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

NIKHIL R. DEVANUR, Microsoft Research, USA
KAMAL JAIN, Faira, USA
BALASUBRAMANIAN SIVAN, Google Research, USA
CHRISTOPHER A. WILKENS, Facebook Research, USA

We present prior robust algorithms for a large class of resource allocation problems where requests arrive one-by-one (online), drawn independently from an *unknown* distribution at every step. We design a single algorithm that, for every possible underlying distribution, obtains a $1-\epsilon$ fraction of the profit obtained by an algorithm that knows the entire request sequence ahead of time. The factor ϵ approaches 0 when no single request consumes/contributes a significant fraction of the global consumption/contribution by all requests together. We show that the tradeoff we obtain here that determines how fast ϵ approaches 0, is near optimal: We give a nearly matching lower bound showing that the tradeoff cannot be improved much beyond what we obtain.

Going beyond the model of a static underlying distribution, we introduce the *adversarial stochastic input* model, where an adversary, possibly in an adaptive manner, controls the distributions from which the requests are drawn at each step. Placing no restriction on the adversary, we design an algorithm that obtains a $1-\epsilon$ fraction of the optimal profit obtainable w.r.t. the worst distribution in the adversarial sequence. Further, if the algorithm is given one number per distribution, namely the optimal profit possible for each of the adversary's distribution, then we design an algorithm that achieves a $1-\epsilon$ fraction of the weighted average of the optimal profit of each distribution the adversary picks.

In the offline setting we give a fast algorithm to solve very large linear programs (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 quantifies how significant a single request is when compared to the sum total of all requests.

We discuss implications of our results to several special cases including online combinatorial auctions, network routing, and the adwords problem.

CCS Concepts: • **Theory of computation → Design and analysis of algorithms**; **Online algorithms**; *Adversary models*;

Additional Key Words and Phrases: Online algorithms, unknown distribution, approximation algorithms, greedy algorithm

Part of this work was done when the second author was a researcher at Microsoft Research, Redmond, and the third and fourth authors were interns at Microsoft Research, Redmond.

Authors' addresses: N. R. Devanur, Microsoft Research, 1 Microsoft Way, Redmond, WA 98052; email: nikdev@microsoft.com; K. Jain, Faira, 608 State St S Kirkland, WA 98033; email: kamaljain@gmail.com; B. Sivan, Google Research, 111 8th Ave, New York, NY 10011; email: balusivan@google.com; C. A. Wilkens, Facebook Research, 1 Hacker Way, Menlo Park, CA 94025; email: cwilkens@berkeley.edu.

0004-5411/2019/01-ART7 $15.00
https://doi.org/10.1145/3284177

ACM Reference format:
Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. 2019. Near Optimal Online Algorithms and Fast Approximation Algorithms for Resource Allocation Problems. *J. ACM* 66, 1, Article 7 (January 2019), 41 pages.
https://doi.org/10.1145/3284177

1 INTRODUCTION AND SUMMARY OF RESULTS

There has been an increasing interest in online algorithms for resource allocation problems motivated by their wide variety of applications in Internet advertising, allocating multi-leg flight seats for customers online, allocating goods to customers arriving online in a combinatorial auction, and so on. Designing efficient resource allocation algorithms has significant scientific and commercial value. The traditional computer science approach to deal with uncertain future inputs has been the worst-case competitive analysis. Here nothing is assumed about the sequence of requests that arrive online, and the benchmark is the optimal algorithm that knows the entire sequence of requests ahead of time. Several problems in this space have been analyzed in the traditional framework, exemplified, for instance, in the well-studied *Adwords* problem introduced by Mehta et al. [19]. While worst-case analysis is a robust framework, for many problems it leads to pessimistic bounds that rule out obtaining more than a constant fraction of the optimal profit. Consequently, there has been a drive in the last few years to go beyond worst-case analysis. A frequently used alternative is to perform stochastic analysis: Assume that the input is drawn from a *known* distribution and optimize the objective w.r.t. this distribution. While stochastic analysis circumvents the impossibility results in worst-case analysis, any error in the knowledge of distribution could render the algorithm suboptimal and sometimes even infeasible.

In this article, we study a middle ground between worst-case and stochastic settings. We assume that the input is drawn from an underlying distribution that is *unknown* to the algorithm designer. We present a single algorithm where for every distribution performs nearly as well as the optimal algorithm that knows the entire sequence of requests ahead of time. In this sense, the algorithm is *prior robust*.

We now give an informal description of the resource allocation framework and our main contributions. See Section 2 for a formal description and theorem statements. We consider a resource allocation setting where requests arrive online; every request can be served by some subset of several available options; each (request, option) pair consumes some amount of every resource, and generates some profit. There is a given budget for each resource. Requests are drawn i.i.d. from an *unknown* distribution. The goal is to maximize the total profit generated while making sure that the total consumption of each resource is no more than the corresponding budget. We compare the profit of the algorithms against the offline optimum and prove competitive ratio bounds. Even for very restricted special cases of this problem, the worst-case setting cannot yield anything beyond a $1 - \frac{1}{e}$ competitive ratio [15, 19]. While the stochastic setting with a fully known distribution can give near optimal performance guarantees, it often leads to very distribution dependent algorithms (e.g., see Reference [3] for the special case of the Adwords problem, which requires knowledge of the entire distribution to perform the optimization). Hence both these approaches are not satisfactory, and this problem lends itself well to the middle ground of prior robust analysis.

Going beyond i.i.d., our work introduces the *adversarial stochastic input* (ASI) model as a more realistic model for analyzing online algorithmic problems. Here the distribution from which the requests arrive is allowed to change over time (unlike i.i.d., where it stays identical for every request). The adversary decides how to pick the distributions and is even allowed to pick them adaptively. For many practical applications such as in display advertising, the distribution of requests shows trends that change over the course of time: mornings are different from evenings and weekdays are different from weekends. Thus a time varying distributional model is more realistic than the i.i.d. model. A keen reader might notice that the above description includes the worst-case setting as well, and therefore we have to make some extra assumptions, either by restricting how these distributions can be picked or by allowing the algorithm some extra information about the distributions. We will describe these in greater detail later.

Apart from the theoretical contribution, *the algorithms we design for the ASI models were used to completely overhaul the display advertising management system at Microsoft*, leading to a significant improvement in revenue ($\approx$10%) and better system manageability and enabling new capabilities.[1] We believe that our results make a significant contribution to the search for "allows-positive-results-yet-realistic" models for online algorithms.

First Result: Near-Optimal Prior Robust Online Algorithms for Resource Allocation Problems. A key parameter on which algorithms for several resource allocation problems depend on is the relative significance of any single request when compared to the entire sequence of requests. For instance, for the special case of the Adwords problem, this is the ratio of a single bid to an advertiser's budget. For the Adwords problem, Mehta et al. [19] and Buchbinder et al. [5] design an algorithm that achieves a worst-case competitive ratio that tends to $1 - 1/e$ as the bid to budget ratio (which we denote by γ) tends to 0.[2] Devanur and Hayes [7] studied the same problem in the random permutation model and showed that 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 tradeoff between γ and the competitive ratio. Devanur and Hayes [7] 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 et al. [2] improved the bound on γ to $O(\frac{\epsilon^2}{n\log(m/\epsilon)})$. The articles of Feldman et al. [10] and Agrawal et al. [2] have also shown that the technique of Devanur and Hayes [7] can be extended to other online problems.

The first main result in this article is the following threefold improvement of previous results (Theorems 2.2 and 2.3), for the i.i.d. with unknown distributions model. All our results apply to the general class of 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 including online network routing and online combinatorial auctions is presented in Section 7.

(1) We give an algorithm that guarantees a $1 - \epsilon$ approximation factor when $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$.
(2) We show that our bound on γ is *almost optimal*; we show that no algorithm, even if it knew the distribution, can guarantee a $1 - \epsilon$ approximation when $\gamma = \omega(\frac{\epsilon^2}{\log(n)})$.
(3) Our algorithms lend themselves to natural generalizations that provide identical guarantees in the more general ASI model that was described earlier. We provide three different versions of the ASI model in Section 3.5.

Significance.

(1) Regarding the bound on γ, we remove a factor of n from γ, making the algorithm more practical. Consider for instance the Adwords problem and suppose that the bids are all

[1]This system had been globally operational from 2011 to 2015, when Microsoft made a deal with AOL to allow AOL to sell the display advertisement on behalf of Microsoft.
[2]Note that γ approaching zero is the easiest case. Even with γ approaching zero, $1 - 1/e$ is the best competitive ratio that any randomized algorithm can achieve in the worst case, illustrating how worst-case analysis leads to pessimistic bounds.

in [0,1]. The earlier bound implies that the advertiser budgets need to be of the order of $n \log n/\epsilon^2$ 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 article, we only need the advertiser 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 more general resource allocation framework, the previous best upper bound on γ is from Agrawal, Wang, and Ye [2] 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.

(2) Our ASI models are realistic models of time varying distributions for which we present algorithms with asymptotically optimal performance guarantees. We consider three different benchmarks, each progressively stronger than the previous, and require different levels of information about the distributions to achieve near optimal performance guarantees. For the weakest benchmark, we need just one parameter from the distribution, while for the strongest benchmark, we still need only $2mn$ parameters. Note that the distributions themselves can have an arbitrarily large support size,[3] and hence the amount of information we need is much smaller than the description of all the distributions. Our results for the ASI model can be thought of as generalizations of the "Prophet Inequality."[4] Finally, as mentioned earlier, our algorithms for this model have made a significant impact on the practice of display advertising management at Microsoft.

Second Result: Prior Robust $1-1/e$ *Approximation Greedy Algorithm for Adwords.* A natural algorithm for the Adwords problem that is widely used for its simplicity is the greedy algorithm: Always match an incoming query to the advertiser that has the maximum effective bid (the minimum of bid and remaining budget) for that query. Because of its wide use, previously the performance of the greedy algorithm has been analyzed by Goel and Mehta [13], 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 that is essentially that γ tends to 0.

It has been an important open problem to analyze the performance of greedy algorithm in a stochastic setting for unbounded γ, i.e., for all $0 \leq \gamma \leq 1$. The best factor known so far is 1/2, and this works for the worst case also. Nothing better was known, even in the stochastic models. *The second result in this article is that for the Adwords problem in the i.i.d. unknown distributions model, with no assumption on γ (i.e., γ could be as big as 1), the greedy algorithm gets an approximation factor of* $1-1/e$ *against the optimal fractional solution to the expected instance (Theorem 2.4).*

Our proof technique for this result has been subsequently used to prove a similar result for the greedy algorithm in online submodular welfare maximization [17]. We note here that there are other algorithms that achieve a $1-1/e$ approximation for the Adwords problem with unbounded γ, but the greedy algorithm is the only prior robust (i.e., distribution independent) algorithm known, and it is quite simple, too. For example Alaei, Hajiaghayi, and Liaghat [3] design a randomized algorithm that obtains a $1-1/e$ approximation but requires the knowledge of the entire distribution. Devanur, Sivan, and Azar [8] design a deterministic algorithm that obtains a $1-1/e$ approximation but requires a few parameters from the distribution.

[3]We even allow continuous distributions that have an infinite support.

[4]The Prophet Inequality is essentially a 1/2-competitive algorithm for the following problem: A sequence of values is drawn independently from different distributions and presented one at a time. The algorithm may pick at most one of these values in an online manner, given some knowledge of the distributions, such as the expectation of the maximum of these values. The goal is to maximize the value picked. See Reference [21].

Third Result: Fast Approximation Algorithms for Mixed Packing and Covering Integer Programs. Charles et al. [6] 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 $M : L \rightarrow R$ 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 Charles et al. [6] 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 article is a generalization of this result to get fast approximation algorithms for a wide class of mixed packing and covering integer programs (IPs) inspired by problems in the resource allocation framework (Theorem 2.5). Problems in the resource allocation framework where the instances are too large to use traditional algorithms occur fairly often, in particular in the management of display advertising systems, where these algorithms are being used. Formal statements and a more detailed discussion are presented in Section 2.4.

High-level Description of Techniques. The underlying idea used for all these results can be summarized at a high level thusly: Consider a hypothetical algorithm called *Hypothetical-Oblivious* 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 Hypothetical-Oblivious 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 Hypothetical-Oblivious, and hence we obtain the same guarantee as that for Hypothetical-Oblivious. The choice of potential function varies across the results; also, whether we minimize or maximize the the potential function varies across the results.

For instance, in our first result (Theorem 2.2), the performance of Hypothetical-Oblivious 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 to which we apply the Chernoff bounds. 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; e.g., the proof of the second result mentioned above (that greedy is $1 - 1/e$ competitive, Theorem 2.4) 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.

Multiplicative-Weight Updates. Our techniques and the resulting algorithms for our first and third results (Theorem 2.2 and Theorem 2.5) are similar to the algorithms of Young [22, 23] for derandomizing randomized rounding and the fast approximation algorithms for solving covering/packing linear programs (LPs) of Plotkin, Shmoys, and Tardos [20], Garg and Koenemann [12], and Fleischer [11]. In fact, Arora et al. [4] 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 that uses Chernoff bounds is also a multiplicative update algorithm. Our algorithm is closer in spirit to Young's algorithms than others. The main difference is that our algorithm solves an online problem, rather than an offline one, and hence will run short of essential distribution dependent parameters to run the multiplicative weights-based algorithm directly: We show that these parameters can be estimated near optimally. And, further, we introduce more adversarial models of online input, namely the varying ASI models, and come up with varying levels of knowledge of the distribution that are sufficient to be able to design good algorithms for these models. And for the offline case, a basic difference of our algorithm from this previous set of results is that our algorithm uses the special structure of the polytope $\sum_k x_{j,k} \leq 1$ (as against the more general polytopes in these works) in giving a more efficient solution. For instance, for our offline problem the number of oracle calls required will have a quadratic dependence on γm if we used the Plotkin et al. [20] algorithm, whereas using the special structure of the polytope, we obtain a linear dependence on γm.

It is possible that our algorithm can also be interpreted as an algorithm for the experts problem. In fact, Mehta et al. [19] 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, but we do not provide formal connections to the experts framework. This was done in follow-up works [1, 14] that showed that essentially the same algorithm as ours can be thought of as using a subroutine of Multiplicative-weight updates on a suitably defined learning with experts problem.

Follow-up Work. There has been a number of follow-up articles since the conference version of this article has been published. Alaei et al. [3] show that for the Adwords problem with a *known* distribution, it is enough for γ to be $O(\epsilon^2)$ to get a $1 - \epsilon$ approximation. Simultaneously, Devanur et al. [8] showed the same dependence of $\gamma = O(\epsilon^2)$ for the Adwords problem but requiring only a few parameters from the distribution. Kapralov et al. [17] study a generalized version of the adwords problem where an advertiser's profit, instead of being budget additive, could be an arbitrary submodular function of the queries assigned to him. For this problem in the worst-case setting, they show that no algorithm can obtain better than a $\frac{1}{2}$ approximation, which the greedy algorithm already achieves. For the same problem in the i.i.d. setting, they show, using techniques we develop in this work, that the greedy algorithm obtains a $1 - \frac{1}{e}$ approximation. Kesselheim et al. [18] gave similar guarantees as us, for the random permutation model (i.i.d. without replacement), and also get the improved bound of $\gamma = O(\epsilon^2)$ for the special case of the Adwords problem. However, the algorithms in Reference [18] are computationally expensive, requiring us to solve a linear program for serving every single request, whereas our algorithm performs a much simpler optimization in every step: For the adwords problem, for instance, it takes only a linear time to perform each step's optimization. Both Agrawal and Devanur [1] and Gupta and Molinaro [14] showed that essentially the same algorithm as ours also works for the random permutation model, with the same guarantees, while also relating it formally to the learning from experts framework. Agrawal and Devanur [1] also greatly generalize the resource allocation framework to handle arbitrary concave objectives and convex constraints. Eghbali et al. [9] interpret our algorithm as an exponentiated sub-gradient algorithm, show that it works for the random permutation model, and give a slight generalization to handle additively separable concave reward functions.

2 PRELIMINARIES AND MAIN RESULTS

2.1 Resource Allocation Framework

We consider the following framework of optimization problems. There are n resources, with resource $i \in \mathcal{A}$ having a capacity of c_i. There are m requests; each request $j \in \mathcal{J}$ can be satisfied by a vector $\mathbf{x}_j \in \{0,1\}^K$, with coordinates $x_{j,k}$, such that $\sum_k x_{j,k} \leq 1$. Think of vector $\mathbf{x}_j$ as picking a single option to satisfy a request from a total of K options. We use $\mathcal{K}$ to denote the set of options. The vector $\mathbf{x}_j$ consumes $\mathbf{a}_{i,j} \cdot \mathbf{x}_j$ amount of resource i and gives $\mathbf{w}_{i,j} \cdot \mathbf{x}_j$ amount of type i profit.[5] The $\mathbf{a}_{i,j}$'s and $\mathbf{w}_{i,j}$'s are non-negative vectors of length K (and so are the $\mathbf{x}_j$'s). The co-ordinates of the vectors $\mathbf{a}_{i,j}$ and $\mathbf{w}_{i,j}$ will be denoted by a_{ijk} and w_{ijk}, respectively, i.e., the kth option consumes a_{ijk} amount of resource i and gives a type i profit of w_{ijk}. The objective is to maximize the minimum among all types of profit subject to the capacity constraints on the resources. The following is the linear program relaxation of the resource allocation problem:

$$
\begin{aligned}
&\text{Maximize } \min_{i \in \mathcal{A}} \sum_{j \in \mathcal{J}} \mathbf{w}_{i,j} \cdot \mathbf{x}_j \text{ s.t.} \\
&\forall\, i \in \mathcal{A}, \sum_{j \in \mathcal{J}} \mathbf{a}_{i,j} \cdot \mathbf{x}_j \leq c_i \\
&\forall\, j \in \mathcal{J}, \sum_{k \in \mathcal{K}} x_{j,k} \leq 1 \\
&\forall\, j \in \mathcal{J}, k \in \mathcal{K}, x_{j,k} \geq 0.
\end{aligned}
$$

Note that dropping a request by not picking any option at all is feasible, too. For expositional convenience, we will denote not picking any option at all as having picked the $\perp$ option ($\perp$ may not be in the set $\mathcal{K}$) for which $a_{ij\perp} = 0$ and $w_{ij\perp} = 0$ for all i, j.

