Online Algorithms with Stochastic Input

NIKHIL R. DEVANUR
Microsoft Research

Categories and Subject Descriptors: F.2.m [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Miscellaneous

General Terms: Algorithms

Additional Key Words and Phrases: Online, Matching, Adwords, Learning, Random Permutation

1. INTRODUCTION

The design and analysis of *online* algorithms, where the input to the algorithm is revealed over time and the algorithm has to make decisions immediately without knowing the future input, has received a revived interest in the last few years primarily due to their application to online advertising. The canonical problem is the *Adwords* problem, which is motivated by the problem of optimally allocating ad slots on search queries to budget constrained advertisers. It involves simplifications that ignore certain aspects of the actual way this allocation is done. For instance, it assumes a "first-price" pay-per-impression scheme, ignoring the game theoretic aspects, and considers only one slot per query. To be precise, the Adwords problem is as follows.

The Adwords problem:
Input:

—n = number of advertisers

—m = number of queries

—$\forall i = 1..n, B_i$ = Budget of advertiser i

For $j = 1..m$

—Input: $\forall i = 1..n, b_{ij}$ = bid of advertiser i for query j

—Algorithm: allocate query j to some advertiser $i \in \{1..n\}$. The allocation is denoted by indicator variables x_{ij} where x_{ij} is 1 if j is allocated to i and is 0 otherwise.

The revenue generated by the algorithm is ALG $:= \sum_i \min\{B_i, \sum_j b_{ij}x_{ij}\}$. Let OPT $:= \max_x\{\sum_i \min\{B_i, \sum_j b_{ij}x_{ij}\}\}$ be the optimal revenue on hindsight. The maximum is taken over all valid allocations, that is all $x_{ij} \in \{0, 1\}$ such that for each j, $\sum_i x_{ij} \leq 1$. An algorithm is said to have a (worst-case) competitive ratio of α if ALG $\geq \alpha$OPT for all inputs. In case the algorithm is randomized, then ALG is replaced by its expectation over the coin tosses of the algorithm.

The worst-case competitive ratio for Adwords was shown to be $1 - 1/e$ by Mehta et. al. [Mehta et al. 2005]. This is in fact tight, that is no randomized algorithm can

Author's addresses: nikdev@microsoft.com

achieve a better ratio. The worst-case model seems too pessimistic; the instances occurring in practice may be "well-behaved" and it may be possible to exploit this behavior to get a better revenue. One way to model this well behavedness is to assume that the input is generated by some stochastic process, as is done in the following models.

Definition 1.1. The random permutation model. Let the adversary pick n, m, B_i $\forall i \in \{1..n\}$ and a_{ij} $\forall i \in \{1..n\}$ and $\forall j \in \{1..m\}$. A permutation π of $\{1..m\}$ is chosen uniformly at random (by nature). In step j of the algorithm, the bids input to the algorithm are $b_{ij} = a_{i\pi(j)}$. The algorithm has a competitive ratio of α if $E[\text{ALG}] \geq \alpha\text{OPT}$ for all choices of the adversary, where the expectation is taken over the choice of the permutation and the coin tosses of the algorithm (if any).

Definition 1.2. The i.i.d model with unknown distribution. Let the adversary pick n, m, B_i $\forall i \in \{1..n\}$ and a probability distribution on *bid vectors*. For simplicity assume that the distribution is uniform over the m bid vectors given by a_{ij} $\forall i \in \{1..n\}$ and $\forall j \in \{1..m\}$. The distribution is not part of the input to the algorithm. In step j of the algorithm, $j' \in \{1..m\}$ is chosen uniformly at random (by nature) and the bids input to the algorithm are set to $b_{ij} = a_{ij'}$. The algorithm has a competitive ratio of α if $E[\text{ALG}] \geq \alpha E[\text{OPT}]$ for all choices of the adversary, where the expectation is taken over the coin tosses of nature and those of the algorithm (if any).

Definition 1.3. The i.i.d model with known distribution. This is the same as before, except that the distribution is part of the input to the algorithm.

