ONLINE ALGORITHMS FOR ADWORDS AND BIPARTITE MATCHING IN THE STOCHASTIC SETTING

CS6170 COURSE PROJECT: LITERATURE REVIEW

Swathi Narashiman **Nitin G** **Shri Prathaa M**

ABSTRACT

This review examines key developments in online bipartite matching, beginning with the foundational algorithms introduced by KVV(1990). We summarize the performance of Greedy, Random, and Ranking algorithms in adversarial settings, and explore enhancements under stochastic models like the random permutation and the i.i.d. arrival model. By leveraging probabilistic bounds and flow-based constructions, recent approaches have surpassed classical limits offering improved approximation guarantees under realistic assumptions. The review highlights these advancements and their implications for practical matching applications.

Keywords Online Bipartite Matching · Ranking · Adwaords · Randomized Algorithms

1 Introduction

Online bipartite matching is a foundational problem with wide-ranging applications in areas like online advertising and dynamic resource allocation. In the classical setting, one side of the bipartite graph (e.g., advertisers) is fixed and known in advance, while the other side (e.g., users or impressions) arrives online, one vertex at a time. Upon the arrival of each vertex, the algorithm must make an immediate and irrevocable decision to match it to an eligible neighbor, if one exists.

A landmark contribution to this area was made by Karp, Vazirani, and Vazirani (KVV, 1990), who introduced and analyzed several fundamental algorithms for online bipartite matching, including Greedy, Random, and the celebrated Ranking algorithm. These algorithms differ significantly in their strategies and approximation guarantees. While Greedy and Random algorithms yield only half the optimal expected matching in worst-case scenarios, the Ranking algorithm achieves an expected competitive ratio of $1 - \frac{1}{e}$, which is provably optimal in the adversarial arrival model.

Subsequent research has extended these ideas to stochastic arrival models, such as the random permutation model and the i.i.d. model. In such settings, the assumptions differ notably, edges are typically unweighted and small-bid assumptions hold. While greedy still maintains a $1 - \frac{1}{e}$ guarantee, researchers have proposed boosted max-flow based techniques and "power of two choices"-style algorithms to push the competitive ratio even further.

1.1 Online Bipartite Matching

There is a bipartite graph $G(A, I, E)$ with advertisers A (or girls) and impressions I (or boys), and a set E of edges between them. Advertisers in A are fixed and known. Impressions (or requests) in I (along with their incident edges) arrive online. Upon the arrival of an impression $i \in I$, we must assign i to any advertiser $a \in A$ where $(i, a) \in E(G)$ (or pair the arriving boy with a girl). At all times, the set of assigned edges must form a matching.

1.2 Adwords Problem

The AdWords problem is a fundamental problem in Online Algorithms that models online allocation scenarios, such as ad auctions in search engines. Here, a sequence of items (or queries) arrives one at a time, and each item must be assigned immediately to one of several bidders.
Each bidder has a specified **value (bid)** for the item and a fixed **budget** that limits their total spending. The objective is to maximize the total value of the allocation while ensuring that no bidder exceeds their budget. A special case of

this problem is the Online Bipartite Matching problem, where all budgets are exactly 1, and bids are binary (0 or 1), meaning each bidder can accept at most one item, and each item can only be assigned to an eligible bidder.

2 Models

These problems can be analyzed in various models, and check the performance in each of these models. Every model captures the essence of the setting we are trying to analyze and fixes the context of performance that we are interested in. The models that we are looking to investigate the algorithms in, include:

1. **Adversarial Model**: An adversary decides upon what requests and what order to send them in. This essentially becomes the worst case setting. In this setting, the lower bound on competitive ratio should hold for any input.
2. **Stochastic Models**: These are models where the inputs are sampled from a distribution. These could be of various types:
    (a) IID Model: Each request is chosen from an arbitrary probability distribution that is known to the algorithm.
    (b) Random Order Model: An adversary chooses the requests that are to be made. But the adversary cannot control the order in which these requests are made. The requests chosen by the adversary are sent in a random order.

The stochastic models are weaker in the constraints it imposes on the algorithm. This allows us to get better results than the adversarial model. And these might approximate real life use cases more accurately than the worst case analysis.

3 Adversarial Model

We review the results from the classic work [KVV90].

Greedy Algorithm
The greedy algorithm tries to match any vertex on the LHS to any vertex on the RHS. Greedy always guarantees a maximal matching (i.e, no more eligible edges can be added to the graph). But the maximal matching need not be the maximum matching.

$$E[A(Greedy)] \geq \frac{n}{2}$$

Random Algorithm
When any vertex v on the RHS arrives, the algorithms picks any eligible vertex (i.e, N(v)) on the LHS at random to be matched to v. But Random performs nearly as poorly as greedy deterministic algorithm giving

$$E[\#\,\textit{matching}] = \frac{n}{2} + O(\log n)$$

Ranking Algorithm
We start with a fixed permutation of vertices on LHS (the boys). This order corresponds to their priority. When vertices on RHS arrive, we match it to the eligible vertex (N(v)) of highest rank. Ranking algorithm is optimal and achieves,

$$E[\#\,\textit{matching}] = n\left(1 - \frac{1}{e}\right) + O(n)$$

4 Random Permutation Model

4.1 `Greedy` Algorithm for Online Bipartite Matching

The greedy algorithm for online bipartite matching is as described below.

