Online Resource Allocation with Matching Constraints[*]

John P. Dickerson
University of Maryland, College Park
College Park, MD, USA
john@cs.umd.edu

Karthik Abinav Sankararaman
University of Maryland, College Park
College Park, MD, USA
kabinav@cs.umd.edu

Kanthi Kiran Sarpatwar
IBM Research AI
Yorktown Heights, NY, USA
sarpatwa@us.ibm.com

Aravind Srinivasan
University of Maryland, College Park
College Park, MD, USA
srin@cs.umd.edu

Kun-Lung Wu
IBM Research AI
Yorktown Heights, NY, USA
klwu@us.ibm.com

Pan Xu
University of Maryland, College Park
College Park, MD, USA
panxu@cs.umd.edu

ABSTRACT

Matching markets with historical data are abundant in many applications, *e.g.,* matching candidates to jobs in hiring, workers to tasks in crowdsourcing markets, and jobs to servers in cloud services. In all these applications, a match consumes one or more *shared* and *limited* resources and the goal is to best utilize these to maximize a global objective. Additionally, one often has historical data and hence some statistics (usually first-order moments) of the arriving agents (*e.g.,* candidates, workers, and jobs) can be learnt. To model these scenarios, we propose a unifying framework, called *Multi-Budgeted Online Assignment with Known Adversarial Distributions*. In this model, we have a set of *offline* servers with different deadlines and a set of *online* job types. At each time, a job of type j arrives. Assigning this job to a server i yields a profit $w_{i,j}$ while consuming $\mathbf{a}_e \in [0,1]^K$ quantities of distinct resources. The goal is to design an (online) assignment policy that maximizes the total expected profit without violating the (hard) budget constraint. We propose and theoretically analyze two linear programming (LP) based algorithms which are almost optimal among all LP-based approaches. We also propose several heuristics adapted from our algorithms and compare them to other LP-agnostic algorithms using both synthetic as well as real-time cloud scheduling and public safety datasets. Experimental results show that our proposed algorithms are effective and *significantly* out-perform the baselines. Moreover, we show empirically the trade-off between *fairness* and *efficiency* of our algorithms which does well even on fairness metrics without explicitly optimizing for it.

[*]Part of this work is done when Pan Xu was an intern at the IBM T. J. Watson Research Center during the summer of 2016. Aravind Srinivasan's research was supported in part by NSF Awards CNS-1010789, CCF-1422569 and CCF-1749864, and by research awards from Adobe, Inc. Karthik Sankararaman's research was supported in part by NSF Awards CNS-1010789 and CCF-1422569. John Dickerson's research was supported by NSF IIS RI CAREER Award #1846237. Pan Xu's research was supported by NSF Awards CNS-1010789, CCF-1422569 and NSF IIS RI CAREER Award #1846237. The authors also like to thank Google for a generous gift support. We thank Aditya Parameswaran and Xingjie Liu for useful discussions on datasets related to our problem.

CCS CONCEPTS

• **Theory of computation** → **Scheduling algorithms**; **Packing and covering problems**; **Stochastic approximation**; **Online algorithms**; • **Mathematics of computing** → *Matchings and factors*;

KEYWORDS

Online Scheduling, Online Matching, Randomized Algorithms, Fairness

ACM Reference Format:
John P. Dickerson, Karthik Abinav Sankararaman, Kanthi Kiran Sarpatwar, Aravind Srinivasan, Kun-Lung Wu, and Pan Xu. 2019. Online Resource Allocation with Matching Constraints. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Large-scale matching markets are abundant in many modern applications. A canonical example is the online advertising market, which is the main source of revenue for internet companies like Google. Online bipartite matching models and their variants provide mathematical insight into the design and analysis of these ubiquitous markets. In the basic version, we are given a bipartite graph $G = (U, V, E)$ where U and V represent sets of advertisers and keywords, respectively. There is an edge $e = (u, v)$ if and only if the advertisement of u is relevant to a keyword v. Keywords arrive one-by-one in an online manner and must be matched to a potential advertiser *immediately and irrevocably.* Matching a keyword v to an advertiser u gives a profit of $w_{u,v}$.

However, the above abstraction for online advertising can be used to model the more general *assignment* problem in various emerging applications, ranging from crowdsourcing marketplaces (*e.g.,* Amazon Mechanical Turk [49], matching online workers to offline tasks), ride-sharing platforms (*e.g.,* Uber, matching online requests to drivers [40]), to assignment of jobs to servers in cloud services [27]. There are several other applications of online matching models in advance admission scheduling and online recommendations (*e.g.,* matching online users to service providers or offline products [17, 42, 52]). These problems can be abstracted as a variant of online bipartite matching where (1) there are two sets of agents with at least one coming online; in any online step an immediate and irrevocable decision has to be made; (2) there is a set of offline (limited) resources with each having a given total budget; every match consumes a subset of these resources.