In all the models, one can ask for a high-probability result instead of a result in expectation, that is, one can ask that with probability $1 - \delta$ for some given δ, $\text{ALG} \geq \alpha\text{OPT}$.

An important parameter that affects the performance of the algorithms is the bid-to-budget ratio, denote by γ.

$$\gamma := \max_{i,j} \left\{ \frac{b_{ij}}{B_i} \right\}.$$

For simplicity, we will assume that $b_{ij} \in [0, 1]$ $\forall$ i, j. Thus $\gamma = 1/\min_i\{B_i\}$. Table 1 summarizes the best known competitive ratios for the various models and assumptions about γ. The first column corresponds to the results that assume that $\gamma \to 0$. These are valid for "sufficiently small" γ. The second column corresponds to results that do not make any assumption[1] about γ. The third column corresponds to the special case of bipartite matching, where the bids are either 0 or 1 and the budgets are all 1.

For the stochastic models one can get a competitive ratio that tends to 1 as γ tends to 0. This means that as each individual query becomes relatively insignificant compared to the overall budget, the online algorithm can almost match the offline optimum. This presents a significant improvement over the worst-case. In this article I will present the main ideas behind these algorithms. In particular, in Sections 2 and 3 I will present the algorithm of Devanur and Hayes [Devanur and

[1]We can always assume that $\gamma \leq 1$.

Table I. Table summarizing the known results for the Adwords problem

Model	Small γ	Large γ	Bipartite matching
Worst Case	$1-1/e$ [Mehta et al. 2005; Buchbinder et al. 2007]	1/2	$1-1/e$ [Karp et al. 1990]
Random Permutation	$1-O(\sqrt{\gamma n \log(mn/\epsilon)})$ [Devanur and Hayes 2009; Agrawal et al. 2009]	1/2	.696 [Mahdian and Yan 2011; Karande et al. 2011]
i.i.d unknown	$1-O(\sqrt{\gamma \log(n/\epsilon)})$ [Devanur et al. 2011]	$1-1/e$ [Devanur et al. 2011]	.696
i.i.d known	$1-O(\sqrt{\gamma \log n})$	$1-1/e$	.702 [Manshadi et al. 2011; Feldman et al. 2009; Bahmani and Kapralov 2010]

Hayes 2009] for the random permutaion model and in Secion 4 the algorithm of Devanur et. al. [Devanur et al. 2011] for the i.i.d model with unknown distributions.

An important point to note is that these algorithms need to know the (approximate) number of queries in advance.[2] In particular, the following example shows that if m is not known in advance, then the competitive ratio is bounded away from 1. The example has only 2 bidders and 2 keywords, a and b, each of which occurs $m/2$ times. The bids are $(1,0)$ for a and $(2,1)$ for b. Each bidder has a budget of 150. Now if $m = 100$, then OPT gives all 50 b's to bidder 1 (in addition to all the a's). However, if $m = 200$ then OPT only gives 25 b's to bidder 1 and 75 b's to bidder 2. After seeing 100 keywords, any allocation has to significantly deviate from OPT (and have a significantly lower revenue) in at least one of the two values of m. Thus, an online algorithm that does not know m has to have a competitive ratio bounded away from 1.

2. AN ALGORITHM FOR THE RANDOM PERMUTATION MODEL

The algorithm is similar to the *Occam's razor* [Kearns and Vazirani 1994] algorithm for PAC learning applied to an online setting. We use the first few queries as the sample, find the "best fit" for the sample from among a class of allocation algorithms, and simply use it for the rest of the queries. Before we give more details of the algorithm, we present some preliminaries. Consider the following LP relaxation of the problem.

$$\begin{aligned}
&\max \quad \sum_{i,j} b_{ij} x_{ij} \\
&\text{s.t. for all } i, \sum_j b_{ij} x_{ij} \le B_i \\
&\text{and for all } j, \sum_i x_{ij} \le 1. \\
&x_{ij} \ge 0.
\end{aligned}$$

[2]In the worst case, knowing the number of queries does not give any advantage. The adversary can always say that this number is infinite and start giving queries where the bids are all zero after some point.

