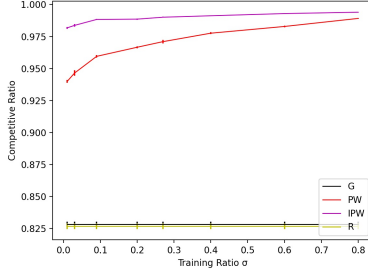
Random Order. The performance of each algorithm in random order is shown in Figure 1, where each plot corresponds to one of the three quota allocation rules[7]. All results are obtained by taking the average performance of ten days' instances, while the performance of each day is evaluated on 4 separate runs. The x-axis of each plot is the training ratio σ, indicating that σ proportion of impressions are sampled to serve as the training data. Note that in these figures, the learned weights from the σ proportion are evaluated on the whole data set for the same day. As σ increases, the baseline greedy (G) and ranking (R) algorithms'
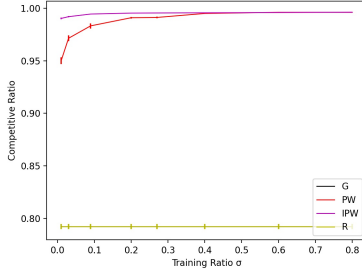
[7]Note the varying scales in the Y-axes in many of the figures.
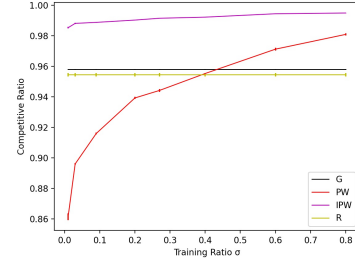
(a) Random Quota
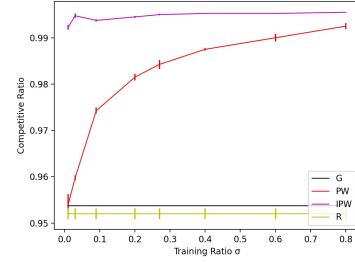

(b) Max-min Quota

(c) Least-degree Quota

Figure 2: The worst performance of each algorithm on the test data among five impression arriving orders tested, plotted as a function of the training ratio.

(a) Random Quota

(b) Max-min Quota

(c) Least-degree Quota

Figure 3: The performance of each algorithm when impression arrives in a daily order, as a function of the training ratio. Impressions within a day are assumed to arrive in random order.

performance remain unchanged since they do not use the training data, while algorithms PW and IPW show improving performances. In Figure 1a and Figure 1b, algorithm PW has a worse performance than algorithm G and algorithm R initially, but obtains a better performance when σ becomes 0.6 and 0.1 respectively. Compared to other algorithms, IPW gives the best performance. We see that even with one percent of the data ($\sigma = 0.01$), IPW outperforms the traditional online algorithms.
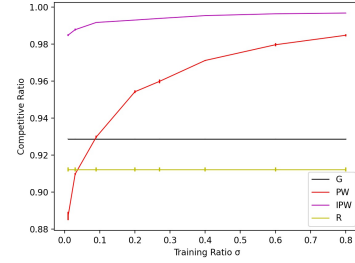
Approximating Adversarial Orders. To study the performance of these algorithms when impressions arrive in an adversarial order, we set up following five arriving orders and check the worst performance.

1. **Random Order:** Impressions arrive randomly.

2. C_i**-descending Order:** Sort all impressions in the descending order of their capacities and let impressions arrive in this order.

3. C_i**-ascending Order:** All impressions arrive in the opposite order of the C_i-descending order (i.e. non-descending order of their capacities).

