Near Optimal Online Algorithms and Fast Approximation Algorithms for Resource Allocation Problems

Nikhil R Devanur* Kamal Jain† Balasubramanian Sivan‡ Christopher A. Wilkens§

Abstract

We present algorithms for a class of resource allocation problems both in the online setting with stochastic input and in the offline setting. This class of problems contains many interesting special cases such as the Adwords problem. In the online setting we introduce a new distributional model called the adversarial stochastic input model, which is a generalization of the i.i.d model with unknown distributions, where the distributions can change over time. In this model we give a $1 - O(\epsilon)$ approximation algorithm for the resource allocation problem, with almost the weakest possible assumption: the ratio of the maximum amount of resource consumed by any single request to the total capacity of the resource, and the ratio of the profit contributed by any single request to the optimal profit is at most $O\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$ where n is the number of resources available. There are instances where this ratio is $\epsilon^2/\log n$ such that no randomized algorithm can have a competitive ratio of $1 - o(\epsilon)$ even in the i.i.d model. The upper bound on ratio that we require improves on the previous upper-bound for the i.i.d case by a factor of n.

Our proof technique also gives a very simple proof that the greedy algorithm has a competitive ratio of $1 - 1/e$ for the Adwords problem in the i.i.d model with unknown distributions, and more generally in the adversarial stochastic input model, when there is no bound on the bid to budget ratio. All the previous proofs assume that either bids are very small compared to budgets or something very similar to this.

In the offline setting we give a fast algorithm to solve very large LPs with both packing and covering constraints. We give algorithms to approximately solve (within a factor of $1 + \epsilon$) the mixed packing-covering problem with $O(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ oracle calls where the constraint matrix of this LP has dimension $n \times m$, the success probability of the algorithm is $1 - \delta$, and γ is a parameter which is very similar to the ratio described for the online setting.

We discuss several applications, and how our algorithms improve existing results in some of these applications.

*Microsoft Research, 1 Microsoft Way, Redmond. Email: nikdev@microsoft.com.

†Microsoft Research, 1 Microsoft Way, Redmond. Email: kamalj@microsoft.com.

‡Computer Sciences Dept., University of Wisconsin-Madison. Email: balu2901@cs.wisc.edu. Part of this work was done while the author was at Microsoft Research, Redmond.

§Computer Science Division, University of California at Berkeley. Email: cwilkens@cs.berkeley.edu. Part of this work was done while the author was at Microsoft Research, Redmond.

1 Introduction

The results in this paper fall into distinct categories of competitive algorithms for online problems and fast approximation algorithms for offline problems. We have two main results in the online framework and one result in the offline setting. However they all share common techniques.

There has been an increasing interest in online algorithms motivated by applications to online advertising. The most well known is the *Adwords* problem introduced by Mehta et. al. [MSVV05], where the algorithm needs to assign keywords arriving online to bidders to maximize profit, subject to budget constraints for the bidders. The problem has been analyzed in the traditional framework for online algorithms: worst-case competitive analysis. As with many online problems, the worst-case competitive analysis is not entirely satisfactory and there has been a drive in the last few years to go beyond the worst-case analysis. The predominant approach has been to assume that the input satisfies some stochastic property. For instance the *random permutation* model (introduced by Goel and Mehta [GM08]) assumes that the adversary picks the set of keywords, but the order in which the keywords arrive is chosen uniformly at random. A closely related model is the *i.i.d* model: assume that the keywords are i.i.d samples from a fixed distribution, which is *unknown* to the algorithm. Stronger assumptions such as i.i.d samples from a *known* distribution have also been considered.

First Result. A key parameter on which many of the algorithms for Adwords depend is the bid to budget ratio. For instance in Mehta et. al. [MSVV05] and Buchbinder, Jain and Naor [BJN07] the algorithm achieves a worst case competitive ratio that tends to $1 - 1/e$ as the bid to budget ratio (let's call it γ) tends to 0. ($1 - 1/e$ is also the best competitive ratio that any randomized algorithm can achieve in the worst case.) Devanur and Hayes [DH09] showed that in the random permutation model, the competitive ratio tends to 1 as γ tends to 0. This result showed that competitive ratio of algorithms in stochastic models could be much better than that of algorithms in the worst case. The important question since then has been to determine the optimal trade-off between γ and the competitive ratio. [DH09] showed how to get a 1- $O(\epsilon)$ competitive ratio when γ is at most $O(\frac{\epsilon^3}{n \log(mn/\epsilon)})$ where n is the number of advertisers and m is the number of keywords. Subsequently Agrawal, Wang and Ye [AWY09] improved the bound on γ to $O(\frac{\epsilon^2}{n \log(m/\epsilon)})$. The papers of Feldman et. al. [FHK$^+$10] and Agrawal, Wang and Ye [AWY09] have also shown that the technique of [DH09] can be extended to other online problems.

The first main result in this paper is the following 3-fold improvement of previous results: (Theorems 2 - 4)

1. We give an algorithm which improves the bound on γ to $O(\frac{\epsilon^2}{\log(n/\epsilon)})$ This is almost *optimal*; we show a lower bound of $\frac{\epsilon^2}{\log(n)}$.
2. The bound applies to a more general model of stochastic input, called the *adversarial stochastic input* model. This is a generalization of the i.i.d model with unknown distribution, but is incomparable to the random permutation model.
3. It applies to a more general class of online problems that we call the *resource allocation framework*. A formal definition of the framework is presented in Section 2.2 and a discussion of many interesting special cases is presented in Section 7.

Regarding the bound on γ, the removal of the factor of n is significant. Consider for instance the Adwords problem and suppose that the bids are all in [0,1]. The earlier bound implies that the budgets need to be of the order of n/ϵ^2 in order to get a $1 - \epsilon$ competitive algorithm, where n is the number of advertisers. With realistic values for these parameters, it seems unlikely that this condition would be met. While with the improved bounds presented in this paper, we only need the budget to be of the order of $\log n/\epsilon^2$ and this condition is met for reasonable values of the parameters. Furthermore, in the resource allocation framework, the current highest upper bound on γ is from Agrawal, Wang and Ye [AWY09] and equals $O(\frac{\epsilon^2}{n \log(mk/\epsilon)})$. Here k is the number of available "options" (see Section 2.2) and in typical applications like network routing, k could be exponential in n, and thus, the factor saved by our algorithm becomes quadratic in n.

We note here that so far, all the algorithms for the i.i.d model (with unknown distribution) were actually designed for the random permutation model. It seems that any algorithm that works for one should also work for the other. However

we can only show that our algorithm works in the i.i.d model, so the natural question is if our algorithm works for the random permutation model. It would be very surprising if it didn't.

One drawback of the stochastic models considered so far is that they are time invariant, that is the input distribution does not change over time. The adversarial stochastic input model allows the input distribution to change over time. The model is as follows: in every step the adversary picks a distribution, possibly adaptively depending on what the algorithm has done so far, and the actual keyword in that step is drawn from this distribution. The competitive ratio is defined with respect to the optimum fractional solution for an *offline* instance of the problem, called the *distribution instance*, which is defined by the distribution. In Section 2.2, where we define the distribution instance, we also prove that the optimal fractional solution for the distribution instance is at least as good as the commonly used benchmark of expected value of optimal fractional solution, where the expectation is with respect to the distribution. A detailed description of this model, how the adversary is constrained to pick its distributions and how it differs from the worst-case model is presented in Section 2.2.

Second Result. Another important open problem is to improve the competitive ratio for the Adwords problem when there is no bound on γ. The best competitive ratio known for this problem is $1/2$ in the worst case. Nothing better was known, even in the stochastic models. (For the special case of online bipartite matching, in the case of i.i.d input with a *known* distribution, recent series of results achieve a ratio of better than 1-1/e, for instance by Feldman et. al. [FMMM09] and Bahmani and Kapralov [BK10]. The best ratio so far is .702 by Manshadi, Gharan and Saberi [MGS11]. The same online bipartite matching has been recently studied in the random permutation model by Mahdian and Yan [MY11] and by Karande, Mehta and Tripathi [KMT11]. The best ratio so far is 0.696 by Mahdian and Yan [MY11].) *The second result in this paper is that for the Adwords problem in the adversarial stochastic input model, with no assumption on γ, the greedy algorithm gets a competitive ratio of $1 - 1/e$ against the optimal fractional solution to the distribution instance (Theorem 5).* The greedy algorithm is particularly interesting since it is a natural algorithm that is used widely for its simplicity. Because of its wide use, previously the performance of the greedy algorithm has been analyzed by Goel and Mehta [GM08] who showed that in the random permutation and the i.i.d models, it has a competitive ratio of $1 - 1/e$ with an assumption which is essentially that γ tends to 0.

Third Result. Charles et. al. [CCD$^+$10] considered the following (offline) problem: given a lopsided bipartite graph $G = (L, R, E)$, that is a bipartite graph where $m = |L| \gg |R| = n$, does there exist an assignment $M : L \to R$ with $(j, M(j)) \in E$ for all $j \in L$, and such that for every vertex $i \in R$, $|M^{-1}(i)| \geq B_i$ for some given values B_i. Even though this is a classic problem in combinatorial optimization with well known polynomial time algorithms, the instances of interest are too large to use traditional approaches to solve this problem. (The value of m in particular is very large.) The approach used by [CCD$^+$10] was to essentially design an online algorithm in the i.i.d model: choose vertices from L uniformly at random and assign them to vertices in R in an online fashion. The online algorithm is guaranteed to be close to optimal, as long as sufficiently many samples are drawn. Therefore it can be used to solve the original problem (approximately): the online algorithm gets an almost satisfying assignment if and only if the original graph has a satisfying assignment (with high probability).

The third result in this paper is a generalization of this result to get fast approximation algorithms for a wide class of problems in the resource allocation framework (Theorem 6). Problems in the resource allocation framework where the instances are too large to use traditional algorithms occur fairly often, especially in the context of online advertising. Formal statements and a more detailed discussion are presented in Section 2.3.

The underlying idea used for all these results can be summarized at a high level as thus: consider a hypothetical algorithm called *Pure-random* that knows the distribution from which the input is drawn and uses an optimal solution w.r.t this distribution. Now suppose that we can analyze the performance of Pure-random by considering a potential function and showing that it decreases by a certain amount in each step. Now we can design an algorithm that does not know the distribution as follows: consider the same potential function, and in every step choose the option that minimizes the potential function. Since the algorithm minimizes the potential in each step, the decrease in the potential for this algorithm is better than that for Pure-random and hence we obtain the same guarantee as that for Pure-random.