Assadi et al. [9] considered the Online Task Assignment (OTA) problem arising in crowdsourcing marketplaces. They assumed a global budget on a single resource. Each match of an online worker to an offline task will require a payment to the worker. The goal is to design an online matching policy such that the expected number of tasks completed is maximized without violating the budget constraint. Ho and Vaughan [32] studied a capacitated OTA where every task has several copies and thus can be matched multiple times. Hence the number of copies of each task is an offline resource. Huang et al. [33], Ma et al. [40], Tong et al. [51] considered OTA emerging in the real-time spatial crowdsourcing platforms (*e.g.,* Grubhub in the online food-ordering business). In this context, we can assign multiple online orders to a single worker where each worker has two kinds of budgets: the number of orders they can handle in each trip and the total working hours and/or travel distance over all trips.

In this paper, we propose a unifying framework to handle the various budget constraints in the above applications. Additionally, we consider a realistic arrival assumption inspired from real datasets; these help us get much better provable performance. The following are the two distinctive features in the model.

Multi-Budgeted Constraints. We have a set of K resources with each resource having a known total budget. Each online match (or assignment) is associated with a vector-valued cost of dimension K with the k^{th} element denoting the amount of resource k the match consumes. We call a resource integral if and only if the value consumed by all possible matches is integral (*e.g.,* number of sub-jobs); otherwise we call it non-integral (*e.g.,* the total running time).

Known Adversarial Distributions. Common assumptions on the arrival sequence include Adversarial Order (AO) where the arrival sequence is fixed by an adversary (*e.g.,* [9]), Random Arrival Order (RAO) where the arrival sequence forms a random permutation over the set of online agents (unknown but fixed) (*e.g.,* [50, 56]) and Known Independent and Identical Distribution (KIID) where online agents present themselves, in every time-step, as a sample (with replacement) from a known and identical distribution (*e.g.,* [24, 47, 48]). In this paper, we consider a generalization of KIID, called Known Adversarial Distributions (KAD), where the arrival distributions are allowed to change over time [23]. We motivate KIID and its generalization KAD as follows. In practice, allocation algorithms are implemented in episodes. We have L episodes (where an episode could last a few hours as in cloud platforms to a day as in ride-sharing). Within each episode, algorithms use the information from the past episodes to "learn" the arrival patterns which are then used as an estimate for the current episode. This model has a two-fold challenge; first is to learn the patterns across episodes and the second is to have an efficient allocation mechanism within an episode (which is the focus of this paper).

We now give a few concrete examples and show how our model can be used to capture these with experiments on real world datasets in the experiments section.

Resource allocation in datacenters. In modern data centers, one of the challenges is to allocate various resources such as CPU, memory, clusters, to various heterogenous tasks with different requirements. The tasks usually fall into broad *categories* with very structured demands for various resources [36]. A number of recent works in the systems literature have empirically studied both efficient allocation as well fair division of resources [20, 26, 27, 35]. Our model can efficiently capture this multi-resource setting where we have multiple shared resources with high *sparsity* (*e.g.,* every computer is associated with its CPU, while the set of resources are CPU's for all computers). The tasks arrive online and should be allocated to a machine immediately and irrevocably. We look at a small time-span of one-two minutes where multiple tasks are run simultaneously on a machine with *hard* constraints on limited resources (*i.e.,* every machine has a finite amount of CPU and memory to simultaneously be utilized). The goal is to maximize the number of tasks performed and/or to drop as few requests as possible.

Public safety. Law enforcement in a given city or region is an important task for any government. The challenge is to allocate limited resources such as cops, vehicles, breath analyzers, etc. to various regions where potential violations can occur while reducing the response time and maximizing the efficiency (see, *e.g.,* [16, 28, 39, 46] for some work in this area). This general problem can be captured as an online resource allocation problem and naturally fits in our model. We have a set of offline vertices which corresponds to patrol team with multiple resources (*e.g.,* number of cops, type of vehicle, chase capabilities) stationed at various locations. When a potential violation occurs, it has to immediately be matched to a certain patrol team where the match fetches a *reward* proportional to the nature of the violation.

