Approximation Algorithms Workshop June 13-17, 2011, Princeton

Lecture : Online Matching and Adwords June 16, 2011

Aranyak Mehta *Scribe: Pushkar Tripathi*

1 Overview

Online matching is a core algorithmic problem in Internet ad allocation. The question of precisely which ads to show on a page is a problem of finding an optimal matching between supply (e.g., page views) and demand (e.g., advertiser budgets).

The biggest example is that of *search based advertising*: In this system the advertisers submit bids to the search engine, stating how much they are willing to pay to get an ad placed for a certain keyword search. This is done offline. The search engine then gets requests online, in the form of queries from users. For each arriving query, the search engine finds the ad candidates, scores them, and runs an auction to determine which ads to show, and how much does each ad pay (on a click). One important aspect of search based advertising is the existence of daily budgets which an advertiser can specify to cap the maximum amount of money that it would pay through a day. With the introduction of budgets the system is no longer stateless, and this induces online matching and allocation problems.

Another important example is that of *display advertising*, on external webpages. Ad contracts are often determined offline by negotiations between publishers and advertisers based on requirements such as targeting and serving quantity. The supply arrives online in the form of page views. Again the problem for the display advertising platform becomes an online matching problem, that of matching the online supply to the offline demand (with prices predetermined via the contracts). Other examples, like *ad exchanges* also involve online matching or allocations.

In each of these examples, the objective function used in the online matching problem is important for the business motivation. Since there are three players – users, advertisers and search engine (or platform, exchange, or publishers) – there are three objective functions: user happiness, advertiser ROI and pubmisher's short term revenue. The first two objectives are important for long term growth, and in practice one would maximize some well-chosen convex combination of the three. In the abstractions below, we will be maximizing the *efficiency of the matching*, namely, the total value that is matched. This can be a reasonable proxy for all three objective functions. Furthermore, in the real problems, there are several domain specifications, e.g., in search ads, advertisers pay only when a user clicks, and pays the price determined by a second price auction. In our abstractions, we shall work with first-price, pay on impression settings.

1.1 Models for Studying Online Allocation Problems

Online allocation problems such as the ones described above have been studied in 4 models. These models make progressively stronger assumptions about the available information about the arriving requests.

1. **Adversarial:** In this model we assume we are presented with the worst possible sequence of requests in adversarial order.
2. **Random Order Arrival (ROA):** Here we assume that the requests are presented to us in random order though they may be adversarially chosen.
3. **Unknown Distribution:** In this model we assume that the requests are drawn from as distribution that is not known to us before-hand.

4. **Known Distribution:** Here we assume that the requests are drawn from a distribution that is available to us a blackbox.

Note that the adversarial model has the least amount of information about the arriving requests while the known distribution model assumes the maximum amount of information about the query stream. Despite their apparent differences there is no known separation result differentiating the Unknown distribution model from the Random Order model. In the next sections, we will consider different matching and allocation problems, under these different arrival models. Table 1.1 provides a preview summary of the results surveyed here.

	Adversarial Input	Unknown Distribution / Random Order	Known Distribution
Bipartite Matching	$1 - \frac{1}{e}$ (optimal)	0.696 (0.823)	0.702 (0.823)
Vertex Weighted Matching	$1 - \frac{1}{e}$ (optimal)	?	?
Adwords	$1 - 1/e$ (optimal)	$1 - \epsilon$ (optimal)	$1 - \epsilon$ (optimal)

Table 1: Summary of Results for Online Matching Problems in different arrival models. Rows correspond to problem and columns to arrival model. The entries are the best known factors, and the corresponding upper bounds (in parenthesis). Citations are provided in the text.

2 Online Bipartite Matching

This is the simplest and most basic problem in online matching. In this problem, first introduced by Karp et al. in [KVV90], we are given a bipartite graph $G(L, R, E)$ whose left side L is known to us in advance and the right R arrives online one vertex at a time. For each incoming vertex we are revealed its neighbors among vertices on the left side. The arriving vertex needs to be matched irrevocably to an available neighbor (if any). The objective is to maximize the number of vertices that get matched. In terms of the motivating problem, the left-hand side corresponds to advertisers and the right refers to page views.