For instance, for the case where γ is small, the performance of Pure-random is analyzed using Chernoff bounds. The Chernoff bounds are proven by showing bounds on the expectation of the moment generating function of a random variable. Thus the potential function is the sum of the moment generating functions for all the random variables that we apply the Chernoff bounds to. The proof shows that in each step this potential function decreases by some multiplicative

factor. The algorithm is then designed to achieve the same decrease in the potential function. A particularly pleasing aspect about this technique is that we obtain very simple proofs. For instance, the proof of Theorem 5 is extremely simple: the potential function in this case is simply the total amount of unused budgets and we show that this amount (in expectation) decreases by a factor of $1 - 1/m$ in each step where there are m steps in all.

On the surface, this technique and the resulting algorithms[1] bear a close resemblance to the algorithms of Young [You95] for derandomizing randomized rounding and the fast approximation algorithms for solving covering/packing LPs of Plotkin, Shmoys and Tardos [PST91], Garg and Könemann [GK98] and Fleischer [Fle00]. In fact Arora, Hazan and Kale [AHK05] showed that all these algorithms are related to the multiplicative weights update method for solving the *experts* problem and especially highlighted the similarity between the potential function used in the analysis of the multiplicative update method and the moment generating function used in the proof of Chernoff bounds and Young's algorithms. Hence it is no surprise that our algorithm is also a multiplicative update algorithm. It seems that our algorithm is closer in spirit to Young's algorithms than others. It is possible that our algorithm can also be interpreted as an algorithm for the experts problem. In fact Mehta et. al. [MSVV05] asked if there is a $1 - o(1)$ competitive algorithm for Adwords in the i.i.d model with small bid to budget ratio, and in particular if the algorithms for experts could be used. They also conjectured that such an algorithm would iteratively adjust a budget discount factor based on the rate at which the budget is spent. Our algorithms for resource allocation problem when specialized for Adwords look exactly like that and with the connections to the experts framework, we answer the questions in [MSVV05] in the positive.

Organization. The rest of the paper is organized as follows. In Section 2, we define the resource allocation framework, the adversarial stochastic model and state our results formally as theorems. We also discuss one special case of the resource allocation framework — the adwords problem and formally state our results. In Section 3, we consider a simplified "min-max" version of the resource allocation framework and present the proofs for this version. The other results build upon this simple version. In Section 4 we give a fast approximation algorithm for the mixed covering-packing problem (Theorem 6). The $1 - O(\epsilon)$ competitive online algorithm for the resource allocation framework with stochastic input (Theorem 2) is in Section 5. The $1-1/e$ competitive algorithm (Theorem 5) for the Adwords problem is in Section 6. Several special cases of the resource allocation framework are considered in Section 7. Section 8 concludes with some open problems and directions for future research.

2 Preliminaries & Main Results

2.1 Resource allocation framework

We consider the following framework of optimization problems. There are n resources, with resource i having a capacity of c_i. There are m requests; each request j can be satisfied by a vector $\mathbf{x}_j$ that is constrained to be in a polytope $\mathcal{P}_j$. (We refer to the vector $\mathbf{x}_j$ as an option to satisfy a request, and the polytope $\mathcal{P}_j$ as the set of options.) The vector $\mathbf{x}_j$ consumes $\mathbf{a}_{i,j} \cdot \mathbf{x}_j$ amount of resource i, and gives a profit of $\mathbf{w}_j \cdot \mathbf{x}_j$. Note that $\mathbf{a}_{i,j},\ \mathbf{w}_j$ and $\mathbf{x}_j$ are all vectors. The objective is to maximize the total profit subject to the capacity constraints on the resources. The following LP describes the problem:

$$\begin{aligned}
&\text{maximize } \sum_j \mathbf{w}_j \cdot \mathbf{x}_j \text{ s.t.} \\
&\forall\, i, \sum_j \mathbf{a}_{i,j} \cdot \mathbf{x}_j \leq c_i \\
&\forall\, j, \mathbf{x}_j \in \mathcal{P}_j.
\end{aligned}$$

We assume that we have the following oracle available to us: given a request j and a vector $\mathbf{v}$, the oracle returns the vector $\mathbf{x}_j$ that maximizes $\mathbf{v}.\mathbf{x}_j$ among all vectors in $\mathcal{P}_j$. Let $\gamma = \max\left(\left\{\frac{\mathbf{a}_{i,j}.\mathbf{x}_j}{c_i}\right\}_{i,j} \cup \left\{\frac{\mathbf{w}_j.\mathbf{x}_j}{W^*}\right\}_j\right)$ be the notion corresponding to the bid to budget ratio for Adwords. Here W^* is the optimal offline objective to the distribution instance, defined in Section 2.2.

[1] For the case of small γ. It is not clear if this discussion applies to the case of large γ, that is to Theorem 5

The canonical case is where each $\mathcal{P}_j$ is a unit simplex in $\mathbf{R}^K$, i.e. $\mathcal{P}_j = \{\mathbf{x}_j \in \mathbf{R}^K : \sum_k x_{j,k} = 1\}$. This captures the case where there are K discrete options, each with a given profit and consumption. This case captures most of the applications we are interested in, which are described in Section 7. All the proofs will be presented for this special case, for ease of exposition. The co-ordinates of the vectors $\mathbf{a}_{i,j}$ and $\mathbf{w}_j$ will be denoted by $a(i,j,k)$ and $w_{j,k}$ respectively, i.e., the k^{th} option consumes $a(i,j,k)$ amount of resource i and gives a profit of $w_{j,k}$. For an example of an application that needs more general polytopes see Section 7.5.

We consider two versions of the above problem. The first is an online version with stochastic input: requests are drawn from an unknown distribution. The second is when the number of requests is much larger than the number of resources, and our goal is to design a fast PTAS for the problem.

2.2 Online Algorithms with Stochastic Input

We now consider an online version of the resource allocation framework. Here requests arrive online. We consider the i.i.d. model, where each request is drawn independently from a given distribution. The distribution is unknown to the algorithm. The algorithm knows m, the total number of requests. The competitive ratios we give for resource allocation problems with bounded γ are with respect to an upper bound on the expected value of fractional optimal solution, namely, the fractional optimal solution of the distribution instance, defined below.

Consider the following *distribution instance* of the problem. It is an offline instance defined for any given distribution over requests and the total number of requests m. The capacities of the resources in this instance are the same as in the original instance. Every request in the support of the distribution is also a request in this instance. Suppose request j occurs with probability p_j. The resource consumption of j in the distribution instance is given by $mp_j\mathbf{a}_{i,j}$ for all i and the profit is $mp_j\mathbf{w}_j$. The intuition is that if the requests were drawn from this distribution then the expected number of times request j is seen is mp_j and this is represented in the distribution instance by scaling the consumption and the profit vectors by mp_j. To summarize, the distribution instance is as follows.

$$\begin{aligned}
&\text{maximize} \sum_{j \text{ in the support}} mp_j \mathbf{w}_j.\mathbf{x}_j \text{ s.t.} \\
&\forall\, i, \sum_j mp_j \mathbf{a}_{i,j}.\mathbf{x}_j \le c_i \\
&\forall\, j, \mathbf{x}_j \in \mathcal{P}_j.
\end{aligned}$$

We now prove that the fractional optimal solution to the distribution instance is an upper bound on the expectation of OPT, where OPT is the offline fractional optimum of the actual sequence of requests.

Lemma 1 $\text{OPT}[\text{Distribution instance}] \ge \mathbf{E}[\text{OPT}]$

Proof: The average of optimal solutions for all possible sequences of requests should give a feasible solution to the distribution instance with a profit equal to $E[\text{OPT}]$. Thus the optimal profit for the distribution instance could only be larger. □

The *competitive ratio* of an algorithm in the i.i.d model is defined as the ratio of the expected profit of the algorithm to the fractional optimal profit for the distribution instance. The main result is that as γ tends to zero, the competitive ratio tends to 1. In fact, we give the almost optimal trade-off.

Theorem 2 *For any $\epsilon > 0$, we give an algorithm such that if $\gamma = O\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$ then the competitive ratio of the algorithm is $1 - O(\epsilon)$.*

Theorem 3 *There exist instances with $\gamma = \frac{\epsilon^2}{\log(n)}$ such that no algorithm can get a competitive ratio of $1 - o(\epsilon)$.*[2]

Also, our algorithm works when the polytope $\mathcal{P}_j$ is obtained as an LP relaxation of the actual problem.[3] To be precise, suppose that the set of options that could be used to satisfy a given request corresponds to some set of vectors,

[2] The proof of this theorem is obtained by a modification of a similar theorem for random permutations presented in [AWY09].

[3] There may be trivial ways of defining $\mathcal{P}_j$ such that its vertices correspond to the actual options. The motivation for allowing non-trivial relaxations is computational: recall that we need to be able to optimize linear functions over $\mathcal{P}_j$.

say $\mathcal{I}_j$. Let the polytope $\mathcal{P}_j \supseteq \mathcal{I}_j$ be an α *approximate relaxation* of $\mathcal{I}_j$ if for the profit vector $\mathbf{w}_j$ and for all $\mathbf{x}_j \in \mathcal{P}_j$, there is an oracle that returns a $\mathbf{y}_j \in \mathcal{I}_j$ such that $\mathbf{w}_j.\mathbf{y}_j \geq \alpha \mathbf{w}_j.\mathbf{x}_j$. Given such an oracle, our algorithm achieves a competitive ratio of $\alpha - O(\epsilon)$.

Theorem 4 *Given a resource allocation problem with an α approximate relaxation, and for any $\epsilon > 0$, we give an algorithm such that if $\gamma = O\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$ then the competitive ratio of the algorithm is $\alpha - O(\epsilon)$.*

Proof:(Sketch.) Consider the problem in the resource allocation framework defined by the relaxation, that is request j can actually be satisfied by the polytope $\mathcal{P}_j$. The optimal solution to the relaxation is an upper bound on the optimal solution to the original problem. Now run Algorithm 1 on the relaxation, and when the algorithm picks a vector $\mathbf{x}_j$ to serve request j, use the rounding oracle to round it to a solution $\mathbf{y}_j$ and use this solution. Since the sequence of $\mathbf{x}_j$'s picked by the algorithm are within $1 - O(\epsilon)$ of the optimum for the relaxation, and for all j $\mathbf{w}_j.\mathbf{y}_j \geq \alpha \mathbf{w}_j.\mathbf{x}_j$, this algorithm is $\alpha(1 - O(\epsilon))$ competitive. □

In fact, our results hold for the following more general model, the *adversarial stochastic input* model. In each step, the adversary adaptively chooses a distribution from which the request in that step is drawn. The adversary is constrained to pick the distributions in one of the following two ways. In the first case, we assume that a target objective value OPT_T is given to the algorithm [4], and that the adversary is constrained to pick distributions such that the fractional optimum solution of *each* of the corresponding distribution instances is at least OPT_T (or at most OPT_T for minimization problems). The competitive ratio is defined with respect to OPT_T. In the second case, we are not given a target, but the adversary is constrained to pick distributions so that the fractional optimum of each of the corresponding distribution instances is the same, which is the benchmark with respect to which the competitive ratio is defined.