Hiring candidates to jobs. Consider the scenario where an HR division wants to hire new employees for a set of job positions (see, *e.g.,* [19, 54]). In practice they can hire at most one or two new employees per post (due to a total budget) and need to train each new employee to acquire the other skills needed. The training process demands several kinds of related resources such as experienced staff, time, money (paid to trainers), machines, to name a few. Suppose we have a finite budget allotted for each resource and every successful hire fetches a monetary reward to the HR. Then the problem facing the HR division can be precisely captured by our model: each match of a candidate j to a post i will consume various related resources due to the training process for j to acquire skills required by i while absent from j.

Our contributions. We make several contributions in this paper. First, we propose a general model that effectively captures the online resource allocation problems in various matching markets with historical data. Our model exploits the fact that historical data can be used to learn the arrival distributions of online agents at various times (*e.g.,* [19] in case of hiring). Second, we present algorithms that are provably correct and yield improved performance over models where distributions are assumed to be unknown. In particular, we first consider the simple case when only integral resources are involved with sparse resource consumption. We show

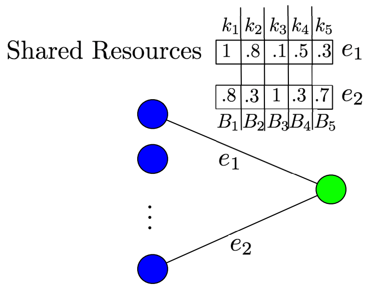


Figure 1: Five shared resources with budgets $B_1, B_2, \ldots, B_5$**.**

that in this case, our algorithms are near-optimal among all LP-based algorithms. Next we consider the general case when both integral and non-integral resources are involved. We show that to achieve a target competitive ratio, our model admits algorithms which have significantly improved lower-bound requirements over the budgets of non-integral resources, compared to previous ones when arrival distributions are unknown. In particular, our results completely eliminate the dependence on the ratio of largest to the smallest bid [9]. This is crucial since this ratio can typically be very large if the bids are non-uniform. Third, we consider a special case when each assignment consumes a single non-integral resource with no assumptions on its budget. We devise an algorithm with theoretical guarantees and also show hardness results. Finally, we give an empirical study of our algorithms and compare them with natural heuristics on real datasets to validate and complement our theoretical results. We also define two natural metrics for fairness for this setting and explore how efficiency maximizing algorithms perform on these fairness metrics.

2 PRELIMINARIES

We first formally define the model considered in this paper and then describe the required background for the technical sections of this paper. As a notation, denote $[k] := \{1, 2, \ldots, k\}$ for any positive integer k.

Multi-Budgeted Online Assignment (MBOA-KAD**).** Let $I = \{i \in [m]\}$ be the set of (*offline*) servers, $J = \{j \in [n]\}$ be the set of types of (*online*) jobs and T be the time horizon. Every server i has a (hard) time-out $d_i \in [T]$ after which it shuts down. Let $G = (I, J, E)$ be the bipartite graph with an edge $e = (i, j)$ iff job-type j can be run on server i. Let $N(j) = \{i : (i, j) \in E\}$ be the set of servers that can handle job-type j and $N(i) = \{j : (i, j) \in E\}$ be the set of job-types that can run on server i. Each edge $e = (i, j)$ has a weight w_e denoting the profit obtained by allocating server i to job-type j. Each assignment $e = (i, j)$ consumes one or more of a given set of K resources. The cost of an allocation e is given by a K-dimensional vector $\mathbf{a}_e \in [0, 1]^K$, where the k^{th} dimension $a_{e,k}$ represents the amount of resource k consumed by assignment e. Each resource k has a budget $B_k \in \mathbb{R}_+$ that must not be exceeded. For each e, let $S_e = \{k \in [K] : a_{e,k} > 0\}$, *i.e.,* the set of resources it consumes.

At any instant $t \in [T]$, a job of type j arrives with a probability p_{jt} such that $\sum_j p_{jt} \le 1$ (thus, with probability $1 - \sum_j p_{jt}$, no job arrives at time t). Let $E_{jt} = \{e = (i, j), i \in N(j) : d_i \ge t\}$ denote the set of *available* assignments (*i.e.,* the corresponding servers should be active at time t) for the job-type j at time t.[1] For each $e \in E_{jt}$, we say e is *safe or valid* iff for each $k \in S_e$, resource k has a remaining budget larger or equal to $a_{e,k}$. When a job of type j arrives at t, we have to make an immediate and irrevocable decision: either reject it or choose a safe option $e \in E_{jt}$ and get a resultant profit w_e. Once a safe assignment e is scheduled, the budget of each resource $k \in S_e$ will be reduced by $a_{e,k}$. Our goal is to design an online assignment policy such that the expected profit is maximized.