It is proven that this algorithm has a worst case competitive ratio of $\frac{1}{2}$. We will show that the competitive ratio for `Greedy` is $1 - \frac{1}{e}$ in the random permutation model. In [KVV90], it was shown that the `Ranking` in the worst case model is equialent to the `Greedy` in the random permutation model. Therefore, through this analysis we also provide an alternative proof for the $1 - \frac{1}{e}$ ratio for `Ranking`.

To do this, we will still use the Dual LP method as proposed in [KVV90]. But we will impose additional constraints that will improve the ratio.

Algorithm 1 Greedy Algorithm for Online Bipartite Matching

```
1: **Initialization:**
2: Mark all girls in $L$ as unmatched.
3: **Online Phase:**
4: **for** each arriving boy $b \in R$ **do**
5:     Let $N(b)$ be the set of neighboring girls of $b$
6:     Let $U(b) \subseteq N(b)$ be the set of unmatched girls in $N(b)$
7:     **if** $U(b) \neq \emptyset$ **then**
8:         Select an arbitrary girl $g \in U(b)$
9:         Match $b$ to $g$ and mark $g$ as matched
10:    **else**
11:        Leave $b$ unmatched
12:    **end if**
13: **end for**
```

4.1.1 Notation

Label the boys and girls from 1 to n. Let $p, q \in [n]$ be the variables that we use to denote the label of a boy or a girl and $s, t \in [n]$ be the variables that are used to denote their position. Also we say time t to denote the event when the algorithm is trying to allocate the boy at position t.

Let Ω be the set of all permutations of $[n]$. Any permutation $\sigma \in \Omega$, $\sigma(s)$ denotes the label of the boy at position s. And $\sigma^{-1}(p)$ denotes the position of the boy labelled p.

4.1.2 Proof Sketch

For simplicity, we will first consider the case where where the optimal matching is the perfect matching (size n). And that the optimal algorithm (offline) matches boy p with girl p for all p.

The intuition here is that, whether a boy is matched or not depends on how late he arrives. Say the greedy algorithm does not match a boy p, then this is only because a boy p' was already matched with his girl p. This way, we are able to map each miss of one boy to the match of another. This immediately gives the factor of $\frac{1}{2}$ (infact for every permutation).

We will now classify matches of a boy p' with a girl p into two types, *good* and *bad*. If boy p arrived before p', then it is a good match, but if not, then it is bad. Our goal is to prove that, on the average over all permutations, there are many matches, in particular, many good matches. We do this by showing that every miss is necessarily caused by a larger fraction of bad and good matches.

Let us first define these more carefully (as done in [GM08]),

For each $\sigma \in \Omega$, $p \in [n]$ and $s \in [n]$, define:

$$miss_\sigma(s,p) = \begin{cases} 1 & \text{if } \sigma(s) = p \text{ and boy } p \text{ remains} \\ & \quad \text{unmatched at the end of Greedy,} \\ 0 & \text{otherwise.} \end{cases}$$

$$good_\sigma(s,q) = \begin{cases} 1 & \text{if girl } q \text{ is matched to boy } \sigma(s), \\ & \quad \text{and } \sigma^{-1}(\text{boy } q) \leq s. \\ 0 & \text{otherwise.} \end{cases}$$

$$bad_\sigma(s,q) = \begin{cases} 1 & \text{if girl } q \text{ is matched to boy } \sigma(s), \\ & \quad \text{and } \sigma^{-1}(\text{boy } q) > s. \\ 0 & \text{otherwise.} \end{cases}$$

Define also the following *partitions* for every boy p. Partition the set Ω into groups of permutations such that in each group the relative order of all but boy p is fixed. Let Ω_p be one such group. Let $\sigma_k \in \Omega_p$ be the permutation in which boy p occurs at position k.

Now we will state a series of lemmas and properties that lead to one another and finally help in capturing the fact that every miss implies a good fraction of matches, when averaged over the permutations.

Prefix Property

Let σ_1 and σ_2 be two permutations, and let $t \in [n]$. If

$$\forall\, s \leq t: \ \sigma_1^{-1}(s) = \sigma_2^{-1}(s),$$

then the runs of **Greedy** on σ_1 and σ_2 are identical from time 1 to time t.

This follows from the fact that the permutations look similar to each other until time t and there is no way for `Greedy` to differentiate among them before time t and therefore will behave the same.

Let $G_s(t)$ be the set of girls that are matched by `greedy` up until time t on permutation σ_s (permutation where boy p is at position s).

Monotonicity

For all $s, m \in [n]$ with $s < m$:

$$\begin{cases} (1) \quad \forall\, t \in [1, s-1]: \quad G_m(t) = G_s(t) \\ (2) \quad \forall\, t \in [s-1, m-1]: \quad G_m(t) \subseteq G_s(t+1) \end{cases}$$

This lemma basically states that, if a boy is moved up the queue without changing the order of the other boys, then the set of girls who were matched by time t previously will now be matched at max by time $t+1$. The rigorous proof can be found in the paper. But the idea is that, by induction, if the girl is not matched till a point, then in one time step she has to be matched by the boy, or else she could not have been matched till that point in the original run of `greedy`(without moving up the position of the boy p). This property is very important to make the analysis go through.