The dual of the above LP is

$$\begin{aligned} \min \quad & \sum_i \alpha_i B_i + \sum_j p_j \\ \text{s.t. } \forall\ i,j, \quad & p_j \geq b_{ij}(1-\alpha_i). \end{aligned}$$

Note that, at the optimum, $\forall\ j,\ \ p_j = \max_i\{b_{ij}(1-\alpha_i)\}$. Thus, one can think of the dual objective as just a function of the α_i's. Thus we define

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

By complementary slackness, if (α, p) minimizes the dual LP, and x is the optimal allocation to the primal LP, then

$$x_{ij} > 0 \text{ implies } p_j = \max_i\{b_{ij}(1-\alpha_i)\},$$

and hence, given the optimal α, we should allocate item j to bidder $\arg\max_i\{b_{ij}(1-\alpha_i)\}$. In fact, every vector $\alpha \in [0,1]^n$ gives an allocation, assign j to $\arg\max_i\{b_{ij}(1-\alpha_i)\}$. This is the class of allocations we will consider. From the above observation, we know that this class is rich enough to contain at least one allocation that is optimal.

The algorithm uses the first $k = \epsilon m$ queries as a sample to learn the vector α, in a way reminiscent of PAC-learning algorithms: it selects the α that minimizes the restriction of D to the observed bids. More precisely, for all subsets S, of size $k = \epsilon m$, define

$$D(\alpha, S) := \sum_i \alpha_i \epsilon B_i + \sum_{j\in S} \max_i\{b_{ij}(1-\alpha_i)\}.$$

We now present our core algorithm, Learn-Weights.

Algorithm 2.1: LEARN-WEIGHTS(ϵ)

for $j \leftarrow 1$ **to** $k = \epsilon m$
do { Observe the bids b_{ij}.
Allocate item j arbitrarily (e.g. all $x_{ij} = 0$).
Let $\alpha^* := \arg\min_\alpha\{D(\alpha, S)\}$.
for $j \leftarrow \epsilon m + 1$ **to** m
do { Observe the bids b_{ij}.
Give item j to the bidder maximizing $b_{ij}(1-\alpha_i^*)$.

It can happen that there are ties; that is, $\arg\max_i\{b_{ij}(1-\alpha_i^*)\}$ may not be uniquely defined. For now, we will ignore this issue, and pretend that such ties never occur (this would be the case if, for instance, the bid vectors were in general position). We will discuss ways to remove this assumption later.

THEOREM 2.1. *The algorithm Learn-Weights*(ϵ) *has a competitive ratio of* $1 - O(\epsilon)$ *given that for all* i, $B_i \geq n\log(mn)/\epsilon^3$

3. ANALYSIS OF THE ALGORITHM

Here, instead of providing a complete proof of correctness, we draw parallels with PAC learning, and highlight the similarities and differences. A brief overview of (a simple version of) PAC learning is first presented.

Let a concept class C on a domain X be a set of boolean functions $f : X \to \{0, 1\}$. Let D be a probability distribution on X. Given an unkown concept $c \in C$, you observe k pairs $(x, c(x))$ where x's are i.i.d samples from D. C is PAC learnable with sample complexity k if we can find a hypothesis $h \in C$ such that with probability $\geq 1 - \delta$ (over the choice of the k samples and the internal randomness of the algorithm if used),

$$error(h) := \Pr_{x \sim D}[h(x) \neq c(x)] \leq \epsilon.$$

The Occam's razor algorithm for PAC learning is as follows. Call the sample set S. Output an $h \in C$ such that $h(x) = c(x)$ for all $x \in S$.

THEOREM 3.1. *C is PAC learnable by Occam's razor algorithm with sample complexity*

$$a \frac{\log(|C|) + \log(1/\delta)}{\epsilon}$$

for some universal constant a.