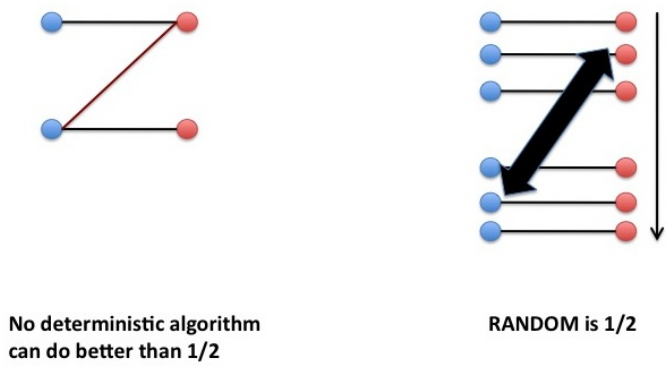

Figure 1: The core difficulty for online bipartite matching

The core difficulty in the problem is the following (see left side of Figure 1). Suppose we have two vertices u and v, on the left (given) side, and suppose the first vertex to arrive has edges to both of them and we match it to v. This decision would turn out to be a sub-optimal choice if the next vertex to arrive is only

adjacent to v which is already matched. In fact this example shows that no deterministic online algorithm can attain a factor better than $1/2$.

Uncorrelated Randomness: One might be tempted to think that a simple application of randomization will overcome this difficulty: Consider the algorithm RANDOM, which lets each arriving vertex pick one of its available neighbors at random. It performs with factor $3/4$ in this example. However this strategy does not generalize to any better than $1/2$, as the following graph shows (see Figure 1, right side). Each side of the bipartition has n vertices such that there is a perfect matching between them. There is also a complete bipartite graph between the first $n/2$ vertices to arrive on the right and the "incorrect" set of $n/2$ vertices on the left side. It can be shown that RANDOM attains a factor of $1/2 - o(1)$ for this instance, essentially since almost all the first half of arriving vertices match to the incorrect set. Next we present an algorithm that is able to significantly improve the $1/2$ bound in the adversarial model using *correlated randomness*.

2.1 Adversarial Order Model

In [KVV90] the authors introduced the RANKING algorithm for the Online Bipartite Matching problem in the adversarial model. This algorithm uses correlated randomness to get an expected competitive ratio of $1 - 1/e$.

The RANKING Algorithm: In this algorithm we begin by permuting the vertices on the left side according to a random permutation. We match each incoming vertex to the first available neighbor according to this permutation. The vertex remains unmatched if none of its neighbors are available.

We will now briefly analyze this algorithm based on a proof by Birnbaum and Matheiu [BM08] (this is the only proof provided in this talk).

Theorem 1. *The* RANKING *algorithm attains a factor of* $1-1/e$ *for the Online Bipartite Matching Problem in the adversarial arrival model.*

Proof. The analysis is based on the following observations:

1. If a vertex u at the t^{th} position on the left side, is left unmatched in an execution of the algorithm then its neighbor (u^*) in the optimal solution should surely be matched (for simplicity, assume that OPT is a perfect matching, so every u has a corresponding u^*). In fact we can show something stronger than this: not only should u^* be matched, it should be matched to a vertex placed higher than u in the permutation for this execution.
2. Furthermore if we consider the the n permutations produced by moving u to all possible positions keeping the relative order of all other vertices fixed, then u^* continues to be matched above position t, in each of these permutations.

This yields the following equation relating miss-events and matches.

$$\Pr\left[\text{miss at position } \ t\right] \leq \frac{\sum_{s \leq t} \Pr\left[\text{match at position } s\right]}{n}$$

The equation can solved to show that the algorithm attains a factor of at least $1 - 1/e$. □

In [KVV90] the authors also gave an example to show that their bound is essentially tight in this model. They considered the graph shown in Figure 2 and showed that if the vertices arrive from right to left in the adjacency matrix as shown, then no online algorithm can hope to achieve a factor better than $1 - 1/e$ for this problem. Intuitively the first few vertices to arrive have lots of alternatives and most of them do not end up matching their partner in the optimal solution. This exhausts the options for the vertices arriving at the end and they are left unmatched.