Note that all algorithms presented in this paper are applicable to a more general setting where each successful match e yields a *random* profit W_e (independent from others). All the algorithms need to know is $w_e \doteq \mathbb{E}[W_e]$ for each e [2].

The performance of online algorithms is usually measured using the notion of competitive ratio (see [13]). For our problem, we define the competitive ratio as follows.

Definition 2.1 (Competitive Ratio). Let ALG denote a given online algorithm whose performance we want to measure. Consider an instance $\mathcal{I}$ of the problem. Let $\mathbb{E}[\mathsf{ALG}(\mathcal{I})]$ denote the expected profit obtained by ALG for this instance (here expectation is over the randomness in the input as well as any randomness the algorithm uses). Similarly, let $\mathbb{E}[\mathsf{OPT}(\mathcal{I})]$ denote the expected value of the optimal offline solution (*i.e.,* the expected value of the optimal solution on seeing the entire arrival sequence). The competitive ratio is defined as $\inf_{\mathcal{I}} \mathbb{E}[\mathsf{ALG}(\mathcal{I})]/\mathbb{E}[\mathsf{OPT}(\mathcal{I})]$.

For any maximization problem like the one studied here, we say ALG achieves a ratio at least $\alpha \in (0, 1)$ if for any instance of the problem the expected profit obtained by ALG is at least a fraction α of the offline optimal solution. Typically computing the value of $\mathbb{E}[\mathsf{OPT}(\mathcal{I})]$ directly is hard. A common methodology to bypass this is to construct a linear program (called *benchmark* LP) whose optimal value is an upper bound on $\mathbb{E}[\mathsf{OPT}(\mathcal{I})]$. Hence comparing $\mathbb{E}[\mathsf{ALG}(\mathcal{I})]$ to the optimal value of this LP gives a lower bound on the competitive ratio. We will now describe the benchmark LP used in this paper.

Recall E_{jt} be the set of available assignments for a job of type j arriving at time-step t. For any t, let $E_t = \bigcup_j E_{jt}$ be the set of all possible assignments (a-priori before the execution of any online algorithm) at t. Further, for each t and $e \in E_t$, let $x_{e,t}$ be the probability that an assignment e is made at round t in an offline optimal algorithm. Then the benchmark LP we use is as follows.

$$\text{maximize } \sum_t \sum_j \sum_{e \in E_{j,t}} w_e x_{e,t} \tag{1}$$

$$\text{subject to } \sum_{e \in E_{jt}} x_{e,t} \le p_{jt} \qquad \forall j \in J, t \in [T] \tag{2}$$

$$\sum_t \sum_{e \in E_t} x_{e,t} a_{e,k} \le B_k \qquad \forall k \in [K] \tag{3}$$

$$0 \le x_{e,t} \le 1 \qquad \forall e \in E, t \in [T] \tag{4}$$

This LP can be interpreted as follows. Constraint (2) — for any given job of type j and time t, the probability that we assign a server to j is at most the probability that j arrives at step t. Constraint (3) — for any (integral or non-integral) resource k, the expected consumption cannot be larger than its budget (B_k). The last constraint (4)

[1]In this paper, we assume w.l.o.g. that each server can be allocated for an arbitrary number of times before its shutdown. Any potential restriction on the number of allocations can easily be modeled by an additional budget constraint.

[2]An allocation does not imply a completion of a job, hence this uncertainty can be handled in our model.

is due to the fact that all $\{x_{e,t}\}$ are probability values and hence should lie in the interval $[0,1]$. The above analysis suggests that any offline optimal solution $\{x_{e,t}\}$ should be feasible for the above LP. Formally, we have Lemma 2.2 which claims that the optimal solution of this LP is an upper bound on the expected offline optimal value.

Lemma 2.2. *The optimal value to* LP-(1) *is a valid upper bound for the offline optimal solution.*

The benchmark LP-(1) has previously been used in [7] and the proof of Lemma 2.2 can be found there.

Integral and non-integral resources. For an integral resource k, we have that for any $e \in E$, $a_{e,k} \in \{0,1\}$ while for a non-integral resource k, we have that for any $e \in E$, $a_{e,k} \in [0,1]$. For any integral resource k, WLOG we assume that $B_k \in \mathbb{Z}_+$. Let $\mathcal{K}_1 = \{1, 2, \cdots, K_1\}$ and $\mathcal{K}_2 = \{K_1+1, \cdots, K_1+K_2\}$ denote the set of integral and non-integral resources respectively. For any assignment e, we assume $|S_e \cap \mathcal{K}_1| \le \ell_1$ and $|S_e \cap \mathcal{K}_2| \le \ell_2$, where ℓ_1 and ℓ_2 are the integral and non-integral sparsity respectively.