PROOF. If for some fixed h, $error(h) > \epsilon$, then for a given sample x, $Pr[h(x) = c(x)] \leq 1 - \epsilon$. Therefore $\Pr[h(x) = c(x) \ \ \forall x \in S] \leq (1 - \epsilon)^k$. Further by union bound on all $h \in C$ such that $error(h) > \epsilon$,

$$\Pr[\exists h \in C : error(h) > \epsilon, h(x) = c(x) \ \ \forall x \in S] \leq |C|(1 - \epsilon)^k.$$

This is less than δ if $k \geq a \frac{\log(|C|)+\log(1/\delta)}{\epsilon}$ for some constant a. □

To draw a parallel, the notion of a concept class is equivalent to the class of all allocations that we consider, or equivalently to the set of all dual vectors. Here each element in the concept class defines an allocation, which is a function with the range $\{1, 2, \ldots, n\}$ rather than a boolean function. There is no distribution on the domain, instead we simply have the set of all queries. The concept c we are targeting is the optimal dual, which gives an optimal allocation. Also, although there are uncountably many α's, it can be shown that the number of distinct allocations are only $(mn)^{O(n)}$ and this will be the size of our concept class, $|C|$.

The first k queries serve as the sample S which is picked uniformly at random from the set of all queries. The goal is to find a dual vector α (a hypothesis) such that with probability $\geq 1 - \delta$,

$$error(\alpha) := 1 - ALG(\alpha)/\text{OPT} \leq \epsilon.$$

For the purpose of exposition, we now consider a setting that is more like PAC learning. Suppose that the algorithm is given $k = \epsilon m$ samples chosen uniformly at random from the set of all queries. The algorithm is required to output a dual vector which will then be used to allocate all the queries. Now, the equivalent of Occam's razor is to output an optimal dual on the sample S.

Analogous to the proof of Theorem 3.1, we need to show that if for some fixed α,

$$error(\alpha) > \epsilon \Rightarrow Pr[\alpha \text{ is optimal on } S] \leq \delta/|C| =: \delta'. \quad (1)$$

This is the heart of the proof. Once we have this, we know that

$$\Pr[\exists \alpha : error(\alpha) > \epsilon, \alpha \text{ is optimal on } S] \leq \delta.$$

However, showing (1) is more complicated in our case. For instance, we cannot analyze sample by sample as we did earlier. Also, the quantity $error(\alpha)$ is not very easy to work with directly. So we define a new function, $d_i(\alpha)$. Instead of measuring how far is the allocation corresponding to α from the optimum (like *error* does), d_i measures how far is α from satisfying the *complementary slackness condition*. Let $x_{ij}(\alpha)$ be the allocation given by α. Let $R_i(\alpha) := \sum_j b_{ij} x_{ij}(\alpha)$. For all i such that $\alpha_i > 0$, let $d_i(\alpha) := |R_i(\alpha) - B_i|$. For all i such that $\alpha_i = 0$, let $d_i(\alpha) := \max\{0, R_i(\alpha) - B_i\}$. Also let $D_i(\alpha) = \alpha_i B_i + (1 - \alpha_i) R_i(\alpha)$ be the contribution of i towards the dual. We show (1) by showing that

(1) if $error(\alpha) > \epsilon$, that is α is far from optimal, then $\exists i : d_i(\alpha) > \epsilon D_i(\alpha)$, that is α is far from satisfying one of the complementary slackness conditions on the entire set.

(2) if $\exists i : d_i(\alpha) > \epsilon D_i(\alpha)$, that is if α is far from satisfying one of the complementary slackness conditions on the entire set, then $Pr[\alpha \text{ is optimal on } S] \leq \delta'$, α is unlikely to satisfy the complementary slackness conditions on a random subset S of size ϵm, and hence is unlikely to be optimal on S.

For the first part, the following fact is easy to check: $D(\alpha) - \text{ALG}(\alpha) \leq \sum_i d_i(\alpha)$. Hence if for all $i, d_i(\alpha) \leq \epsilon D_i(\alpha)$ then $D(\alpha) - \text{ALG}(\alpha) \leq \epsilon D(\alpha)$ and $error(\alpha) \leq \epsilon$.