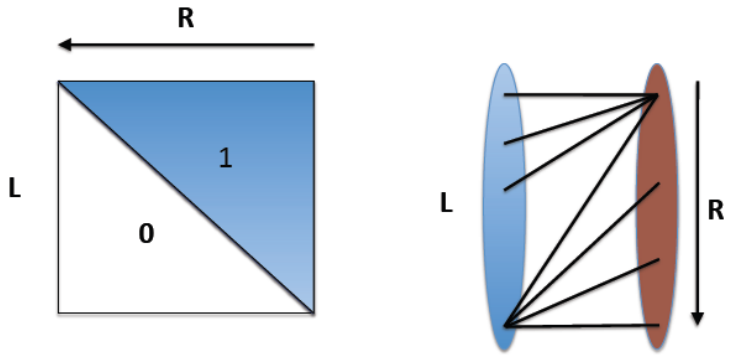

Figure 2: Tight Example for the RANKING Algorithm

2.2 Random Order Arrival Model

In this model (shortened to ROA) we assume that the graph is adversarial, but the vertices on the right arrive online in random order. The model was introduced by Goel and Mehta in [GM08]. They showed that the greedy algorithm that matches each incoming vertex to the first available neighbor on the left side (breaking ties consistently), attains a factor of $1 - 1/e$. In fact this is easy to prove once we have the analysis of Ranking in the adversarial model, since Greedy with random order input simulates the RANKING algorithm. The paper also provided an upper bound of 5/6 for this model.

Since $1 - 1/e$ is no longer an upper bound on the performance of an online algorithm in this model, an intriguing unanswered question in the work of Goel and Mehta was whether it was possible to beat this bound in the ROA (and hence Unknown Distribution) model. This question was answered in the affirmative by Karande et al. [KMT11] and Mahdian and Yan [MY11] where they showed that the RANKING algorithm attains a factor strictly greater than $1 - 1/e$ in the ROA model. The analysis is based on the observation that the output of the RANKING algorithm is independent of which side is considered as the streaming side and which one is assumed to be given offline, i.e. we can switch the roles of the known and streaming side in the ROA model and the output would remain unchanged. Comparing to the proof on KVV in the adversarial model, the increase in factor is provided by considering amplified maps to matches, not only from miss events, but also from certain types of matches themselves. Their results are summarized in the following theorems.

Theorem 2 ([KMT11, MY11]). *The* RANKING *algorithm attains a factor of at least* 0.656 *[KMT11] (*0.696 *[MY11]) for the online bipartite matching problem in the ROA model.*

The 0.696 factor uses a computer aided proof to solve a "strong" factor revealing LP.

In [KMT11] the authors also showed that if the underlying graph has k disjoint (nearly) perfect matchings then the RANKING algorithm attains a factor of at least $1 - 1/\sqrt{k}$. Intuitively this result states that if the given graph is internally robust then it is easy to find a large matching in it. This theorem explains the intriguing empirical observation seen in experiments that indicated that the performance of the RANKING algorithm goes to 1 for the worst case graph for the adversarial model shown in Figure 2 (note that this graph has a large number of nearly perfect matchings).

In terms of tight examples for RANKING, [KMT11] provided two examples for which the algorithm does significantly worse than 1 and attains a factor of 0.75 and 0.726 respectively. These graphs are shown in Figure 3. Using a result by Mahshadi et al. [MOGS11] we know that no online algorithm can attain a competitive ratio better than 0.83 even in the known distribution model (next section). Since the known distribution model is a special case of the ROA model the upper bound carries over to this setting.

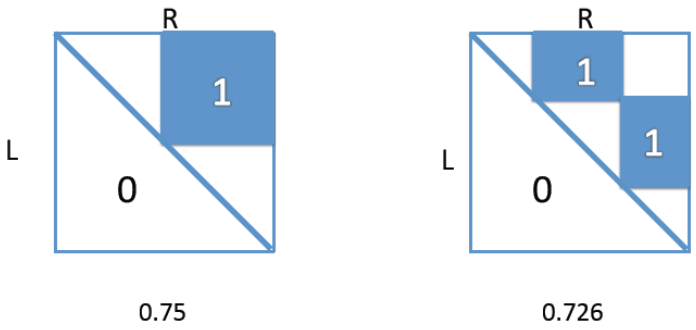

Figure 3: Tight Example for RANKING in the ROA model.

2.3 Known Distribution Model

This model was first introduced by Feldman et al. in [FMMM09]. Here we assume that there are a polynomially bounded number of types of vertices and this is known to us in advance. Vertices are drawn independently and with replacement from these types and have to be matched irrevocably upon arrival. More formally: We are given a *base graph* $G(L, \hat{R}, \hat{E})$ in advance, and also a distribution $\mathcal{D}$ on $\hat{R}$. The outcome graph, $G(L, R, E)$, is obtained through the process in which the next vertex $v \in R$ to arrive is picked iid from $\mathcal{D}$, with the neighbors of v being identical to that of the chosen base vertex $\hat{v} \in \hat{R}$. Let us begin by considering a simple algorithm for this problem that is based on the idea of using offline estimates to guide online decisions.

