Using Predicted Weights for Ad Delivery*

Thomas Lavastida¹, Benjamin Moseley¹, R. Ravi†, and Chenyang Xu‡

¹Carnegie Mellon University, USA
tlavasti,moseleyb,ravi@andrew.cmu.edu

‡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 1/2-competitive. Allowing for randomization, the ranking algorithm is known to be (1 − 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]

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

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.

$$
\begin{align*}
x_{ia}(\alpha) = \frac{\alpha_a}{\sum_{a' \in N_i} \alpha_{a'}}
\end{align*}
$$

That is, an impression is assigned to the advertisers in its neighborhood proportionally according to $\alpha$. 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 $\epsilon^2$. 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 yesterday’s 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-
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 [13], which fractionally allocates the current impression so that it maximizes the minimum occupied proportion of its capacity among its neighbors. 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 \( \sigma \) 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 \( m = \Omega(\frac{n}{\sigma \epsilon} \log(\frac{n}{\delta}))\) and \( \text{OPT} \geq \epsilon m \) 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 \((1 - \epsilon)\text{OPT}\) impressions on a given instance defined by \( C \), then the same weights have value at least \((1 - \epsilon)\text{OPT} - 2\eta\) on any instance \( C' \) where \( \eta = \sum_i |C_i - C'_i| + \sum_a |C'_a - C_a| \). Here \( \eta \) 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, 10, 11, 27, 13, 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 \( \alpha \), 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 \( A(T, \epsilon) = \{\alpha \in \mathbb{R}^A_+ \mid \alpha_a = (1+\epsilon)^k, k \in |T|\} \).

**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 \( \alpha \in A(T, \epsilon) \) such that

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

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

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 \( \gamma \). If \( \gamma \) 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 \( I \):

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

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 \( \gamma \). We will use \( \text{OPT} \) instead of \( \text{OPT}(I) \) when the context is clear.

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

Denote the first \( \sigma m \) 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 \( \alpha \) 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 \( \sigma m \) impressions in \( I(\gamma) \)

Set \( x_{ia} = 0 \) for each \( i \in S, a \in N_i \)

Set \( T = \Theta(\frac{1}{\epsilon^2} \log(\frac{2}{\delta})) \)

Compute weights \( \alpha \in A(T, \epsilon) \) on \( G' = (S, A, E) \) with capacities \( C'_a = \sigma C_a \) \( \forall a \in A \)

for each remaining impression \( i \)

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 \( x_{ia} = \frac{\alpha_a}{\sum_{a' \in N_i} \alpha_{a'}} \).

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

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 \( \alpha \) with high probability. This involves a union bound over all possible weights in \( A(R, \epsilon) \). 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. Proofs have been omitted from this version of the paper.

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. 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 \( \hat{I} \) has a competitive ratio at least \( 1 - \epsilon - 2\eta/OPT \) on instance \( I \), where OPT is the optimal value of instance \( I \) and \( \eta \) is \( \ell_1 \) 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 \( \hat{I} \), algorithm IPW has a competitive ratio at least

\[
1 - \epsilon - \frac{2\eta}{OPT}
\]

on instance \( I \), where OPT is the optimal value of instance \( I \) and \( \eta \) is \( \ell_1 \) norm between the impression vectors of these two instances plus the \( \ell_1 \) 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 [1] 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]. 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 \((f(P), i)\) 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 \( a \).

**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. 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 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

---

**Footnotes:**

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 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 \( \sigma \) 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 any 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.

![Random Quota](image1)
![Max-min Quota](image2)
![Least-degree Quota](image3)

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. 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 \( \sigma \), indicating that \( \sigma \) proportion of impressions are sampled to serve as the training data. Note that in these figures, the learned weights from the \( \sigma \) proportion are evaluated on the whole data set for the same day. As \( \sigma \) increases, the baseline greedy (G) and ranking (R) algorithms’

---


7. Note the varying scales in the Y-axes in many of the figures.
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 $\sigma$ 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. **$C_n$-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.
5. **Cα-ascending Order**: All impressions arrive in the opposite order of the Cα-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 can be found in a full version of this paper). 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 Ci-descending order and Cα-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 l1 norm between the impressions on the first day (Day 0) and the normalized impression 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 show empirically that weights are robust across days despite 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.

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 lead 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