In order to prove the second part, we need the following measure concentration inequality which is a corollary of Bernstein's inequality. Suppose that $b_1, b_2, \ldots, b_m \in [0, 1]$. S is a random subset of $[m]$ of size k, and $X = \sum_{j \in S} b_j$. $\mu = E[X] = \frac{k}{m} \sum_j b_j$. Then with probability $\geq 1 - \delta$,

$$|X - \mu| \leq O\left(\sqrt{\mu \ln\left(\frac{1}{\delta}\right)} + \ln\left(\frac{1}{\delta}\right)\right). \quad (2)$$

In other words, the probability that X deviates from μ by more than $O(\sqrt{\mu \ln\left(\frac{1}{\delta}\right)} + \ln\left(\frac{1}{\delta}\right))$ is less than δ. What we show is that α is optimal on S implies such a deviation. In order for α to be optimal on S, it has to satisfy the complementary slackness conditions on S. Thus if we set $b_j = b_{ij} x_{ij}(\alpha)$ (and suppose that $\alpha_i > 0$) the complementary slackness conditions imply that $X = \epsilon B_i$, where as $\mu = \epsilon R_i(\alpha)$. Therefore we have that $|X - \mu| = \epsilon d_i(\alpha)$. The facts that $|C| = (mn)^{O(n)}$, $d_i(\alpha) > \epsilon D_i(\alpha)$ and the assumption that $B_i \geq \Omega(n \log(mn)/\epsilon^3)$ imply that

$$\epsilon d_i(\alpha) > \Omega(\sqrt{\epsilon R_i(\alpha) \ln(1/\delta')} + \ln(1/\delta')).$$

Agrawal, Wang and Ye [Agrawal et al. 2009] use the "doubling trick" to achieve an improvement on the dependence on ϵ (from ϵ^3 to ϵ^2). This involves re-training the α's throughout the algorithm, every time the number of queries seen doubles. That is, you retrain the α's whenever $j = \epsilon m, 2\epsilon m, 4\epsilon m, \ldots, m/2$.

Breaking Ties

In the description of our algorithm, we assumed that there are never any ties, where $\arg\max_i b_{ij}(1-\alpha_i^*)$ is not uniquely defined. In fact, it is important to our arguments that such ties not be allowed to occur. However, in practice, such ties might occur frequently, for various possible reasons.

Our approach to resolving this problem is based on the observation that, if the bid vectors are in general position in $\mathbf{R}^n$, then for any α, there can be at most $n-1$ ties. Assume that we break ties greedily, that is assign to the highest bidder. Since all bids are in $[0,1]$ and $n \leq \epsilon B_i$ these ties don't introduce much of an error. Unfortunately, we cannot assume the bid vectors are in general position.

To get around this, suppose we choose, in advance, a tiny perturbation $\xi_{i,j}$ to be added to each bid b_{ij}. These will be chosen independently and uniformly at random from the interval $[0,\zeta]$, where $\zeta = O(\epsilon/m)$. Because the perturbations are chosen independently from continuous distributions, the perturbed bid vectors will be in general position with probability one. Because the perturbations are small, the exact amounts of the perturbations will have a negligible effect on the profit for any α. Indeed, if so desired, this effect can be made exactly zero by instead defining ζ as an infinitesimal.

Note that, since the perturbations are independently sampled, it makes no difference whether we think of these perturbations as being chosen before or after the query order is randomized.

4. I.I.D MODEL WITH UNKNOWN DISTRIBUTIONS

In this section, we consider a problem that is slightly different from the Adwords problem and present only the main ideas behind the technique, sacrificing details for clarity. The problem is as before, except that we now wish to simplify satisfy the constraints that $\forall i, \sum_j b_{ij}x_{ij} \leq B_i$ and for each j, it is allocated to *exactly* one i. Failing that, we wish to minimize the maximum violation, that is we wish to minimize $\max_i\{\sum_j b_{ij}x_{ij}/B_i\}$. In fact, for simplicity, we assume that $B_i = B$ for all i.