4. C_a**-descending Order:** Define the neighbourhood capacity of an impression to be the sum of the supplies of its neighbouring advertisers. Sort all impressions in the descending order of their neighbourhood capacities and let impressions arrive in this order.
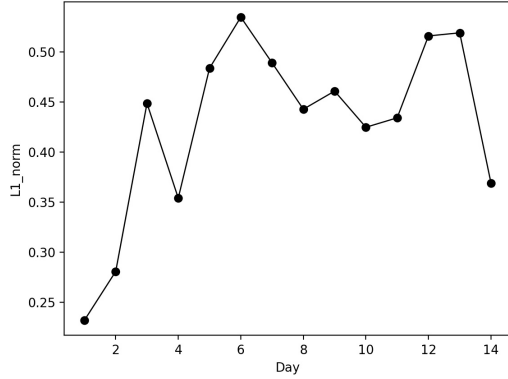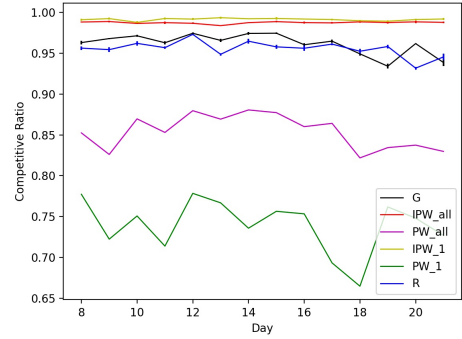
Figure 4: The ℓ_1 norm between the impression vectors on day 0 and day $i > 0$.

5. **C_a-ascending Order:** All impressions arrive in the opposite order of the C_a-descending order.
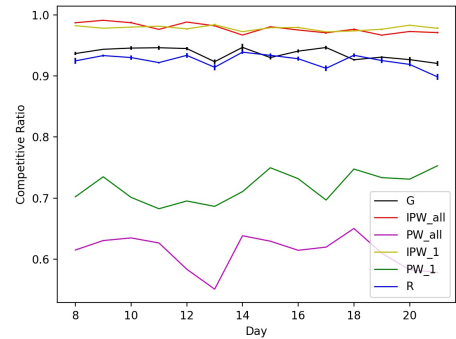
The worst performance of each algorithm (over these five orders) is shown in Figure 2. We observe that the algorithms based on proportional weights are much more stable than other algorithms over different arrival orders (More experimental results are provided in Appendix D). The performance of algorithms PW and IPW mostly remain unchanged because their allocation policies are unrelated to the impression arrival order, unlike greedy and ranking, and IPW remains the best. The performances of algorithm G and R vary significantly when the arriving order changes. Their worst performances are usually realized in the C_i-descending order and C_a-descending order respectively.

Daily Order. Finally, we show each algorithm's performance when impressions arrive in a daily order in Figure 3. We combine the instances on 7 days and get a stacked instance, where the vertex set is the union of vertex sets in these graphs and each vertex's capacity is the sum of its capacities in the instances. Say that impressions arrive in daily order if all impressions on day d arrive before the impressions on day $d + 1$ while inside any day d, impressions arrive in a random order. The results show a similar trend as in Figure 1 with one difference being that the performances of algorithm G and R decrease slightly. Thus, the value σ where algorithm PW surpasses algorithm G and R becomes slightly smaller.

6.3 Robustness. This subsection considers the robustness. As mentioned above, the instances on different days are quite different from each other. We visualize this by showing the ℓ_1 norm between the impressions on the first day (Day 0) and the normalized impression

(a) Max-min Quota

(b) Least-degree Quota

Figure 5: The performance of algorithms on different days where PW and IPW use weights from previous days on the current day, and the impression arrival order is random within each day. Note that the starting day is set to be day 8 in order to collect more training data for algorithm PW_all and IPW_all.

vectors on following days, using the impressions from the given data set across the first 15 days. In our setting, the vector has dimension $2^{20} - 1$. As shown in Figure. 4, the difference between two days could be very large. This suggests it could be hard for proportional weights to work well when learned on one day and used on another. Surprisingly, our experiments shows empirically that weights are robust across days despite these large differences in the problem instances. Since we wish to model some level of correlation between the instances on different days, the random quota is not tested in this experiment.

The robustness experiment simulates how the proportional weights may be used in practice. We predict the weights based on previous day(s) and use the weights to allocate impressions for a different day. We consider either computing weights on an instance that

is just the previous day or computing the weights from *all* prior days impressions. We use "_all" and "_1" to denote the weights of all previous days and yesterday respectively. The results are shown in Figure 5 [8].

As mentioned in the beginning of this section, algorithm PW will not be robust if the predicted weights are inaccurate. The performance of algorithm PW_all and PW_1 imply a large error in the prediction. However, even with large prediction error IPW achieves the best performance.

6.4 Conclusions We see the following trends from the experiments.

- The Improved Proportional Weights algorithm is consistently the best algorithm considered, giving near optimal performance on all instances tested.

- The weights are learnable given a random sample of a problem instance. In particular, such weights leads to the improved proportional weights algorithm having near-optimal performance.