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 an offline problem 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 Near-Optimal Online Algorithm for Resource Allocation

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. To define our benchmark, we now define the expected instance.

Expected Instance. Consider the following *expected instance* of the problem, where everything happens as per expectation. It is a single offline instance that is a function of the given distribution over requests and the total number of requests m. Every request in the support of the distribution is also a request in this instance. The capacities of the resources in this instance are the same as in the original instance. Suppose request j has a probability p_j of arriving in the given distribution. The resource consumption of j in the expected instance is given by $mp_j\mathbf{a}_{i,j}$ for all i and the type i profit is $mp_j\mathbf{w}_{i,j}$. 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. To summarize, the *LP relaxations* of a random instance with set of requests R, and the expected instance E are as follows (slightly rewritten for convenience):

[5]While this notation seems to imply that the number of resource types is equal to the number/set of profit types, namely n, this choice was made purely to reduce clutter in notation. In general the number/set of resource types could be different from that of the number of profit types, and it is straightforward to verify that our proofs go through for the general case.

LP relaxations for random and expected instances (1)

Random Instance R	**Expected Instance E**
Maximize λ s.t.	Maximize λ s.t.
$\forall\ i \in \mathcal{A}, \sum_{j\in R, k\in\mathcal{K}} w_{ijk}x_{j,k} \ge \lambda$	$\forall\ i \in \mathcal{A}, \sum_{j\in\mathcal{J}, k\in\mathcal{K}} mp_j w_{ijk}x_{j,k} \ge \lambda$
$\forall\ i \in \mathcal{A}, \sum_{j\in R, k\in\mathcal{K}} a_{ijk}x_{j,k} \le c_i$	$\forall\ i \in \mathcal{A}, \sum_{j\in\mathcal{J}, k\in\mathcal{K}} mp_j a_{ijk}x_{j,k} \le c_i$
$\forall\ j \in R, \sum_{k\in\mathcal{K}} x_{j,k} \le 1$	$\forall\ j \in \mathcal{J}, \sum_{k\in\mathcal{K}} x_{j,k} \le 1$
$\forall\ j \in R, k \in \mathcal{K}, x_{j,k} \ge 0.$	$\forall\ j \in \mathcal{J}, k \in \mathcal{K}, x_{j,k} \ge 0.$

We now prove that the fractional optimal solution to the expected instance W_E is an upper bound on the expectation of W_R, where W_R is the offline fractional optimum of the actual sequence of requests in a random instance R.

Lemma 2.1. $W_E \ge \mathbf{E}[W_R]$.

Proof. The average of optimal solutions for all possible sequences of requests is a feasible solution to the expected instance with a profit equal to $\mathbf{E}[W_R]$. Thus the optimal profit for the expected instance could only be larger. □

The approximation factor 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 W_E for the expected instance. Let $\gamma = \max(\{\frac{a_{ijk}}{c_i}\}_{i,j,k} \cup \{\frac{w_{ijk}}{W_E}\}_{i,j,k})$ be the parameter capturing the significance of any one request when compared to the total set of requests that arrive online. The main result is that as γ tends to zero, the approximation factor ratio tends to 1. In fact, we give the almost optimal tradeoff.

Theorem 2.2. *For any $\epsilon \ge 1/m$, Algorithm 2 achieves an objective value of $W_E(1 - O(\epsilon))$ for the online resource allocation problem with probability at least $1 - \epsilon$, assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$. Algorithm 2 does not require any knowledge of the distribution at all.*

Theorem 2.3. *There exist instances with $\gamma = \frac{\epsilon^2}{\log(n)}$ such that no algorithm, even with complete knowledge of the distribution, can get a $1 - o(\epsilon)$ approximation factor.*

Oracle Assumption. 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 $\mathbf{x}_j$s in $\{0,1\}^K$ satisfying $\sum_{k\in\mathcal{K}} x_{j,k} \le 1$. This assumption boils down to being able to find the maximum among K numbers, but K may be exponential in some cases. For the Adwords and display ads problem (described below), K is actually equal to n, and this is trivial. For network routing (described in Section 7), K could be exponential in the size of the network, and this assumption corresponds to being able to find the shortest path in a graph in polynomial time. For combinatorial auctions (described in Section 7), this corresponds to the demand query assumption: Given prices on various items, the buyer should be able to decide in polynomial time which bundle gives her the maximum utility. (While this is not always achievable in polynomial time, there cannot be any hope of a posted pricing solution for combinatorial auction without this minimum assumption.)

Extensions and Special Cases. The extensions of Theorem 2.2 to the various generalizations of the i.i.d. model, including the adversarial stochastic input model, are presented in Section 3.5. 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 two special cases: the Adwords problem and the display ads problem.

(1) **Adwords.** In the adwords problem, there are n advertisers with a daily budget of B_i for advertiser i. There are m keywords/queries that arrive online, and advertiser i has a bid of b_{ij} for query j. This is a special case of the resource allocation framework where the set of options $\mathcal{K}$ matches the set of resources/advertisers $\mathcal{A}$, i.e., each query can be given to at most one advertiser and will consume budget just from that advertiser. Let x_{ij} denote the indicator variable for whether or not query j was allocated to agent i. After all allocation is over, agent i pays $\min(\sum_{j \in \mathcal{J}} b_{ij} x_{ij}, B_i)$, i.e., the minimum of the sum of the bids for queries allocated to i and his budget B_i. The objective is to maximize the sum of the payments from all advertisers—this is again a special case of the resource allocation framework where this only a single profit type, and we just want to maximize it. One could raise a technical objection that this is not a special case of the resource allocation framework, because the budget constraint is not binding: The value of the allocated bids to an advertiser can exceed his budget, although the total payment from the advertiser will be at most the budget. But it is not difficult to see that the LP relaxation of the offline problem can be written as in LP (2), which is clearly a special case of resource allocation framework LP. Note that the benchmark is anyway an upper bound even on the expected optimal fractional solution. Therefore, any algorithm that gets an α approximation factor for resource allocation is also guaranteed to get the same approximation factor for Adwords. The only notable thing being that an algorithm for resource allocation when used for adwords will treat the budget constraints as binding and obtain the guarantee promised in Theorem 2.2. (In our $1 - 1/e$ approximation algorithm for adwords in Section 5 that holds for all values of γ (≤ 1 of course), we use this facility to exceed budget.)

(2) **Display Ads.** In the display ads problem, there are n advertisers and m impressions arrive online. Advertiser i has wants c_i impressions in total and pays v_{ij} for impression j and will get paid a penalty of ρ_i for every undelivered impression. If over-delivered, then he will pay his bid for the first c_i impressions delivered. Letting $b_{ij} = v_{ij} + \rho_i$, we can write the LP relaxation of the offline display ads problem as in LP (2), which is clearly a special case of the resource allocation LP, where just like the Adwords special case the set of options $\mathcal{K}$ is equal to the set of resources/advertisers $\mathcal{A}$, and there is only a single profit type.

LP relaxations for Adwords and Display Ads (2)

<u>**Adwords**</u>	<u>**Display Ads**</u>
Maximize $\sum_{i \in \mathcal{A}, j \in \mathcal{J}} b_{ij} x_{ij}$ s.t.	Maximize $\sum_{i \in \mathcal{A}, j \in \mathcal{J}} b_{ij} x_{ij}$ s.t.
$\forall\, i \in \mathcal{A}, \sum_{j \in \mathcal{J}} b_{ij} x_{ij} \leq B_i$	$\forall\, i \in \mathcal{A}, \sum_{j \in \mathcal{J}} x_{ij} \leq c_i$
$\forall\, j \in \mathcal{J}, \sum_{i \in \mathcal{A}} x_{ij} \leq 1$	$\forall\, j \in \mathcal{J}, \sum_{i \in \mathcal{A}} x_{ij} \leq 1$
$\forall\, i \in \mathcal{A}, j \in \mathcal{J}, x_{ij} \geq 0.$	$\forall\, i \in \mathcal{A}, j \in \mathcal{J}, x_{ij} \geq 0.$

2.3 Greedy Algorithm for Adwords

As noted in the Introduction, the greedy algorithm is widely implemented due to its simplicity, but its performance was known to be only a $1/2$ approximation even in stochastic models. We show that the greedy algorithm obtains a $1 - 1/e$ approximation for all γ, i.e., $0 \leq \gamma \leq 1$.

Theorem 2.4. *The greedy algorithm achieves an approximation factor of* $1 - 1/e$ *for the Adwords problem in the i.i.d. unknown distributions model for all* γ, *i.e.,* $0 \le \gamma \le 1$.

We note here that the competitive ratio of $1 - 1/e$ is tight for the greedy algorithm [13]. It is, however, not known to be tight for an arbitrary algorithm.

2.4 Fast Approximation Algorithms for Large Mixed Packing and Covering Integer Programs

Charles et al. [6] 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 $i \in R$, $|M^{-1}(i)| \ge B_i$ for some given values B_i. Since m is very large classic matching algorithms are not useful. Charles et al. [6] gave an algorithm that runs in time linear[6] in the number of edges of an induced subgraph obtained by taking a random sample from L of size $O(\frac{m \log n}{\min_i \{B_i\} \epsilon^2})$, for a gap-version of the problem with gap ϵ. Such an algorithm is very useful in a variety of applications involving ad assignment for online advertising, particularly when $\min_i \{B_i\}$ is large.

We consider a generalization of the above problem inspired by the resource allocation framework. In fact, we consider the following mixed covering-packing integer program. Suppose that there are n packing constraints, one for each $i \in [n]$ of the form $\sum_{j=1}^{m} \mathbf{a}_{i,j} \cdot \mathbf{x}_j \le c_i$ and n covering constraints, one for each $i \in [n]$ of the form $\sum_{j=1}^{m} \mathbf{w}_{i,j} \cdot \mathbf{x}_j \ge d_i$. Each $\mathbf{x}_j$ (with coordinates $x_{j,k}$) is constrained to be in $\{0, 1\}^K$ and satisfy $\sum_k x_{j,k} \le 1$. The $\mathbf{a}_{i,j}$'s and $\mathbf{w}_{i,j}$'s (and hence $\mathbf{x}_j$'s) are non-negative vectors of length K with coordinates a_{ijk} and w_{ijk}. Does there exist 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 the c_i's are multiplied by $1 + \epsilon$ and all the d_i's are 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,

Let $\gamma = \max(\{\frac{a_{ijk}}{c_i}\}_{i,j,k} \cup \{\frac{w_{ijk}}{d_i}\}_{i,j,k})$.

Theorem 2.5. *For any* $\epsilon > 0$, *assuming* $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$, *Algorithm 5 solves the gap version of the mixed covering-packing integer program with* $\Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ *oracle calls.*

2.5 Chernoff Bounds

We present here the form of Chernoff bounds that we use throughout the rest of this article. 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 $\epsilon > 0$,

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

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

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

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

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

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

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

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

For $\epsilon > 2e-1$,

$$\mathbf{Pr}[X > \mu(1+\epsilon)] < 2^{-(1+\epsilon)\mu/B}.$$

3 NEAR-OPTIMAL PRIOR ROBUST ONLINE ALGORITHMS FOR RESOURCE ALLOCATION

For convenience, we begin by rewriting the LP relaxation of a random instance R of the online resource allocation problem and the expected instance (already defined in Section 2.2 as LP (1)).

LPs for random and expected instances (3)

<u>**Random Instance** R</u>	<u>**Expected Instance** E</u>
Maximize λ s.t.	Maximize λ s.t.
$\forall\, i \in \mathcal{A}, \sum_{j\in R, k\in\mathcal{K}} w_{ijk}x_{j,k} \ge \lambda$	$\forall\, i \in \mathcal{A}, \sum_{j\in\mathcal{J}, k\in\mathcal{K}} mp_j w_{ijk}x_{j,k} \ge \lambda$
$\forall\, i \in \mathcal{A}, \sum_{j\in R, k\in\mathcal{K}} a_{ijk}x_{j,k} \le c_i$	$\forall\, i \in \mathcal{A}, \sum_{j\in\mathcal{J}, k\in\mathcal{K}} mp_j a_{ijk}x_{j,k} \le c_i$
$\forall\, j \in R, \sum_{k\in\mathcal{K}} x_{j,k} \le 1$	$\forall\, j \in \mathcal{J}, \sum_{k\in\mathcal{K}} x_{j,k} \le 1$
$\forall\, j \in R, k \in \mathcal{K}, x_{j,k} \ge 0.$	$\forall\, j \in \mathcal{J}, k \in \mathcal{K}, x_{j,k} \ge 0.$

We showed in Lemma 2.1 that $W_E \ge \mathbf{E}[W_R]$. All our approximation guarantees are w.r.t. the stronger benchmark of W_E, which is the optimal fractional solution of the expected instance. We would like to remind the reader that while the benchmark is allowed to be fractional, the online algorithm of course is allowed to find only integral solutions.

We divide the rest of this section into four subsections. The subsections progressively weaken the assumptions on knowledge of the distribution of the input.

(1) In Section 3.1, we develop a hypothetical algorithm called Hypothetical-Oblivious-Conservative, denoted by $\widetilde{P}$, that achieves an objective value of $W_E(1-2\epsilon)$ w.p. at least $1-\epsilon$ assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$. Theorem 3.1 is the main result of this section. The algorithm is hypothetical, because it assumes knowledge of the entire distribution, whereas the goal of this article is to develop algorithms that work without distributional knowledge.

(2) In Section 3.2, we design an algorithm for the online resource allocation problem that achieves the same guarantee as the Hypothetical-Oblivious-Conservative algorithm $\widetilde{P}$, without any knowledge of the distribution except for a single parameter of the distribution—the value of W_E. Theorem 3.2 is the main result of this section.

(3) In Section 3.3 we design an algorithm for the online resource allocation problem that achieves an objective value of at least $W_E(1-O(\epsilon))$ w.p. at least $1-\epsilon$ assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$ without any knowledge at all about the distribution. The algorithm in

Section 3.2 serves as a good warm-up for the algorithm in this section. Theorem 2.2 is the main result of this section.

(4) In Section 3.5, we relax the assumption that the distribution from which the requests are drawn is i.i.d.; we give three different generalizations of the i.i.d. model with strong revenue guarantees as in the i.i.d. model.

3.1 Completely Known Distributions

When the distributions are completely known, we first compute the expected instance and solve its LP relaxation (LP (3)) optimally. Let x^*_{jk} denote the optimal solution to the expected LP (3). The Hypothetical-Oblivious algorithm P works as follows: When request j arrives, it serves it using option k with probability x^*_{jk}. Let $X^*_{i,t}$ denote the amount of resource i consumed in step t for the algorithm P. Thus the total amount of resource i consumed over the entire m steps of algorithm P is $\sum_{t=1}^m X^*_{i,t}$. Note that $\mathbf{E}[X^*_{i,t}] = \sum_{j,k} p_j a_{ijk} x^*_{jk} \le \frac{c_i}{m}$. Thus, we can bound the probability that $\mathbf{Pr}[\sum_{t=1}^m X^*_{i,t} \ge c_i(1+\epsilon)]$ using Chernoff bounds. We explicitly derive this bound here, since we use this derivation in designing the algorithm in Section 3.2.

Since we cannot exceed c_i amount of resource consumption by any non-zero amount, we need to be more conservative than P. So we analyze the following algorithm $\widetilde{P}$, called Hypothetical-Oblivious-Conservative, instead of P: When request j arrives, it serves it using option k with probability $\frac{x^*_{jk}}{1+\epsilon}$, where ϵ is an error parameter of algorithm designer's choice. Let $\widetilde{X}_{i,t}$ denote the amount of resource i consumed in step t for the algorithm $\widetilde{P}$. Note that $\mathbf{E}[\widetilde{X}_{i,t}] \le \frac{c_i}{(1+\epsilon)m}$. Thus, even with a $(1+\epsilon)$ deviation using Chernoff bounds, the resource consumption is at most c_i.