Recall that in the i.i.d model, the adversary picks a distribution, given by the a_{ij}'s and in every step j, j' is chosen uniformly at random and b_{ij} is set to $a_{ij'}$. Suppose that we know the a_{ij}'s. Let x_{ij}^* be a solution to the following system of inequalities (assuming that there is one solution).

$$\text{for all } i, \sum_j a_{ij}x_{ij} \leq B$$

$$\text{for all } j, \sum_i x_{ij} = 1.$$

$$x_{ij} \in \{0,1\}.$$

Now consider the following algorithm, which we call *Pure Random*: for all j in which j' was chosen, set $x_{ij} = x_{ij'}^*$. Let $Y_{ij} = b_{ij}x_{ij}$ and $Y_i = \sum_j Y_{ij}$. It is easy to show that for all $i, Y_i \leq B(1+\epsilon)$ with probability $1-\delta$ as long as $B \geq \Omega(\log(n/\delta)/\epsilon^2)$. The proof is by a straight-forward application of Chernoff bounds. Going into the proof of the Chernoff bounds, one bounds the expectation of the moment generating function $\exp(\lambda Y_i)$ for a suitable $\lambda(= \ln(1+\epsilon))$. That is, show

that $E[\exp(\lambda Y_i)] \leq \exp(\epsilon B)$. Then we apply Markov's inequality on $\exp(\lambda Y_i)$ to conclude that

$$Pr[Y_i > B(1+\epsilon)] = Pr[\exp(\lambda Y_i) > \exp(\lambda B(1+\epsilon))]$$

$$\leq \exp(\epsilon B)/\exp(\lambda B(1+\epsilon)) \approx \exp(-O(B\epsilon^2)) \leq \delta/n.$$

Finally, we take the union bound over all i.

The main result of Devanur et. al. [Devanur et al. 2011] (building upon the ideas of Charles et. al. [Charles et al. 2010]) is that one can actually achieve the same result without knowing the distribution! First, in the analysis of Pure Random, one can bound the sum of the moment generating functions for all i, instead of bounding them separately and then using the union bound, to get the same result. In other words, it is sufficient to show that $\sum_i E[\exp(\lambda Y_i)] \leq n \exp(\epsilon B)$. What the algorithm in [Devanur et al. 2011] does is to assign each j to greedily minimize the sum of these moment generating functions at every step. To be precise, let $X_{ij} = b_{ij}x_{ij}$ where x_{ij} is the allocation of the algorithm. The algorithm allocates j to

$$\arg\min_i \left\{ \sum_{i' \neq i} \exp(\lambda \sum_{j'<j} X_{i'j'}) + \exp(\lambda \sum_{j'<j} X_{ij'} + \lambda b_{ij}) \right\}. \tag{3}$$

We claim that $\sum_i E[\exp(\lambda X_i)] \leq n \exp(\epsilon B)$, where $X_i = \sum_j X_{ij}$. Let $\phi_i^j = \exp(\lambda \sum_{j' \leq j} X_{ij'})$. In each step the algorithm picks the i that minimizes $\sum_i \phi_i^j$. We show a bound on $E[\phi_i^j | \phi_i^{j-1}]$ that holds for an oblivious choice of i in that step, and hence the same bound should also hold for the choice of the algorithm. The oblivious choice that we consider is the allocation of Pure Random, that is suppose we used $x_{ij} = x^*_{ij'}$ where j' was the result of the random choice in step j. Then

$$E[\phi_i^j | \phi_i^{j-1}] = \phi_i^{j-1} E[\exp(\lambda b_{ij} x^*_{ij})] \leq \phi_i^{j-1} \exp(\epsilon B/m).$$

Thus $E[\sum_i \phi_i^j | \phi_i^{j-1}] \leq \sum_i \phi_i^{j-1} \exp(\epsilon B/m)$. The claim follows easily from this bound. From the claim, the following theorem follows as in the proof of Chernoff bounds.

THEOREM 4.1.

$$Pr[\ \exists\ i, X_i > B(1+\epsilon)] \leq \delta$$

given that $B \geq \Omega(\log(n/\delta)/\epsilon^2)$.