Partition

Fix p, and a partition Ω_p. Consider the run of **Greedy** on the permutation σ_n. If girl p gets matched to the boy at position t, for some $t \in [n]$, then:

$$\begin{cases} (1) \quad \forall\, x \in [1, t]: \quad \sum_{s: s \leq t+1} good_{\sigma_x}(s, p) = 1 \\ (2) \quad \forall\, x \in [t+1, n]: \quad bad_{\sigma_x}(t, p) = 1 \end{cases}$$

This can be put into words. If boy p comes after time t, then girl p is already matched, and that would be a bad match (case 2). And if boy p comes before time t, then by monotonicity, we know that girl p would be matched by time $t+1$ and that is a good match.

These properties will help in proving the lemmas below. The full proofs are given in the paper. Here we try and give a reasoning.

Lemma 4.1

$$\forall \sigma \in \Omega,\ \forall t \in [n]: \quad \sum_p good_\sigma(t, p) + \sum_p bad_\sigma(t, p) + \sum_p miss_\sigma(t, p) = 1$$

This comes from the fact that, at any moment, the decision could either lead to a miss or a match. And the matches can be either good or bad.

Lemma 4.2

$$\forall t \in [n]: \quad miss_{\sigma_t}(t, p) \leq \sum_{\sigma \in \Omega_p} \sum_{s: s < t} \frac{bad_\sigma(s, p)}{n - s}$$

If $miss_{\sigma_t}(t, p) = 0$, then it holds trivially. So, the person p at time t was not matched. This happens only if there is another boy that the girl p matched with before t. This with the Partition property helps prove the lemma. This lemma says that there is atleast one bad match before time t over all Ω_p if $miss_{\sigma_t}(t, p) = 1$.

Lemma 4.3

$$\forall t \in [n-1]: \quad \sum_{\sigma \in \Omega_p} \sum_{s \leq t} bad_\sigma(s,p) \cdot \frac{s}{n-s} \leq \sum_{\sigma \in \Omega_p} \sum_{s \leq t+1} good_\sigma(s,p)$$

We can see this in cases. First if the girl p is not matched in σ_n. Then from the prefix property, when the boy is moved forward, then girl could only be matched with a boy after p. This means $bad_\sigma(s,p) = 0$ and the lemma holds trivially.

If the girl is matched in σ_n, We consider that she was matched to a boy at t^*. If $t < t^*$, then there is no bad match and again the lemma holds trivially.

If $t \geq t^*$, for all $s \leq t+1$, summing the good matches for p would give one (from the partition property. With this the inequality can be shown. The takeaway here is that, summing over Ω_p, if there is some number of bad matches until time t, then there has to be a fraction of good matches contained within time $t+1$.

We need to find the minimum revenue, as we want the expected revenue for the worst case of queries. We can proceed to do a factor revealing LP method of analysis. This works by constructing a dual LP and finding a solution to it to bound the optimum of the primal LP. The three lemmas above constraint the variables miss, bad and good. In some sense it says that many misses needs to have many bad matches, and that needs to have many good matches (over all the permutations). So when we take the revenue as the expectation over all permutations, and construct an LP with these constraints, we are able to show a better competitive ratio of $1 - \frac{1}{e}$.

We have seen a the simple case of 2 boys 2 girls, where the first boy can be matched with both the girls, and the second boy will only be matched with the same girl that was matched to boy 1 per the algorithm (this can be determined as it is a deterministic algorithm). This would give a competitive ratio of $\frac{1}{2}$ in the worst case model. In the random permutation model, if boy 2, is the first to arrive then algorithm gets a competitive ratio of 1 if not then $\frac{1}{2}$. Therefore on average, this has a competitive ratio of $\frac{3}{4}$ in this model.

This gap between the observed competitive ratio and the lowerbound suggests that there are smarter algorithms that would be able to capitalize on this model. We will these later.

4.2 `Greedy` Algorithm for General Adwords Problem

The analysis presented here works for a setting which is slightly more lenient than the adwords problem. This class includes the small bids model and the bipartite matching model. It is described (as presented in [GM08]) below:

For each bidder i, consider an allocation of some subset of the queries to bidder i, say $(q_1, q_2, \ldots, q_k)$, which straddles the budget in the following sense: $\sum_{j=1}^{k-1} bid_{ij} < B_i \leq \sum_{j=1}^{k} bid_{ij}$. For every such allocation we are guaranteed that the overspending is small compared to the budget, i.e., $\sum_{j=1}^{k} bid_{ij} - B_i \ll B_i$.

The `Greedy` algorithm assigns each query to the highest bidder with a non-zero remaining balance. And in the case of a tie, decides on the basis of an arbitrary order that is hardcoded.

Let the query p be assigned to bidder i_p by OPT. Previously in the matchings case, we tried to make constraints by having a mapping between the boy p and the boy who stole his girl. Here it is more complicated. `Greedy` could assign many queries with different bid values to this bidder i_p, and the relation between these queries and the query p is not so straightforward. This is taken care of by the tagging procedure.