We begin by noting that $\widetilde{X}_{i,t} \le \gamma c_i$ by the definition of γ. For all $\epsilon \in [0,1]$, we have

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=1}^m \widetilde{X}_{i,t} \ge \frac{c_i}{1+\epsilon}(1+\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^m \widetilde{X}_{i,t}}{\gamma c_i} \ge \frac{1}{\gamma}\right] \\
&= \mathbf{Pr}\left[(1+\epsilon)^{\frac{\sum_{t=1}^m \widetilde{X}_{i,t}}{\gamma c_i}} \ge (1+\epsilon)^{\frac{1}{\gamma}}\right] \\
&\le \mathbf{E}\left[(1+\epsilon)^{\frac{\sum_{t=1}^m \widetilde{X}_{i,t}}{\gamma c_i}}\right] / (1+\epsilon)^{\frac{1}{\gamma}} \\
&= \mathbf{E}\left[\prod_{t=1}^m (1+\epsilon)^{\frac{\widetilde{X}_{i,t}}{\gamma c_i}}\right] / (1+\epsilon)^{\frac{1}{\gamma}} \\
&\le \mathbf{E}\left[\prod_{t=1}^m \left(1+\epsilon\frac{\widetilde{X}_{i,t}}{\gamma c_i}\right)\right] / (1+\epsilon)^{\frac{1}{\gamma}} \\
&\le \left[\prod_{t=1}^m \left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)\right] / (1+\epsilon)^{\frac{1}{\gamma}} \\
&\le \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}}\right)^{\frac{1}{\gamma(1+\epsilon)}} \\
&\le e^{\frac{-\epsilon^2}{4\gamma}\frac{1}{1+\epsilon}} \\
&\le \frac{\epsilon}{2n},
\end{aligned}$$

where the first inequality follows from Markov's inequality, the second from convexity of exponential function together with with the fact that $\widetilde{X}_{i,t} \leq \gamma c_i$, the third from $\mathbf{E}[\widetilde{X}_{i,t}] \leq \frac{c_i}{(1+\epsilon)m}$, and the fourth from $1 + x \leq e^x$; the fifth is standard for all $\epsilon \in [0,1]$, and the sixth follows from $\gamma = O(\epsilon^2 / \log(n/\epsilon))$ for an appropriate choice of constant inside the big O coupled with $n \geq 2$.

Remark 1. At first sight, this bound might seem anomalous—the bound $\frac{\epsilon}{2n}$ is increasing in ϵ, i.e., the probability of a smaller deviation is smaller than the probability of a larger deviation. The reason for this anomaly is that γ is related to ϵ as $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$, and the smaller the γ, the better revenue we can get (i.e., more granular requests leads to lesser wastage from errors, and hence more revenue). Thus a small deviation for small γ has a smaller probability than a larger deviation for a larger γ.

Similarly, let $\widetilde{Y}_{i,t}$ denote the revenue obtained from type i profit in step t for the algorithm $\widetilde{P}$. Note that $\mathbf{E}[\widetilde{Y}_{i,t}] = \sum_{j,k} p_j w_{ijk} \frac{x^*_{jk}}{1+\epsilon} \geq \frac{W_E}{(1+\epsilon)m}$. By the definition of γ, we have $\widetilde{Y}_{i,t} \leq \gamma W_E$. For all $\epsilon \in [0,1]$ we have

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=1}^{m} \widetilde{Y}_{i,t} \leq \frac{W_E}{1+\epsilon}(1-\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^{m} \widetilde{Y}_{i,t}}{\gamma W_E} \leq \frac{1-\epsilon}{\gamma(1+\epsilon)}\right] \\
&= \mathbf{Pr}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{m} \widetilde{Y}_{i,t}}{\gamma W_E}} \geq (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}\right] \\
&\leq \mathbf{E}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{m} \widetilde{Y}_{i,t}}{\gamma W_E}}\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}} \\
&= \mathbf{E}\left[\prod_{t=1}^{m} (1-\epsilon)^{\frac{\widetilde{Y}_{i,t}}{\gamma W_E}}\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}} \\
&\leq \mathbf{E}\left[\prod_{t=1}^{m} \left(1 - \epsilon \frac{\widetilde{Y}_{i,t}}{\gamma W_E}\right)\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}} \\
&\leq \left[\prod_{t=1}^{m} \left(1 - \frac{\epsilon}{(1+\epsilon)\gamma m}\right)\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}} \\
&\leq \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{1-\epsilon}}\right)^{\frac{1}{\gamma(1+\epsilon)}} \\
&\leq e^{\frac{-\epsilon^2}{2\gamma} \frac{1}{1+\epsilon}} \\
&\leq \frac{\epsilon}{2n}.
\end{aligned}$$

Thus, we have all the capacity constraints satisfied, (i.e., $\sum_i \widetilde{X}_{i,t} \leq c_i$), and all resource profits are at least $\frac{W_E}{1+\epsilon}(1-\epsilon)$ (i.e., $\sum_i \widetilde{Y}_{i,t} \geq \frac{W_E}{1+\epsilon}(1-\epsilon) \geq W_E(1-2\epsilon)$), with probability at least $1 - 2n \cdot \epsilon/2n = 1 - \epsilon$. This proves the following theorem:

THEOREM 3.1. *For any $\epsilon > 0$, the Hypothetical-Oblivious-Conservative algorithm $\widetilde{P}$ achieves an objective value of $W_E(1-2\epsilon)$ for the online resource allocation problem with probability at least $1-\epsilon$, assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$.*

3.2 Unknown Distribution, Known W_E

We now design an algorithm[7] A without knowledge of the distribution but just knowing a single parameter W_E. Let $A^s\widetilde{P}^{m-s}$ be a hybrid algorithm that runs A for the first s steps and $\widetilde{P}$ for the remaining $m-s$ steps. Let $\epsilon \in [0,1]$ be the error parameter, which is the algorithm designer's choice. Call the algorithm a failure if at least one of the following fails:

(1) For all i, $\sum_{t=1}^{m} X_{i,t}^A \leq c_i$.
(2) For all i, $\sum_{t=1}^{m} Y_{i,t}^A \geq W_E(1-2\epsilon)$.

For any algorithm A, let the amount of resource i consumed in the tth step be denoted by $X_{i,t}^A$ and the amount of resource i profit be denoted by $Y_{i,t}^A$. Let $S_s(X_i^A) = \sum_{t=1}^{s} X_{i,t}^A$ denote the amount of resource i consumed in the first s steps, and let $S_s(Y_i^A) = \sum_{t=1}^{s} Y_{i,t}^A$ denote the resource i profit in the first s steps. Similarly to the derivation in Section 3.1 that bounded the failure probability of $\widetilde{P}$, we can bound the failure probability of any algorithm A, i.e.,

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=1}^{m} X_{i,t}^A \geq \frac{c_i}{1+\epsilon}(1+\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^{m} X_{i,t}^A}{\gamma c_i} \geq \frac{1}{\gamma}\right] \\
&= \mathbf{Pr}\left[(1+\epsilon)^{\frac{\sum_{t=1}^{m} X_{i,t}^A}{\gamma c_i}} \geq (1+\epsilon)^{\frac{1}{\gamma}}\right] \\
&\leq \mathbf{E}\left[(1+\epsilon)^{\frac{\sum_{t=1}^{m} X_{i,t}^A}{\gamma c_i}}\right] / (1+\epsilon)^{\frac{1}{\gamma}} \\
&= \mathbf{E}\left[\prod_{t=1}^{m}(1+\epsilon)^{\frac{X_{i,t}^A}{\gamma c_i}}\right] / (1+\epsilon)^{\frac{1}{\gamma}} \qquad (4)
\end{aligned}$$

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=1}^{m} Y_{i,t}^A \leq \frac{W_E}{1+\epsilon}(1-\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^{m} Y_{i,t}^A}{\gamma W_E} \leq \frac{1-\epsilon}{\gamma(1+\epsilon)}\right] \\
&= \mathbf{Pr}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{m} Y_{i,t}^A}{\gamma W_E}} \geq (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}\right] \\
&\leq \mathbf{E}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{m} Y_{i,t}^A}{\gamma W_E}}\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}} \\
&= \mathbf{E}\left[\prod_{t=1}^{m}(1-\epsilon)^{\frac{Y_{i,t}^A}{\gamma W_E}}\right] / (1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}. \qquad (5)
\end{aligned}$$

In Section 3.1, our algorithm A was $\widetilde{P}$ (and therefore we can use $\mathbf{E}[\widetilde{X}_{i,t}] \leq \frac{c_i}{(1+\epsilon)m}$ and $\mathbf{E}[\widetilde{Y}_{i,t}] \geq \frac{W_E}{(1+\epsilon)m}$), the total failure probability that is the sum of Equations (4) and (5) for all the i's would have been $n \cdot [\frac{\epsilon}{2n} + \frac{\epsilon}{2n}] = \epsilon$. The goal is to design an algorithm A for stage r that, unlike P, does not know the distribution and knows just W_E but obtains the same ϵ failure probability. That is we

[7]Note that the notation $\mathcal{A}$ that we use for the set of advertisers/resources is different from the non-calligraphic A that we use for an algorithm. Also, it is immediate from the context which one we are referring to.

want to show that the sum of Equations (4) and (5) over all i's is at most ϵ:

$$\sum_i \frac{\mathbf{E}\left[\prod_{t=1}^m (1+\epsilon)^{\frac{X_{i,t}^A}{\gamma c_i}}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} + \sum_i \frac{\mathbf{E}\left[\prod_{t=1}^m (1-\epsilon)^{\frac{Y_{i,t}^A}{\gamma W_E}}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}} \leq \epsilon.$$

For the algorithm $A^s\widetilde{P}^{m-s}$, the above quantity can be rewritten as

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \prod_{t=s+1}^m (1+\epsilon)^{\frac{\widetilde{X}_{i,t}}{\gamma c_i}}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma W_E}} \prod_{t=s+1}^m (1-\epsilon)^{\frac{\widetilde{Y}_{i,t}}{\gamma W_E}}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}},$$

which, by using $(1+\epsilon)^x \leq 1+\epsilon x$ for $0 \leq x \leq 1$, is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \prod_{t=s+1}^m \left(1+\epsilon\frac{\widetilde{X}_{i,t}}{\gamma c_i}\right)\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma W_E}} \prod_{t=s+1}^m \left(1-\epsilon\frac{\widetilde{Y}_{i,t}}{\gamma W_E}\right)\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}.$$

Since for all t, the random variables $\widetilde{X}_{i,t}, X_{i,t}^A, \widetilde{Y}_{i,t}$, and $Y_{i,t}^A$ are all independent, and $\mathbf{E}[\widetilde{X}_{i,t}] \leq \frac{c_i}{(1+\epsilon)m}$ and $\mathbf{E}[\widetilde{Y}_{i,t}] \geq \frac{W_E}{(1+\epsilon)m}$, the above is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-s}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma W_E}} \left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-s}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}. \tag{6}$$

Let $\mathcal{F}[A^s\widetilde{P}^{m-s}]$ denote the quantity in Equation (6), which is an upper bound on failure probability of the hybrid algorithm $A^s\widetilde{P}^{m-s}$. By Theorem 3.1, we know that $\mathcal{F}[\widetilde{P}^m] \leq \epsilon$. We now prove that for all $s \in \{0, 1, \ldots, m-1\}$, $\mathcal{F}[A^{s+1}\widetilde{P}^{m-s-1}] \leq \mathcal{F}[A^s\widetilde{P}^{m-s}]$, thus proving that $\mathcal{F}[A^m] \leq \epsilon$, i.e., running the algorithm A for all the m steps results in a failure with probability at most ϵ. To design such an A, we closely follow the derivation of Chernoff bounds, which is what established that $\mathcal{F}[\widetilde{P}^m] \leq \epsilon$ in Theorem 3.1. However the design process will reveal that, unlike algorithm $\widetilde{P}$ that needs the entire distribution, just the knowledge of W_E will do for bounding the failure probability by ϵ.

Assuming that for all $s < p$, the algorithm A has been defined for the first $s+1$ steps in such a way that $\mathcal{F}[A^{s+1}\widetilde{P}^{m-s-1}] \leq \mathcal{F}[A^s\widetilde{P}^{m-s}]$, we now define A for the $p+1$th step in a way that will ensure that $\mathcal{F}[A^{p+1}\widetilde{P}^{m-p-1}] \leq \mathcal{F}[A^p\widetilde{P}^{m-p}]$. We have

$$\begin{aligned}\mathcal{F}[A^{p+1}\widetilde{P}^{m-p-1}] &= \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_{p+1}(X_i^A)}{\gamma c_i}} \left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_{p+1}(Y_i^A)}{\gamma W_E}} \left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}} \\ &\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}} \left(1+\epsilon\frac{X_{i,p+1}^A}{\gamma c_i}\right) \left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}}\end{aligned}$$

$$+\sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p\left(Y_i^A\right)}{\gamma W_E}}\left(1-\epsilon\frac{Y_{i,p+1}^A}{\gamma W_E}\right)\left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}. \tag{7}$$

Define

$$\phi_{i,s} = \frac{1}{c_i}\left[\frac{(1+\epsilon)^{\frac{S_s\left(X_i^A\right)}{\gamma c_i}}\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-s-1}}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$$
$$\psi_{i,s} = \frac{1}{W_E}\left[\frac{(1-\epsilon)^{\frac{S_s\left(Y_i^A\right)}{\gamma W_E}}\left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-s-1}}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right].$$

Define the step $p+1$ of algorithm A as picking the following option k^* for request j, where

$$k^* = \arg\min_k\left\{\sum_i a_{ijk}\cdot\phi_{i,p} - \sum_i w_{ijk}\cdot\psi_{i,p}\right\}. \tag{8}$$

For the sake of clarity, the entire algorithm is presented in Algorithm 1.

ALGORITHM 1: Algorithm for stochastic online resource allocation with unknown distribution, known W_E

Input: Capacities c_i for $i \in [n]$, the total number of requests m, the values of γ and W_E, an error parameter $\epsilon > 0$.

Output: An online allocation of resources to requests

```
1: Initialize $\phi_{i,0} = \frac{1}{c_i}\left[\frac{\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-1}}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$, and, $\psi_{i,0} = \frac{1}{W_E}\left[\frac{\left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-1}}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right]$
2: for $s = 1$ to $m$ do
3:     If the incoming request is $j$, then use the following option $k^*$:
           $k^* = \arg\min_{k\in\mathcal{K}\cup\{\perp\}}\left\{\sum_i a_{ijk}\cdot\phi_{i,s-1} - \sum_i w_{ijk}\cdot\psi_{i,s-1}\right\}.$
4:     $X_{i,s}^A = a_{ijk^*}$, $Y_{i,s}^A = w_{ijk^*}$
5:     Update $\phi_{i,s} = \phi_{i,s-1}\cdot\left[\frac{(1+\epsilon)^{\frac{X_{i,s}^A}{\gamma c_i}}}{1+\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$, and, $\psi_{i,s} = \psi_{i,s-1}\cdot\left[\frac{(1-\epsilon)^{\frac{Y_{i,s}^A}{\gamma W_E}}}{1-\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$
6: end for
```

By the definition of step $p+1$ of algorithm A given in Equation (8), it follows that for any two algorithms with the first p steps being identical, and the last $m-p-1$ steps following the Hypothetical-Oblivious-Conservative algorithm $\widetilde{P}$, algorithm A's $p+1$th step is the one that minimizes expression (7). In particular it follows that expression (7) is upper bounded by the same expression where the $p+1$th step is according to $\widetilde{X}_{i,p+1}$ and $\widetilde{Y}_{i,p+1}$, i.e., we replace $X_{i,p+1}^A$ by $\widetilde{X}_{i,p+1}$

and $Y^A_{i,p+1}$ by $\widetilde{Y}_{i,p+1}$. Therefore, we have

$$\begin{aligned}
\mathcal{F}[A^{p+1}\widetilde{P}^{m-p-1}] &\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}}\left(1+\epsilon\frac{\widetilde{X}_{i,p+1}}{\gamma c_i}\right)\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} \\
&\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p(Y_i^A)}{\gamma W_E}}\left(1-\epsilon\frac{\widetilde{Y}_{i,p+1}}{\gamma W_E}\right)\left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}} \\
&\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}}\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1+\epsilon)^{\frac{1}{\gamma}}} \\
&\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p(Y_i^A)}{\gamma W_E}}\left(1-\frac{\epsilon}{(1+\epsilon)\gamma W_E}\right)\left(1-\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-p-1}\right]}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}} \\
&= \mathcal{F}[A^p\widetilde{P}^{m-p}].
\end{aligned}$$

This completes the proof of the following theorem.

Theorem 3.2. *For any $\epsilon > 0$, Algorithm 1 achieves an objective value of $W_E(1-2\epsilon)$ for the online resource allocation problem with probability at least $1-\epsilon$, assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$. The algorithm A does not require any knowledge of the distribution except for the single parameter W_E.*

3.3 Completely Unknown Distribution

We first give a high-level overview of this section before going into the details. In this section, we design an algorithm A without any knowledge of the distribution at all. The algorithm is similar in spirit to the one in Section 3.2 except that since we do not have knowledge of W_E, we divide the algorithm into many stages. In each stage, we run an algorithm similar to the one in Section 3.2 except that instead of W_E, we use an estimate of W_E that gets increasingly accurate with each successive stage.