- The weights are robust to large changes in the problem instance. Across days, the advertiser capacities and the supply of the impressions change by large margins. Still, the Improved Proportional Weights algorithm has strong performance.

These experiments show that (1) the proportional weights algorithm is a strong algorithm for online matching and (2) the theoretical results on the weights can be seen empirically. This empirically shows a connection between the algorithms augmented with predictions model and practice. We hope these results stimulate further experimental investigation of algorithms augmented with predictions.

References

[1] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Oper. Res.*, 62(4):876–890, 2014.

[2] Shipra Agrawal, Morteza Zadimoghaddam, and Vahab Mirrokni. Proportional allocation: Simple, distributed, and diverse matching with high entropy. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 99–108, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] Keerti Anand, Rong Ge, and Debmalya Panigrahi. Customizing ml predictions for online algorithms. *ICML 2020*, 2020.

[4] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 345–355. PMLR, 13–18 Jul 2020.

[5] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7933–7944. Curran Associates, Inc., 2020.

[6] Maria-Florina Balcan. Data-driven algorithm design, 2020.

[7] Maria-Florina Balcan, Dan F. DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? *CoRR*, abs/1908.02894, 2019.

[8] Aditya Bhaskara, Ashok Cutkosky, Ravi Kumar, and Manish Purohit. Online learning with imperfect hints. *CoRR*, abs/2002.04726, 2020.

[9] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 1307–1315, New York, NY, USA, 2011. Association for Computing Machinery.

[10] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings 10th ACM Conference on Electronic Commerce (EC-2009), Stanford, California, USA, July 6–10, 2009*, pages 71–78, 2009.

[11] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *J. ACM*, 66(1):7:1–7:41, 2019.

[12] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, volume 6346 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2010.

[13] Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of*

---

[8]If one considers the worst performance among the five orders, the performance of algorithm G and R will decrease significantly as in previous experiments, while PW and IPW remain stable.

Machine Learning Research, pages 2319–2327. PMLR, 2019.

[14] Anupam Gupta and Marco Molinaro. How the experts algorithm can help solve lps online. *Math. Oper. Res.*, 41(4):1404–1431, 2016.

[15] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM J. Comput.*, 46(3):992–1017, 2017.

[16] Piotr Indyk, Frederik Mallmann-Trenn, Slobodan Mitrovic, and Ronitt Rubinfeld. Online page migration with ML advice. *CoRR*, abs/2006.05028, 2020.

[17] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 69:1–69:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[18] Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

[19] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358. ACM, 1990.

[20] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. *SIAM J. Comput.*, 47(5):1939–1964, 2018.

[21] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877. SIAM, 2020.

[22] Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing, 2020.

[23] Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[24] Will Ma and David Simchi-Levi. Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *Oper. Res.*, 68(6):1787–1803, 2020.

[25] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

[26] Marco Molinaro and R. Ravi. The geometry of online packing linear programs. *Math. Oper. Res.*, 39(1):46–59, 2014.

[27] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 9684–9693, 2018.

[28] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1834–1845. SIAM, 2020.

[29] Yahoo! Webscope. Yahoo! search marketing advertiser bid-impression-click dataset version 1.0. `http://research.yahoo.com/Academic_Relations`. Accessed 2020-12-1.

[30] Yu-Hang Zhou, Chen Liang, Nan Li, Cheng Yang, Shenghuo Zhu, and Rong Jin. Robust online matching with user arrival distribution drift. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):459–466, Jul. 2019.

[31] Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling, 2020.

[32] Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms, 2020.

A   Concentration Inequalities

THEOREM A.1. (COROLLARY 2.4 IN [14]) *Let* $Y = \{Y_1, ..., Y_n\}$ *be a set of real numbers in the interval* $[0, 1]$. *Let* S *be a random subset of* Y *of size* s *and let* $Y_S = \sum_{i \in S} Y_i$. *Setting* $\mu = \frac{1}{n} \sum_i Y_i$, *we have that for every* $\tau > 0$,

$$\Pr[|Y_S - s\mu| \geq \tau] \leq 2 \exp\left(-\min\left\{\frac{\tau^2}{8s\mu}, \frac{\tau}{2}\right\}\right).$$

B   Proofs for the Random Order Model

LEMMA B.1. *Suppose that the weights* α *computed in Algorithm 1 satisfy*

1. $\Pr\left[\sum_a R_a(\alpha, S) > \sigma \sum_a R_a(\alpha) + \epsilon^2 \sigma m\right] \leq \frac{\delta}{2}$

2. $\Pr\left[\text{OPT}(S) < \sigma \text{OPT} - \epsilon^2 \sigma m\right] \leq \frac{\delta}{2}$

Then Algorithm 1 is $(1 - O(\epsilon))$-*competitive with probability* $1 - \delta$ *whenever* $\text{OPT} \geq \epsilon m$.