Note that while the i.i.d model can be reduced to the random permutation model, these generalizations are incomparable to the random permutation model as they allow the input to vary over time. Also the constraint that each of the distribution instances has a large optimum value distinguishes this from the worst-case model. This constraint in general implies that the distribution must contain sufficiently rich variety of requests in order for the corresponding distribution instance to have a high optimum. To truly simulate the worst-case model, in every step the adversary would chose a "deterministic distribution", that is a distribution supported on a single request. Then the distribution instance will simply have m copies of this single request and hence will not have a high optimum. For instance consider online bipartite b-matching where each resource is a node on one side of a bipartite graph with the capacity c_i denoting the number of nodes it can be matched to and the requests are nodes on the other side of the graph and can be matched to at most one node. A deterministic distribution in this case corresponds to a single online node and if that node is repeated m times then the optimum for that instance is just the weighted (by c_i) degree of that node. If the adversary only picks such deterministic distributions then he is constrained to pick nodes of very high degree thus making it easy for the algorithm to match them.

We refer the reader to Section 7 for a discussion on several problems that are special cases of the resource allocation framework and have been previously considered. Here, we discuss one special case — the adwords problem.

2.2.1 The Adwords problem

In the i.i.d *Adwords* problem, there are n bidders, and each bidder i has a daily budget of B_i dollars. *Keywords* arrive online with keyword j having an (unknown) probability p_j of arriving in any given step. For every keyword j, each bidder submits a bid, b_{ij}, which is the profit obtained by the algorithm on allocating keyword j to bidder i. The objective is to maximize the profit, subject to the constraint that no bidder is charged more than his budget. Here, the resources are the daily budgets of the bidders, the requests are the keywords, and the options are once again the bidders. The amount of resource consumed and the profit are both b_{ij}.

For this problem, with no bounds on γ, we show that the greedy algorithm has a competitive ratio of $1 - 1/e$. For our results for the adwords problem with bounded γ, see Section 7.1

Theorem 5 *The greedy algorithm achieves a competitive ratio of $1 - 1/e$ for the Adwords problem in the adversarial stochastic input model with no assumptions on the bid to budget ratio.*

We note here that the competitive ratio of $1 - 1/e$ is tight for the greedy algorithm [GM08]. It is however not known to be tight for an arbitrary algorithm.

[4] In this case, we require $\gamma = \max\left(\left\{\frac{\mathbf{a}_{i,j}.\mathbf{x}_j}{c_i}\right\}_{i,j} \cup \left\{\frac{\mathbf{w}_j.\mathbf{x}_j}{\text{OPT}_T}\right\}_j\right)$, i.e., OPT_T takes the place of W^* in the definition of γ.

2.3 Fast algorithms for very large LPs

Charles et al. [CCD+10] consider the following problem: given a bipartite graph $G = (L, R, E)$ where $m = |L| \gg |R| = n$, does there exist an assignment $M : L \to R$ with $(j, M(j)) \in E$ for all $j \in L$, and such that for every vertex $i \in R$, $|M^{-1}(i)| \geq B_i$ for some given values B_i. They gave an algorithm that runs in time linear[5] in the number of edges of an induced subgraph obtained by taking a random sample from R of size $O\left(\frac{m \log n}{\min_i\{B_i\}\epsilon^2}\right)$, for a gap-version of the problem with gap ϵ. When $\min_i\{B_i\}$ is reasonably large, such an algorithm is very useful in a variety of applications involving ad assignment for online advertising.

We consider a generalization of the above problem (that corresponds to the resource allocation framework). In fact, we consider the following mixed covering-packing problem. Suppose that there are n_1 packing constraints, one for each $i \in \{1..n_1\}$ of the form $\sum_{j=1}^{m} \mathbf{a}_{i,j}\mathbf{x}_j \leq c_i$ and n_2 covering constraints, one for each $i \in \{1..n_2\}$ of the form $\sum_{j=1}^{m} \mathbf{b}_{i,j}\mathbf{x}_j \geq d_i$. Each $\mathbf{x}_j$ is constrained to be in $\mathcal{P}_j$. Does there exists a feasible solution to this system of constraints? The gap-version of this problem is as follows. Distinguish between the two cases, with a high probability, say $1 - \delta$:

YES: There is a feasible solution.

NO: There is no feasible solution even if all of the c_i's are multiplied by $1 + \epsilon$ and all of the d_i's is multiplied by $1 - \epsilon$.

We note that solving (offline) an optimization problem in the resource allocation framework can be reduced to the above problem through a binary search on the objective function value.

Suppose as in [CCD+10] that m is much larger than n. Assume that solving the following costs unit time: given j and $\mathbf{v}$, find $\mathbf{x}_j \in \mathcal{P}_j$ that maximizes $\mathbf{v}.\mathbf{x}_j$. Let $\gamma = \max\{i \in [n_1], j \in [m] : \frac{\mathbf{a}_{i,j}.\mathbf{x}_j}{c_i}\} \cup \{i \in [n_2], j \in [m] : \frac{\mathbf{b}_{i,j}.\mathbf{x}_j}{d_i}\}$.

Theorem 6 *For any $\epsilon > 0$, there is an algorithm that solves the gap version of the mixed covering-packing problem with a running time of $O\left(\frac{\gamma m \log(n/\delta)}{\epsilon^2}\right)$.*

Applications to online advertising:

The matching problem introduced by [CCD+10] was motivated by the problem of computing the available inventory for display ad allocation (see the original paper for details). In fact, the matching problem was a simplified version of the real problem, which fits into the resource allocation framework. Moreover, such algorithms are used in multiple ways. For instance, although the technique of Devanur and Hayes [DH09] was originally designed to solve the purely online problem, it can be used in the PAC model where the algorithm can make use of a prediction of the future arrival of requests (see for instance Vee, Vassilvitskii and Shanmugasundaram [VVS10]). The key technique is to formulate an LP relaxation of the problem and learn the optimal dual variables using the prediction, and these duals can then be used for the allocation online. Even if the prediction is not entirely accurate, we note that such an approach has certain advantages. This motivates the problem of finding the optimal duals. We observe that our algorithm can also be used to compute near optimal duals which can then be used to do the allocation online. Many problems (for instance the Display ad allocation problem) can benefit from such an algorithm.

A similar approach was considered by Abrams, Mendelevitch and Tomlin [AMT07] for the following problem motivated by sponsored search auctions: for each *query* j, one can show an advertiser in each of the K slots. Each advertiser i bids a certain amount on each query j, and has a daily budget. However, the cost to an advertiser depends on the entire ordered set of advertisers shown (called a *slate*), based on the rules of the auction. Given the set of queries that arrive in a day (which in practice is an estimate of the queries expected rather than the actual queries), the goal is to schedule a slate of advertisers for each query such that the total cost to each advertiser is within the budget and maximize a given objective such as the total revenue, or the social welfare. This problem is modeled as an LP and a column-generation approach is suggested to solve it. Also, many compromises are made, in terms of limiting the number of queries, etc. due to the difficulties in solving an LP of very large size. We observe that this LP fits in the resource allocation framework and thus can be solved quickly using our algorithm.

Chernoff bounds. We present here the form of Chernoff bounds that we use throughout the rest of this paper. Let $X = \sum_i X_i$, where $X_i \in [0, B]$ are i.i.d random variables. Let $\mathbf{E}[X] = \mu$. Then, for all $\epsilon > 0$,

$$\Pr[X < \mu(1 - \epsilon)] < \exp\left(\frac{-\epsilon^2\mu}{2B}\right).$$

[5] In fact, the algorithm makes a single pass through this graph.

Consequently, for all $\delta > 0$, with probability at least $1 - \delta$,

$$X - \mu \geq -\sqrt{2\mu B \ln(1/\delta)}.$$

Similarly, for all $\epsilon \in [0, 2e - 1]$,

$$\Pr[X > \mu(1 + \epsilon)] < \exp\left(\frac{-\epsilon^2 \mu}{4B}\right).$$

Consequently, for all $\delta > \exp(\frac{-(2e-1)^2\mu}{4B})$, with probability at least $1 - \delta$,

$$X - \mu \leq \sqrt{4\mu B \ln(1/\delta)}.$$

3 Min-Max version

In this section, we solve a slightly simplified version of the general online resource allocation problem, which we call the min-max version. In this problem, m requests arrive online, and each of them must be served. The objective is to minimize the maximum fraction of any resource consumed. (There is no profit.) The following LP describes it formally.

$$\begin{aligned}
&\text{minimize } \lambda \text{ s.t.} \\
&\forall\, i, \sum_{j,k} a(i,j,k) x_{j,k} \leq \lambda c_i \\
&\forall\, j, \sum_{k} x_{j,k} = 1, \\
&\forall\, j, k, x_{j,k} \geq 0.
\end{aligned}$$

For ease of illustration, we assume that the requests arrive i.i.d (unknown distribution) in the following proof. At the end of this section, we show that the proof holds for the adversarial stochastic input model also.

The algorithm proceeds in steps. Let λ^* denote the fractional optimal objective value of the distribution instance of this problem. Let X_i^t be the random variable indicating the amount of resource i consumed during step t, that is, $X_i^t = a(i,j,k)$ if in step t, request j was chosen and was served using option k. Let $S_i^T = \sum_{t=1}^{T} X_i^t$ be the total amount of resource i consumed in the first T steps. Let $\gamma = \max_{i,j}\{\frac{a(i,j,k)}{c_i}\}$, which implies that for all i, j and k, $a(i,j,k) \leq \gamma c_i$. Let $\phi_i^t = (1+\epsilon)^{S_i^t/(\gamma c_i)}$. For the sake of convenience, we let $S_i^0 = 0$ and $\phi_i^0 = 1$ for all i. The algorithm is as follows.