More formally, the algorithm runs in l stages $\{0, 1, \ldots, l-1\}$, where l is such that $\epsilon 2^l = 1$, and $\epsilon \in [1/m, 1/2]$ (we need $\epsilon \leq 1/2$ so that l is at least 1) is the error parameter of algorithm designer's choice. Further, we need $m \geq \frac{1}{\epsilon}$ so that $\epsilon m \geq 1$. We assume that ϵm is an integer for clarity of exposition. Stage r handles $t_r = \epsilon m 2^r$ requests for $r \in \{0, \ldots l-1\}$. The first ϵm requests are used just for future estimation, and none of them are served. For convenience, we sometimes call this pre-zero stage as stage -1 and let $t_{-1} = \epsilon m$. Stage $r \geq 0$ serves $t \in [t_r + 1, t_{r+1}]$. Note that in the optimal solution to the expected instance of stage r, no resource i gets consumed by more than $\frac{t_r c_i}{m}$, and every resource i gets a profit of $\frac{t_r W_E}{m}$, i.e., consumption and profit have been scaled down by a factor of $\frac{t_r}{m}$. As in the previous sections, with a high probability, we can only reach close to $\frac{t_r W_E}{m}$. Further, since stage r consists of only t_r requests, which is much smaller than m for small r, it follows that for small r, our error in how close to we get to $\frac{t_r W_E}{m}$ will be higher. Indeed, instead of having the same error parameter of ϵ in every stage, we set stage-specific error parameters that get progressively smaller and become close to ϵ in the final stages. These parameters are chosen such that the overall error is still $O(\epsilon)$, because the later stages having more requests matter more than the former. There are two sources of error/failure that we detail below.

(1) The first source of failure stems from not knowing W_E. Instead, we estimate a quantity Z_r that is an approximation we use for W_E in stage r, and the approximation gets better as r increases. We use Z_r to set a profit target of $\frac{t_r Z_r}{m}$ for stage r. Since Z_r could be much smaller than W_E, our algorithm could become very suboptimal. We prove that with a probability of at least $1 - 2\delta$, we have $W_E(1 - 3\epsilon_{x,r-1}) \le Z_r \le W_E$ (see next for what $\epsilon_{x,r}$ is), where $\delta = \frac{\epsilon}{3l}$. Thus for all the l stages, these bounds are violated with probability at most $2l\delta = 2\epsilon/3$.
(2) The second source of failure stems from achieving this profit target in every stage. We set error parameters $\epsilon_{x,r}$ and $\epsilon_{y,r}$ such that for every i stage r consumes at most $\frac{t_r c_i}{m}(1 + \epsilon_{x,r})$ amount of resource i, and for every i we get a profit of at least $\frac{t_r Z_r}{m}(1 - \epsilon_{y,r})$, with probability at least $1 - \delta$. Thus the overall failure probability, as regards falling short of the target $\frac{t_r Z_r}{m}$ by more than $\epsilon_{y,r}$ and exceeding $\frac{t_r c_i}{m}$ by more than $\epsilon_{x,r}$, for all the l stages together is at most $\delta \cdot l = \epsilon/3$.

Thus summing over the failure probabilities we get $\epsilon/3 + 2\epsilon/3 = \epsilon$. We have that with probability at least $1 - \epsilon$, for every i, the total consumption of resource i is at most $\sum_{r=0}^{l-1} \frac{t_r c_i}{m}(1 + \epsilon_{x,r})$, and total profit from resource i is at least $\sum_{r=0}^{l-1} \frac{t_r W_E}{m}(1 - 3\epsilon_{x,r-1})(1 - \epsilon_{y,r})$. We set $\epsilon_{x,r} = \sqrt{\frac{4\gamma m \ln(2n/\delta)}{t_r}}$ for $r \in \{-1, 0, 1, \ldots, l-1\}$ and $\epsilon_{y,r} = \sqrt{\frac{2 w_{\max} m \ln(2n/\delta)}{t_r Z_r}}$ for $r \in \{0, 1, \ldots, l-1\}$ (we define $\epsilon_{x,r}$ starting from $r = -1$, with $t_{-1} = \epsilon m$, just for technical convenience). From this it follows that $\sum_{r=0}^{l-1} \frac{t_r c_i}{m}(1 + \epsilon_{x,r}) \le c_i$ and $\sum_{r=0}^{l-1} \frac{t_r W_E}{m}(1 - 3\epsilon_{x,r-1})(1 - \epsilon_{y,r}) \ge W_E(1 - O(\epsilon))$, assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$. The algorithm is described in Algorithm 2. This completes the high-level overview of the proof. All that is left to prove is the points 1 and 2 above, on which we would have proved our main theorem, namely Theorem 2.2, which we recall below. **Theorem 2.2** For any $\epsilon \ge 1/m$, Algorithm 2 achieves an objective value of $W_E(1 - O(\epsilon))$ for the online resource allocation problem with probability at least $1 - \epsilon$, assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$. Algorithm 2 does not require any knowledge of the distribution at all.

Detailed Description and Proof. We begin with the first point in our high-level overview above, namely by describing how Z_r is estimated and proving its concentration around W_E. After stage r (including stage -1), the algorithm computes the optimal fractional objective value e_r to the following instance $\mathcal{I}_r$: The instance has the t_r requests of stage r, and the capacity of resource i is capped at $\frac{t_r c_i}{m}$. Using the optimal fractional objective value e_r of this instance, we set $Z_{r+1} = \frac{e_r}{1+\epsilon_{x,r}} \cdot \frac{m}{t_r}$. The first task now is to prove that Z_{r+1} as estimated above is concentrated enough around W_E. This basically requires proving concentration of e_r.

Lemma 3.3. *With a probability at least* $1 - 2\delta$, *we have*

$$\frac{t_r W_E}{m}(1 - 2\epsilon_{x,r}) \le e_r \le \frac{t_r W_E}{m}(1 + \epsilon_{x,r}).$$

Proof. We prove that the lower and upper bound hold with probability $1 - \delta$ each, thus proving the lemma.

We begin with the lower bound on e_r. Note that the expected instance of the instance $\mathcal{I}_r$ has the same optimal solution x^*_{jk} as the optimal solution to the full expected instance (i.e., the one without scaling down by $\frac{t_r}{m}$). Now consider the algorithm $\widetilde{P}(r)$, which is the same as the $\widetilde{P}$ defined in Section 3.1 except that ϵ is replaced by $\epsilon_{x,r}$, i.e., it serves request j with option k with probability $\frac{x^*_{jk}}{1+\epsilon_{x,r}}$. When $\widetilde{P}(r)$ is run on instance $\mathcal{I}_r$, with a probability at least $1 - \frac{\delta}{2n}$, at most $\frac{t_r c_i}{m}$ amount of resource i is consumed, and with probability at least $1 - \frac{\delta}{2n}$, at least $\frac{t_r W_E}{m}\frac{1-\epsilon_{x,r}}{1+\epsilon_{x,r}}$ resource i profit is

ALGORITHM 2: Algorithm for stochastic online resource allocation with unknown distribution

Input: Capacities c_i for $i \in [n]$, the total number of requests m, the value of γ, an error parameter $\epsilon > 1/m$.

Output: An online allocation of resources to requests

```
1: Set $l = \log(1/\epsilon)$
2: Initialize $t_{-1} : t_{-1} \leftarrow \epsilon m$
3: for $r = 0$ to $l - 1$ do
4:     Compute $e_{r-1}$: the optimal solution to the $t_{r-1}$ requests of stage $r - 1$ with capacities capped at $\frac{t_{r-1} c_i}{m}$.
5:     Set $Z_r = \frac{e_{r-1}}{1+\epsilon_{x,r-1}} \cdot \frac{m}{t_{r-1}}$
6:     Set $\epsilon_{x,r} = \sqrt{\frac{4\gamma m \ln(2n/\delta)}{t_r}}$, $\epsilon_{y,r} = \sqrt{\frac{2w_{\max} m \ln(2n/\delta)}{t_r Z_r}}$
7:     Set $\phi^r_{i,0} = \frac{\epsilon_{x,r}}{\gamma c_i}\left[\frac{\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r - 1}}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}\right]$, and, $\psi^r_{i,0} = \frac{\epsilon_{y,r}}{w_{\max}}\left[\frac{\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r - 1}}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}\right]$
8:     for $s = 1$ to $t_r$ do
9:         If the incoming request is $j$, then use the following option $k^*$:
               $$k^* = \arg\min_{k \in \mathcal{K} \cup \{\perp\}} \left\{\sum_i a_{ijk} \cdot \phi^r_{i,s-1} - \sum_i w_{ijk} \cdot \psi^r_{i,s-1}\right\}.$$
10:        $X^A_{i,t_r+s} = a_{ijk^*}$, $Y^A_{i,t_r+s} = w_{ijk^*}$
11:        Update $\phi^r_{i,s} = \phi^r_{i,s-1} \cdot \left[\frac{(1+\epsilon_{x,r})^{\frac{X^A_{i,t_r+s}}{\gamma c_i}}}{1+\frac{\epsilon_{x,r}}{m\gamma}}\right]$, and, $\psi^r_{i,s} = \psi^r_{i,s-1} \cdot \left[\frac{(1-\epsilon_{y,r})^{\frac{Y^A_{i,t_r+s}}{w_{\max}}}}{1-\frac{\epsilon_{y,r}}{m\gamma}}\right]$
12:    end for
13: end for
```

obtained. Thus with a probability at least $1 - 2n \cdot \frac{\delta}{2n} = 1 - \delta$, $\widetilde{P}(r)$ achieves an objective value of at least $\frac{t_r W_E}{m}(1 - 2\epsilon_{x,r})$. Therefore the optimal objective value e_r will also be at least $\frac{t_r W_E}{m}(1 - 2\epsilon_{x,r})$.

To prove the upper bound, we consider the primal and dual LPs that define e_r in LP (9) and the primal and dual LPs defining the expected instance in LP (10). In the latter, for convenience, we use $mp_j\beta_j$ as the dual multiplier instead of just β_j.

Primal and dual LPs defining e_r (9)

Primal defining e_r	**Dual defining** e_r
Maximize λ s.t.	Minimize $\sum_{j \in \mathcal{I}_r} \beta_j + \frac{t_r}{m}\sum_i \alpha_i c_i$ s.t.
$\forall\, i,\ \sum_{j \in \mathcal{I}_r, k} w_{ijk} x_{j,k} \geq \lambda$	$\forall\, j \in \mathcal{I}_r, k,\ \beta_j + \sum_i(\alpha_i a_{ijk} - \rho_i w_{ijk}) \geq 0$
$\forall\, i,\ \sum_{j \in \mathcal{I}_r, k} a_{ijk} x_{j,k} \leq \frac{t_r c_i}{m}$	$\sum_i \rho_i \geq 1$
$\forall\, j \in \mathcal{I}_r,\ \sum_k x_{j,k} \leq 1$	$\forall\, i,\ \rho_i \geq 0, \alpha_i \geq 0$
$\forall\, j \in \mathcal{I}_r, k,\ x_{j,k} \geq 0.$	$\forall\, j \in \mathcal{I}_r,\ \beta_j \geq 0.$

Primal and dual LPs defining the expected instance (10)

Primal for the expected instance	**Dual for the expected instance**
Maximize λ s.t.	Minimize $\sum_j mp_j\beta_j + \sum_i \alpha_i c_i$ s.t.
$\forall\, i,\ \sum_{j,k} mp_j w_{ijk} x_{j,k} \geq \lambda$	$\forall\, j,k,\ mp_j\left(\beta_j + \sum_i(\alpha_i a_{ijk} - \rho_i w_{ijk})\right) \geq 0$
$\forall\, i,\ \sum_{j,k} mp_j a_{ijk} x_{j,k} \leq c_i$	$\sum_i \rho_i \geq 1$
$\forall\, j,\ \sum_k x_{j,k} \leq 1$	$\forall\, i,\ \rho_i \geq 0, \alpha_i \geq 0$
$\forall\, j,k,\ x_{j,k} \geq 0.$	$\forall\, j,\ \beta_j \geq 0,$

Note that the set of constraints in the dual of LP (10) is a superset of the set of constraints in the dual of LP (9), making any feasible solution to dual of LP (10) also feasible to dual of LP (9). In in particular, the optimal solution to dual of LP (10) given by β_j^*'s, α_i^*'s, and ρ_i^*'s is feasible for dual of LP (9). Hence, the value of e_r is upper bounded the objective of dual of LP (9) at β_j^*'s, α_i^*'s, and ρ_i^*'s. That is, we have

$$e_r \leq \sum_{j \in \mathcal{I}_r} \beta_j^* + \frac{t_r}{m}\sum_i \alpha_i^* c_i.$$

We now upper bound the right-hand side by applying Chernoff bounds on $\sum_{j\in\mathcal{I}_r}\beta_j^*$. Since the dual LP in LP (10) is a minimization LP, the constraints there imply that $\beta_j^* \leq w_{\max}$. Applying Chernoff bounds, we have

$$\begin{aligned} e_r &\leq t_r \sum_j p_j\beta_j^* + \sqrt{4t_r\left(\sum_j p_j\beta_j^*\right) w_{\max}\ln(1/\delta)} + \frac{t_r}{m}\sum_i \alpha_i^* c_i \\ &\leq \frac{t_r W_E}{m} + \frac{t_r W_E}{m}\epsilon_{x,r} \end{aligned}$$

where the first inequality holds with probability at least $1-\delta$, and the second inequality uses the fact the optimal value of the expected instance (dual of LP (10)) is W_E. This proves the required upper bound on e_r that $e_r \leq \frac{t_r W_E}{m}(1+\epsilon_{x,r})$ with probability at least $1-\delta$.

Going back to our application of Chernoff bounds above, to apply it in the form above, we require that the multiplicative deviation from mean $\sqrt{\frac{4w_{\max}\ln(1/\delta)}{t_r\sum_j p_j\beta_j^*}} \in [0, 2e-1]$. If $\sum_j p_j\beta_j^* \geq \frac{\epsilon W_E}{m}$, then this requirement would follow. Suppose, however, that $\sum_j p_j\beta_j^* < \frac{\epsilon W_E}{m}$. Since we are happy if the excess over mean is at most $\frac{t_r W_E}{m}\epsilon_{x,r}$, let us look for a multiplicative error of $\frac{\frac{t_r W_E \epsilon_{x,r}}{m}}{t_r\sum_j p_j\beta_j^*}$. Based on the fact that $\sum_j p_j\beta_j^* < \frac{\epsilon W_E}{m}$ and that $\epsilon_{x,r} > \epsilon$ for all r, the multiplicative error can be seen to be at least a constant and can be made larger than $2e-1$ depending on the constant inside the big O of γ. We now use the version of Chernoff bounds for multiplicative error larger than $2e-1$, which gives us that a deviation of $\frac{t_r W_E}{m}\epsilon_{x,r}$ occurs with a probability at most $2^{-\left(1+\frac{\frac{t_r W_E\epsilon_{x,r}}{m}}{t_r\sum_j p_j\beta_j^*}\right)\frac{t_r\sum_j p_j\beta_j^*}{w_{\max}}}$, where the division by $w_{\max}$ is because of the fact that $\beta_j^* \leq w_{\max}$. Noting that $w_{\max} \leq \gamma W_E$, we get that this probability is at most δ/n that is at most δ. □

Lemma 3.3 implies that $W_E(1-3\epsilon_{x,r-1}) \le Z_r \le W_E$, $\forall r \in \{0,1,\dots,l-1\}$. The rest of the proof is similar to that of Section 3.2 and is focused on the second point in the high-level overview we gave in the beginning of Section 3.3. In Section 3.2 we knew W_E and obtained a $W_E(1-2\epsilon)$ approximation with no resource i consumed beyond c_i with probability $1-\epsilon$. Here, instead of W_E we have an approximation for W_E in the form of Z_r that gets increasingly accurate as r increases. We set a target of $\frac{t_r Z_r}{m}$ for stage r and show that with a probability of at least $1-\delta$ we get a profit of $\frac{t_r Z_r}{m}(1-\epsilon_{y,r})$ from every resource i and no resource i consumed beyond $\frac{t_r c_i}{m}(1+\epsilon_{x,r})$ capacity.[8]

As in Section 3.2, call stage r of algorithm A a failure if at least one of the following fails:

(1) For all i, $\sum_{t=t_r+1}^{t_{r+1}} X_{i,t}^A \le \frac{t_r c_i}{m}(1+\epsilon_{x,r})$.
(2) For all i, $\sum_{t=t_r+1}^{t_{r+1}} Y_{i,t}^A \ge \frac{t_r Z_r}{m}(1-\epsilon_{y,r})$.