4.2.1 Tagging Procedure

For the analysis, we will fix the OPT allocation. And fix an arbitrary ordering of queries for each bidder.

Now consider any query q and the bidder i to whom OPT assigned q. In the fixed ordering, when OPT assigned q to i, suppose that the money spent by i increased from x to y. Then we will call $[x,y]_i$, the Opt-Interval of the query q: i.e., defineOpt-Interval(q) $= [x,y]_i$. Greedy, run on some permutation σ, makes its own allocation, leading to a similar Alg-Interval$_\sigma(q)$ for each query q. We define

$$tagset_\sigma(q_1, q) = \text{Opt-interval}(q) \cap \text{Alg-Interval}_\sigma(q)(q_1)$$

And each tagset can be a goodset or a badset. It is good if $\sigma^{-1}(q) \leq \sigma^{-1}(q_1)$ and bad otherwise.

This is how we keep track of which queries are displacing and affecting which. Note that one query in the optimal solution can be tagged by many queries in the greedy run. These can be good or bad.

4.2.2 Proof Sketch

We first define the notion of constant-tie breaking. This property is when more than one bidder ties for the highest bid, they are chosen in the same arbitrary manner. That is, let there be a group of bidders who are tied for a query in one permutation and i^* is chosen as the winner. Even in another permutation, when a subset of this group is tied, and when the subset includes i^*, the algorithm will always chose this i^*.

We define the prefix property in the exact same way as with bipartite matching. The monotonicity property is also in the same vein, but here it makes a statement on the amount spent by a bidder till time t. It says that, if a query p is moved forward, then the amount of money spent by any bidder till time $t+1$ is greater than the money spent till time t in the original case.

Lemma 4.4 (Partition Property) *Fix p, and a partition Ω_p.*

1. $\forall x \in [1, t]$:
$$\bigcup_{s: x \leq s \leq t+1} goodset_{\sigma_x}(s, p) \supseteq tagset_{\sigma_n}(t, p)$$

2. $\forall x \in [t+1, n]$:
$$badset_{\sigma_x}(t, p) = tagset_{\sigma_n}(t, p)$$

The proof of this property follows from the prefix and monotonicity, much like the bipartite case.

Let us define $good_\sigma(s, p) = |goodset_\sigma(s, p)|$ [1] and $bad_\sigma(s, p) = |badset_\sigma(s, p)|$. Also define:

$$extra_\sigma(q) = Alg\text{-}Interval_\sigma(q) \setminus \bigcup_p Opt\text{-}Interval(p)$$

$$miss_\sigma(s, p) = \begin{cases} \max(0, opt(p) - alg_\sigma(p)) & \text{if } \sigma(s) = p \\ 0 & \text{otherwise.} \end{cases}$$

Now we will use these properties to derive 3 lemmas, very similiar to the bipartite case. Though here, it is not discrete, but we talk about the amount of money that was spent in a *good* manner, a *bad* manner and the amount that could have been gained (miss) or the amount that was left untagged (extra). We bound these amounts to show, again, that miss neccesarily only happens when there is some amount of bad, and that happens only when there is a significant fraction of good.

The proofs of these lemmas are quite involved, so we only state them.

Key Lemmas

Lemma 1. For all $s \in [n]$ and all $\sigma \in \Omega$,

$$\sum_p bad_\sigma(s, p) + \sum_p good_\sigma(s, p) + |extra_\sigma(s)| + \sum_p miss_\sigma(s, p) \geq opt(\sigma(s))$$

Lemma 2. For all $t \in [n]$,

$$miss_{\sigma_t}(t, p) \leq \sum_{\sigma \in \Omega_p} \sum_{s<t} \frac{bad(s, p)}{n - s}$$

Lemma 3. For all $t \in [n-1]$,

$$\sum_{\sigma \in \Omega_p} \sum_{s \leq t} bad(s, p) \frac{s}{n - s} \leq \sum_{\sigma \in \Omega_p} \sum_{s \leq t+1} good(s, p)$$

We can then do the same procedure with the dual LP to reveal the competitive ratio. Turns out that the LP we end up with is the same one. Thus we have show a competitive ratio of $1 - \frac{1}{e}$. An example where `Greedy` has a ratio of $1 - \frac{1}{e}$ is can also be shown to make the analysis tight.

[1]we will define $tagset_\sigma(s, p)$ to mean $tagset_\sigma(\sigma(s), p)$

4.3 Learn-Weights Algorithm

The problem revolves around a bunch of n bidders, $B_1, B_2, B_3, ..., B_n$ and **m** (previously known) queries. When query j is allocated to bidder i a total revenue of u_{ij} is collected. Though the set of queries is known prior the order in which they arrive is a random permutation. The random permutation model is very similar to the iid stochastic model where bid vectors are sampled from an unknown distribution without replacement.

4.3.1 Algorithm outline

PAC (Probably Approximately Correct) algorithm is a useful tool in machine learning that offers insights on how much data is required for a model to generalize to unseen data. In the PAC framework, generalization is quantified by the probability that the chosen hypothesis will have an error rate within an acceptable range on new samples.