ALG Min-max In step $t+1$, on receiving request j, use option $\arg\min_k \left\{\sum_i \frac{a(i,j,k)\phi_i^t}{c_i}\right\}$.

Lemma 7 *For any $\epsilon \in (0,1]$, The algorithm **ALG Min-max** described above approximates λ^* within a factor of $(1+\epsilon)$, with a probability at least $1 - \delta$, where $\delta = n \exp\left(\frac{-\epsilon^2 \lambda^*}{4\gamma}\right)$*

We will prove Lemma 7 through a series of lemmas, namely Lemmas 8, 9 and 10. Before we begin the proof, we give some intuition. Consider a hypothetical algorithm, call it *Pure-random*, that knows the distribution. Let $\mathbf{x}_j^*$ denote the optimal fractional solution to the distribution instance. Pure-random is a non-adaptive algorithm which uses $\mathbf{x}_j^*$ to satisfy request j, i.e., it serves request j using option k with probability x_{jk}. Suppose we wanted to prove a bound on the performance of Pure-random, that is show that with high probability, Pure-random is within $1 + O(\epsilon)$ of the optimum, say λ^*. This can be done using Chernoff bounds: for each resource separately bound the probability that the total consumption is more than $\lambda^* c_i(1 + O(\epsilon))$ using Chernoff bounds and take a union bound. Note that the Chernoff bounds are shown by proving an upper bound on the expectation of the moment generating function of the random variables. If we could show the same bound for our algorithm, then we would be done. Let $\phi^t = \sum_i \phi_i^t$. We wish to upper bound the expectation of ϕ^m.

Consider the state of the algorithm after the first t steps. Let $\tilde{X}_i^{t+1}$ denote the amount of resource i consumed in step $t+1$ had we served the request at step $t+1$ using the Pure-random algorithm instead of our algorithm. Then we show that the expectation of ϕ^{t+1} is upper bounded by $\sum_i \phi_i^t \left(1 + \epsilon \frac{\tilde{X}_i^{t+1}}{\gamma c_i}\right)$, and the rest of the proof is along the lines of the Chernoff bound proof.

Lemma 8 *For all t,*

$$\phi^{t+1} \leq \sum_i \phi_i^t \left(1 + \epsilon \frac{\tilde{X}_i^{t+1}}{\gamma c_i}\right).$$

Proof:

$$\begin{aligned}
\phi^{t+1} &= \sum_i \phi_i^{t+1} = \sum_i \phi_i^t (1+\epsilon)^{\frac{X_i^{t+1}}{\gamma c_i}} \\
&\leq \sum_i \phi_i^t \left(1 + \epsilon \frac{X_i^{t+1}}{\gamma c_i}\right) \leq \sum_i \phi_i^t \left(1 + \epsilon \frac{\tilde{X}_i^{t+1}}{\gamma c_i}\right)
\end{aligned}$$

The first inequality is because the convexity of the function $(1+\epsilon)^x$ can be used to upper bound it by $1 + \epsilon x$ for all $x \in [0,1]$, and $X_i^t \leq \max_{j,k} a(i,j,k) \leq \gamma c_i$. The second inequality follows from the definition of our algorithm as it chooses the option that minimizes $\left\{\sum_i \frac{a(i,j,k)\phi_i^t}{c_i}\right\}$ $\square$

Lemma 9 *For all T, $\mathbf{E}[\phi^T] \leq n \exp\left(\frac{\epsilon \lambda^* T}{\gamma m}\right)$, where λ^* is the optimal solution to the LP.*

Proof: From Lemma 8, it follows that

$$\begin{aligned}
\mathbf{E}\left[\phi^{t+1} \mid \phi_i^t \text{ for all } i\right] &\leq \mathbf{E}\left[\sum_i \phi_i^t \left(1 + \epsilon \frac{\tilde{X}_i^{t+1}}{\gamma c_i}\right)\right] \\
&\leq \sum_i \phi_i^t \left(1 + \epsilon \frac{\lambda^*}{\gamma m}\right) \\
&= \phi^t \left(1 + \epsilon \frac{\lambda^*}{\gamma m}\right) \\
&\leq \phi^t \exp\left(\frac{\epsilon \lambda^*}{\gamma m}\right)
\end{aligned}$$

and hence the lemma follows since $\phi^0 = n$. The second inequality follows from the fact that $\mathbf{E}[\tilde{X}_i^{t+1}] \leq \frac{\lambda^* c_i}{m}$ for all i. This is because requests are drawn i.i.d, and hence the optimal value of the distribution instance is the same for all time steps and is equal to λ^*. $\square$

Lemma 10

$$\mathbf{Pr}\left[\max_i \left\{\frac{S_i^T}{c_i}\right\} > \frac{T}{m}\lambda^*(1+\epsilon)\right] \leq n \exp\left(\frac{-\epsilon^2 T \lambda^*}{4\gamma m}\right).$$

Proof:

$$\begin{aligned}
\mathbf{Pr}\left[\max_i \left\{\frac{S_i^T}{c_i}\right\} > \frac{T}{m}\lambda^*(1+\epsilon)\right] &\leq \mathbf{Pr}\left[\phi^T > (1+\epsilon)^{\frac{T\lambda^*(1+\epsilon)}{\gamma m}}\right] \\
&\leq \mathbf{E}[\phi^T]/(1+\epsilon)^{\frac{T\lambda^*(1+\epsilon)}{\gamma m}} \\
&\leq n \exp\left(\frac{\epsilon \lambda^* T}{\gamma m}\right)/(1+\epsilon)^{\frac{T\lambda^*(1+\epsilon)}{\gamma m}} \\
&\leq n \exp\left(\frac{-\epsilon^2 T \lambda^*}{4\gamma m}\right).
\end{aligned}$$

The inequality in the first line is because $\max_i \phi_i^T \leq \phi_T$. The rest is similar to proofs of Chernoff bounds. The second line follows from Markov's inequality, the third line from Lemma 9 and the fourth line is a well known algebraic inequality whenever $\epsilon \in (0, 2e-1]$, and in particular when $\epsilon \in (0,1]$. $\square$

Substituting $T = m$ in Lemma 10, we get Lemma 7.

Adversarial stochastic input model In the above lemmas, we assumed that the requests are drawn i.i.d, i.e., we used the fact that $\mathbf{E}[\tilde{X}_i^t] \leq \lambda^* c_i/m$ for all t. But in an adversarial stochastic model, since the distribution from which a request is drawn changes each step, the optimal objective of the distribution instance also changes every step, i.e., it could be λ_t^* at step t. So, in the proof Lemma 9, where we proved that

$$\mathbf{E}\left[\phi^{t+1} \mid \phi_i^t \text{ for all } i\right] \leq \phi^t \exp\left(\frac{\epsilon\lambda^*}{\gamma m}\right),$$

we would instead have $\mathbf{E}\left[\phi^{t+1} \mid \phi_i^t \text{ for all } i\right] \leq \phi^t \exp\left(\frac{\epsilon\lambda_t^*}{\gamma m}\right)$. But given a target λ^*, we know the adversary is constrained to pick distributions whose distribution instance has an optimum objective *at most* λ^* (recall that this is a minimization problem). Therefore, we can upper bound $\phi^t \exp\left(\frac{\epsilon\lambda_t^*}{\gamma m}\right)$ by $\phi^t \exp\left(\frac{\epsilon\lambda^*}{\gamma m}\right)$. The rest of the steps in the proof remain the same. Thus, the adversary is not constrained to pick requests from the same distribution at every time step. All we require is that, whatever distribution it uses for drawing its request, the corresponding distribution instance has an optimum objective value at most λ^*, which is the target value we aim for.

In the following sections, we illustrate all our proofs in the i.i.d model with unknown distribution and it is easy to convert them to proofs for the adversarial stochastic input model.

4 Mixed Covering-Packing and Online Resource Allocation

4.1 Mixed Covering-Packing

In this section, we consider the mixed packing-covering problem stated in Section 2.3. and prove Theorem 6. We restate the LP for the mixed covering-packing problem here.

$$\begin{aligned}
&\forall\, i, \sum_{j,k} a(i,j,k)x_{j,k} \leq c_i \\
&\forall\, i, \sum_{j,k} b(i,j,k)x_{j,k} \geq d_i \\
&\forall\, j, \sum_{k} x_{j,k} \leq 1, \\
&\forall\, j,k, x_{j,k} \geq 0.
\end{aligned}$$

The goal is to check if there is a feasible solution to this LP. We solve a gap version of this problem. Distinguish between the two cases with a high probability, say $1-\delta$:

YES: There is a feasible solution.

NO: There is no feasible solution even if all of the c_i's are multiplied by $1+\epsilon$ and all of the d_i's are multiplied by $1-\epsilon$.

For convenience of description, we refer to the quantities indexed by j as requests, those indexed by i as resources and those indexed by k as options.

As before, the algorithm proceeds in steps. In each step, the algorithm samples a request i.i.d from the total of m possible requests. We will prove that if the number of samples $T \geq \Theta(\frac{\gamma m \ln(n/\delta))}{\epsilon^2})$, then the algorithm solves the gap version with probability at least $(1-\delta)$. Since the time taken for serving any given request is one (by taking the time consumed by a single oracle call to be one), this proves that the total run-time is $O(\frac{\gamma m \ln(n/\delta))}{\epsilon^2})$. This proves Theorem 6.

Let X_i^t, $\tilde{X}_i^t$, S_i^t be as defined in Section 3. Let Y_i^t be the random variable indicating the amount of demand i satisfied during step t, that is, $Y_i^t = b(i,j,k)$ if in step t, request j was chosen and was served using option k. Let $\tilde{Y_i}^t$ denote the amount of demand i satisfied during step t by the optimal algorithm for the distribution instance of this problem. Let $V_i^T = \sum_{t=1}^T Y_i^t$. Let $\phi_i^t = \eta_c(1+\frac{\epsilon}{2})^{\frac{S_i^t}{\gamma c_i}}(1+\frac{\epsilon}{2\gamma m})^{T-t}$, where $\eta_c = (1+\frac{\epsilon}{2})^{-(1+\frac{\epsilon}{2})\frac{T}{\gamma m}}$. Let $\psi_i^t = \eta_d(1-\frac{\epsilon}{2})^{\frac{V_i^t}{\gamma d_i}}(1-\frac{\epsilon}{2\gamma m})^{T-t}$, where $\eta_d = (1-\frac{\epsilon}{2})^{-(1-\frac{\epsilon}{2})\frac{T}{\gamma m}}$. Let $\phi^t = \sum_i \phi_i^t$, let $\psi^t = \sum_i \psi_i^t$ and $\Phi^t = \phi^t + \psi^t$. As before, we let $S_i^0 = 0$ and $V_i^0 = 0$. The algorithm is as follows.