Let $S_s^r(X_i) = \sum_{t=t_r+1}^{t_r+s} X_{i,t}$ denote the amount of resource i consumed in the first s steps of stage r, and let $S_s^r(Y_i) = \sum_{t=t_r+1}^{t_r+s} Y_{i,t}$ denote the resource i profit in the first s steps of stage r,

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=t_r+1}^{t_{r+1}} X_{i,t}^A \ge \frac{t_r c_i}{m}(1+\epsilon_{x,r})\right] &= \mathbf{Pr}\left[\frac{\sum_{t=t_r+1}^{t_{r+1}} X_{i,t}^A}{\gamma c_i} \ge \frac{t_r}{m\gamma}(1+\epsilon_{x,r})\right] \\
&= \mathbf{Pr}\left[(1+\epsilon_{x,r})^{\frac{\sum_{t=t_r+1}^{t_{r+1}} X_{i,t}^A}{\gamma c_i}} \ge (1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}\right] \\
&\le \mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{\sum_{t=t_r+1}^{t_{r+1}} X_{i,t}^A}{\gamma c_i}}\right] / (1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}} \\
&= \frac{\mathbf{E}\left[\prod_{t=t_r+1}^{t_{r+1}} (1+\epsilon_{x,r})^{\frac{X_{i,t}^A}{\gamma c_i}}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}.
\end{aligned} \tag{11}$$

$$\begin{aligned}
\mathbf{Pr}\left[\sum_{t=t_r+1}^{t_{r+1}} Y_{i,t}^A \le \frac{t_r Z_r}{m}(1-\epsilon_{y,r})\right] &= \mathbf{Pr}\left[\frac{\sum_{t=t_r+1}^{t_{r+1}} Y_{i,t}^A}{w_{\max}} \le \frac{t_r Z_r}{m w_{\max}}(1-\epsilon_{y,r})\right] \\
&= \mathbf{Pr}\left[(1-\epsilon_{y,r})^{\frac{\sum_{t=t_r+1}^{t_{r+1}} Y_{i,t}^A}{w_{\max}}} \ge (1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}\right] \\
&\le \mathbf{E}\left[(1-\epsilon_{y,r})^{\frac{\sum_{t=t_r+1}^{t_{r+1}} Y_{i,t}^A}{w_{\max}}}\right] / (1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}} \\
&= \frac{\mathbf{E}\left[\prod_{t=t_r+1}^{t_{r+1}} (1-\epsilon_{y,r})^{\frac{Y_{i,t}^A}{w_{\max}}}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}.
\end{aligned} \tag{12}$$

If our algorithm A was P (and therefore we can use $\mathbf{E}[X_{i,t}^*] \le \frac{c_i}{m}$ and $\mathbf{E}[Y_{i,t}^*] \ge \frac{W_E}{m} \ge \frac{Z_r}{m}$), then the total failure probability for each stage r that is the sum of Equations (11) and (12) for all the i's would have been $n \cdot [e^{\frac{-\epsilon_{x,r}^2}{4\gamma}\frac{t_r}{m}} + e^{\frac{-\epsilon_{y,r}^2}{2}\frac{t_r Z_r}{m w_{\max}}}] = n \cdot [\frac{\delta}{2n} + \frac{\delta}{2n}] = \delta$. The goal is to design an

[8]Note that we are allowed to consume a bit beyond $\frac{t_r c_i}{m}$, because our goal is just that overall we do not consume beyond c_i and not that for every stage we respect the $\frac{t_r c_i}{m}$ constraint. In spite of this $(1+\epsilon_{x,r})$ excess consumption in all stages, since stage -1 consumes nothing at all, we will see that no excess consumption occurs at the end.

algorithm A for stage r that, unlike P, does not know the distribution but also obtains the same δ failure probability, just as we did in Section 3.2. That is, we want to show that the sum of Equations (11) and (12) over all i's is at most δ:

$$\sum_i \frac{\mathbf{E}\left[\prod_{t=t_r+1}^{t_{r+1}} (1+\epsilon_{x,r})^{\frac{X_{i,t}^A}{\gamma c_i}}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}} + \sum_i \frac{\mathbf{E}\left[\prod_{t=t_r+1}^{t_{r+1}} (1-\epsilon_{y,r})^{\frac{Y_{i,t}^A}{w_{\max}}}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}} \leq \delta.$$

For the algorithm $A^s P^{t_r-s}$, the above quantity can be rewritten as

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S_s^r(X_i^A)}{\gamma c_i}} \prod_{t=t_r+s+1}^{t_{r+1}} (1+\epsilon_{x,r})^{\frac{X_{i,t}^*}{\gamma c_i}}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}$$
$$+ \sum_i \frac{\mathbf{E}\left[\left(1-\epsilon_{y,r}\right)^{\frac{S_s^r(Y_i^A)}{w_{\max}}} \prod_{t=t_r+s+1}^{t_{r+1}} \left(1-\epsilon_{y,r}\right)^{\frac{Y_{i,t}^*}{w_{\max}}}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}},$$

which, by using $(1+\epsilon)^x \leq 1+\epsilon x$ for $0 \leq x \leq 1$, is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S_s^r(X_i^A)}{\gamma c_i}} \prod_{t=t_r+s+1}^{t_{r+1}} \left(1+\epsilon_{x,r}\frac{X_{i,t}^*}{\gamma c_i}\right)\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}$$
$$+ \sum_i \frac{\mathbf{E}\left[\left(1-\epsilon_{y,r}\right)^{\frac{S_s^r(Y_i^A)}{w_{\max}}} \prod_{t=t_r+s+1}^{t_{r+1}} \left(1-\epsilon_{y,r}\frac{Y_{i,t}^*}{w_{\max}}\right)\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}.$$

Since for all t, the random variables $X_{i,t}^*$, $X_{i,t}^A$, $Y_{i,t}^*$, and $Y_{i,t}^A$ are all independent, and $\mathbf{E}[X_{i,t}^*] \leq \frac{c_i}{m}$, $\mathbf{E}[Y_{i,t}^*] \geq \frac{W_E}{m}$, and $\frac{W_E}{w_{\max}} \geq \frac{1}{\gamma}$, the above is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S_s^r(X_i^A)}{\gamma c_i}} \left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-s}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}} + \sum_i \frac{\mathbf{E}\left[\left(1-\epsilon_{y,r}\right)^{\frac{S_s^r(Y_i^A)}{w_{\max}}} \left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-s}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}. \tag{13}$$

Let $\mathcal{F}_r[A^s P^{t_r-s}]$ denote the quantity in Equation (13), which is an upper bound on failure probability of the hybrid algorithm $A^s P^{t_r-s}$ for stage r. We just showed that $\mathcal{F}_r[P^{t_r}] \leq \delta$. We now prove that for all $s \in \{0, 1, \ldots, t_r - 1\}$, $\mathcal{F}_r[A^{s+1}P^{t_r-s-1}] \leq \mathcal{F}_r[A^s P^{t_r-s}]$, thus proving that $\mathcal{F}_r[A^{t_r}] \leq \delta$, i.e., running the algorithm A for all the t_r steps of stage r results in a failure with probability at most δ.

Assuming that for all $s < p$, the algorithm A has been defined for the first $s+1$ steps in such a way that $\mathcal{F}_r[A^{s+1}P^{t_r-s-1}] \leq \mathcal{F}_r[A^s P^{t_r-s}]$, we now define A for the $p+1$th step of stage r in a way that will ensure that $\mathcal{F}_r[A^{p+1}P^{t_r-p-1}] \leq \mathcal{F}_r[A^p P^{t_r-p}]$. We have

$$\begin{aligned}\mathcal{F}_r[A^{p+1}P^{m-p-1}] &= \sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S^r_{p+1}(X^A_i)}{\gamma c_i}}\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon_{y,r})^{\frac{S^r_{p+1}(Y^A_i)}{w_{\max}}}\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}} \\ &\le \sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S^r_p(X^A_i)}{\gamma c_i}}\left(1+\epsilon_{x,r}\frac{X^A_{i,t_r+p+1}}{\gamma c_i}\right)\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon_{y,r})^{\frac{S^r_p(Y^A_i)}{w_{\max}}}\left(1-\epsilon_{y,r}\frac{Y^A_{i,t_r+p+1}}{w_{\max}}\right)\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}. \qquad (14)\end{aligned}$$

Define

$$\begin{aligned}\phi^r_{i,s} &= \frac{\epsilon_{x,r}}{\gamma c_i}\left[\frac{(1+\epsilon_{x,r})^{\frac{S^r_s(X^A_i)}{\gamma c_i}}\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-s-1}}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}\right] \\ \psi^r_{i,s} &= \frac{\epsilon_{y,r}}{w_{\max}}\left[\frac{(1-\epsilon_{y,r})^{\frac{S^r_s(Y^A_i)}{w_{\max}}}\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-s-1}}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}\right].\end{aligned}$$

Define the step $p+1$ of algorithm A as picking the following option k^* for request j:

$$k^* = \arg\min_k \left\{\sum_i a_{ijk}\cdot\phi^r_{i,p} - \sum_i w_{ijk}\cdot\psi^r_{i,p}\right\}.$$

By the above definition of step $p+1$ of algorithm A (for stage r), it follows that for any two algorithms with the first p steps being identical, and the last t_r-p-1 steps following the Hypothetical-Oblivious algorithm P, algorithm A's $p+1$th step is the one that minimizes expression (14). In particular, it follows that expression (14) is upper bounded by the same expression where the $p+1$th step is according to X^*_{i,t_r+p+1} and Y^*_{i,t_r+p+1}, i.e., we replace X^A_{i,t_r+p+1} by X^*_{i,t_r+p+1} and Y^A_{i,t_r+p+1} by Y^*_{i,t_r+p+1}. Therefore, we have

$$\begin{aligned}\mathcal{F}_r[A^{p+1}P^{m-p-1}] &\le \sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S^r_p(X^A_i)}{\gamma c_i}}\left(1+\epsilon_{x,r}\frac{X^*_{i,tr+p+1}}{\gamma c_i}\right)\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon_{y,r})^{\frac{S^r_p(Y^A_i)}{w_{\max}}}\left(1-\epsilon_{y,r}\frac{Y^*_{i,tr+p+1}}{w_{\max}}\right)\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}} \\ &\le \sum_i \frac{\mathbf{E}\left[(1+\epsilon_{x,r})^{\frac{S^r_p(X^A_i)}{\gamma c_i}}\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)\left(1+\frac{\epsilon_{x,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1+\epsilon_{x,r})^{(1+\epsilon_{x,r})\frac{t_r}{m\gamma}}}\end{aligned}$$

$$+ \sum_i \frac{\mathbf{E}\left[(1-\epsilon_{y,r})^{\frac{S_p^r\left(Y_i^A\right)}{w_{\max}}}\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)\left(1-\frac{\epsilon_{y,r}}{m\gamma}\right)^{t_r-p-1}\right]}{(1-\epsilon_{y,r})^{(1-\epsilon_{y,r})\frac{t_r Z_r}{m w_{\max}}}}$$
$$= \mathcal{F}_r[A^p P^{t_r-p}].$$

This completes the proof of Theorem 2.2.

3.4 Approximate Estimations

Our Algorithm 2 in Section 3.3 required periodically computing the optimal solution to an offline instance. Similarly, our Algorithm 1 in Section 3.2 requires the value of W_E to be given. Suppose we could only approximately estimate these quantities, do our results carry through approximately? That is, suppose the solution to the offline instance is guaranteed to be at least $\frac{1}{\alpha}$ of the optimal, and the stand-in that we are given for W_E is guaranteed to be at least $\frac{1}{\alpha}$ of W_E. Both our Theorem 2.2 and Theorem 3.2 go through with just the W_E replaced by W_E/α. Every step of the proof of the exact version goes through in this approximate version, and so we skip the formal proof for this statement.

3.5 Adversarial Stochastic Input

In this section, we relax the assumption that requests are drawn i.i.d. every time step. Namely the distribution for each time step need not be identical, but an adversary gets to decide which distribution to sample a request from. The adversary could even use how the algorithm has performed in the first $t-1$ steps in picking the distribution for a given time step t. The relevance of this model for the real world is that for settings like display ads, the distribution fluctuates over the day. In general, a day is divided into many chunks, and within a chunk, the distribution is assumed to remain i.i.d. This is exactly captured by this model.

We give algorithms that give guarantees against three different benchmarks in the three models below. The benchmarks get successively stronger, and hence the information sought by the algorithm also increases successively.

3.5.1 ASI Model 1. In this model, the guarantee we give is against the worst distribution over all time steps picked by the adversary. More formally, let $W_E(t)$ denote the optimal profit for the expected instance of distribution of time step t. Our benchmark will be $W_E = \min_t W_E(t)$. Given just the single number W_E, our Algorithm 1 in Section 3.2 will guarantee a revenue of $W_E(1-2\epsilon)$ with a probability of at least $1-\epsilon$ assuming $\gamma = O(\frac{\epsilon^2}{\log(n/\epsilon)})$, just like the guarantee in Theorem 3.2.

Algorithm 1 works for this ASI model, because the proof did not use the similarity of the distributions beyond the fact that $\mathbf{E}[X_{i,t}^*|X_{i,t'}^*\forall t' < t] \leq \frac{c_i}{m}$ for all values of $X_{i,t'}^*$, and $\mathbf{E}[Y_{i,t}^*|Y_{i,t'}^*\forall t' < t] \geq \frac{W_E}{m}$ for all values of $Y_{i,t'}^*$ (Here $X_{i,t}^*$ and $Y_{i,t}^*$ denote the random variables for resource consumption and profit at time t following from allocation according the optimal solution to the expected instance of the distribution used in stage t). In other words, distributions being identical and independent is not crucial, but the fact that the expected instances of these distributions have a minimum profit guarantee in spite of all the dependencies between the distributions is sufficient. Both of these inequalities remain true in this model of ASI also, and thus it easy to verify that Algorithm 1 works for this model.

3.5.2 ASI Model 2. In this model, which is otherwise identical to model 1, our benchmark is stronger, namely $W_E = \frac{\sum_{t=1}^m W_E(t)}{m}$: This is clearly a much stronger benchmark than $\min_t W_E(t)$.

Correspondingly, our algorithm requires more information than in model 1: We ask for $W_E(t)$ for every t, at the beginning of the algorithm.

A slight modification of our Algorithm 1 in Section 3.2 will give a revenue of $\frac{\sum_{t=1}^m W_E(t)}{m}(1-2\epsilon)$ with probability at least $1-\epsilon$, i.e., $W_E(1-2\epsilon)$ w.p. at least $(1-\epsilon)$. Among the two potential functions $\phi_{i,s}$ and $\psi_{i,s}$, we modify $\psi_{i,s}$ in the most natural way to account for the fact that distributions change every step.

Define

$$\phi_{i,s} = \frac{1}{c_i}\left[\frac{(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}}\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-s-1}}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$$

$$\psi_{i,s} = \frac{1}{W_E}\left[\frac{(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma W_E}}\prod_{t=s+2}^m\left(1-\frac{\epsilon W_E(t)}{(1+\epsilon)W_E\gamma m}\right)}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right].$$

Note that when $W_E(t) = W_E$ for all t, then we get precisely the $\psi_{i,s}$ defined in Section 3.2 for Algorithm 1. We present our algorithm below in Algorithm 3.

Algorithm 3 works for this ASI model much for the same reason Algorithm 1 worked for ASI model 2: All the proof needs is that $\mathbf{E}[X^*_{i,t}|X^*_{i,t'}\forall t' < t] \le \frac{c_i}{m}$ for all values of $X^*_{i,t'}$, and $\mathbf{E}[Y^*_{i,t}|Y^*_{i,t'}\forall t' < t] = \frac{W_E(t)}{m}$ for all values of $Y^*_{i,t'}$ (Here $X^*_{i,t}$ and $Y^*_{i,t}$ denote the random variables for resource consumption and profit at time t following from allocation according the optimal solution to the expected instance of the distribution used in stage t).

ALGORITHM 3: Algorithm for stochastic online resource allocation in ASI model 2

Input: Capacities c_i for $i \in [n]$, the total number of requests m, the values of γ and $W_E(t)$ for $t \in [m]$, an error parameter $\epsilon > 0$.
Output: An online allocation of resources to requests

```
1: Initialize $\phi_{i,0} = \frac{1}{c_i}\left[\frac{\left(1+\frac{\epsilon}{(1+\epsilon)\gamma m}\right)^{m-1}}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$, and, $\psi_{i,0} = \frac{1}{W_E}\left[\frac{\prod_{t=2}^m\left(1-\frac{\epsilon W_E(t)}{(1+\epsilon)W_E\gamma m}\right)}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right]$
2: for $s = 1$ to $m$ do
3:     If the incoming request is $j$, then use the following option $k^*$:
           $k^* = \arg\min_{k\in\mathcal{K}\cup\{\perp\}}\left\{\sum_i a_{ijk}\cdot\phi_{i,s-1} - \sum_i w_{ijk}\cdot\psi_{i,s-1}\right\}.$
4:     $X^A_{i,s} = a_{ijk^*}$, $Y^A_{i,s} = w_{ijk^*}$
5:     Update $\phi_{i,s} = \phi_{i,s-1}\cdot\left[\frac{(1+\epsilon)^{\frac{X^A_{i,s}}{\gamma c_i}}}{1+\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$, and, $\psi_{i,s} = \psi_{i,s-1}\cdot\left[\frac{(1-\epsilon)^{\frac{Y^A_{i,s}}{\gamma W_E}}}{1-\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$
6: end for
```

We skip the proof for the profit guarantee of $W_E(1-2\epsilon)$, since it is almost identical to the proof in Section 3.2 for Algorithm 1.