Primal and Dual LPs

Primal (P):

$$\begin{array}{lll} \textit{maximize} & \sum_{i,j} u_{ij}x_{ij} & \\ \textit{subject to} & \sum_j u_{ij}x_{ij} \le B_i & \forall i \\ & x_{ij} \le 1 & \forall i,j \\ & x_{ij} \in \{0,1\} & \forall i,j \end{array}$$

Dual (D):

$$\begin{array}{lll} \textit{minimize} & \sum_i \alpha_i B_i + \sum_j p_j & \\ \textit{subject to} & p_j \ge u_{ij}(1-\alpha_i) & \forall i,j \\ & \alpha_i \ge 0 & \forall i \\ & p_j \ge 0 & \forall j \end{array}$$

Equivalent Objective Function

$$D(\alpha) = \sum_i \alpha_i B_i + \sum_j \max_i u_{ij}(1-\alpha_i)$$

The learn-weights algorithm proposed by ***Devanur & Hayes*** proves that there exists an $1-\epsilon$ competitive algorithm for all inputs such that

$$\frac{OPT}{b_{max}} \ge \left(\frac{n^2 \log(\lambda/\epsilon)}{\epsilon^3}\right)$$

Where, b_{max} is the maximum bid on any query, λ is the upper bound on the ratio of maximum to minimum non zero bid n any query.

$$b_{max} = \max_{i,j} u_{ij} \qquad \lambda = \max_{i,i',j}\{\frac{u_{i'j}}{u_{ij}} : u_{ij} \neq 0\}$$

In layman terms, the algorithm adds a multiplicative factor $(1-\alpha_i)$ to each bidding value and maximises the new bidding value while allocating a query j to a bidder i. To learn the weights α_i we use a PAC framework (similar to machine learning) where we optimize the objective function on a set S of ϵm queries while ensuring that it generalizes well for the universal set.

Algorithm 2 Learn-Weights Algorithm

```
1: Input: Number of queries $m$, exploration fraction $\epsilon$
2: for $j \leftarrow 1$ to $\epsilon m$ do
3:     Allocate query $j$ arbitrarily
4: end for
5: $\alpha^* \leftarrow \arg\min_\alpha D(\alpha, S)$        {Objective over first $\epsilon m$ queries}
6: for $j \leftarrow \epsilon m + 1$ to $m$ do
7:     Allocate query $j$ to bidder $i \leftarrow \arg\max_i [u_{ij}(1-\alpha_i^*)]$
8: end for
```

4.3.2 Novelties in the algorithm

1. Complementary slackness property is used in the LP duality problem

   Primal (P): $\begin{array}{l} \textit{maximize } c^T x \\ \textit{subject to } Ax \le b,\ x \ge 0 \end{array}$

   Dual (D): $\begin{array}{l} \textit{minimize } b^T y \\ \textit{subject to } A^T y \ge c,\ y \ge 0 \end{array}$

   Complementary Slackness Conditions:

$$y_i^*(a_i^T x^* - b_i) = 0 \quad \textit{for all } i$$
$$x_j^*\left((A^T y^*)_j - c_j\right) = 0 \quad \textit{for all } j$$

2. Non uniform error rates in functions ensuring that the total error is bound
3. **Resolving Ties** using random perturbation to the bidding values and smoothing using Lagrangain Duality.

4.3.3 Analysis

Let's define the following notations,

$$x_{ij}(\alpha) = \begin{cases} 1, & \textit{if} \quad \textit{arg}\max_i u_{ij}(1-\alpha_i) \\ 0, & \textit{otherwise} \end{cases}$$

$$R_i(\alpha, S) = \sum_{j\in S} u_{ij} x_{ij}(\alpha); \quad R(\alpha) = \sum_i R_i(\alpha)$$

An $\mathcal{L} \subseteq [0,1]^n$ is an (x,ϵ)-net for an allocation rule $x_{ij}()$ and an $\epsilon > 0$, if for all $\alpha \in [0,1]^n$, there exists an $\alpha' \in \mathcal{L}$ such that

$$|x_{ij}(\alpha) - x_{ij}(\alpha')| \le \epsilon \quad \text{for all } i, j.$$

This means the net $\mathcal{L}$ gives a good approximation of the whole space instead of searching over the infinite space, you can search over the smaller set and still be confident that the function values don't vary too much within each ϵ-ball.

The follwing lemma's help us in deriving the bounds of the algorithm (stated without rigorous mathematical proof).

Concentration Lemma

If for all i, $|R_i(\alpha^*, S) - \epsilon R_i(\alpha^*)| \le t_i$, and $\sum_i t_i \le \epsilon^2 \max\{\text{OPT}, R\alpha^*\}$, then $P(\alpha^*, S^c) \ge (1 - O(\epsilon))\text{OPT}$.

The proof comes from the property that α^* satisfies the complementary slackness property. The next part uses the **Bernstein Inequality**. Bernstein's inequality is a probabilistic bound used to estimate how much the sum of independent random variables deviates from its expected value. It's especially useful when the variables are bounded or sub-exponential.