The SUGGESTED-MATCHING **algorithm**: This is the simplest idea in this setting: find the optimal matching in the base graph and for each arriving vertex v match it to the optimal match for its base type $\hat{v}$, according to this matching. However, the core difficulty in this approach is that the types of vertices can repeat across different samples drawn from the underlying graph, i.e., we may get several draws $v_1, v_2, ..., v_k \in R$ for the same base type $\hat{v} \in \hat{R}$. Since the algorithm tries to match each such v_i to the same vertex in L (the match of $\hat{v}$), this will result in all but the first arriving vertex, v_1, remaining unmatched. In fact by a standard *balls-in-bins* argument we can show that SUGGESTED-MATCHING algorithm attains a factor of $1-1/e$ (when D is the uniform distribution).

We will now discuss an algorithm that does better than $1 - 1/e$ in this model.

The TWO-SUGGESTED-MATCHING **(TSM) algorithm**, is based on the idea of the *power of two choices*. As above, it exploits offline statistics to guide its choices as the vertices arrive online. However instead of having one suggested matching to guide in decision making, it uses two matchings for this purpose. We use maximum 2-flow on the base graph network to calculate two large disjoint matchings (say M_1 and M_2) in the base graph. The TSM algorithm tries to match each arriving vertex it to the match of its type in M_1; if it fails then it attempts to match according to M_2. If both these attempts fail then the vertex remains unmatched. Their result is summarized in the following theorem. Thus each arriving vertex now has two attempts at finding a match. The difficulty introduced is that two vertices of different types now interact, since the M_1 match of one may be the M_2 match of the other.

Theorem 3. *The Two-Suggested-Matching algorithm attains a factor of* $\frac{1-\frac{2}{e^2}}{\frac{4}{3}-\frac{2}{3e}} \simeq 0.67$ *for the Online Bipartite Matching problem in the Known Distribution Model.*

The above theorem holds for the case of integral rates, i.e., when the distribution $\mathcal{D}$ is such that the expected number of arrivals of each type is integral. In [FMMM09] the authors also showed that their analysis was tight by showing an example for which the TSM algorithm attains exactly this factor. They also showed that even in this model, with complete information about the distribution from which the vertices are drawn, no online algorithm can achieve a competitive ratio of $1 - o(1)$. Their results were later improved by Bahmani

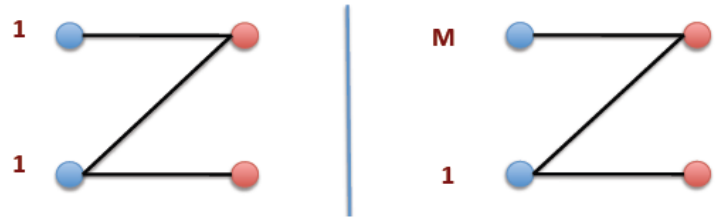

Figure 4: Two extreme examples for vertex weighted matching.

and Kapralov [BK10] to a 0.70 factor algorithm (which is a modification of TSM), and an upper bound of 0.902. Mahshadi et al. [MOGS11] provided a different algorithm, based on finding two matchings, as in TSM, but with the difference in how the two matchings are chosen: they choose them by simply sampling from the optimal fractional solution. They also provide an adaptive algorithm using a new idea of using a joint distribution on matchings obtained from the optimal fractional matching. They showed that this algorithm achieves a factor of 0.702, even for the case when the rates are not integral. They also improved the upper bound to 0.83.

In the next two sections, we will deal with two related, but different variants of matching with weights. The optimal algorithms for these two problems, although very different, have related intuition and techniques.

3 Weighted Vertex Matching

We will now focus our attention on the vertex weighted version of the online matching problem discussed earlier. In this problem the vertices on the right arrive online and reveal their neighbors on the left side, as before. The left side is known in advance and each vertex (say v) on the left also has a weight ($w(v)$) associated with it. We wish to maximize the sum of weights of vertices on the left that get matched.

Intuition: Let us consider two cases for this problem. Consider the graph shown on the left in Figure 4. Here the two vertices on the left have the same weight. For this instance RANKING, attains a factor of $3/4$ and $1 - 1/e$ in general (as we saw in Section 2). Also, Greedy (which matches to the highest weighted neighbor, breaking ties arbitrarily) only attains a factor of $1/2$ for such instances. Next consider the graph on the right in Figure 4. Here the weight of one of the vertices is much greater than that of the other. In this setting RANKING attains a factor of $1/2$ in the examples (which goes to 0 for larger examples) whereas GREEDY is almost optimal (factor ≈ 1). Thus while RANKING works well for uniformly weighted graphs, it fails badly when the vertex weights are highly skewed. On the other hand the other hand Greedy does well for highly skewed weights but fails when the weights are equal.