Proof. The key fact we use is that for any S, by Theorem 3.1, $\sum_a R_a(\alpha, S) \geq (1 - \epsilon)\text{OPT}(S)$ since we computed α using S. If neither event above occurs, then

we have

$$\sum_a R_a(\alpha) \geq \frac{1}{\sigma}\sum_a R_a(\alpha, S) - \epsilon^2 m$$

$$\geq \frac{(1-\epsilon)}{\sigma}\mathrm{OPT}(S) - \epsilon^2 m$$

$$\geq (1-\epsilon)\mathrm{OPT} - 2\epsilon^2 m$$

$$\geq (1 - O(\epsilon))\mathrm{OPT}.$$

Indeed, neither event occurs with probability at least $1 - \delta$ by a union bound. □

Our goal is to now show that the two properties above hold whenever $m = \Omega(\frac{n^2}{\sigma\epsilon^2}\log(\frac{n}{\delta}))$. Let's start with the first property. In order to show this property we need the following lemma.

LEMMA B.2. *Let S be a subset I of size σm and $\alpha \in \mathcal{A}(T,\epsilon)$. For any $a \in A$, if $R_a(\alpha, S) \geq \sigma R_a(\alpha)$ then $\sum_{i \in S} x_{ia}(\alpha) \geq \sigma \sum_{i \in I} x_{ia}(\alpha)$.*

Proof. Fix $a \in A$ and suppose that $R_a(\alpha, S) \geq \sigma R_a(\alpha)$. Using the definition of $R_a(\alpha, S)$ and $R_a(\alpha)$ we have

$$\min\left\{\sum_{i \in S} x_{ia}(\alpha), \sigma C_a\right\} = R_a(\alpha, S) \geq \sigma R_a(\alpha)$$

$$= \sigma \min\left\{\sum_{i \in I} x_{ia}(\alpha), C_a\right\}$$

$$= \min\left\{\sigma\sum_{i \in I} x_{ia}(\alpha), \sigma C_a\right\}$$

A case analysis of this shows that $\sum_{i \in S} x_{ia}(\alpha) \geq \sigma \sum_{i \in I} x_{ia}(\alpha)$. □

This lemma allows us to bound the probability that $R_a(\alpha, S) > (1+\epsilon)\sigma R_a(\alpha)$ by instead bounding the probability that $\sum_{i \in S} x_{ia}(\alpha) \geq (1+\epsilon)\sigma \sum_{i \in I} x_{ia}(\alpha)$, which is accomplished via standard concentration inequalities.

LEMMA B.3. *For each $\alpha \in \mathcal{A}(T,\epsilon)$ and $a \in A$, if $m = \Omega(\frac{n}{\epsilon^2\sigma}\log(\frac{2n|\mathcal{A}(T,\epsilon)|}{\delta}))$ then*

$$\Pr[R_a(\alpha, S) > \sigma R_a(\alpha) + \epsilon^2\sigma\frac{m}{n}] \leq \frac{\delta}{2n|\mathcal{A}(T,\epsilon)|}.$$

Proof. By Lemma B.2, we have

$$\Pr[R_a(\alpha, S) > \sigma R_a(\alpha) + \epsilon^2\sigma\frac{m}{n}]$$

$$\leq \Pr[\sum_{i \in S} x_{ia}(\alpha) > \sigma\sum_{i \in I} x_{ia}(\alpha) + \epsilon^2\sigma\frac{m}{n}].$$