ALG Packing-Covering Given request j in step $t+1$, use the option

$$\arg\min_k \left\{ \frac{1}{\left(1+\frac{\epsilon}{2\gamma m}\right)} \sum_i \phi_i^t \frac{a(i,j,k)}{c_i} - \frac{1}{\left(1-\frac{\epsilon}{2\gamma m}\right)} \sum_i \psi_i^t \frac{b(i,j,k)}{d_i} \right\}$$

At the end of T steps, the algorithm checks if $\max_i \frac{S_i^T}{c_i} < \frac{T}{m}(1+\frac{\epsilon}{2})$ and if $\min_i \frac{V_i^T}{d_i} > \frac{T}{m}(1-\frac{\epsilon}{2})$. If true, the algorithm answers YES. Else it says NO. We now proceed to prove that, whenever the real answer is YES, the algorithm says YES with a high probability. Lemmas 11 and 12 prove this case.

Lemma 11 *For a* YES *instance* $\mathbf{E}\left[\Phi^T\right] \leq \Phi^0$.

Proof: Similar to the proof of Lemma 8, we have

$$\Phi^{t+1} \leq \sum_i \phi_i^t \frac{\left(1+\frac{\epsilon \tilde{X}_i^{t+1}}{2\gamma c_i}\right)}{\left(1+\frac{\epsilon}{2\gamma m}\right)} + \sum_i \psi_i^t \frac{\left(1-\frac{\epsilon \tilde{Y_i}^{t+1}}{2\gamma d_i}\right)}{\left(1-\frac{\epsilon}{2\gamma m}\right)}.$$

$$\mathbf{E}\left[\Phi^{t+1} | \phi_i^t,\ \psi_i^t \text{ for all } i\right] \leq \sum_i \phi_i^t \frac{\left(1+\frac{\epsilon}{2\gamma m}\right)}{\left(1+\frac{\epsilon}{2\gamma m}\right)} + \sum_i \psi_i^t \frac{\left(1-\frac{\epsilon}{2\gamma m}\right)}{\left(1-\frac{\epsilon}{2\gamma m}\right)} = \Phi^t$$

where the inequality follows from the fact that, when the real answer is YES, $\mathbf{E}[\tilde{X}_i^t] \leq \frac{c_i}{m}$ and $\mathbf{E}[\tilde{Y_i}^t] \geq \frac{d_i}{m}$ for all i. Since the above sequence of inequalities holds for every t, the lemma follows. □

Lemma 12 *For a* YES *instance*

$$\mathbf{Pr}\left[\max_i \frac{S_i^T}{c_i} \geq \frac{T}{m}(1+\frac{\epsilon}{2})\right] + \mathbf{Pr}\left[\min_i \frac{V_i^T}{d_i} \leq \frac{T}{m}(1-\frac{\epsilon}{2})\right] \leq \Phi^0$$

Proof: As in proof of Lemma 10

$$\mathbf{Pr}\left[\max_i \frac{S_i^T}{c_i} \geq \frac{T}{m}(1+\frac{\epsilon}{2})\right] \leq \mathbf{Pr}\left[\frac{\phi^T}{\eta_c} \geq \frac{1}{\eta_c}\right] \leq \mathbf{E}[\phi^T]$$

where the inequality in the first line follows from $\phi_i^T \leq \phi^T$ for all i, and the next line follows from Markov's inequality. Similarly, we have

$$\mathbf{Pr}\left[\min_i \frac{V_i^T}{d_i} \leq \frac{T}{m}(1-\frac{\epsilon}{2})\right] \leq \mathbf{Pr}\left[\frac{\psi^T}{\eta_d} \geq \frac{1}{\eta_d}\right] \leq \mathbf{E}[\psi^T]$$

Thus the sum of these probabilities is at most $\mathbf{E}[\phi^T] + \mathbf{E}[\psi^T] = \mathbf{E}[\Phi^T]$, which is at most Φ^0 from Lemma 11. □

Observe that Φ^0, the failure probability equals $n\left(\eta_c(1+\frac{\epsilon}{2\gamma m})^T + \eta_d(1-\frac{\epsilon}{2\gamma m})^T\right)$, which is upper bounded by $n\left(\exp\left(\frac{-\epsilon^2 T}{16\gamma m}\right) + \exp\left(\frac{-\epsilon^2 T}{8\gamma m}\right)\right)$. If $T = O(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$, we have the failure probability to be at most δ. Thus Lemma 12 proves that the algorithm **ALG Packing-Covering** says YES with a probability at least $1-\delta$ when the real answer is YES.

We now proceed to prove that when the real answer is NO, our algorithm says NO with a probability at least $1-\delta$, i.e.,

Lemma 13 *For a* NO *instance, if* $T \geq \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$*, then*

$$\mathbf{Pr}\left[\max_i \frac{S_i^T}{c_i} < \frac{T}{m}(1+\frac{\epsilon}{2}) \ \&\ \min_i \frac{V_i^T}{d_i} > \frac{T}{m}(1-\frac{\epsilon}{2})\right] < \delta.$$

Proof: Let S denote the set of requests sampled. Consider the following LP.

$$\begin{aligned}
&\text{minimize } \lambda \qquad (1)\\
&\forall\, i, \lambda - \sum_{j\in S,k} \frac{a(i,j,k)x_{j,k}}{c_i} \geq -\frac{T}{m}\\
&\forall\, i, \lambda + \sum_{j\in S,k} \frac{b(i,j,k)x_{j,k}}{d_i} \geq \frac{T}{m}\\
&\forall\, j \in S, \sum_k x_{j,k} \leq 1\\
&\forall\, j,k, x_{j,k} \geq 0\\
&\lambda \geq 0.
\end{aligned}$$

If the above LP has an optimal objective value at least $\frac{T\epsilon}{2m}$, then our algorithm would have declared NO. We now show that by picking $T = \Theta(\frac{\gamma m \ln(n/\delta)}{\epsilon^2})$, the above LP will have its optimal objective value at least $\frac{T\epsilon}{2m}$, with a probability at least $1-\delta$. This makes our algorithm answer NO with a probability at least $1-\delta$.

Consider the dual of LP (1):

$$\begin{aligned}
&\text{maximize} \sum_{j\in S} \beta_j + \frac{T}{m}\sum_i(\rho_i - \alpha_i) \qquad (2)\\
&\forall\, j \in S,k,\ \beta_j \leq \sum_i \left(\alpha_i \frac{a(i,j,k)}{c_i} - \rho_i \frac{b(i,j,k)}{d_i}\right)\\
&\sum_i(\alpha_i + \rho_i) \leq 1\\
&\forall\, i, \alpha_i, \rho_i \geq 0\\
.\quad &\forall\, j \in S, \beta_j \geq 0.
\end{aligned}$$

The optimal value of LP (1) is equal to the optimal value to LP (2), which in turn is lower bounded by the value of LP (2) at any feasible solution. One such feasible solution is $\alpha^*, \beta^*, \rho^*$, which is the optimal solution to the full version of LP (2), namely the one with $S = [m], T = m$. Thus, the optimal value of LP (1) is lower bounded by value of LP (2) at $\alpha^*, \beta^*, \rho^*$, which is

$$= \sum_{j\in S} \beta_j^* + \frac{T}{m}(\sum_i \rho_i^* - \alpha_i^*) \qquad (3)$$

For proceeding further in lower bounding (3), we apply Chernoff bounds to $\sum_{j\in S} \beta_j^*$. In order to get useful Chernoff bounds, we first prove that β_j^* resides in a small interval. Consider the full version of LP (2), i.e., $S = [m]$ and $T = m$. In this version, since according to the second constraint the optimal solution must satisfy $\sum_i(\alpha_i^* + \rho_i^*) \leq 1$, it follows that $0 \leq \beta_j^* \leq \gamma$ according to the first constraint. Further, let τ^* denote the optimal value of the full version of LP (2). Recall that since we are in the NO case $\tau^* \geq \epsilon$. Now, because of the constraint $\sum_i(\alpha_i^* + \rho_i^*) \leq 1$, it follows that $\sum_i(\rho_i^* - \alpha_i^*) \geq -1$ and thus it follows that $\sum_j \beta_j^* \leq \tau^* + 1 \leq 2\max(\tau^*, 1)$. We are now ready to lower bound the quantity in (3). We have the optimal solution to LP (2)

$$\begin{aligned}
&\geq \sum_{j\in S} \beta_j^* + \frac{T}{m}(\sum_i \rho_i^* - \alpha_i^*)\\
&\geq \frac{T\sum_j \beta_j^*}{m} - \sqrt{\frac{2T(\sum_j \beta_j^*)\gamma\ln(1/\delta)}{m}} + \frac{T}{m}\sum_i(\rho_i^* - \alpha_i^*) \left(\text{Since } \beta_j^* \in [0,\gamma]\right)\\
&\geq \frac{T\tau^*}{m} - \sqrt{\frac{4T\max(\tau^*,1)\gamma\ln(1/\delta)}{m}} \qquad (4)
\end{aligned}$$

where the second inequality is a "with probability at least $1-\delta$" inequality, i.e., we apply Chernoff bounds for $\sum_{j\in S}\beta_j^*$, along with the observation that each $\beta_j^* \in [0,\gamma]$

We now set T so that the quantity in (4) is at least $\frac{T\epsilon}{2m}$. Noting that $\tau^* \geq \epsilon$, setting $T = \Theta(\frac{\gamma m \ln(n/\delta)}{\epsilon^2})$ will ensure this inequality and hence proves the lemma. □

Lemmas 11, 12 and 13 prove that the gap-version of the mixed covering-packing problem can be solved in time $O(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$, thus proving Theorem 6.