Bernstein-type Inequality

Let $Y = (Y_1, \dots, Y_m)$ be a vector of real numbers, and let $0 < \epsilon < 1$. Let S be a random subset of $[m]$ of size ϵm, and set $Y_S := \sum_{j\in S} Y_j$. Then, for every $0 < \delta < 1$,

$$\Pr\left(|Y_S - \mathbb{E}Y_S| \ge \frac{2}{3}\|Y\|_\infty \ln\left(\frac{2}{\delta}\right) + \|Y\|_2\sqrt{2\epsilon \ln\left(\frac{2}{\delta}\right)}\right) \le \delta.$$

The following lemma is the core of the algorithm that guarentees near optimal competitive ratio.

Core Lemma

If $\mathcal{L}$ is an (x, ϵ)-net and

$$\frac{OPT}{b_{\max}} \geq \left(\frac{n \log(n|\mathcal{L}|/\epsilon)}{\epsilon^3}\right),$$

then

$$ALG \geq (1-\epsilon)OPT.$$

4.3.4 Avoiding Ties

The $arg\max_i u_{ij}(1-\alpha_i^*)$ need not be a unique value. If different bidders compete for the same query, we need to resolve the tie among them to efficiently allocate the query. To solve this problem the *Devanur & Hayes* propose two strategies.

Random Perturbations

There can be atmost n-1 ties. To get around this, we add a small perturbation τ_{ij} to each bidding value u_{ij}. Since these perturbations are small, the effect of these distortions in the optimization is minimal, hence α^* remain nearly same. If the perturbations are multiplicative and are chosen from $[1-O(\epsilon), 1+O(\epsilon)]$ then we can ensure to achieve a competitve ratio of $1-O(\epsilon)$.

Algorithm 3 Learn-Weights Perturbed Algorithm

```
1: Input: Number of queries $m$, exploration fraction $\epsilon$, perturbation bound $\eta$
2: for $j \leftarrow 1$ to $\epsilon m$ do
3:     for each bidder $i$ do
4:         Sample $\tau_{ij}$ uniformly from $[0, \eta]$
5:         Set $\hat{u}_{ij} \leftarrow (1-\tau_{ij})u_{ij}$
6:     end for
7:     Allocate query $j$ arbitrarily
8: end for
9: $\alpha^* \leftarrow \arg\min_\alpha D(\alpha, S)$
10:     where $D(\alpha, S) = \sum_i \alpha_i B_i + \sum_{j=1}^{\epsilon m} \max_i \hat{u}_{ij}(1-\alpha_i)$
11: for $j \leftarrow \epsilon m + 1$ to $m$ do
12:     for each bidder $i$ do
13:         Sample $\tau_{ij}$ uniformly from $[0, \eta]$
14:         Set $\hat{u}_{ij} \leftarrow (1-\tau_{ij})u_{ij}$
15:     end for
16:     Allocate query $j$ to bidder $i \leftarrow \arg\max_i [\hat{u}_{ij}(1-\alpha_i^*)]$
17: end for
```

Smoothing

Another approach to solve the ties problem is taking a smooth version of the dual LP. The sub-gradient of the function $x_{ij}(\alpha)u_{ij}$ is not unique when there is a tie and hence it is not differentiable, whereas the smooth version of the dual LP is a convex funtion and the gradient determines the allocation. The smooth version of the dual LP is obtained using the Lagrangian Multipliers.

Primal and Dual for smooth LP

Primal Program

[h]

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} u_{ij}x_{ij} - \frac{(u_{ij}x_{ij})^2}{2C_j} \\ \text{subject to} \quad & \sum_j u_{ij}x_{ij} \leq B_i \quad \forall i \\ & \sum_i x_{ij} \leq 1 \quad \forall j \\ & x_{ij} \geq 0 \end{aligned}$$

Dual Program

$$\begin{aligned} \text{minimize} \quad & \sum_i \alpha_i B_i + \sum_{j \in S} p_j + \sum_{i,j \in S} \frac{C_j}{2}\theta_{ij}^2 \\ \text{subject to} \quad & p_j \geq u_{ij}(1-\alpha_i-\theta_{ij}) \quad \forall i, j \in S \\ & \alpha_i \geq 0 \\ & p_j \geq 0 \end{aligned}$$

We define the algorithm for the smoothed dual LP as follows

Algorithm 4 Learn-Weights Smoothed Algorithm

```
1: Input: Number of queries $m$, exploration fraction $\epsilon$
2: for $j \leftarrow 1$ to $\epsilon m$ do
3:     Allocate query $j$ arbitrarily
4: end for
5: $\alpha^* \leftarrow \arg\min_\alpha D^*(\alpha, S)$
6:     where $D^*(\alpha, S) = \sum_i \alpha_i B_i + \sum_{j \in S} p_j + \sum_{i,j \in S} \frac{C_j}{2} \theta_{ij}^2$
7: for $j \leftarrow \epsilon m + 1$ to $m$ do
8:     Allocate query $j$ to bidder $i \leftarrow \arg\max_i [u_{ij}(1 - \alpha_i^*)]$
9: end for
```

5 IID Model

5.1 `Greedy` Algorithm for Online Bipartite Matching (Unknown IID)

In this model, there is a fixed but unknown distribution on Q (set of Queries). We claim that this model gives lesser power to the adversary, so if `Greedy` had a competitive ratio of $1 - \frac{1}{e}$ in the random permutation model, then, in the IID model, its competitive ratio will be $1 - \frac{1}{e}$ or higher. Then an example where the competitive ratio is exactly $1 - \frac{1}{e}$ is shown in [GM08], thus making the analysis tight.