Since S is a random subset of I of size σm, $x_{ia} \in [0,1]$, and $\mathbb{E}[\sum_{i \in S} x_{ia}(\alpha)] = \sigma \sum_{i \in I} x_{ia}(\alpha)$, we can apply Theorem A.1 to the right hand side to get

$$\Pr\left[\sum_{i \in S} x_{ia}(\alpha) > \sigma\sum_{i \in I} x_{ia}(\alpha) + \epsilon^2\sigma\frac{m}{n}\right]$$

$$\leq \exp\left(-\epsilon^2\sigma\frac{m}{2n}\right) \leq \frac{\delta}{2n|\mathcal{A}(T,\epsilon)|}$$

where in the last step we use the condition on m. □

We use a similar strategy for showing the second condition which regards OPT and OPT(S): find a quantity which can be captured as a sum and apply concentration. In this case we use the fact that there exists a cut/vertex cover which equals the size of the maximum cardinality (fractional) matching. The following theorem is folklore (see e.g. Claim 1 in [2]). For $B \subseteq A$ let $N(B) = \bigcup_{a \in B} N_a$.

THEOREM B.1. *Let $G = (I, A, E)$ be a bipartite graph with capacities $C \in Z_+^A$ on A. If OPT is the optimum value of (1.1), then for all partitions of A into $A' \cup A''$ we have $\mathrm{OPT} \leq |N(A')| + \sum_{a \in A''} C_a$. Moreover, there exists a partition of A into $A_0 \cup A_1$ such that $\mathrm{OPT} = |N(A_1)| + \sum_{a \in A_1} C_a$.*

We can apply this theorem directly to G as well as to $G' = (S, A, E)$ as in Algorithm 1. The capacities for G' are $C'_a = \sigma C_a$, so for any $S \subseteq I$ of size σm, we get that there is a partition $A = A_0 \cup A_1$ such that $\mathrm{OPT}(S) = |N(A_0) \cap S| + \sum_{a \in A_1} \sigma C_a$. Now we can write $|N(A_0) \cap S| = \sum_{i \in S} Y_i$, where $Y_i = \mathbf{1}_{\{i \in N(A_0)\}}$, which will allow us to apply Theorem A.1.

LEMMA B.4. *If $m = \Omega(\frac{1}{\epsilon^2\sigma}(n + \log\frac{2}{\delta}))$, then*

$$\Pr[\mathrm{OPT}(S) < \sigma\mathrm{OPT} - \epsilon^2\sigma m] \leq \frac{\delta}{2^{n+1}}.$$

Proof. By Theorem B.1 there is a partition such that $A = A_0 \cup A_1$ such that $\mathrm{OPT}(S) = |N(A_0) \cap S| + \sum_{a \in A_1} \sigma C_a$. Now write $|N(A_0) \cap S| = \sum_{i \in S} Y_i$, where $Y_i = \mathbf{1}_{\{i \in N(A_0)\}}$. Computing the expectation of this sum we have $\mathbb{E}[\sum_{i \in S} Y_i] = \sigma|N(A_0)|$. Now we have

$$\Pr[\mathrm{OPT}(S) < \sigma\mathrm{OPT} - \epsilon^2\sigma m]$$

$$\leq \Pr\left[\sum_{i \in S} Y_i \leq \sigma|N(A_0)| - \epsilon^2\sigma m\right]$$

$$\leq \exp\left(-\epsilon^2\sigma\frac{m}{2}\right) \leq \frac{\delta}{2^{n+1}}$$

where we use the condition on m in the last step. □

Now we can prove the main result about Algorithm 1.

Proof. [of Theorem 4.1] Our goal is to show that the properties in Lemma B.1 hold, then apply its conclusion. Recall that we assume $m = \Omega(\frac{n^2}{\sigma\epsilon^2}\log(\frac{n}{\delta}))$ and that OPT $\geq \epsilon m$. For the first property, from Lemma B.3, we have that for each $\alpha \in \mathcal{A}(T,\epsilon)$ and $a \in A$

$$\Pr\left[R_a(\alpha, S) > \sigma R_a(\alpha) + \epsilon^2\sigma\frac{m}{n}\right] \leq \frac{\delta}{2n|\mathcal{A}(T,\epsilon)|}.$$