Adaptive and non-adaptive algorithms. For an LP-based ALG, we say ALG is *non-adaptive* if for a given LP solution, the computation of strategy in each round t does not depend on the strategies in the previous rounds from $1, 2, \ldots, t-1$. Otherwise, we call it "adaptive". Here we distinguish "adaptive" and "non-adaptive" to highlight the computations of strategies in the "online" phase.

3 OTHER RELATED WORK

We now describe some related works other than those already described. Our problem falls under the *online packing* family of problems. Some representative works on this include [4, 6, 14, 15, 22, 37]. The most relevant to this paper is that of Devanur et al. [22] who study it in the unknown i.i.d. setting. We have the following important distinctions. First, our work assumes the known i.i.d. setting. Second, Devanur et al. [22] study this problem in the large budget regime (*i.e.,* $B \ge \Omega(\frac{1}{\epsilon^2})$) and obtain $1-\epsilon$ approximation. This paper however considers regimes where the lower bound on the budget is only $\Omega(\frac{1}{\log \epsilon})$. We circumvent the necessity for a large lower-bound on the budget is by assuming sparsity in the packing program. Closely related to the online packing problems literature is another line of work on a family of problems called *bandits with knapsacks* [2, 3, 5, 10, 11, 45]. These works consider the learning variant of the online packing problems and obtain approximation ratios that are comparable to [22]. Many special cases of online packing problems and bandits with knapsacks have been studied across communities including dynamic pricing ([21] and references within), network routing and optimization ([10] and references within), network revenue management ([12] and references within).

4 ALGORITHMS

In this section, we describe our two main algorithms NADAP and ADAP.

Non-adapative algorithm (NADAP)**.** Algorithm NADAP is a non-adaptive algorithm based on LP. Suppose $\{x^*_{e,t} | t \in [T], e \in E_t\}$ is an optimal solution to the LP-(1). The main idea behind NADAP (described in Algorithm 1) is as follows. Suppose a job of type j arrives at time t: sample a server i from E_{jt} with probability $\alpha x^*_{e,t}/p_{jt}$, where $\alpha \in (0,1]$ is a parameter optimized in the analysis. Make the assignment $e=(i,j)$ iff e is safe (*i.e.,* it will not violate any budget constraint at t).

ALGORITHM 1: The non-adaptive algorithm (NADAP)

```
For each time $t$, let the arriving job be denoted by $j$.
Let $\hat{E}_{jt} \subseteq E_{jt}$ be the set of safe assignments available for $j$.
If $\hat{E}_{jt} = \emptyset$, then reject $j$; else sample an assignment $e \in \hat{E}_{jt}$ with
  probability $\alpha x^*_{e,t}/p_{jt}$.
```

The last step of Algorithm 1 is well defined since we have $\sum_{e \in \hat{E}_{jt}} \alpha x^*_{e,t}/p_{jt} \le \sum_{e \in E_{jt}} x^*_{e,t}/p_{jt}$, which is at most 1.[3]

Adapative algorithm (ADAP)**.** Algorithm ADAP is an adaptive algorithm which uses Monte-Carlo simulations. The main idea is as follows. Suppose we aim to develop an online algorithm achieving a competitive ratio of $\gamma \in [0,1]$. Consider an assignment $e=(i,j) \in E_t$ for a job j at time t. Let $\mathcal{S}_{e,t}$ be the event that e is safe (*i.e.,* we can choose this assignment without budget violation for all resources) conditioning on the arrival of e at t. By using Monte-Carlo simulation of the strategy up to t, we can get a sharp estimate of $\Pr[\mathcal{S}_{e,t}]$, say $\beta_{e,t}$, with polynomial number of samples. Therefore if e is safe at t, we choose it with probability $\frac{x_{e,t}}{p_{j,t}} \frac{\gamma}{\beta_{e,t}}$, which implies that e is chosen with probability $\gamma x_{e,t}$ unconditionally.

The simulation-based attenuation technique has been used previously in other stochastic optimization problems (*e.g.,* stochastic knapsack [41], stochastic matching [1]). We assume that the sharp estimate $\beta_{e,t}$ of $\Pr[\mathcal{S}_{e,t}]$ for all t and e is *exact*, since the sampling error can be accounted as a multiplicative factor of $(1-\epsilon)$ in the competitive ratio by a standard Chernoff bound argument. Formally our algorithm, denoted by ADAP, is described in Algorithm 2. The running time of this algorithm is polynomial in $1/\varepsilon$.