Let the algorithm get n queries that are sampled from the distribution. Conditioned on the fact that the set of queries that are sampled are same and equal to this n queries, we can say that it simplifies to the random permutation model. And thereby claim that any algorithm in the iid model is atleast as good as the random permutation model. Note, we do not make any claims saying that the models are equivalent. If we design an algorithm to take advantage of the fact that it is the iid model, then we will not be able to give such a tight analysis, and the competitive ratio in this model can be better than the random permutation.

But this leaves room for a better algorithm to have a competitive ratio better than $1 - \frac{1}{e}$.

5.2 Boosted Flow Graph Algorithm for Online Bipartite Matching (Known IID)

In the IID model, the arrival of impressions is not adversarial but is random. It is also independent of previous impressions that arrived and $i \in I$ arrive online according to a known probability distribution $\mathcal{D}$ over I.
The greedy algorithm achieves a competitive ratio of $1 - 1/e$.
Through boosted maxflow graph we can achieve a 0.67-approximation online algorithm breaking $1 - 1/e$.
The assumptions in this setup are that:

1. The edges are unweighted
2. OPT is not necesarily much larger than each bid(small bid assumption)

There exists a family of instances where no algorithm achieves expected approximation better than $1 - o(1)$ as $n \to \infty$. (Extended Proof in appendix B of Feldman et al (2009))

Note: We could have a $1 - \epsilon$-approximation for Adwords assignment when opt is larger than $O\left(\frac{n^2}{\epsilon^3}\right)$ times each bid in the i.i.d. and random permutation models.

Fact 1

Suppose we throw n balls into n bins, where each ball is placed independently and uniformly at random. Let $B \subseteq \{1, \ldots, n\}$ be a subset of bins and S be the number of bins in B that receive at least one ball.
Then, with probability at least $1 - 2e^{-\epsilon n/2}$, for any $\epsilon > 0$, we have:

$$|B|\left(1 - \frac{1}{e}\right) - \epsilon n \leq S \leq |B|\left(1 - \frac{1}{e} + \frac{1}{en}\right) + \epsilon n.$$

(This comes from $1 - \frac{1}{e} \leq 1 - (1 - \frac{1}{n})^n \leq 1 - \frac{1}{e}(1 - \frac{1}{n})$)

Fact 2

Fix some arbitrary subset $\mathcal{R} \subseteq \{1, \ldots, c\}$. A bin sequence is satisfied if:(i) at least one of its bins b_i with $i \notin \mathcal{R}$ has at least one ball in it; or, (ii) at least one of its bins b_i with $i \in \mathcal{R}$ has at least two balls in it. Let S be a random variable that equals the number of satisfied bin sequences. With probability at least $1 - 2e^{-\epsilon^2 n/2}$, we have

$$S \geq \ell \left(1 - \frac{2^{|\mathcal{R}|}}{e^c}\right) - \epsilon dn - \frac{2^{|\mathcal{R}|} c^2}{e^c} \cdot \frac{\ell}{n - c^2}.$$

Suggested Matching

If B is the set of advertisers that are part of the optimal matching (i.e., $|B| = n$) and S, the number of advertisers in B that are matched in the online algorithm with n impressions, then, by **Fact 1**, with high probability,

$$S \geq |B| \left(1 - \frac{1}{e}\right) - \epsilon n.$$

Hence, the suggested algorithm achieves at least a $(1 - \frac{1}{e} - \epsilon)$ fraction of the optimal number of matches, with high probability. As $\epsilon \to 0$, the guarantee becomes $1 - \frac{1}{e}$.

Two Suggested Matching

We adapt the *power of two choices* from load balancing by computing two disjoint matchings on the expected instance and using them in a fixed preference order during the online phase. These matchings are obtained via a max-flow in a boosted flow graph and decomposed into two edge-disjoint near-matchings. They not only guide the online algorithm but also help upper-bound the offline optimum via a bounded cut in the flow graph.

Algorithm 5 Two Suggested Matchings (TSM) Algorithm

```
Require: Bipartite graph $G = (A, I, E)$, distribution $\mathcal{D}$ over $I$, total impressions $n$
Ensure: Online matching of impressions to advertisers
 1: Offline Phase:
 2: Construct flow graph $G_f$ with capacities:
        • $c(s, a) = 2$ for all $a \in A$
        • $c(a, i) = 1$ for all $(a, i) \in E$
        • $c(i, t) = 2$ for all $i \in I$
 3: Compute a maximum flow in $G_f$
 4: Decompose flow into:
        • Blue edges: first-choice matches
        • Red edges: fallback matches
 5: Online Phase:
 6: Initialize counters $x_i \leftarrow 0$ for all $i \in I$
 7: for each arriving impression $i'$ of type $i \sim \mathcal{D}$ do
 8:     $x_i \leftarrow x_i + 1$
 9:     if $x_i = 1$ and a blue edge $(i, a)$ exists and $a$ is unmatched then
10:         Assign $i'$ to $a$
11:     else if $x_i = 2$ and a red edge $(i, a)$ exists and $a$ is unmatched then
12:         Assign $i'$ to $a$
13:     else
14:         Skip assignment
15:     end if
16: end for
```