3.5.3 ASI Model 3. In this model, which is otherwise identical to models 1 and 2, our benchmark is even stronger: namely the optimal profit of the expected instance with all the time varying distributions (explicitly spelled out in LP (15)). This benchmark W_E is the strongest benchmark possible. Correspondingly, our algorithm requires more information than in model 2: We ask for

$W_{E,i}(t)$ for every i and t and $c_i(t)$ for every i and t at the beginning of the algorithm, where $W_{E,i}(t)$ and $c_i(t)$ are the amount of type i profit obtained and type i resource consumed by the optimal solution to the expected instance in LP (15) at step t. Namely $W_{E,i}(t) = \sum_{j,k} p_{j,t} w_{ijk} x^*_{j,k,t}$ and $c_i(t) = \sum_{j,k} p_{j,t} a_{ijk} x^*_{j,k,t}$, where $x^*_{j,k,t}$'s are the optimal solution to LP (15).

Primal and dual LPs defining the expected instance (15)

Primal for ASI model 3	**Dual for ASI model 3**
Maximize λ s.t.	Minimize $\sum_{j,t} p_{j,t}\beta_j, t + \sum_i \alpha_i c_i$ s.t.
$\forall\, i,\ \sum_{t,j,k} p_{j,t} w_{ijk} x_{j,k,t} \geq \lambda$	$\forall\, j,k,\ p_{j,t}\left(\beta_{j,t} + \sum_i (\alpha_i a_{ijk} - \rho_i w_{ijk})\right) \geq 0$
$\forall\, i,\ \sum_{t,j,k} p_{j,t} a_{ijk} x_{j,k,t} \leq c_i$	$\sum_i \rho_i \geq 1$
$\forall\, j,t,\ \sum_k x_{j,k,t} \leq 1$	$\forall\, i,\ \rho_i \geq 0, \alpha_i \geq 0$
$\forall\, j,k,t\ x_{j,k,t} \geq 0.$	$\forall\, j,\ \beta_j \geq 0.$

A slight modification of our Algorithm 1 in Section 3.2 will give a revenue of $W_E(1-2\epsilon)$ with probability at least $1-\epsilon$. We modify the two potential functions $\phi_{i,s}$ and $\psi_{i,s}$ in the most natural way to account for the fact that distributions change every step. Let $W_{E,i} = \sum_{t=1}^m W_{E,i}(t)$, and thus, our benchmark W_E is simply $\min_i W_{E,i}$. Note also that $\sum_{t=1}^m c_i(t)$, call it c_i^*, is at most c_i by the feasibility of the optimal solution to LP (15).

Define

$$\phi_{i,s} = \frac{1}{c_i}\left[\frac{(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \prod_{t=s+2}^m \left(1 + \frac{\epsilon c_i(t)}{(1+\epsilon)c_i\gamma}\right)}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$$

$$\psi_{i,s} = \frac{1}{W_{E,i}}\left[\frac{(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma W_{E,i}}} \prod_{t=s+2}^m \left(1 - \frac{\epsilon W_{E,i}(t)}{(1+\epsilon)W_{E,i}\gamma}\right)}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right].$$

We present our algorithm below in Algorithm 4.

Algorithm 4 works for this ASI model much for the same reason Algorithm 3 worked for ASI model 2: All the proof needs is that $\mathbf{E}[X^*_{i,t} | X^*_{i,t'} \forall t' < t] = \frac{c_i(t)}{m}$ for all values of $X^*_{i,t'}$ and $\mathbf{E}[Y^*_{i,t} | Y^*_{i,t'} \forall t' < t] = \frac{W_{E,i}(t)}{m}$ for all values of $Y^*_{i,t'}$ (Here $X^*_{i,t}$ and $Y^*_{i,t}$ denote the random variables for resource consumption and profit at time t following from allocation according the optimal solution to the expected instance captured by LP (15)).

We skip the proof for the profit guarantee of $W_E(1-2\epsilon)$, since it is almost identical to the proof in Section 3.2 for Algorithm 1.

4 PROOF OF NEAR-OPTIMALITY OF ONLINE ALGORITHM FOR RESOURCE ALLOCATION

In this section, we construct a family of instances of the resource allocation problem in the i.i.d. setting for which $\gamma = \omega(\epsilon^2/\log n)$ will rule out a competitive ratio of $1 - O(\epsilon)$. The construction

ALGORITHM 4: Algorithm for stochastic online resource allocation in ASI model 3

Input: Capacities $c_i(t)$ and c_i, and profits $W_{E,i}(t)$ for $i \in [n],\ t \in [m]$, the total number of requests m, the values of γ and an error parameter $\epsilon > 0$.
Output: An online allocation of resources to requests

```
1: Initialize $\phi_{i,0} = \frac{1}{c_i}\left[\frac{\prod_{t=2}^{m}\left(1+\frac{\epsilon c_i(t)}{(1+\epsilon)c_i\gamma}\right)}{(1+\epsilon)^{\frac{1}{\gamma}}}\right]$, and, $\psi_{i,0} = \frac{1}{W_{E,i}}\left[\frac{\prod_{t=2}^{m}\left(1-\frac{\epsilon W_{E,i}(t)}{(1+\epsilon)W_{E,i}\gamma}\right)}{(1-\epsilon)^{\frac{1-\epsilon}{\gamma(1+\epsilon)}}}\right]$
2: for $s = 1$ to $m$ do
3:     If the incoming request is $j$, then use the following option $k^*$:
           $$k^* = \arg\min_{k\in\mathcal{K}\cup\{\perp\}}\left\{\sum_i a_{ijk}\cdot\phi_{i,s-1} - \sum_i w_{ijk}\cdot\psi_{i,s-1}\right\}.$$
4:     $X^A_{i,s} = a_{ijk^*},\ Y^A_{i,s} = w_{ijk^*}$
5:     Update $\phi_{i,s} = \phi_{i,s-1}\cdot\left[\frac{(1+\epsilon)^{\frac{X^A_{i,s}}{\gamma c_i}}}{1+\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$, and, $\psi_{i,s} = \psi_{i,s-1}\cdot\left[\frac{(1-\epsilon)^{\frac{Y^A_{i,s}}{\gamma W_{E,i}}}}{1-\frac{\epsilon}{(1+\epsilon)\gamma m}}\right]$
6: end for
```

closely follows the construction by Agrawal et al. [2] for proving a similar result in the random-permutation model.

The instance has $n = 2^z$ resources with B units of each resource, and $Bz(2 + 1/\alpha) + \sqrt{Bz}$ requests where $\alpha < 1$ is some scalar. Each request has only one "option," i.e., each request can either be dropped or, if served, consumes the same number of units of a specific subset of resources (which we construct below). This means that a request is simply a scalar times a binary string of length 2^z, with the ones (or the scalars) representing the coordinates of resources that are consumed by this request if served.

The requests are classified into z categories. Each category in expectation consists of $m/z = B(2 + 1/\alpha) + \sqrt{B/z}$ requests. A category, indexed by i, is composed of two different binary vectors v_i and w_i (each of length 2^z). The easiest way to visualize these vectors is to construct two $2^z \times z$ $0-1$ matrices, with each matrix consisting of all possible binary strings of length z, written one string in a row. The first matrix lists the strings in ascending order and the second matrix in descending order. The ith column of the first matrix multiplied by the scalar α is the vector v_i and the ith column of the second matrix is the vector w_i. There are two properties of these vectors that are useful for us:

(1) The vectors v_i/α and w_i are complements of one another.
(2) Any matrix of z columns, with column i being either v_i/α or w_i has exactly one row with all ones in it.

We are ready to construct the i.i.d. instance now. Each request is drawn from the following distribution. A given request could be, for each i, of type:

(1) v_i and profit 4α with probability $\frac{B}{\alpha zm}$,
(2) w_i and profit 3 with probability $\frac{B}{zm}$,
(3) w_i and profit 2 with probability $\sqrt{\frac{B}{zm}}$,

(4) w_i and profit 1 with probability $\frac{B}{zm}$,
(5) Zero vector with zero profit with probability $1 - \frac{2B}{zm} - \sqrt{\frac{B}{zm}} - \frac{B}{\alpha zm}$.

We use the following notation for request types: A $(2, w_i)$ request stands for a w_i type request of profit 2. Observe that the expected instance has an optimal profit of OPT $= 7B$. This is obtained by picking for each i the $\frac{B}{\alpha z}$ vectors of type v_i and profit 4α, along with $\frac{B}{z}$ vectors of type w_i with profit 3. Note that this exhausts every unit of every item, and thus, combined with the fact that the most profitable requests have been served, the value of $7B$ is indeed the optimal value. This means that any algorithm that obtains a $1-\epsilon$ competitive ratio must have an expected profit of at least $7B - 7\epsilon B$.

Let $r_i(w)$ and $r_i(v)$ be the random variables denoting the number of vectors of type w_i and v_i picked by some $1-\epsilon$ competitive algorithm *ALG*. Let $a_i(v)$ denote the total number of vectors of type v_i that arrived in this instance.

Lemma 4.1. *For some constant k, the $r_i(w)$'s satisfy*

$$\sum_i \mathbf{E}[|r_i(w) - B/z|] \le 7\epsilon B + 4\sqrt{\alpha k B z}.$$

Proof. Let Y denote the set of indices i for which $r_i(w) > B/z$. One way to upper bound the total number of vectors of type v picked by *ALG* is the following. Split the set of indices into Y and $X = [z] \setminus Y$. The number of v's from Y is, by chosen notation, $\sum_{i\in Y} r_i(v)$. The number of v's from X, we show, is at most $\frac{B - \sum_{i\in Y} r_i(w)}{\alpha}$. Note that since there are only B copies of every item, it follows that $\alpha[\sum_i r_i(v)] \le B$ and $\sum_i r_i(w) \le B$. Further, by property 2 of v_i's and w_i's, we have that $\alpha[\sum_{i\in X} r_i(v)] + \sum_{i\in Y} r_i(w) \le B$. This means that the number of v's from X is at most $\frac{B - \sum_{i\in Y} r_i(w)}{\alpha}$.

Let $P = \sum_{i\in Y}(r_i(w) - B/z)$ and $M = \sum_{i\in X}(B/z - r_i(w))$. Showing $\mathbf{E}[P+M] \le 7\epsilon B + 4\sqrt{\alpha k B z}$ proves the lemma. By an abuse of notation, let *ALG* also be the profit obtained by the algorithm *ALG* and let best$w_i(t)$ denote the most profitable t requests of type w_i in a given instance. Note that $4B + \sum_{i=1}^{z}\text{best}w_i(B/z) \le 7B = \text{OPT}$. We upper bound $\mathbf{E}[ALG]$ as

$$\begin{aligned}
\mathbf{E}[ALG] &\le \mathbf{E}\left[\sum_{i=1}^{z} \text{best}w_i(r_i(w))\right] + 4\alpha\left[\frac{B - \sum_{i\in Y}\mathbf{E}[r_i(w)]}{\alpha} + \sum_{i\in Y}\mathbf{E}[r_i(v)]\right] \\
&\le \mathbf{E}\left[\sum_{i=1}^{z}\text{best}_i(B/z) + 3P - M\right] + 4\left(B - \mathbf{E}\left[\sum_{i\in Y}(r_i(w) - B/z) + |Y|B/z\right]\right) \\
&\quad + 4\alpha\,\mathbf{E}\left[\sum_{i\in Y} r_i(v)\right] \\
&\le \text{OPT} - \mathbf{E}[P+M] + 4\alpha\,\mathbf{E}\left[\sum_{i\in Y}\left(r_i(v) - \frac{B}{\alpha z}\right)\right] \\
&\left(\text{Since } P = \sum_{i\in Y}(r_i(w) - B/z)\right) \\
&\le \text{OPT} - \mathbf{E}[P+M] + 4\alpha\,\mathbf{E}\left[\sum_{i\in Y}\left(a_i(v) - \frac{B}{\alpha z}\right)\right] \text{ (Since } r_i(v) \le a_i(v)) \\
&\le \text{OPT} - \mathbf{E}[P+M] + 4\alpha\,\mathbf{E}\left[\sum_{i:a_i(v)\ge\frac{B}{\alpha z}}\left(a_i(v) - \frac{B}{\alpha z}\right)\right]
\end{aligned}$$

$$\leq \text{OPT} - \mathbf{E}[P + M] + 4\alpha \cdot z \cdot k' \cdot \sqrt{\frac{B}{\alpha z}}$$
(where k' is some constant from Central Limit Theorem)
$$\leq \text{OPT} - \mathbf{E}[P + M] + 4\sqrt{\alpha k B z} \text{ (where } k \text{ is } k'^2).$$

The inequality that follows from CLT uses the fact that for a random variable $X \sim (m, c/m)$ (X is binomially distributed with success probability of c/m), whenever $c = \omega(1)$, and $c \leq m$, we have that $E[X|X \geq c] = c + k'\sqrt{c}$, for some constant k'. In this case, we have $\frac{B}{\alpha z}$ in place of c. For example, if $n = \log(m)$ (and thus $z = \log n = \log\log m$), as long as $B = \omega(\log\log m)$ and $B \leq m$, then the CLT inequality will hold. Note that α could have been any constant and this argument still holds. □

We are now ready to prove Theorem 2.3, which we restate here for convenience. **Theorem 2.3** There exist instances with $\gamma = \frac{\epsilon^2}{\log(n)}$ such that no algorithm, even with complete knowledge of the distribution, can get a $1 - o(\epsilon)$ approximation factor.

Proof. We first give the overview of the proof before providing a detailed argument.

Overview. Lemma 4.1 says that $r_i(w)$ has to be almost always close to B/z for all i. In particular, the probability that $\sum_i |r_i(w) - B/z| \leq 4(7\epsilon B + 4\sqrt{\alpha k B z})$ is at least 3/4. In this proof, we show, in an argument similar to the one in Agrawal et al. [2], that if this has to be true, then one has to lose a revenue of $\Omega(\sqrt{Bz}) - 4(7\epsilon B + 4\sqrt{\alpha k B z})$. Since α can be set to any arbitrary constant, this means that we lose a revenue of $\Omega(\sqrt{Bz}) - 28\epsilon B$. Since OPT is $7B$, to get a $1 - \epsilon$ approximation, we require that $\Omega(\sqrt{Bz}) - 28\epsilon B \leq 7\epsilon B$. Thus, we need $B \geq \Omega(\frac{\log m}{\epsilon^2})$. In other words, we require $\gamma = \frac{1}{B} \leq O(\frac{\epsilon^2}{\log m})$.

In Detail. We now proceed to prove the claim that a revenue loss of $\Omega(\sqrt{Bz}) - 4(7\epsilon B + \sqrt{\alpha k B z})$ is inevitable. We just showed that with a probability of at least 3/4, $\sum_i |r_i(w) - B/z| \leq 4(7\epsilon B + 4\sqrt{\alpha k B z})$. For now, we assume that $r_i(w)$ should be exactly B/z and later account for the probability 1/4 leeway and also the $4(7\epsilon B + 4\sqrt{\alpha k B z})$ error that is allowed by Lemma 4.1. With this assumption, we show that for each i there is a loss of $\Omega(\sqrt{B/z})$.

For each i let o_i denote the number of $(1, w_i)$ requests that the algorithm served in total. With a constant probability the number of 3's and 2's (of type w_i) exceed B/z. If $o_i = \Omega(\sqrt{B/z})$, then there is a loss of at least $\Omega(\sqrt{B/z})$ because of picking 1's instead of 2's or 3's. This establishes the $\Omega(\sqrt{B/z})$ loss that we wanted to prove for this case.

Suppose $o_i < \Omega(\sqrt{Bz})$. For each i, let R_i be the set of requests of type w_i with profit either 1 or 3. For every i, with a constant probability $2B/z - 2\sqrt{B/z} \leq |R_i| \leq 2B/z + 2\sqrt{B/z}$. Conditional on the set R_i, we make the following two observations:

- the types of requests in R_i are independent random variables that take value 1 or 3 with equal probability.
- the order of requests in R_i is a uniformly random permutation of R_i.

Now consider any $(2, w_i)$ request, say, the tth request, of profit 2. With a constant probability, this request can be served without violating any capacity constraints, and thus, the algorithm has to decide whether or not to serve this request. In at least 1/2 of the random permutations of R_i, the number of bids from set R_i before the bid t is less than B/z. Conditional on this event, the profits of requests in R_i before t, with a constant probability could:

(1) take values such that there are enough $(3, w_i)$ requests after t to make the total number of w_i requests picked by the algorithm to be at least B/z;
(2) take values such that even if all the $(3, w_i)$ requests after t were picked, the total number of w_i requests picked is at most $B/z - \sqrt{B/z}$ with a constant probability.

In the first kind of instances (where number of $(3, w_i)$ requests are more than B/z) retaining $(2, w_i)$ causes a loss of 1 as we could have picked a 3 instead. In the second kind, skipping $(2, w_i)$ causes a loss of 1, since we could have picked that 2 instead of a 1. Thus there is an inevitable constant probability loss of 1 per $(2, w_i)$ request. Thus, in expectation, there is a $\Omega(\sqrt{B/z})$ loss.

Thus, whether $o_i = \sqrt{B/z}$ or $o_i < \sqrt{B/z}$, we have established a loss of $\Omega(\sqrt{B/z})$ per i and thus a total expected loss of $\Omega(\sqrt{Bz})$. This is under the assumption that $r_i(w)$ is exactly B/z. There is a leeway of $4(7\epsilon B + 4\sqrt{\alpha k B z})$ granted by Lemma 4.1. Even after that leeway, since α can be made an arbitrarily small constant and Lemma 4.1 still holds, we have the loss at $\Omega(\sqrt{Bz}) - 28\epsilon B$. Now after the leeway, the statement $\sum_i |r_i(w) - B/z| \leq 4(7\epsilon B + 4\sqrt{\alpha k B z})$ has to hold only with probability 3/4. But even this puts the loss at $\Omega(\sqrt{Bz}) - 21\epsilon B$.