ALGORITHM 2: The adaptive algorithm (ADAP)

```
At time $t$, let $j$ be the job that arrives.
Let $\hat{E}_{j,t} \subseteq E_{j,t}$ be the set of safe assignments available for $j$.
If $\hat{E}_{j,t} = \emptyset$, then reject $j$; else sample an assignment $e \in \hat{E}_{j,t}$ with
  probability $\frac{x^*_{e,t}}{p_{j,t}} \frac{\gamma}{\beta_{e,t}}$.
```

To ensure the above algorithm is mathematically well-defined with parameter γ, we need to show that $\beta_{e,t} \ge \gamma$ for every t and e.

Lemma 4.1 (Validity of ADAP). *By choosing $\gamma = 1/(\ell+1)$, we have $\beta_{e,t} \ge \gamma$ for all $t \in [T]$ and $e \in E_t$.*

Both the algorithms presented do not work in the regime when B is small. To overcome this, we propose a new algorithm with additional restrictions when $B_k = 1$. Consider MBOA-S which has the following setting: (1) all resources are non-integral and have a unit budget $B_k = 1$; (2) each assignment requires only one

[3] In other words, with probability $1 - \sum_{e \in \hat{E}_{jt}} \alpha x^*_{e,t}/p_{jt}$, we will do nothing and reject job j.

single resource ($\ell = 1$); (3) servers have no deadline ($d_i = T$); (4) the arrival distributions over all job-types are *identical* across all rounds, *i.e.,* for each job-type j, $p_{jt} = p_j$ for all $t \in [T]$. It can be shown that the performance of ALG_1 and ALG_2 can be arbitrarily bad (see full version).

This new setting requires a modified benchmark LP. For each $e \in E$, let x_e be the expected number of times the assignment e is made in the offline optimal over the T rounds. For each e, we use a_e to denote the cost of e for the unique resource it consumes. For a given threshold $\alpha = 1/2$, we say e is *big* if $a_e > \alpha$ and *small* otherwise. For each resource k, let BIG(k) (SM(k)) refers to the set of big assignments (small) participating on constraint (or resource) k. We use the following benchmark LP.

$$\max \textstyle\sum_{e \in E} w_e x_e \tag{5}$$
$$\text{s.t.} \textstyle\sum_{e \in E_j} x_e \le p_j * T \qquad \forall j \in J, t \in [T] \tag{6}$$
$$\textstyle\sum_{e \in \text{BIG}(k)} x_e a_e + \sum_{e \in \text{SM}(k)} x_e a_e \le 1 \qquad \forall k \in [K] \tag{7}$$
$$\textstyle\sum_{e \in \text{BIG}(k)} x_e \le 1 \qquad \forall k \in [K] \tag{8}$$

Constraint (8) is valid since in any offline optimal, at most one big assignment from BIG(k) can be made, for each k. We design a new algorithm for this setting, whose main idea is as follows. We split the whole T rounds into two stages, where the first stage consists of the first $T * \beta$ rounds and the second stage consists of the remaining rounds. In the first stage, our algorithm only considers big assignments and drops any small assignment while in the second stage it considers only small ones while dropping the big ones. For each job-type j, let BIG(j) (SM(j)) be the set of big (small) assignment with respect to j. Suppose $\{x_e^*\}$ is an optimal solution to the new benchmark LP (5). We can then formally describe the algorithm as in Algorithm 3.

```
ALGORITHM 3: MBOA-S($\beta$, $\gamma_1$, $\gamma_2$)
The first stage:
For each time $t \in [T * \beta]$, assume some job $j$ arrives.
Sample a big assignment $e \in \text{BIG}(j)$ with probability $\frac{\gamma_1 x_e^*}{p_j * T}$.
If $e$ is safe, then make it; otherwise reject it.
The second stage:
For each time $t \in \{T * \beta + 1, T * \beta + 2, \ldots, T\}$, assume some job $j$ arrives.
Sample a small assignment $e \in \text{SM}(j)$ with probability $\frac{\gamma_2 x_e^*}{p_j * T}$.
If $e$ is safe, then make it; otherwise reject it.
```

5 MAIN RESULTS AND TECHNIQUES

We now describe the main results and theoretical techniques used. Detailed proofs are deferred to the full version.