The examples explore two different aspects of the Weighted Vertex Matching problem and suggest two candidate algorithms.

1. **Ranking with non-uniform permutations:** Instead of using a random permutation for the RANKING algorithm, we can draw a permutation for a distribution that depends on the weights on the vertices.
2. **Perturbed Greedy:** Run Greedy with a "soft-max": allow vertex weights to flip with some probability. (As discussed in Section 2 no deterministic algorithm can attains a competitive ratio better than $1/2$, thus we can hope to use randomness in the greedy algorithm to circumvent this problem).

In the next section we will present an algorithm that has both these properties.

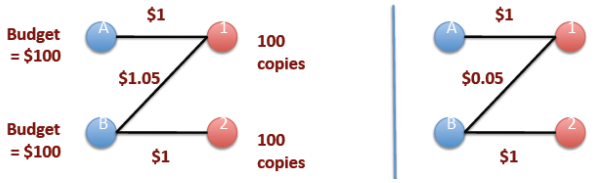

Figure 5: Two extreme examples for the Adwords problem.

3.1 Algorithm for weighted vertex matching

In [AGKM10] the authors presented the following algorithm for the weighted vertex matching problem.

1. For every vertex v pick a random number r_v between 0 and 1.
2. For every vertex v, define its perturbed weight as $W^*(v) = W(v)\psi(r(v))$, where $\psi(x) = 1 - e^{-x}$
3. For each arriving vertex on the right, match it to the available neighbor with the highest perturbed weight.

Note that the above algorithm is exactly equal to the RANKING algorithm when all weights are equal and morphs in to the greedy algorithm when the weights are highly skewed. In [AGKM10] the authors proved the following theorem.

Theorem 4. *The Perturbed-Greedy / Non-uniform-KVV algorithm above attains a factor of* $1 - 1/e$ *for the Weighted Vertex Matching problem. This is optimal (follows from the unweighted case).*

Remark: It is not known whether one can better this $1 - 1/e$ bound for the adversarial model in other models such as the ROA model and the Known Distribution model.

4 Adwords Problem

Finally we move on to Adwords problem. We are given a set of advertisers each of who have a daily budget B_i. Ad requests arrive online and the advertisers place a (possibly different) bid on the ad-request (so that the bid of advertiser i for request q is b_{iq}). We assume that the bids are small with respect to the daily budgets of the agents (i.e., $\forall\ i, q : b_{iq} \ll B_i$). Each ad slot can be allocated to at most one advertiser and the advertiser is charged his bid for the item from his budget. The objective is to maximize the amount of money spent by the advertisers. We will assume that the advertisers' bids are much smaller than their budget. Thus, one may still consider this as a generalized matching (or allocation) problem, with each vertex on the left (known) side having a budget constraint, and the edges having weights.

Intuition: We will consider two algorithms - GREEDY and LOAD-BALANCE. GREEDY assigns each ad-slot to the advertiser with the highest bid who still has not exhausted his budget. On the other hand LOAD-BALANCE gives it to the advertiser who has spent the least fraction of his budget. Consider the example on the left in Figure 5. Here GREEDY will assign the first 100 copies to the second advertiser and the latter 100 copies would remain unsold. Whereas LOAD-BALANCE would attain a factor of 3/4.

On the other hand for the setting on the right GREEDY does well attaining a factor of 1 while the LOAD-BALANCE gets a factor of 1/2. The main intuition in solving the Adwords problem is to find a hybrid algorithm that combines these ideas and performs well on all instances. This is achieved by perturbing the bids for the agents to reflect the fraction of their budget spent. The complete algorithm is described in the following section.

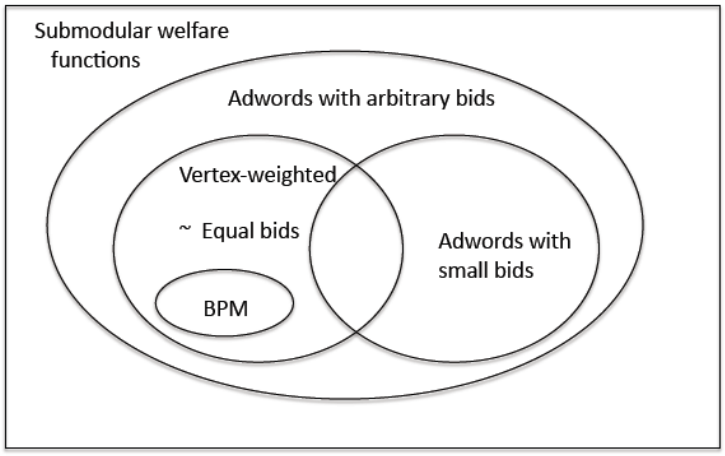

Figure 6: Landscape of problems.