5 Online Algorithms with Stochastic Input

In this section, we use the potential function based algorithm to solve the online version of the resource allocation problem introduced in Section 2.2. The following LP describes the resource allocation problem.

$$\begin{aligned} &\text{maximize } \sum_{j,k} w_{j,k}x_{j,k} \text{ s.t.} \\ &\forall\, i, \sum_{j,k} a(i,j,k)x_{j,k} \leq c_i \\ &\forall\, j, \sum_{k} x_{j,k} \leq 1, \\ &\forall\, j,k, x_{j,k} \geq 0. \end{aligned} \tag{5}$$

Our algorithm computes increasingly better estimates of the objective value by computing the optimal solution for the observed requests, and uses it to guide future allocations. This is similar to the algorithm in [AWY09], except that we only need to estimate the value of the optimal solution as against the entire solution itself. Through Lemmas 14 and **??**, we show that our algorithm achieves a competitive ratio of $1 - O(\epsilon)$ thus proving Theorem 2. We assume that the number of requests m is known in advance. Algorithm 1 describes our algorithm.

Algorithm 1 : Algorithm for stochastic online resource allocation

Initialize $t_0 : t_0 \leftarrow \lceil \epsilon m \rceil$,
for r = 0 to $l-1$ **do**
  for $t = t_r + 1$ to t_{r+1} **do**
    if $t = t_r + 1$ **then**
      If the incoming request is j, use the following option k:

$$\arg\min_k \left\{ \frac{\epsilon_c(r)/\gamma}{\left(1+\frac{\epsilon_c(r)}{\gamma m}\right)} \sum_i \phi_i^{init}(r)\frac{a(i,j,k)}{c_i} - \frac{\epsilon_o(r)/w_{\max}}{\left(1-\frac{\epsilon_o(r)Z(r)}{w_{\max}m}\right)}\phi_{\text{obj}}^{init}(r)w_{j,k} \right\}.$$

      For all i, $S_i^t(r) = X_i^t$, and $V^t(r) = Y^t$.
    else
      If the incoming request is j, use the following option k:

$$\arg\min_k \left\{ \frac{\epsilon_c(r)/\gamma}{\left(1+\frac{\epsilon_c(r)}{\gamma m}\right)} \sum_i \phi_i^{t-1}\frac{a(i,j,k)}{c_i} - \frac{\epsilon_o(r)/w_{\max}}{\left(1-\frac{\epsilon_o(r)Z(r)}{w_{\max}m}\right)}\phi_{\text{obj}}^{t-1}w_{j,k} \right\}.$$

      For all i, $S_i^t(r) = S_i^{t-1}(r) + X_i^t$, and, $V^t(r) = V^{t-1}(r) + Y^t$.
    end if
  end for
end for

The first ϵm requests are not served but used just for computational purposes. After these first ϵm requests, the algorithm proceeds in l stages, namely $0,\ 1,\ \ldots,\ l-1$, where l is such that $\epsilon 2^l = 1$ and ϵ is a positive number between

0 and 1 that the algorithm designer gets to choose. In stage r the algorithm serves $t_r = \epsilon m 2^r$ requests. Note that the stage r consists of all steps $t \in (t_r, t_{r+1}]$.

Let W^* denote the optimal solution to the distribution instance of the problem. Let X_i^t be as defined in Section 3. Let Y^t be the amount of profit earned during step t, i.e., $Y^t = w_{j,k}$, if in step t, request j was served using option k. Instead of the usual S_i^t, we now define $S_i^t(r) = \sum_{u=t_r+1}^{t} X_i^u$, which is the sum of X_i^u's till t for u's belonging to stage r alone, i.e., $u \in (t_r, t_{r+1}]$. Similarly, $V^t(r) = \sum_{u=t_r+1}^{t} Y^u$. Let $w_{\max} = \max_{j,k} w_{j,k}$.

The potential function for constraint i in step t when $t \in (t_r + 1, t_{r+1}]$ is defined by

$$\phi_i^t = \eta_c(r)(1+\epsilon_c(r))^{\frac{S_i^t(r)}{\gamma c_i}} \left(1 + \frac{\epsilon_c(r)}{\gamma m}\right)^{t_r - t},$$

$$\eta_c(r) = (1+\epsilon_c(r))^{-(1+\epsilon_c(r))\frac{t_r}{\gamma m}},$$

$$\epsilon_c(r) = \sqrt{\frac{4\gamma m \ln((n+1)/\delta)}{t_r}}.$$

Similarly, the potential function for objective at step t is,

$$\phi_{\text{obj}}^t = \eta_{\text{obj}}(r)(1-\epsilon_o(r))^{\frac{V^t(r)}{w_{\max}}} \left(1 - \frac{\epsilon_o(r) Z(r)}{w_{\max} m}\right)^{t_r - t},$$

$$\eta_{\text{obj}}(r) = (1-\epsilon_o(r))^{-(1-\epsilon_o(r))\frac{t_r Z(r)}{m w_{\max}}}.$$

$$\epsilon_o(r) = \sqrt{\frac{2 w_{\max} m \ln((n+1)/\delta)}{t_r Z(r)}}.$$

When $t = t_r + 1$, which is a special case marking the beginning of a new stage, the potential function for constraint i is defined by

$$\phi_i^{init}(r) = \eta_c(r) \left(1 + \frac{\epsilon_c(r)}{\gamma m}\right)^{t_r}$$

and the potential function for the objective function is given by

$$\phi_{obj}^{init}(r) = \eta_{\text{obj}}(r) \left(1 - \frac{\epsilon_o(r) Z(r)}{w_{\max} m}\right)^{t_r}.$$

Note that apart from constants, the only difference between $\epsilon_o(r)$ and $\epsilon_c(r)$ is that instead of γ, $\epsilon_o(r)$ has $w_{\max}/Z(r)$. The value $Z(r)$, as we define below, gets progressively updated, but within a single stage r remains the same. After stage r, the algorithm computes the optimal objective value e_r to the following instance $\mathcal{I}_r$: the instance $\mathcal{I}_r$ has the t_r requests of stage-r, and the capacity of resource i is $\frac{t_r c_i (1+\epsilon_c(r))}{m}$, i.e., the capacity of resources are scaled down according to the number of requests by a factor of $\frac{t_r}{m}$, along with a slight extra allowance by a factor of $(1+\epsilon_c(r))$. It uses e_r to compute the value $Z(r+1)$ to be used in the potential function for objective in stage $r+1$.

Lemma 14 *With probability at least* $1 - 2\delta$*,*

$$\frac{t_r W^*(1-\epsilon_c(r))}{m} \leq e_r \leq \frac{t_r W^*(1+2\epsilon_c(r))}{m}.$$

Proof: Our goal is to establish bounds on e_r, which is the optimal solution to the instance $\mathcal{I}_r$. For this we consider the distribution instance of the following instance $\hat{\mathcal{I}}_r$: this instance is the same as $\mathcal{I}_r$, except that each resource i has a capacity of $t_r c_i/m$ instead of $t_r c_i(1+\epsilon_c(r))/m$. We denote this distribution instance of $\hat{\mathcal{I}}_r$ by $D(\hat{\mathcal{I}}_r)$. Let p_r be the objective value obtained by the algorithm Pure-random on the instance $\mathcal{I}_r$, i.e., the objective value obtained by following the allocation used by the optimal solution to $D(\hat{\mathcal{I}}_r)$. Since the resource usage and objective value obtained by the algorithm Pure-random can be seen as the sum of i.i.d random variables, we can apply Chernoff bounds on them. With a probability at least $1-\delta$, Pure-random's usage of resource i will be within a factor of $(1+\epsilon_c(r))$ of $\frac{t_r c_i}{m}$ and

Pure-random's objective value will fall short of $t_r W^*/m$ (which is the optimal objective value for $D(\hat{\mathcal{I}}_r)$) by a factor of at most $\left(1 - \sqrt{\frac{2w_{\max} m \ln((n+1)/\delta)}{t_r W^*}}\right)$. Since the instance $\mathcal{I}_r$ allows for this extra resource usage by having an additional capacity by a factor of $(1 + \epsilon_c(r))$ as compared to $\hat{\mathcal{I}}_r$, the real excess resource usage occurs only with probability δ. Thus, with probability $1 - \delta$, resources are consumed within capacity and the objective value p_r is at least

$$\begin{aligned} p_r &\geq \frac{t_r W^*}{m}\left(1 - \sqrt{\frac{2w_{\max} m \ln((n+1)/\delta)}{t_r W^*}}\right) \\ &\geq \frac{t_r W^*(1-\epsilon_c(r))}{m}. \end{aligned}$$

The solution obtained by Pure-random is just a feasible solution for $\mathcal{I}_r$, and thus p_r is smaller than the optimal objective value for $\mathcal{I}_r$, which is e_r. Thus, with probability more than $1 - \delta$, we have,

$$e_r \geq \frac{t_r W^*(1-\epsilon_c(r))}{m}. \tag{6}$$

We now get an upper-bound on e_r. To do this, we consider the LP which defines e_r, along with its dual. The LP which defines e_r is given by:

$$\begin{aligned} &\text{maximize} \sum_{j \in \mathcal{I}_r} w_{j,k} x_{j,k} \\ &\forall\, i, \sum_{j \in \mathcal{I}_r, k} a(i,j,k) x_{j,k} \leq \frac{t_r c_i (1+\epsilon_c(r))}{m} \\ &\forall\, j \in \mathcal{I}_r, \sum_k x_{j,k} \leq 1 \\ &\forall\, j \in \mathcal{I}_r,\ k, x_{j,k} \geq 0. \end{aligned} \tag{7}$$

Consider the dual of LP (7)

$$\begin{aligned} &\text{minimize} \sum_{j \in \mathcal{I}_r} \beta_j + \frac{t_r(1+\epsilon_c(r))}{m}(\sum_i \alpha_i c_i) \\ &\forall\, j \in \mathcal{I}_r, k,\ \beta_j + \sum_i a(i,j,k)\alpha_i \geq w_{j,k} \\ &\forall\, i, \alpha_i \geq 0 \\ &\forall\, j \in \mathcal{I}_r, \beta_j \geq 0. \end{aligned} \tag{8}$$

.