First, we present two algorithms based on LP-(1), NADAP and ADAP, which are non-adaptive and adaptive respectively. For the integral MBOA-KAD where all resources are integral, we have the following theorems.

Theorem 5.1 (Performance of NADAP for integral case). *For* MBOA-KAD *when all resources are integral with sparsity* ℓ, NADAP *with* $\alpha = \frac{1}{\ell+1}$ *achieves a competitive ratio of at least* $\frac{1}{\ell+1}(1 - \frac{1}{\ell+1})^{\ell} \ge \frac{1}{e(\ell+1)}$ *using* LP-(1) *as the benchmark. The analysis for this is tight.*

Theorem 5.2 (Performance of ADAP for integral case). *For* MBOA-KAD *when all resources are integral with sparsity* ℓ, ADAP *with* $\gamma = \frac{1}{\ell+1}$ *achieves a competitive ratio of at least* $\frac{1-\epsilon}{\ell+1}$, *for any given constant* $\epsilon > 0$. *Moreover, no adaptive algorithm can achieve a ratio better than* $\frac{1}{(\ell-1+1/\ell)}$.

From the above two theorems, we have that NADAP and ADAP are almost optimal among all algorithms that use LP-(1) as benchmark, for the integral MBOA-KAD. Additionally, Theorem 5.1 achieves the best possible ratio that NADAP can get based on LP-(1). For algorithms which do not use LP-(1) as benchmark, the hardness result is $O(\ln \ell/\ell)$ since the inapproxibility result of ℓ-uniform hypergraph matching [30] carries over to this setting.

We now show an example to show that the results proved in Theorem 5.1 is tight.

Example 5.3 (Tight Example for Integral Resources). Consider a star graph $G = (I, J, E)$ where $|I| = 1, |J| = \ell+1, E = \{e_j | j \in [\ell+1]\}$ with $T = \ell + 1$. Let $d_1 = T$, *i.e.,* no deadline constraints. For each $t \in [T], p_{jt} = 1$ iff $j = t$ and 0 otherwise. We use $\mathbf{a}_j$ and x_j^* to denote the terms $\mathbf{a}_{e_j}$ and $x_{e_j, t=j}^*$. Let $K = \ell$ with $B_k = 1$ for each $k \in [\ell]$ and $\mathbf{a}_j = \mathbf{e}_j$ for each $j \le \ell$, where $\mathbf{e}_j$ is the jth standard-basis unit vector of dimension K, and $\mathbf{a}_j = \mathbf{1}$ (of dimension K) for $j = \ell + 1$. Let the optimal solution to LP-(1) be $x_j^* = 1 - \epsilon$ for each $j \le \ell$ and $x_{\ell+1}^* = \epsilon$ for a proper weight vector. Now consider the assignment $e = e_{\ell+1}$ when $j = \ell + 1$ comes at $t = T$. Let us compute the probability $\Pr[\mathcal{S}_{e,T}]$ that e is safe at T in NADAP(α). Assignment e will be safe at $t = T$ iff none of $e_j, j \le \ell$ is made before. At each time $t < T$, NADAP(α) makes the assignment $e_{j=t}$ with probability $\frac{\alpha x_j^*}{p_j} = \alpha(1-\epsilon)$. This implies that $\Pr[\mathcal{S}_{e,T}] = (1 - \alpha(1-\epsilon))^{\ell}$, which matches the lower bound. □

Next, we consider a general case, where we have both integral and non-integral resources while making a mild assumption that the budget of any non-integral resource is large enough. Let B be the minimum budget for any non-integral resource. We then prove the following two theorems.

Theorem 5.4 (Performance of NADAP for the general case). *For* MBOA-KAD *with integral and non-integral sparsity* ℓ_1 *and* ℓ_2, NADAP *with* $\alpha = \frac{1}{\ell_1+1}$ *achieves a competitive ratio of* $\frac{1}{\ell_1+1}\Big((1 - \frac{1}{\ell_1+1})^{\ell_1} - \varepsilon\Big)$, *for any* $\varepsilon > 0$, *assuming* $B \ge 2\ln(\frac{\ell_2}{\varepsilon})\Big(1 + \frac{3\ell_1+2}{\ell_1^2}\Big) + 2$.

Theorem 5.5 (Performance of ADAP for the general case). *For* MBOA-KAD *with integral and non-integral sparsity* ℓ_1 *and* ℓ_2, ADAP *with* $\gamma = \frac{1-\epsilon}{\ell_1+1}$ *achieves a competitive ratio of* $\frac{1-2\varepsilon}{\ell_1+1}$ *for any given* $\varepsilon > 0$, *assuming* $B \ge 3\ln(\frac{\ell_2}{\varepsilon})(1 + \frac{1}{\ell_1}) + 2$.