Note that $|\mathcal{A}(T,\epsilon)| = T^n = (O\left(\frac{1}{\epsilon^2}\log(\frac{n}{\epsilon})\right))^n$. Union bounding over all $\alpha \in \mathcal{A}(T,\epsilon)$ and $a \in A$, we get that this holds for the particular α computed by the algorithm and also we can sum the inequalities to get

$$\Pr\left[\sum_a R_a(\alpha, S) > \sigma\sum_a R_a(\alpha) + \epsilon^2\sigma m\right] \leq \frac{\delta}{2}$$

showing the first property. For the second property, we apply Lemma B.4 and union bound over all partitions of A into $A_0 \cup A_1$, of which there are at most 2^n, yielding

$$\Pr[\mathrm{OPT}(S) < \sigma\mathrm{OPT} - \epsilon^2\sigma m] \leq \frac{\delta}{2}$$

showing the second property. Applying Lemma B.1 proves the theorem. □

C   The Proof for Theorem 5.1

Proof. The basic framework of the proof is similar with the proof in [22]. Let $R(\alpha, \mathcal{I})$ be the objective value obtained by using weights α proportionally on instance $\mathcal{I}$. Thus, our main goal is to prove $R(\hat{\alpha}, \mathcal{I}) \geq (1-\epsilon)\mathrm{OPT} - 2\eta$. According to the property of the proportional weights [1], if we add a dummy source s adjacent to all impressions and dummy sink t adjacent to all advertisers on the graph of instance $\hat{\mathcal{I}}$, there exists a vertex s-t cut $\mathcal{C}$ whose value $C(\mathcal{C}, \hat{\mathcal{I}})$ is at most $(1+\epsilon)R(\hat{\alpha}, \hat{\mathcal{I}})$. As mentioned in Appendix B, the cut $\mathcal{C}$ is formed by $N(A_0) \cup A_1$, where A_0 and A_1 is two partitions of advertiser set A. Use $C(\mathcal{C}, \mathcal{I})$ to represent the value of the cut $\mathcal{C}$ in instance $\mathcal{I}$.

We first build the relationship between the performances of weights $\hat{\alpha}$ in the two instances:

(C.1) $R(\hat{\alpha}, \mathcal{I}) \geq R(\hat{\alpha}, \hat{\mathcal{I}}) - \eta,$

and then prove the values of cut $\mathcal{C}$ in $\hat{\mathcal{I}}$ and $\mathcal{I}$ are also close:

(C.2) $C(\mathcal{C}, \hat{\mathcal{I}}) \geq C(\mathcal{C}, \mathcal{I}) - \eta.$

Since $C(\mathcal{C}, G)$ is the upper bound of OPT, the theorem can be proved directly by Eq. (C.1) and Eq. (C.2).

Note that we can assume that the graph connections in $\hat{\mathcal{I}}$ and $\mathcal{I}$ are the same and the difference is the capacity of each vertex. Create a new instance $\mathcal{I}'$ based on $\hat{\mathcal{I}}$ and $\mathcal{I}$, where the connection is the same and the capacity of each vertex is the minimum value of its capacity in these two instance. Let $C_v(\mathcal{I})$ be the capacity of vertex v in instance $\mathcal{I}$. Namely, for each vertex $v \in \mathcal{I}'$, we have $C_v(\mathcal{I}') = \min\{C_v(\mathcal{I}), C_v(\hat{\mathcal{I}})\}$. Thus, we have

$$\sum_i |C_i(\mathcal{I}') - C_i(\hat{\mathcal{I}})| + \sum_a |C_a(\mathcal{I}') - C_a(\hat{\mathcal{I}})| \leq \eta,$$

indicating that with the same weights, if the capacity of each vertex v increases from $C_v(\mathcal{I}')$ to $c_v(\hat{\mathcal{I}})$, the objective value increases at most η. In other words,

$$R(\hat{\alpha}, \mathcal{I}') \geq R(\hat{\alpha}, \hat{\mathcal{I}}) - \eta.$$

Since the capacity of each vertex in $\mathcal{I}$ is no less than that in $\mathcal{I}'$, we know

$$R(\hat{\alpha}, \mathcal{I}) \geq R(\hat{\alpha}, \mathcal{I}') \geq R(\hat{\alpha}, \hat{\mathcal{I}}) - \eta,$$

completing the proof of Eq. (C.1).