Therefore, $E[ALG] \leq \text{OPT} - \Omega(\sqrt{Bz}) - 21\epsilon B$. Since $\text{OPT} = 7B$, we have $E[ALG] \leq \text{OPT}(1 - \Omega(\sqrt{z/B}) - 21\epsilon)$, and to get the $1 - O(\epsilon)$ approximation we need $\Omega(\sqrt{z/B} - 21\epsilon) \leq O(\epsilon)$, implying that $B \geq \Omega(z/\epsilon^2) = \Omega(\log m/\epsilon^2)$. □

5 GREEDY ALGORITHM FOR ADWORDS

In this section, we give a simple proof of Theorem 2.4, which we restate below for convenience.

Theorem 2.4. The greedy algorithm achieves an approximation factor of $1 - 1/e$ for the Adwords problem in the i.i.d. unknown distributions model for all γ, i.e., $0 \leq \gamma \leq 1$.

As noted in Section 2.2, where the Adwords problem was introduced, the budget constraints are not hard, i.e., when 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 [13] 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 that almost gets down to γ tending to zero, i.e., 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 for all γ.

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 expected instance. Let $w_i(t)$ denote the amount of money spent by advertiser i at time step t, i.e., for the tth 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 expected instance.

Consider the greedy algorithm that allocates the query j arriving at time t to the advertiser who has the maximum effective bid for that query, i.e., $\text{argmax}_i \min\{b_{ij}, B_i - \sum_{r=1}^{t-1} w_i(r)\}$. 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 expected instance. 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 5.1. $\mathbf{E}[w_i^h(t)|f_i^h(t-1)] \geq \frac{f_i^h(t-1)}{m}$.

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

$$\mathbf{E}[w_i^h(t)|f_i^h(t-1)] = \sum_{j \in \text{EXCEED}_i(t)} \left(B_i - \sum_{r=1}^{t-1} w_i^h(r)\right)\frac{x_{ij}y_j}{m} + \sum_{j \notin \text{EXCEED}_i(t)} b_{ij}\frac{x_{ij}y_j}{m}. \tag{16}$$

If $\sum_{j \in \text{EXCEED}_i(t)} x_{ij}y_j \geq 1$, then by Equation (16),

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

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

$$\mathbf{E}[w_i^h(t)|f_i^h(t-1)] = \frac{f_i^h(0)}{m} - \sum_{j \in \text{EXCEED}_i(t)} \left(b_{ij} - \left(B_i - \sum_{r=1}^{t-1} w_i^h(r)\right)\right)\frac{x_{ij}y_j}{m}. \tag{17}$$

Since $b_{ij} \leq B_i$, and $\sum_{j \in \text{EXCEED}_i(t)} x_{ij}y_j < 1$, Equation (17) 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 5.2. *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

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

where the inequality is due to Lemma 5.1. Summing over all i gives the Lemma. □

Lemma 5.3. $\mathbf{E}[\text{GREEDY}] \geq (1-1/e)\sum_{i,j} b_{ij}x_{ij}y_j$.

Proof. Lemma 5.2 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 5.2 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

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

and this proves the lemma. □

Lemma 5.3 proves Theorem 2.4.

6 FAST APPROXIMATION ALGORITHM FOR LARGE MIXED PACKING AND COVERING INTEGER PROGRAMS

In this section, we consider the mixed packing-covering problem stated in Section 2.4 and prove Theorem 2.5. We restate the integer program for the mixed covering-packing problem here,

$$\begin{aligned}
&\forall\, i, \sum_{j,k} a_{ijk}x_{j,k} \le c_i\\
&\forall\, i, \sum_{j,k} w_{ijk}x_{j,k} \ge d_i\\
&\forall\, j, \sum_{k} x_{j,k} \le 1\\
&\forall\, j,k, x_{j,k} \in \{0,1\}.
\end{aligned}\tag{18}$$

The goal is to check whether there is a feasible solution to this IP. 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 with a slack, namely even if all of the c_i's are multiplied by $1+3\epsilon(1+\epsilon)$ and all of the d_i's are multiplied by $1-3\epsilon(1+\epsilon)$.

We use $1+3\epsilon(1+\epsilon)$ and $1-3\epsilon(1+\epsilon)$ for slack instead of just $1+\epsilon$ and $1-\epsilon$ purely to reduce notational clutter in what follows (mainly for the NO case).

Like in the online problem, we refer to the quantities indexed by j as requests, a_{ijk} as resource i consumption, and w_{ijk} as resource i profit, and the quantities indexed by k as options. There are a total of m requests, n resources, and K options, and the "zero" option is denoted by $\perp$. Recall that the parameter γ for this problem is defined by $\gamma = \max(\{\frac{a_{ijk}}{c_i}\}_{i,j,k} \cup \{\frac{w_{ijk}}{d_i}\}_{i,j,k})$. Our algorithm needs the values of m, n, and γ (an upper bound on the value of γ also suffices).

High-level Overview. We solve this offline problem in an online manner via random sampling. We sample $T = \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ requests j from the set of possible requests uniformly at random with replacement and then design an algorithm that allocates resources online for these requests. At the end of serving T requests, we check whether the obtained solution proportionally satisfies the constraints of IP (18). If yes, then we declare YES as the answer and declare NO otherwise. At the core of the solution is the online sampling algorithm we use, which is identical to the techniques used to develop the online algorithm in Sections 3.2 and 3.3. We describe our algorithm in Algorithm 5.

The main theorem of this section is Theorem 2.5, which we restate here: **Theorem 2.5** For any $\epsilon > 0$, Algorithm 5 solves the gap version of the mixed covering-packing problem with $\Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ oracle calls.

Detailed Description and Proof. The proof is in two parts. The first part proves that our algorithm indeed answers YES when the actual answer is YES with a probability at least $1-\delta$. The second part is the identical statement for the NO case.

The YES Case. We begin with the case where the true answer is YES. Let x^*_{jk} denote some feasible solution to the LP relaxation of IP (18). In a spirit similar to that of Sections 3.1, 3.2, and 3.3, we define the algorithm P as follows. It samples a total of $T = \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ requests uniformly at random, with replacement, from the total pool of m requests. When request j is sampled, P serves

ALGORITHM 5: Online sampling algorithm for offline mixed covering-packing problems

Input: The mixed packing and covering IP (18), failure probability $\delta > 0$, and an error parameter $\epsilon > 0$.

Output: Distinguish between the cases "YES" where there is a feasible solution to IP (18), and "NO" where there is no feasible solution to IP (18) even if all the c_i's are multiplied by $1 + 3\epsilon(1+\epsilon)$ and all of the d_i's are multiplied by $1 - 3\epsilon(1+\epsilon)$.

```
1: Set $T = \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$
2: Initialize $\phi_{i,0} = \frac{1}{c_i}\left[\frac{\left(1+\frac{\epsilon}{m\gamma}\right)^{T-1}}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}}\right]$, and, $\psi_{i,0} = \frac{1}{d_i}\left[\frac{\left(1-\frac{\epsilon}{m\gamma}\right)^{T-1}}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}\right]$
3: for $s = 1$ to $T$ do
4:     Sample a request $j$ uniformly at random from the total pool of $m$ requests
5:     If the incoming request is $j$, then use the following option $k^*$:
           $$k^* = \arg\min_{k \in \mathcal{K}\cup\{\perp\}} \left\{\sum_i a_{ijk} \cdot \phi_{i,s-1} - \sum_i w_{ijk} \cdot \psi_{i,s-1}\right\}.$$
6:     $X^A_{i,s} = a_{ijk^*}$, $Y^A_{i,s} = w_{ijk^*}$
7:     Update $\phi_{i,s} = \phi_{i,s-1} \cdot \left[\frac{(1+\epsilon)^{\frac{X^A_{i,s}}{\gamma c_i}}}{1+\frac{\epsilon}{m\gamma}}\right]$, and, $\psi_{i,s} = \psi_{i,s-1} \cdot \left[\frac{(1-\epsilon)^{\frac{Y^A_{i,s}}{\gamma d_i}}}{1-\frac{\epsilon}{m\gamma}}\right]$
8: end for
9: if $\forall i \sum_{t=1}^T X^A_{i,t} < \frac{Tc_i}{m}(1+\epsilon)$, and, $\sum_{t=1}^T Y^A_{i,t} > \frac{Td_i}{m}(1-\epsilon)$ then
10:     Declare YES
11: else
12:     Declare NO
13: end if
```

j using option k with probability x^*_{jk}. Thus, if we denote by $X^*_{i,t}$ the consumption of resource i in step t of P, then we have $\mathbf{E}[X^*_{i,t}] = \sum_{j=1}^m \frac{1}{m}\sum_k a_{ijk}x^*_{jk} \le \frac{c_i}{m}$. This inequality follows from x^*_{jk} being a feasible solution to LP relaxation of Equation (18). Similarly, let $Y^*_{i,t}$ denote the resource i profit in step t of P. We have $\mathbf{E}[Y^*_{i,t}] \ge \frac{d_i}{m}$. We now write the probability that our condition for YES is violated for some algorithm A,

$$\begin{aligned}\mathbf{Pr}\left[\sum_{t=1}^T X^A_{i,t} \ge \frac{Tc_i}{m}(1+\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^T X^A_{i,t}}{\gamma c_i} \ge \frac{T}{m\gamma}(1+\epsilon)\right] \\ &= \mathbf{Pr}\left[(1+\epsilon)^{\frac{\sum_{t=1}^T X^A_{i,t}}{\gamma c_i}} \ge (1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}\right] \\ &\le \mathbf{E}\left[(1+\epsilon)^{\frac{\sum_{t=1}^T X^A_{i,t}}{\gamma c_i}}\right] / (1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}} \\ &= \frac{\mathbf{E}\left[\prod_{t=1}^T (1+\epsilon)^{\frac{X^A_{i,t}}{\gamma c_i}}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} \end{aligned} \quad (19)$$

$$\begin{aligned}\mathbf{Pr}\left[\sum_{t=1}^{T} Y_{i,t}^A \le \frac{Td_i}{m}(1-\epsilon)\right] &= \mathbf{Pr}\left[\frac{\sum_{t=1}^{T} Y_{i,t}^A}{\gamma d_i} \ge \frac{T}{m\gamma}(1-\epsilon)\right]\\ &= \mathbf{Pr}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{T} Y_{i,t}^A}{\gamma d_i}} \ge (1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}\right]\\ &\le \mathbf{E}\left[(1-\epsilon)^{\frac{\sum_{t=1}^{T} Y_{i,t}^A}{\gamma d_i}}\right] / (1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}\\ &= \frac{\mathbf{E}\left[\prod_{t=1}^{T}(1-\epsilon)^{\frac{Y_{i,t}^A}{\gamma d_i}}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}. \end{aligned} \tag{20}$$

If our algorithm A was P (and, therefore, we can use $\mathbf{E}[X_{i,t}^*] \le \frac{c_i}{m}$ and $\mathbf{E}[Y_{i,t}^*] \ge \frac{d_i}{m}$), then the total failure probability in the YES case, which is the sum of Equations (19) and (20) for all the i's would have been at most δ if $T = \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$ for an appropriate constant inside Θ. The goal is to design an algorithm A that, unlike P, does not first solve LP relaxation of IP (18) and then use x_{jk}^*'s to allocate resources but allocates online and also obtains the same δ failure probability, just as we did in Sections 3.2 and 3.3. That is we want to show that the sum of Equations (19) and (20) over all i's is at most δ:

$$\frac{\mathbf{E}\left[\prod_{t=1}^{T}(1+\epsilon)^{\frac{X_{i,t}^A}{\gamma c_i}}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} + \frac{\mathbf{E}\left[\prod_{t=1}^{T}(1-\epsilon)^{\frac{Y_{i,t}^A}{\gamma d_i}}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}} \le \delta.$$

For the algorithm $A^s P^{T-s}$, the above quantity can be rewritten as

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \prod_{t=s+1}^{T}(1+\epsilon)^{\frac{X_{i,t}^*}{\gamma c_i}}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma d_i}} \prod_{t=s+1}^{T}(1-\epsilon)^{\frac{Y_{i,t}^*}{\gamma d_i}}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}},$$

which, by using $(1+\epsilon)^x \le 1+\epsilon x$ for $0 \le x \le 1$, is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \prod_{t=s+1}^{T}\left(1+\epsilon\frac{X_{i,t}^*}{\gamma c_i}\right)\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma d_i}} \prod_{t=s+1}^{T}\left(1-\epsilon\frac{Y_{i,t}^*}{\gamma d_i}\right)\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}.$$

Since for all t, the random variables $X_{i,t}^*, X_{i,t}^A, Y_{i,t}^*$, and $Y_{i,t}^A$ are all independent, and $\mathbf{E}[X_{i,t}^*] \le \frac{c_i}{m}$, $\mathbf{E}[Y_{i,t}^*] \ge \frac{d_i}{m}$, the above is in turn upper bounded by

$$\sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}} \left(1+\frac{\epsilon}{m\gamma}\right)^{T-s}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma d_i}} \left(1-\frac{\epsilon}{m\gamma}\right)^{T-s}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}. \tag{21}$$

Let $\mathcal{F}[A^s P^{T-s}]$ denote the quantity in Equation (21), which is an upper bound on failure probability of the hybrid algorithm $A^s P^{T-s}$. We just said that $\mathcal{F}[P^T] \le \delta$. We now prove that for all $s \in \{0, 1, \ldots, T-1\}$, $\mathcal{F}[A^{s+1}P^{T-s-1}] \le \mathcal{F}[A^s P^{T-s}]$, thus proving that $\mathcal{F}[A^T] \le \delta$, i.e., running the algorithm A for all the T steps of stage r results in a failure with probability at most δ.

Assuming that for all $s < p$, the algorithm A has been defined for the first $s+1$ steps in such a way that $\mathcal{F}[A^{s+1}P^{T-s-1}] \le \mathcal{F}[A^s P^{T-s}]$, we now define A for the $p+1$th step of stage r in a way that will ensure that $\mathcal{F}[A^{p+1}P^{T-p-1}] \leq \mathcal{F}[A^pP^{T-p}]$. We have

$$\begin{aligned}\mathcal{F}[A^{p+1}P^{m-p-1}] &= \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_{p+1}(X_i^A)}{\gamma c_i}}\left(1+\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_{p+1}(Y_i^A)}{\gamma d_i}}\left(1-\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}} \\ &\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}}\left(1+\epsilon\frac{X_{i,p+1}^A}{\gamma c_i}\right)\left(1+\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p(Y_i^A)}{\gamma d_i}}\left(1-\epsilon\frac{Y_{i,p+1}^A}{\gamma d_i}\right)\left(1-\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}. \end{aligned} \tag{22}$$

Define

$$\phi_{i,s} = \frac{1}{c_i}\left[\frac{(1+\epsilon)^{\frac{S_s(X_i^A)}{\gamma c_i}}\left(1+\frac{\epsilon}{m\gamma}\right)^{T-s-1}}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}}\right]; \qquad \psi_{i,s} = \frac{1}{d_i}\left[\frac{(1-\epsilon)^{\frac{S_s(Y_i^A)}{\gamma d_i}}\left(1-\frac{\epsilon}{m\gamma}\right)^{T-s-1}}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}\right].$$

Define the step $p+1$ of algorithm A as picking the following option k for request j:

$$k^* = \arg\min_{k\in K\cup\{\perp\}}\left\{\sum_i a_{ijk}\cdot\phi_{i,p} - \sum_i w_{ijk}\cdot\psi_{i,p}\right\}.$$

By the above definition of step $p+1$ of algorithm A, it follows that for any two algorithms with the first p steps being identical, and the last $T-p-1$ steps following the Hypothetical-Oblivious algorithm P, algorithm A's $p+1$th step is the one that minimizes expression (22). In particular, it follows that expression (22) is upper bounded by the same expression where the $p+1$thstep is according to $X_{i,p+1}^*$ and $Y_{i,p+1}^*$, i.e., we replace $X_{i,p+1}^A$ by $X_{i,p+1}^*$ and $Y_{i,p+1}^A$ by $Y_{i,p+1}^*$. Therefore, we have

$$\begin{aligned}\mathcal{F}[A^{p+1}P^{T-p-1}] &\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}}\left(1+\epsilon\frac{X_{i,p+1}^*}{\gamma c_i}\right)\left(1+\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}} \\ &\quad + \sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p(Y_i^A)}{\gamma d_i}}\left(1-\epsilon\frac{Y_{i,T+p+1}^*}{\gamma d_i}\right)\left(1-\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}\end{aligned}$$

$$\leq \sum_i \frac{\mathbf{E}\left[(1+\epsilon)^{\frac{S_p(X_i^A)}{\gamma c_i}}\left(1+\frac{\epsilon}{m\gamma}\right)\left(1+\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1+\epsilon)^{(1+\epsilon)\frac{T}{m\gamma}}}$$
$$+\sum_i \frac{\mathbf{E}\left[(1-\epsilon)^{\frac{S_p(Y_i^A)}{\gamma d_i}}\left(1-\frac{\epsilon}{m\gamma}\right)\left(1-\frac{\epsilon}{m\gamma}\right)^{T-p-1}\right]}{(1-\epsilon)^{(1-\epsilon)\frac{T}{m\gamma}}}$$
$$= \mathcal{F}[A^p P^{T-p}].$$

The NO Case. We now proceed to prove that when the real answer is NO, our algorithm says NO with a probability at least $1-\delta$. To prove this result (formally stated in Lemma 6.3), we use as a tool the fact that when the integer program in Equation (18) is in the NO case where even a slack of $3\epsilon(1+\epsilon)$ will not make it feasible, then even the LP relaxation of Equation (18) will be infeasible with a slack of 2ϵ. We prove this statement now by proving its contrapositive in Lemma 6.1.