The optimal value of LPs (7) and (8) are the same and equal to e_r. The optimal value of LP (8) is upper bounded by the LP's value at a feasible solution. One such feasible solution is α^*, β^*, which is the optimal solution to the full version of LP (8), namely for which $\mathcal{I}_r$ is replaced by $[m]$ (i.e., contains all the requests), t_r is replaced by m and $\epsilon_c(r)$ is replaced by zero. The value of LP (8) at α^*, β^* is equal to

$$\sum_{j \in \mathcal{I}_r} \beta_j^* + \frac{t_r(1+\epsilon_c(r))}{m}(\sum_i \alpha_i^* c_i) \tag{9}$$

To upper bound the expression in (9), we apply Chernoff bounds to $\sum_{j \in \mathcal{I}_r} \beta_j^*$. Notice that by constraint 1 of (the

full version of) LP (8), we get $\beta_j^* \leq w_{\max}$. Using this, we upper bound the optimal solution to LP (8) as

$$\begin{aligned}
&\leq \sum_{j \in \mathcal{I}_r} \beta_j^* + \frac{t_r(1+\epsilon_c(r))}{m}(\sum_i \alpha_i^* c_i) \\
&\leq \frac{t_r \sum_j \beta_j^*}{m} + \sqrt{\frac{4t_r(\sum_j \beta_j^*) w_{\max} \ln(1/\delta)}{m}} + \frac{t_r(1+\epsilon_c(r))}{m}(\sum_i \alpha_i^* c_i) \\
&\leq \frac{t_r W^*}{m} + \frac{2t_r W^* \epsilon_c(r)}{m}.
\end{aligned}$$

where the second inequality is a "with probability at least $1-\delta$" inequality, i.e., we apply Chernoff bounds for $\sum_{j \in \mathcal{I}_r} \beta_j^*$ along with the observation that $\beta_j^* \leq w_{\max}$. The third inequality follows from noting that $\sum_j \beta_j^* + \sum_i \alpha_i^* c_i = W^*$ and all α_i^*'s and β_j^*'s are non-negative, and thus $\sum_j \beta_j^* \leq W^*$.

Thus $e_r \leq \frac{t_r W^*(1+2\epsilon_c(r))}{m}$ with probability at least $1-\delta$.

Thus we have proved that the upper bound and the lower bound on e_r hold with probability $1-\delta$ each and hence together with a probability at least $1-2\delta$. This proves the lemma. □

Using these e_r's, we define our $Z(r+1)$ as follows:

$$Z(r+1) = \frac{m e_r}{t_r(1+2\epsilon_c(r))}.$$

Using the bounds on e_r in Lemma 14, we note that $Z(r+1) \leq W^*$ and that $Z(r+1) \geq \frac{W^*(1-\epsilon_c(r))}{1+2\epsilon_c(r)} \geq W^*(1-3\epsilon_c(r))$. Thus with probability at least $1 - 2\log(1/\epsilon)\delta$, $Z(r)$ satisfies these bounds for all r. Given the bounds on $Z(r)$, we use Lemma 12 to see that with a probability at least $1-\delta$, the objective value achieved in stage r is at least $\frac{t_r Z(r)}{m}(1-\epsilon_o(r))$, and the amount of resource i consumed in stage r is at most $\frac{t_r c_i}{m}(1+\epsilon_c(r))$. Hence, these bounds are true for all r with probability at least $1 - \log(1/\epsilon)\delta$, since the total number of stages $l = \log(1/\epsilon)$.

The total failure probability is the sum of the failure probability during estimation of $Z(r)$ through e_r, given by $2\log(1/\epsilon)\delta$ and the failure probability of our algorithm in all stages together given by $\log(1/\epsilon)\delta$. Thus, the total failure probability is $3\log(1/\epsilon)\delta$.

With a probability of $1 - 3\log(1/\epsilon)\delta$, the algorithm obtains an objective value of at least

$$\sum_{r=0}^{l-1} \frac{t_r Z(r)(1-\epsilon_o(r))}{m},$$

and for each i, the amount of resource i consumed is at most

$$\sum_{r=0}^{l-1} \frac{t_r c_i (1+\epsilon_c(r))}{m}.$$

On setting $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$, and $\delta = \frac{\epsilon}{\log(1/\epsilon)}$, we get that with a failure probability of at most $O(\epsilon)$, the objective value achieved by the algorithm is $W^*(1-O(\epsilon))$ and all resources are consumed within capacity. Thus, we have a $1-O(\epsilon)$ competitive algorithm, proving Theorem 2.

6 Adwords in i.i.d setting

In this section, we give a simple proof of Theorem 5: greedy algorithm achieves a competitive ratio of $(1-1/e)$ in the adwords problem, where the impressions come from an adversarial stochastic input model. As before, we illustrate our proofs for the i.i.d model with unknown distribution below. We now briefly describe the adwords setting.

Setting. There are a total of n advertisers, and queries arrive online, from some pool of queries. Let the (unknown) number of queries that arrive be m. The queries that appear each day are drawn i.i.d from some unknown distribution. Advertiser i bids an amount b_{ij} on query j. Advertiser i has a budget B_i denoting the maximum amount of money that can be spent on a given day. The bid amounts b_{ij} are revealed online as the queries arrive. The objective is to maximize the sum of the bid amounts successfully allocated, subject to budget constraints. Whenever a query j arrives, with a bid amount $b_{ij} >$ remaining budget of i, we are still allowed to allot that query to advertiser i, but we only earn a revenue of the remaining budget of i, and not the total value b_{ij}.

Goel and Mehta [GM08] prove that the greedy algorithm gives a $(1 - 1/e)$ approximation to the adwords problem when the queries arrive in a random permutation or in i.i.d, but under an assumption which almost gets down to bids being much smaller than budgets. We give a much simpler proof for a $(1 - 1/e)$ approximation by greedy algorithm for the i.i.d unknown distributions case, and our proof works irrespective of the the relation between the size of the bids and the budgets involved.

Let p_j be the probability of query j appearing in any given impression. Let $y_j = mp_j$. Let x_{ij} denote the offline fractional optimal solution for the distribution instance. Let $w_i(t)$ denote the amount of money spent by advertiser i at time step t, i.e., for the t-th query in the greedy algorithm (to be described below). Let $f_i(0) = \sum_j b_{ij}x_{ij}y_j$. Let $f_i(t) = f_i(0) - \sum_{r=1}^{t} w_i(r)$. Let $f(t) = \sum_{i=1}^{n} f_i(t)$. Note that $f_i(0)$ is the amount spent by i in the offline fractional optimal solution to the distribution instance.

Consider the greedy algorithm which allocates the query j arriving at time t to the advertiser who has the maximum effective bid for that query, i.e., $\operatorname{argmax}_i \min\{b_{ij}, B_i - \sum_{r=1}^{t-1} w_i(r)\}$. We prove that this algorithm obtains a revenue of $(1 - 1/e)\sum_{i,j} b_{ij}x_{ij}y_j$ and thus gives the desired $1 - 1/e$ competitive ratio against the fractional optimal solution to the distribution instance. The proof is similar to the proof we presented in Lemma 8 for the resource allocation problem. Consider a hypothetical algorithm that allocates queries to advertisers according to the x_{ij}'s. We prove that this hypothetical algorithm obtains an expected revenue of $(1-1/e)\sum_{i,j} b_{ij}x_{ij}y_j$, and argue that the greedy algorithm only performs better. Let $w_i^h(t)$ and $f_i^h(t)$ denote the quantities analogous to $w_i(t)$ and $f_i(t)$ for the hypothetical algorithm, with the initial value $f_i^h(0) = f_i(0) = \sum_j b_{ij}x_{ij}y_j$. Let $f^h(t) = \sum_{i=1}^{n} f_i^h(t)$. Let $\text{EXCEED}_i(t)$ denote the set of all j such that b_{ij} is strictly greater than the remaining budget at the beginning of time step t, namely $b_{ij} > B_i - \sum_{r=1}^{t-1} w_i^h(r)$.

Lemma 15 $\mathbf{E}[w_i^h(t)|f_i^h(t-1)] \geq \frac{f_i^h(t-1)}{m}$

Proof: The expected amount amount of money spent at time step t, is given by

$$\mathbf{E}[w_i^h(t)|f_i^h(t-1)] = \sum_{j \in \text{EXCEED}_i(t)} \left(B_i - \sum_{r=1}^{t-1} w_i^h(r)\right)\frac{x_{ij}y_j}{m} + \sum_{j \notin \text{EXCEED}_i(t)} b_{ij}\frac{x_{ij}y_j}{m}. \tag{10}$$

If $\sum_{j \in \text{EXCEED}_i(t)} x_{ij}y_j \geq 1$, then by (10),

$$\mathbf{E}[w_i^h(t)|f_i^h(t-1)] \geq \frac{B_i - \sum_{r=1}^{t-1} w_i^h(r)}{m} \geq \frac{f_i^h(0) - \sum_{r=1}^{t-1} w_i^h(r)}{m} = \frac{f_i^h(t-1)}{m}.$$

Suppose on the other hand $\sum_{j \in \text{EXCEED}_i(t)} x_{ij}y_j < 1$. We can write $\mathbf{E}[w_i^h(t)|f_i^h(t-1)]$ as

$$\mathbf{E}[w_i^h(t)|f_i^h(t-1)] = \frac{f_i^h(0)}{m} - \sum_{j \in \text{EXCEED}_i(t)} \left(b_{ij} - (B_i - \sum_{r=1}^{t-1} w_i^h(r))\right)\frac{x_{ij}y_j}{m}. \tag{11}$$

Since $b_{ij} \leq B_i$, and $\sum_{j \in \text{EXCEED}_i(t)} x_{ij}y_j < 1$, (11) can be simplified to

$$\begin{aligned} \mathbf{E}[w_i^h(t)|f_i^h(t-1)] &> \frac{f_i^h(0)}{m} - \frac{\sum_{r=1}^{t-1} w_i^h(r)}{m} \\ &= \frac{f_i^h(t-1)}{m}. \end{aligned}$$

□

Lemma 16 *The hypothetical algorithm satisfies the following:* $\mathbf{E}[f^h(t)|f^h(t-1)] \leq f^h(t-1)(1-1/m)$

Proof: From the definition of $f_i^h(t)$, we have

$$\begin{aligned} f_i^h(t) &= f_i^h(t-1) - w_i^h(t) \\ \mathbf{E}[f_i^h(t)|f_i^h(t-1)] &= f_i^h(t-1) - \mathbf{E}[w_i^h(t)|f_i^h(t-1)] \leq f_i^h(t-1)(1-\frac{1}{m}), \end{aligned}$$

where the inequality is due to Lemma 15. Summing over all i gives the Lemma. □

Lemma 17 $\mathbf{E}[\text{GREEDY}] \geq (1-1/e)\sum_{i,j} b_{ij}x_{ij}y_j$

Proof: Lemma 16 proves that for the hypothetical algorithm, the value of the difference $f^h(t-1) - \mathbf{E}[f^h(t)|f^h(t-1)]$, which is the expected amount spent at time t by all the advertisers together, conditioned on $f^h(t-1)$, is at least $\frac{f^h(t-1)}{m}$. But by definition, conditioned on the amount of money spent in first $t-1$ steps, the greedy algorithm earns the maximum revenue at time step t . Thus, for the greedy algorithm too, the statement of the lemma 16 must hold, namely, $\mathbf{E}[f(t)|f(t-1)] \leq f(t-1)(1-1/m)$. This means that $\mathbf{E}[f(m)] \leq f(0)(1-1/m)^m \leq f(0)(1/e)$. Thus the expected revenue earned is

$$\begin{aligned} \mathbf{E}[\sum_{r=1}^{m} w(r)] &= f(0) - \mathbf{E}[f(m)] \\ &\geq f(0)\,(1-1/e) \\ &= (1-1/e)\sum_{i,j} b_{ij}x_{ij}y_j \end{aligned}$$

and this proves the lemma. □

Lemma 17 proves Theorem 5.