Eq (C.2) can also be proved in the same way by analyzing the capacity of the cut in $\mathcal{I}'$ and comparing this to the capacities in $\hat{\mathcal{I}}$ and $\mathcal{I}$, respectively. Doing so yields the following chain of inequalities:

$$C(\mathcal{C}, \hat{\mathcal{I}}) \geq C(\mathcal{C}, \mathcal{I}') \geq C(\mathcal{C}, \mathcal{I}) - \eta.$$

This now completes the proof as argued above. □

D   Additional Experimental Results

This section shows more results in the experiments. Recall that we used five arrival orders in the learnability experiments and show the performance in the random order and the worst performance of each algorithm among these five orders. Now we present the performance of different algorithms in the remaining four impression arrival orders (other than random) in Fig. 6 - 9. For the robustness experiments in Fig. 5, we give the instance differences under different capacity settings in Fig. 10. In most places, when the instance difference η decreases, algorithm PW_1 tends to increase.
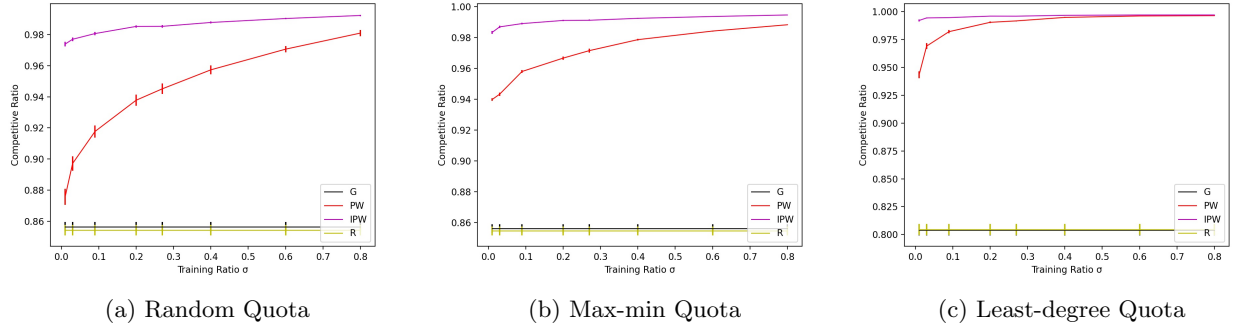
(a) Random Quota   (b) Max-min Quota   (c) Least-degree Quota

Figure 6: Competitive Ratios of different algorithms in the C_i-descending order, plotted as a function of the training ratio.
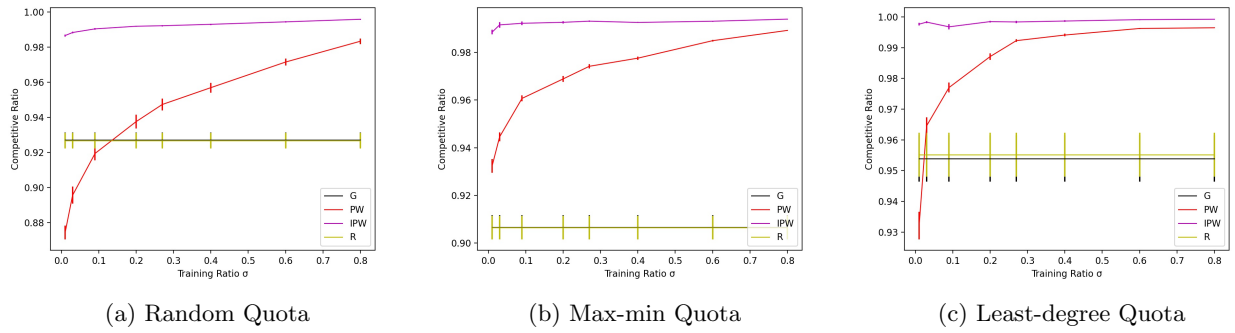

(a) Random Quota   (b) Max-min Quota   (c) Least-degree Quota

Figure 7: Competitive Ratios of different algorithms in the C_i-ascending order, plotted as a function of the training ratio.