4.1 Deterministic Algorithm for the Adwords Problem

As with the algorithm for weighted vertex matching we use a perturbation function to modify the input and then run the greedy algorithm. However the perturbation function used here is deterministic.

The MSVV **Algorithm:** For advertiser i and for the arriving query q define the scaled bid $\hat{b_{iq}} = b_{iq} \times \psi(f_i)$, where f_i is the fraction of unspent budget of i, and $\psi(x) = 1 - e^{-x}$. For each arriving query we allocate it to the advertiser with the highest scaled bid.

In [MSVV05] Mehta et al. showed that this algorithm attains a factor of $1 - 1/e$ and it is optimal. The curious thing is to recall that this is the same perturbation function as the one used for weighted vertex matching in Section 3. Buchbinder et. al [BJN07] gave a different (though related) online algorithm, based on the primal dual programs for the problem.

Theorem 5. *The MSVV algorithm above attains a factor of $1 - 1/e$ for the Adwords problem. This is optimal even among randomized algorithms. The BJN primal-dual online algorithm also achieves an optimal $1 - 1/e$ factor.*

4.2 Adwords in the Random Order Arrival Model

In [DH09] Devanur and Hayes considered the Adwords problem in the ROA model. They presented a factor $1 - \epsilon$ competitive algorithm that was also based on the primal dual method. The main idea behind the algorithm was to use a fraction of the input to approximate the entire query stream. However the primal variables in the linear program corresponding to the offline solution are hard to approximate by sampling a fraction of the query stream. On the other hand the dual variables (as in the BJN algorithm) can be approximated and can be used to guide the allocation problem for the rest of the query stream.

Theorem 6. *The DH dual sampling algorithm attains a factor of $1 - \epsilon$ for the Adwords problem, where ϵ is a function of the ratio of bid to budget.*

5 Discussion

In this lecture we have covered various online allocation problems, starting with the foundational (and simplest) case of online bipartite matching and using those ideas to build towards a solution for the Adwords

problem. In Figure 6 we show the complete landscape of problems. All the problems considered here are special cases of the Adwords problem with arbitrary bids (not necessarily small compared to the budget). All the offline versions of the problems discussed in this lecture are either in P or approximated to $1 - \epsilon$ (for the Adwords problem with small bids). Unlike those, the Adwords problem with arbitrary bids (also known as Maximum Budgeted Allocation) is NP-hard to approximate to a factor better than 15/16 [CG08], and the best known offline algorithm is $\frac{3}{4}$ [CG08, Sri08]. The greedy algorithm attains a factor of $\frac{1}{2}$ in the online setting and nothing better than $1/2$ is known for the online setting. A further generalization of these problems is the Submodular Welfare Maximization problem, where the advertisers have general submodular functions as their objective. Once again this problem is hard to approximate to a factor better than $1 - 1/e$ in the offline setting and the best online algorithm only attains a factor of $\frac{1}{2}$. Proving optimal algorithms or hardness results for these problems remains an exciting open question.

References

[AGKM10] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. *SODA*, 2010.

[BJN07] N. Buchbinder, K. Jain, and J.S. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *Algorithms–ESA 2007 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007: Proceedings*, page 253. Springer, 2007.

[BK10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. *ESA*, 2010.

[BM08] B. Birnbaum and C. Mathieu. On-line bipartite matching made simple. 2008.

[CG08] D. Chakrabarty and G. Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science-Volume 00*, pages 687–696. IEEE Computer Society Washington, DC, USA, 2008.

[DH09] N. Devanur and T Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, 2009.

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS*, pages 117–126, 2009.

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008.

[KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching in the unknown distributional model. In *STOC*, pages 106–117, 2011.

[KVV90] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.

[MOGS11] Vahideh H. Manshadi, Shayan Oveis-Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, 2011.

[MSVV05] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. In *FOCS*, 2005.

[MY11] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In *STOC*, pages 117–126, 2011.

[Sri08] A. Srinivasan. Budgeted allocations in the full-information setting. *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 247–253, 2008.