7 Applications

We now list the problems that are special cases of the resource allocation framework and have been previously considered.

7.1 Adwords Problem

While in Section 2.2.1 we noted that we could get a $1-1/e$ approximation to the adwords problem with unbounded γ, we note here that γ is small, i.e. $\max_{i,j} \frac{b_{ij}}{B_i} \leq O\left(\frac{\epsilon^2}{\log(n/\epsilon)}\right)$, we get a $1-O(\epsilon)$ approximation to the maximum profit through the resource allocation framework.

7.2 Display Ad Allocation

The following problem occurs in the allocation of *display ads*[6] and is rather similar to the Adwords problem. Here, there are *impressions* that arrive online and have to be allocated to advertisers. Each advertiser i has a value v_{ij} for each impression j. The difference is that in this case, the advertisers have a bound on the total *number* of impressions that they can be allocated to. The objective is to maximize the total value of the allocation. The LP formulation for this problem fits directly in our resource allocation framework.

[6]These are shown on web sites, as opposed to search ads that are shown on search engines and were the motivation for the Adwords problem.

7.3 Network Routing and Load Balancing

Consider a graph (either undirected or directed) with edge capacities. Requests arrive online; a request j consists of a source-sink pair, (s_j, t_j) and a bandwidth ρ_j. In order to satisfy a request, a capacity of ρ_j must be allocated to it on every edge along some path from s_j to t_j in the graph. In the *congestion minimization* version, all requests must be satisfied, and the objective is to minimize the maximum (over all edges) congestion, which is the ratio of the allocated bandwidth to the capacity of the edge. In the *throughput maximization* version, the objective is to maximize the number of satisfied requests while not allocating more bandwidth than the available capacity for each edge. (Different requests could have different values on them, and one could also consider maximizing the total value of the satisfied requests.) Both the congestion minimization version and the throughput maximization version can be solved through our algorithm 1 for resource allocation framework. Kamath, Palmon and Plotkin [KPP96] considered a variant of this problem with the requests arriving according to a stationary Poisson process, and show a competitive ratio that is very similar to ours.

7.4 Combinatorial Auctions

Suppose we have n items for sale, with c_i copies of item i. Bidders arrive online, and bidder j has a utility function $U_j : 2^{[n]} \rightarrow \mathbf{R}$. If we posted prices p_i for each item i, then bidder j buys a bundle S that maximizes $U_j(S) - \sum_{i \in S} p_i$. We assume that bidders can compute such a bundle. The goal is to maximize social welfare, the total utility of all the bidders, subject to the supply constraint that there are only c_i copies of item i. Firstly, incentive constraints aside, this problem can be written as an LP in the resource allocation framework. The items are the resources and agents arriving online are the requests. All the different subsets of items form the set of options. The utility $U_j(S)$ represents the profit $w_{j,S}$ of serving agent j through option S, i.e. subset S. If an item $i \in S$, then $a_{i,j,S} = 1$ for all j and zero otherwise. Incentive constraints aside, our algorithm for resource allocation at step t, will choose the option k (or equivalently the bundle S) as specified in point 4 of algorithm 1, i.e., minimize the potential function. That can be equivalently written as,

$$\arg\max_k \left\{ \frac{\epsilon_o(r)/w_{\max}}{\left(1 - \frac{\epsilon_o(r) Z(r)}{w_{\max} m}\right)} \phi_{\text{obj}}^{t-1} w_{j,k} - \frac{\epsilon_c(r)/\gamma}{\left(1 + \frac{\epsilon_c(r)}{\gamma m}\right)} \sum_i \phi_i^{t-1} \frac{a(i,j,k)}{c_i} \right\}$$

Now, maximizing the above expression is the same as picking the k to maximize $w_{j,k} - \sum_i p_i a(i,j,k)$, where

$$p_i = \frac{\frac{\epsilon_c(r)/\gamma}{\left(1+\frac{\epsilon_c(r)}{\gamma m}\right)}}{\frac{\epsilon_o(r)/w_{\max}}{\left(1-\frac{\epsilon_o(r) Z(r)}{w_{\max} m}\right)} \phi_{\text{obj}}^{t-1}} \cdot \frac{\phi_i^{t-1}}{c_i}.$$

Thus, if we post these prices p_i on items, agents will do exactly what the algorithm would have done otherwise. Suppose that the bidders are i.i.d samples from some distribution (or they arrive as in the adversarial stochastic input model). Here $\gamma = 1/\min_i\{c_i\}$ and we can use Theorem 2 to get an incentive compatible posted price auction[7] with a competitive ratio of $1 - O(\epsilon)$ whenever $\min_i\{c_i\} \geq O\left(\frac{\log(n/\epsilon)}{\epsilon^2}\right)$. Further if an analog of Theorem 2 also holds in the random permutation model then we get a similar result for combinatorial auctions in the offline case: we simply consider the bidders one by one in a random order.

7.5 Selective Call-out

Chakraborty et. al. [CEDG$^+$10] formulated the following problem that arises in the design of an ad-exchange. An exchange gets *ad-requests* online; each ad-request may have multiple slots with different *qualities*. Whenever the exchange gets an ad-request, it calls out to a subset of ad-networks for a bid. Given the bids it then allocates the slots to the highest bidders. The ad-networks have constraints on how frequently they want to be called out. In addition, the following assumptions are made: the ad-requests are i.i.d samples from an unknown distribution, and for every ad-network its values for all ad-requests of a certain type are i.i.d from a distribution that is known to the exchange. They consider various objective functions, such as social welfare, revenue of a particular auction, GSP with reserve, and so

[7]Here we assume that each agent reveals his true utility function *after* he makes his purchase. This information is necessary to compute the prices to be charged for future agents.

on. They state their results in the PAC model, where they use an initial sample of impressions to train their algorithm. They give a bound on the number of samples needed in order to get a $1 - 1/e - \epsilon$ competitive algorithm. We can use our algorithms (with an approximate relaxation, Theorem 4) to improve their results in the following two ways. Either we are given the target objective value, in which case we achieve the same competitive ratio in the online setting without the need for an initial sample. If we are not given the target objective value then we need an initial sample to estimate that value. The number of samples we need is less than what is required by [CEDG$^+$10] by a factor of n. Further, our algorithm would also work in the adversarial stochastic input model.

8 Conclusion and Future Work

Our work raises the following open questions.

- As mentioned in the introduction, we can show that our algorithm works in the i.i.d model, so the natural question is if our algorithm works for the random permutation model.
- Currently in our algorithm for the online case, we need to estimate the optimum objective function value periodically. For this we need to solve (at least approximately) an offline instance of the problem repeatedly. Is there an algorithm that avoids this?
- Perhaps the holy grail for an online algorithm for say, Adwords, is to get a guarantee of the following form: if the input is i.i.d from some given distribution then get a competitive ratio that is close to 1, while simultaneously getting a competitive ratio of $1 - 1/e$ if the input is adversarial. Our algorithm for Adwords (or some simple variant) could actually achieve this. At the very least, can we get such a guarantee for online b-matching with different budgets? Note that when all the budgets are the same then our algorithm for the min-max version is equivalent to a simple algorithm called BALANCE that achieves this. (This observation follows from the results in Kalyanasundaram and Pruhs [KP00] and Motwani, Panigrahy and Xu [MP06].)
- A high level goal is to come up with other reasonable definitions that go beyond worst case. The motivation for this is to bridge the gap between the theory and practice of online algorithms.

References

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, 2005.

[AMT07] Zoe Abrams, Ofer Mendelevitch, and John Tomlin. Optimal delivery of sponsored search advertisements subject to budget constraints. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 272–278, New York, NY, USA, 2007. ACM.

[AWY09] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. arXiv:0911.2974v1, 2009.

[BJN07] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA'07: Proceedings of the 15th annual European conference on Algorithms*, pages 253–264, Berlin, Heidelberg, 2007. Springer-Verlag.

[BK10] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *ESA*, pages 170–181, 2010.

[CCD$^+$10] Denis Charles, Max Chickering, Nikhil R. Devanur, Kamal Jain, and Manan Sanghi. 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*, pages 121–128, New York, NY, USA, 2010. ACM.

[CEDG$^+$10] Tanmoy Chakraborty, Eyal Even-Dar, Sudipto Guha, Yishay Mansour, and S. Muthukrishnan. Selective call out and real time bidding. In *WINE*, 2010.

[DH09] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In John Chuang, Lance Fortnow, and Pearl Pu, editors, *ACM Conference on Electronic Commerce*, pages 71–78. ACM, 2009.

[FHK+10] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *ESA*, pages 182–194, 2010.

[Fle00] Lisa K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discret. Math.*, 13(4):505–520, 2000.

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126, Washington, DC, USA, 2009. IEEE Computer Society.

[GK98] Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 300, Washington, DC, USA, 1998. IEEE Computer Society.

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, 2011.

[KP00] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

[KPP96] Anil Kamath, Omri Palmon, and Serge Plotkin. Routing and admission control in general topology networks with poisson arrivals. In *SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 269–278, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[MGS11] Vahideh Manshadi, Shayan Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *To appear in SODA*, 2011.

[MP06] Rajeev Motwani, Rina Panigrahy, and Ying Xu 0002. Fractional matching via balls-and-bins. In *APPROX-RANDOM*, pages 487–498, 2006.

[MSVV05] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized on-line matching. In *In FOCS 05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273. IEEE Computer Society, 2005.

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

[PST91] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *SFCS '91: Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 495–504, Washington, DC, USA, 1991. IEEE Computer Society.

[VVS10] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *EC '10: Proceedings of the 11th ACM conference on Electronic commerce*, pages 109–118, New York, NY, USA, 2010. ACM.

[You95] Neal E. Young. Randomized rounding without solving the linear program. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 170–178, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.