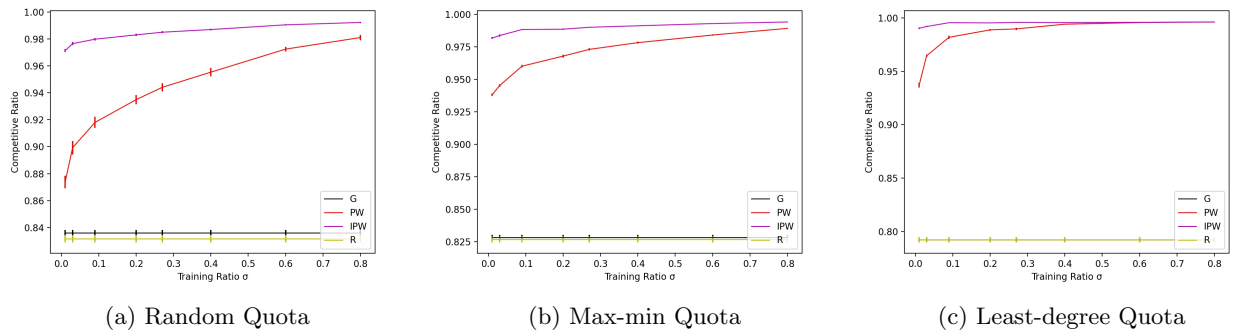(a) Random Quota   (b) Max-min Quota   (c) Least-degree Quota

Figure 8: Competitive Ratios of different algorithms in the C_a-descending order, plotted as a function of the training ratio.

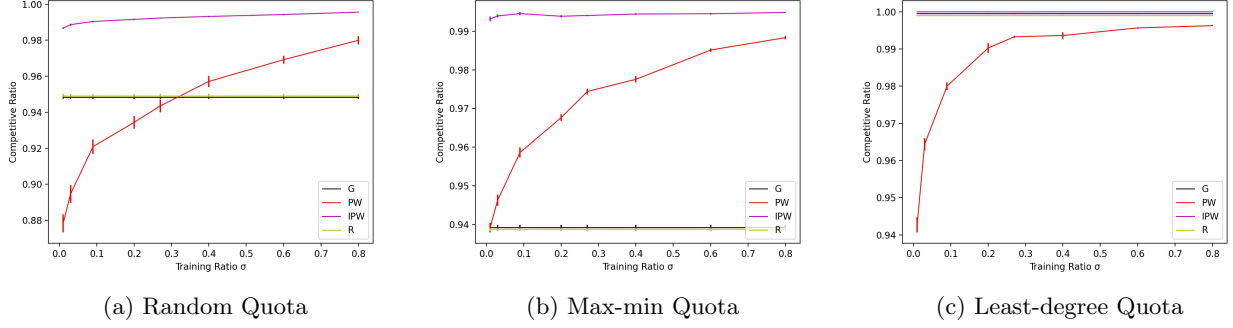(a) Random Quota   (b) Max-min Quota   (c) Least-degree Quota

Figure 9: Competitive Ratios of different algorithms in the C_a-ascending order, plotted as a function of the training ratio.

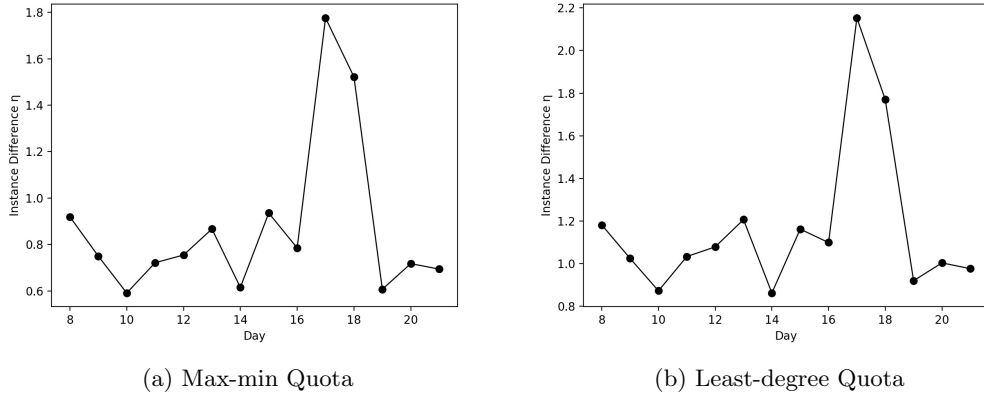(a) Max-min Quota   (b) Least-degree Quota

Figure 10: The instance difference η between day i and day $i-1$ for $i \in [8, 21]$ under two capacity settings, where the instance difference is the ℓ_1 norm between the impression vectors (normalized) of these two instances plus the ℓ_1 norm between two advertiser vectors (normalized). Note the different scales in the Y-axis.