In fact, there is an alternate algorithm that achieves the same bound and is simpler to state and implement. Note that $\exp(\lambda b_{ij}) = (1+\epsilon)^{b_{ij}} \leq (1+\epsilon b_{ij})$. It turns out that (3) can be replaced with $\arg\min_i \left\{ \phi_i^{j-1} b_{ij} \right\}$.

5. EXTENSIONS AND GENERALIZATIONS

The technique presented here can be generalized to solve more general problems, as was shown in Feldman et. al. [Feldman et al. 2010], Agrawal, Wang and Ye [Agrawal et al. 2009] and Devanur et. al. [Devanur et al. 2011]. The class of problems is as follows: let there be n *resources*, with resource i having a capacity

of c_i that is given at the beginning of the algorithm. In each step j, the algorithm is given $a(i,j,k)$, $w_{j,k}$ for each i and for $k = 1..K$, where each k corresponds to one *option* that the algorithm can exercise. If the algorithm exercises option k, then $a(i,j,k)$ amount of resource i is consumed and a profit of $w_{j,k}$ is realized. The goal of the algorithm is to maximize the total profit realized, subject to the capacity constraints on the resources. γ is now defined as $\max_{i,j,k}\{a(i,j,k)/c_i, w_{j,k}/\mathrm{OPT}\}$. It is easy to see how the Adwords problem can be modeled as a special case of this framework.

6. CONCLUSION AND FUTURE DIRECTIONS

Analyzing online algorithms under stochastic assumptions about the input gives an effective way to get around the impossibility results that one encounters in the worst-case model. This also addresses the concerns about the pessimistic perspective of the worst-case model. However, the reality is bound to be somewhere in between; the stochastic assumptions need not hold in practice either. This calls for exploring the territory between these two models.

7. ACKNOWLEDGMENTS

I would like to thank my co-authors Tom Hayes, Denis Charles, Max Chickering, Kamal Jain, Manan Sanghi, Balasubramanian Sivan and Chris Wilkens for letting me write about our results in this article. I would also like to thank Claire Kenyon for her feedback on an earlier draft of the paper.

REFERENCES

Agrawal, S., Wang, Z., and Ye, Y. 2009. A dynamic near-optimal algorithm for online linear programming. arXiv:0911.2974v1.

Bahmani, B. and Kapralov, M. 2010. Improved bounds for online stochastic matching. In *ESA*. 170–181.

Buchbinder, N., Jain, K., and Naor, J. S. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA'07: Proceedings of the 15th annual European conference on Algorithms*. Springer-Verlag, Berlin, Heidelberg, 253–264.

Charles, D., Chickering, M., Devanur, N. R., Jain, K., and Sanghi, M. 2010. Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads. In *EC '10: Proceedings of the 11th ACM conference on Electronic commerce*. ACM, New York, NY, USA, 121–128.

Devanur, N. R. and Hayes, T. P. 2009. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, J. Chuang, L. Fortnow, and P. Pu, Eds. ACM, 71–78.

Devanur, N. R., Jain, K., Sivan, B., and Wilkens, C. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*.

Feldman, J., Henzinger, M., Korula, N., Mirrokni, V. S., and Stein, C. 2010. Online stochastic packing applied to display ad allocation. In *ESA*. 182–194.

Feldman, J., Mehta, A., Mirrokni, V., and Muthukrishnan, S. 2009. Online stochastic matching: Beating 1-1/e. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, 117–126.

Karande, C., Mehta, A., and Tripathi, P. 2011. Online bipartite matching with unknown distributions. In *STOC*.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. 1990. An optimal algorithm for on-line bipartite matching. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 352–358.

Kearns, M. J. and Vazirani, U. V. 1994. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA.

Mahdian, M. and Yan, Q. 2011. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *STOC*.

Manshadi, V., Gharan, S., and Saberi, A. 2011. Online stochastic matching: Online actions based on offline statistics. In *To appear in SODA*.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. 2005. Adwords and generalized on-line matching. In *In FOCS 05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 264–273.