Lemma 6.1. *If the LP relaxation of (18) is feasible with a slack of s, then the integer program in Equation (18) is feasible with a slack of $s(1+\epsilon)+\epsilon$.*

Proof. To prove this, we write the LP relaxation of the integer program in Equation (18) slightly differently below.

Primal and dual LPs corresponding to integer program in Equation (18) (23)

Primal LP corresponding to IP (18)	**Dual LP corresponding to IP (18)**
Minimize λ s.t.	Maximize $\sum_i(\rho_i-\alpha_i)-\sum_j\beta_j$ s.t.
$\forall\, i, \lambda-\sum_{j,k}\frac{a_{ijk}x_{j,k}}{c_i}\geq -1$	$\forall\, j,k,\ \beta_j\geq\sum_i\left(\rho_i\frac{w_{ijk}}{d_i}-\alpha_i\frac{a_{ijk}}{c_i}\right)$
$\forall\, i, \lambda+\sum_{j,k}\frac{w_{ijk}x_{j,k}}{d_i}\geq 1$	$\sum_i(\alpha_i+\rho_i)\leq 1$
$\forall\, j, \sum_k x_{j,k}\leq 1$	$\forall\, i, \alpha_i,\rho_i\geq 0$
$\forall\, j,k, x_{j,k}\geq 0$	$\forall\, j, \beta_j\geq 0.$
$\lambda\geq 0$	

The optimal value λ^* of the primal LP in Equation (23) represents the slack in the YES/NO problem, i.e., if $\lambda^*=0$, then we have zero slack and hence are in the YES case. Else, we are in the NO case with a slack equal to λ^*. Given this, all we have to show is that when the LP in Equation (23) has an optimal value of λ^*, then the corresponding integer program's optimal solution is at most $\lambda^*(1+\epsilon)+\epsilon$. To see this is true, let $x^*_{j,k}$ denote the optimal solution to primal LP (23). Consider the integral solution that does a randomized rounding of the $x^*_{j,k}$'s and allocates according to these rounded integers, and let X_{jk} be the corresponding $\{0,1\}$ random variable. Let random variable $X_{ij}=\sum_k\frac{a_{ijk}X_{jk}}{c_i}$. By the definition of LP (23), we have $\mathbf{E}[\sum_j X_{ij}]\leq 1+\lambda^*$. Noting that each of the X_{ij}'s is at most γ, by Chernoff bounds it follows that $\mathbf{Pr}[\sum_j X_{ij}\geq(1+\lambda^*)(1+\epsilon)]\leq e^{-\frac{\epsilon^2}{4\gamma}}$, which given that $\gamma=O(\frac{\epsilon^2}{\log(n/\epsilon)})$, is at most $\frac{\epsilon}{2n}$ (the probability derivation is just like the derivation in Section 3.1). Likewise, if we define by Y_{ij} the random variable $\sum_k\frac{w_{ijk}X_{jk}}{d_i}$, then by the definition of LP (23), we have $\mathbf{E}[\sum_j Y_{ij}] \geq 1 - \lambda^*$. By Chernoff bounds, we get an identical argument, we get $\mathbf{Pr}[\sum_j Y_{ij} \leq (1-\lambda^*)(1-\epsilon)] \leq \frac{\epsilon}{2n}$. By doing a union bound over the $2n$ Chernoff bounds, we have that the randomized rounding integer solution is feasible with an optimal value of at most $\lambda^* + \epsilon + \lambda^*\epsilon$ with probability at least $1-\epsilon$. This means that there exists an integer solution of value $\lambda^* + \epsilon + \lambda^*\epsilon$, and this proves the lemma. □

Corollary 6.2. *For a* NO *instance with a slack of* $3\epsilon(1+\epsilon)$, *the LP relaxation of the instance is still infeasible with a slack of* 2ϵ. *In particular, this implies that the optimal value of the primal* λ^* *in Equation (23) is at least* 2ϵ, *and likewise the optimal dual value in Equation (23) is* $\sum_i(\rho_i^* - \alpha_i^*) - \sum_j \beta_j^* \geq 2\epsilon$.

Lemma 6.3. *For a* NO *instance, if* $T \geq \Theta(\frac{\gamma m \log(n/\delta)}{\epsilon^2})$, *then*

$$\mathbf{Pr}\left[\max_i \frac{S_T\left(X_i^A\right)}{c_i} < \frac{T}{m}(1+\epsilon) \ \&\ \min_i \frac{S_T\left(Y_i^A\right)}{d_i} > \frac{T}{m}(1-\epsilon)\right] \leq \delta.$$

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

Sampled primal and dual LPs (24)

Sampled primal LP	**Sampled dual LP**
Minimize λ s.t.	Maximize $\frac{T}{m}\sum_i(\rho_i - \alpha_i) - \sum_{j\in R}\beta_j$ s.t.
$\forall\, i, \lambda - \sum_{j\in R,k} \frac{a_{ijk}x_{j,k}}{c_i} \geq -\frac{T}{m}$	$\forall\, j \in R, k,\ \beta_j \geq \sum_i \left(\rho_i \frac{w_{ijk}}{d_i} - \alpha_i \frac{a_{ijk}}{c_i}\right)$
$\forall\, i, \lambda + \sum_{j\in R,k} \frac{w_{ijk}x_{j,k}}{d_i} \geq \frac{T}{m}$	$\sum_i(\alpha_i + \rho_i) \leq 1$
$\forall\, j \in R, \sum_k x_{j,k} \leq 1$	$\forall\, i, \alpha_i, \rho_i \geq 0$
$\forall\, j, k, x_{j,k} \geq 0$	$\forall\, j \in R, \beta_j \geq 0.$
$\lambda \geq 0$	

If the primal in LP (24) has an optimal objective value at least $\frac{T\epsilon}{m}$, then by definition of our Algorithm 5, we would have declared NO, i.e., if the sampled LP itself had a slack of ϵ (scaled by $\frac{T}{m}$), then no integral allocation based on those samples can obtain a smaller slack. We now show that by picking $T = \Theta(\frac{\gamma m \ln(n/\delta)}{\epsilon^2})$, the above LP (24) will have its optimal objective value at least $\frac{T\epsilon}{m}$, with a probability at least $1-\delta$. This makes our algorithm answer NO with a probability at least $1-\delta$.

Now, the primal of LP (24) has an optimal value equal to that of the dual, which in turn is lower bounded by the value of dual at any feasible solution. One such feasible solution is $\alpha^*, \beta^*, \rho^*$, which is the optimal solution to the full version of the dual in LP (24), namely the one written in LP (23), where $R = [m], T = m$. This is because the set of constraints in the full version of the dual is clearly a superset of the constraints in the dual of LP (24). Thus, the optimal value of the primal of LP (24) is lower bounded by value of dual at $\alpha^*, \beta^*, \rho^*$, which is

$$= \frac{T}{m}\left(\sum_i \rho_i^* - \alpha_i^*\right) - \sum_{j\in R}\beta_j^*. \tag{25}$$

For proceeding further in lower bounding (25), we apply Chernoff bounds to $\sum_{j\in R}\beta_j^*$. The fact that the dual of the full version of LP (24) is a maximization LP, coupled with the constraints there in imply that $\beta_j^* \le \gamma$. Further, let τ^* denote the optimal value of the full version of LP (24), i.e., $\sum_i(\rho_i^* - \alpha_i^*) - \sum_j \beta_j^* = \tau^*$. Now, the constraint $\sum_i(\alpha_i^* + \rho_i^*) \le 1$ coupled with the fact that $\tau^* \ge 0$ implies $\sum_j \beta_j^* \le 1$. We are now ready to lower bound the quantity in Equation (25). We have the optimal solution to primal of LP (24)

$$\begin{aligned}
&\ge \frac{T}{m}\left(\sum_i \rho_i^* - \alpha_i^*\right) - \sum_{j\in R}\beta_j^* \\
&\ge \frac{T}{m}\sum_i(\rho_i^* - \alpha_i^*) - \left(\frac{T\sum_j\beta_j^*}{m} + \sqrt{\frac{4T(\sum_j\beta_j^*)\gamma\ln(1/\delta)}{m}}\right) \quad \left(\text{Since } \beta_j^* \in [0,\gamma]\right) \\
&\ge \frac{T\tau^*}{m} - \sqrt{\frac{4T\gamma\ln(1/\delta)}{m}} \\
&= \frac{T\tau^*}{m}\left[1 - \sqrt{\frac{\gamma m\ln(1/\delta)}{T}\cdot\frac{4}{\tau^{*2}}}\right],
\end{aligned} \tag{26}$$

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 $\beta_j^* \in [0,\gamma]$. The third inequality follows from $\sum_j\beta_j^* \le 1$ and $\sum_i(\rho_i^* - \alpha_i^*) - \sum_j\beta_j^* = \tau^*$. Setting $T = \Theta(\frac{\gamma m\ln(n/\delta)}{\epsilon^2})$ with an appropriate constant inside the Θ, coupled with the fact that $\tau^* \ge 2\epsilon$ in the NO case (see Corollary 6.2), it is easy to verify that the quantity in Equation (26) is at least $\frac{T\epsilon}{m}$.

Going back to our application of Chernoff bounds above, to apply it in the form above, we require that the multiplicative deviation from mean $\sqrt{\frac{4\gamma m\ln(1/\delta)}{T\sum_j\beta_j^*}} \in [0, 2e-1]$. If $\sum_j\beta_j^* \ge \epsilon^2$, then this requirement would follow. Suppose, however, that $\sum_j\beta_j^* < \epsilon^2$. Since we are happy if the excess over mean is at most $\frac{T\epsilon}{m}$, let us look for a multiplicative error of $\frac{\frac{T\epsilon}{m}}{\frac{T\sum_j\beta_j^*}{m}}$. Based on the fact that $\sum_j\beta_j^* < \epsilon^2$ the multiplicative error can be seen to be at least $1/\epsilon$ that is larger than $2e-1$ when $\epsilon < \frac{1}{2e-1}$. We now use the version of Chernoff bounds for multiplicative error larger than $2e-1$, which gives us that a deviation of $\frac{T\epsilon}{m}$ occurs with a probability at most $2^{-(1+\frac{\frac{T\epsilon}{m}}{\frac{T\sum_j\beta_j^*}{m}})\frac{T\sum_j\beta_j^*}{m\gamma}}$, where the division by γ is because of the fact that $\beta_j^* \le \gamma$. This probability is at most $(\frac{\delta}{n})^{1/\epsilon}$, which is at most δ. □

The proofs for the YES and NO cases together prove Theorem 2.5.

7 SPECIAL CASES OF THE RESOURCE ALLOCATION FRAMEWORK

We now list the problems that are special cases of the resource allocation framework and have been previously considered. The Adwords and Display ads special cases were already discussed in Section 2.2.

7.1 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. 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 *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). Our Algorithm 2 for resource allocation framework directly applies here and the approximation guarantee there directly carries over. Kamath et al. [16] consider a different version of this problem where requests according to a Poisson process with unknown arrival rates. Each request has an associated *holding time* that is assumed to be exponentially distributed, and once a request has been served, the bandwidth it uses up gets freed after its holding time (this is unlike our setting where once a certain amount of resource capacity has been consumed, it remains unavailable to all future requests). Some aspects of the distribution are assumed to be known, namely that the algorithm knows the average rate of profit generated by all the incoming circuits, the average holding time, and also the target offline optimal offline solution that the online algorithm is aiming to approximate (again this is unlike our setting where no aspect of the request distribution is known to the algorithm, and there could even be some adversarial aspects like in the ASI model). When each request consumes at most γ fraction of any edge's bandwidth, Kamath et al. [16] give an online algorithm that achieves an expected profit of $(1-\epsilon)$ times the optimal offline solution when $\gamma = O(\frac{\epsilon^2}{\log n})$.

7.2 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 $U_j : 2^{[n]} \to \mathbf{R}$. 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. First, 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 s will choose the option k^* (or, equivalently, the bundle S) as specified in point 8 of Algorithm 2, i.e., minimize the potential function. That is, if step s falls in stage r, then

$$k^* = \arg\min_k \left\{ \sum_i a_{ijk} \cdot \phi^r_{i,s-1} - w_{j,k} \cdot \psi^r_{s-1} \right\}$$

(note that unlike Algorithm 2 there is no subscripting for $w_{j,k}$). This can be equivalently written as

$$k^* = \arg\max_k \left\{ w_{j,k} \cdot \psi^r_{s-1} - \sum_i a_{ijk} \cdot \phi^r_{i,s-1} \right\}.$$

Now, maximizing the above expression at step s is the same as picking the k to maximize $w_{j,k} - \sum_i p_i(s) a_{ijk}$, where $p_i(s) = \frac{\phi^r_{i,s-1}}{\psi^r_{s-1}}$. Thus, if we post a price of $p_i(s)$ on item i for bidder number s, then he 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). We can use Theorem 2.2 to get an incentive compatible posted price auction[9] with a competitive ratio of $1 - O(\epsilon)$ whenever $\gamma = \min_i\{c_i\} \geq \Omega(\frac{\log(n/\epsilon)}{\epsilon^2})$. Further, if an analog of Theorem 2.2 also

[9]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.

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.

REFERENCES

[1] Shipra Agrawal and Nikhil R. Devanur. 2015. Fast algorithms for online stochastic convex programming. In *Proceedings of the Symposium on Discrete Algorithms (SODA'15).*

[2] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. 2014. A dynamic near-optimal algorithm for online linear programming. *Operat. Res.* 62, 4 (2014), 876–890.

[3] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. 2012. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the ACM Conference on Electronic Commerce.* 18–35.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2005. *The Multiplicative Weights Update Method: A Meta Algorithm and Applications.* Technical Report.

[5] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Conference on Algorithms (ESA'07).* Springer-Verlag, Berlin, 253–264.

[6] Denis Xavier Charles, Max Chickering, Nikhil R. Devanur, Kamal Jain, and Manan Sanghi. 2010. Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads. In *Proceedings of the ACM Conference on Electronic Commerce.* 121–128.

[7] Nikhil R. Devanur and Thomas P. Hayes. 2009. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC'09).* 71–78.

[8] Nikhil R. Devanur, Balasubramanian Sivan, and Yossi Azar. 2012. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the ACM Conference on Electronic Commerce.* 388–404.

[9] R. Eghbali, J. Swenson, and M. Fazel. 2014. Exponentiated subgradient algorithm for online optimization under the random permutation model. *ArXiv e-prints*1410.7171 (Oct. 2014).

[10] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. 2010. Online stochastic packing applied to display ad allocation. In *Proceedings of the 18th Annual European Conference on Algorithms (ESA'10).* Springer, Berlin, 182–194.

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

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

[13] Gagan Goel and Aranyak Mehta. 2008. Online budgeted matching in random input models with applications to Adwords. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08).* Society for Industrial and Applied Mathematics, Philadelphia, PA, 982–991.

[14] Anupam Gupta and Marco Molinaro. 2014. How experts can solve LPs online. In *Proceedings of the 22nd Annual European Conference on Algorithms (ESA'14).* Springer, Berlin, 517–529.

[15] Bala Kalyanasundaram and Kirk Pruhs. 1996. An optimal deterministic algorithm for online b-matching. In *Foundations of Software Technology and Theoretical Computer Science*, V. Chandru and V. Vinay (Eds.). Lecture Notes in Computer Science, Vol. 1180. Springer, Berlin, 193–199. DOI: https://doi.org/10.1007/3-540-62034-6_49

[16] Anil Kamath, Omri Palmon, and Serge Plotkin. 1996. Routing and admission control in general topology networks with Poisson arrivals. In *Proceedings of the S7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96).* Society for Industrial and Applied Mathematics, Philadelphia, PA, 269–278.

[17] Michael Kapralov, Ian Post, and Jan Vondrák. 2013. Online submodular welfare maximization: Greedy is optimal. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13).* 1216–1225.

[18] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. 2014. Primal beats dual on online packing LPs in the random-order model. In *Proceedings of the Symposium on Theory of Computing (STOC'14).* 303–312.

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

[20] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. 1991. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (SFCS'91).* IEEE Computer Society, Washington, DC, 495–504. DOI: https://doi.org/10.1109/SFCS.1991.185411

[21] Ester Samuel-Cahn. 1984. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *Ann. Probabil.* 12, 4 (1984), 1213–1216.

[22] Neal E. Young. 1995. Randomized rounding without solving the linear program. In *Proceedings of the S6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 170–178.

[23] Neal E. Young. 2001. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*. IEEE Computer Society, Washington, DC, 538–547.

Received August 2015; revised March 2018; accepted September 2018