Let A_{BR} be the ads incident to both a blue and red edge, A_B to only a blue edge, A_R to only a red edge and A_{BB} to 2 blue edges.

$$\text{ALG} \geq \left(1 - \frac{2}{e^2}\right)|A_{BR}| + \left(1 - \frac{1}{e^2}\right)|A_{BB}| + \left(1 - \frac{3}{2e}\right)(|A_B| + |A_R|) - 4\epsilon n.$$

(From fact 2 we get A_{BR}'s contribution as $1 - 2/e^2$, A_B contributes $1 - 1/e$ from fact 1, A_R contributes $1 - 2/e$, A_{BB} contributes $1 - 1/e^2$)

$$\text{OPT} \le \left(\frac{4}{3} - \frac{2}{3e}\right)|A_{BR}| + \left(\frac{5}{3} - \frac{4}{3e}\right)|A_{BB}| + \left(1 - \frac{1}{e}\right)(|A_B| + |A_R|) + \epsilon' n.$$

(OPT has been bounded using the cut of the offline solution)

$$\frac{\text{ALG}}{\text{OPT}} + \epsilon \ge \min\left\{\frac{1-\frac{1}{e^2}}{\frac{5}{3}-\frac{4}{3e}}, \frac{1-\frac{2}{e^2}}{\frac{4}{3}-\frac{2}{3e}}, \frac{1-\frac{3}{2e}}{1-\frac{1}{e}}\right\} = \min\{0.735\ldots, 0.670\ldots, 0.709\ldots\} = \frac{1-\frac{2}{e^2}}{\frac{4}{3}-\frac{2}{3e}} \approx 0.670.$$

The setting where the tightness of above bound is achieved is explained in the paper.

5.3 Novelties

1. Each impression type is preassigned two potential disjoint matchings based on the solution to the expected impression arrival vector (found using distribution)
2. Efficient at runtime as online phase maintains only a simple counter per impression type to give suggestions
3. The boosted flow capacity ensures sufficient flexibility to handle stochastic deviations.

6 Comparing the Algorithms

Table 1: Comparison of Bipartite Algorithms

Algorithm	**Model**	**Competitive Ratio**	**Random Bits**	**Information Known Prior**
Ranking	Adversarial	0.63	$n \log n$	-
Greedy	Stochastic IID	0.63	-	-
Greedy	Random Permutations	0.63	-	-
Boosted Flow Graph	Stochastic IID	0.67	-	Distribution of Impressions

Table 2: Comparison of Adwords Algorithms

Algorithm	**Model**	**Competitive Ratio**	**Random Bits**	**Information Known Prior**
Learn weights(Base)	Random Permutations	$1 - O(\epsilon)$	-	# Queries (m)
Greedy	Stochastic IID	0.63	-	-

7 Recent Development

In the i.i.d. model under an integrality restriction on the expected number of impressions, an LP-guided algorithm with improved rounding achieves a competitive ratio of 0.705 [MGS11], while a flow-based LP algorithm improves this to 0.729 [JL14]. Without the integrality restriction in the same model, the LP-guided algorithm attains a slightly lower competitive ratio of 0.702 [MGS11], and the flow-based LP algorithm achieves 0.706 [JL14]. In the random order model, an Online Contention Resolution Scheme (OCRS) with improved rounding in the vertex-weighted case yields a competitive ratio of 0.6534 [HTWZ19], whereas the classic Ranking algorithm for the unweighted case achieves a ratio of 0.696 [MY11].

8 Exploratory Questions

The Learn Weights Algorithm proposed can be modified for robustness in real world Adwords problem. Instead of modelling the queries as a random permutation other models such as Markovian arrivals and Semi-Markov model can be experimented and competitive ratio can be compared. Instead of the PAC based learning algorithm, we can use Stochastic Gradient Descent (or Mini-batch gradient) to update the parameters alpha at specific time intervals.

In the boosted flow graph method, the best competitive ratio that can be achieved by extending 2 disjoint matchings to k matchings remains an open problem. This is because analyzing the OPT on generalizing to k matchings is not easier.

9 Conclusions

In this review, we explored the theoretical foundations and algorithmic advances in online bipartite matching and the Adwords problem. We explore techniques like Primal-Dual analysis, tagging procedure, PAC learning, Boosted flow graphs to better analyse and design algorithms for stochastic models. Despite substantial progress, challenges remain in extending these models to more dynamic and realistic settings.

References

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 982–991, San Francisco, 2008.

[HTWZ19] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals, 2019.

[JL14] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. 39(3), 2014.

[KVV90] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 352–358, New York, NY, USA, 1990. Association for Computing Machinery.

[MGS11] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics, 2011.

[MY11] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. New York, NY, USA, 2011. Association for Computing Machinery.