Our results imply that with the knowledge about arrival distributions we can obtain significant improvements over the results for the adversarial model. Let us compare our results with those of [9]. The setting in [9] can be viewed as a special case of our model with $\ell_1 = \ell_2 = 1$. From Theorem 5.5, we obtain a $(\frac{1}{2} - \epsilon)$ competitive ratio assuming $B \ge 12\ln(1/\epsilon)$ while [9] obtain a ratio of $O(\frac{1}{R^{\epsilon}\ln R})$, assuming $B \ge \frac{R}{\epsilon}$ and $R \doteq \frac{\max b_{i,j}}{\min b_{i,j}}$ (*i.e.,* the ratio of the largest bid to the smallest bid over all possible assignments). *Note that our results completely removes the dependency on R and also significantly relax* *the lower bound assumption on B.* This is a theoretical evidence to advocate the use of historical data to learn arrival distributions.

Third, we consider the case of MBOA-KAD when both integral and non-integral resources are involved but no lower bound is known for the budgets of non-integral resources. To make the problem tractable, we make the following three assumptions: (1) each assignment consumes only a single resource ($\ell = 1$); (2) deadline constraints on all servers are removed; (3) the arrival distributions over all job types are *identical* across all rounds. We refer to MBOA-KAD under these three simplifying conditions as MBOA-S. Note that MBOA-S still generalizes the well-known online bipartite matching problem and several variants such as Adwords and Display Ads (*e.g.,* [25, 29, 34, 43]). It can be shown that the performance of the two previous algorithms, NADAP and ADAP, can be arbitrarily bad in this case. We propose a strengthened benchmark LP and obtain the following theorem.

Theorem 5.6. *There exists an online algorithm which achieves an online ratio of $\frac{1}{4}$ for* MBOA-S. *Meanwhile, no online algorithm can achieve a ratio better than $\frac{e-1}{2e-1} \sim 0.387$.*

6 EXPERIMENTS

In this section we describe our experimental results. We use the Google cluster trace data [18, 31] which was used by Kash et al. [36]. This dataset contains traces of job allocation to servers within Google's datacenters. We process this dataset for our purposes, which we describe below. To further show the generality of our model, we also run additional experiments modeling an allocation problem in the public safety domain.

Experimental setup. Every machine is characterized by an id, the total CPU capacity and the total memory capacity. Each sample is from an interval of 2 minutes in the dataset and hence is a short enough time-span to consider *hard total budget* on the resources. A job type is characterized by the CPU and memory requirements. Hence there can be multiple jobs of the same type (which we use to construct our arrivals). Every machine consumes two resources and these resources are not shared by other machines (this application has a simpler notion of *shared* resources compared to the generality our model can handle). Therefore we have total of $2m$ resources with a sparsity $\ell = 2$. We assume that all machines are active throughout.

Dataset and preprocessing. Our experimental setup is inspired from the experiments of Kash et al. [36]. We use a random subset of the dataset by sampling m machines and n jobs randomly for $m = 10$ and $n = \{20, 100\}$. Our experiments are run for both the (m, n) pairs. We assign arrival rates randomly to each of the jobs and use it for all the experiments. For every (m, n) pair we generate the compatibility graph by choosing 5 machines at random, that a job j can be run on. All experiments are reported by running 100 independent trials and taking the sample average. Assignment weights are assigned by generating an independent random number between 0 and 1.

Algorithms. We compare our main algorithm NADAP with the following three baselines. These baselines have been used previously in the literature [23]. The baselines are as follows. Suppose job j arrives at time t. (1) SCALED: sample an assignment $e \in E_{jt}$ with probability $\frac{x^*_{e,t}}{\sum_{e \in E_{jt}} x^*_{e,t}}$ and assign e iff safe. (2) USamp: sample an assignment $e \in E_{jt}$ uniformly from E_{jt} and assign e iff safe. (3) Greedy: choose the assignment $e \in E_{jt}$, which has the largest weight w_e among all safe options in E_{jt}. Finally note that ADAP had similar performances as NADAP in our experiments and for clarity we omit those from the figures. Deviating from the conservative estimate predicted by theory for α in NADAP, throughout the experimental section we set $\alpha = 1$. We chose this by tuning for optimal performance on a small holdout of the dataset. The reason this value is different from what theory predicts is that, in theory we are optimizing for the worst case input, while, as will be evident from the results, these datasets do not represent the worst-case graphs.

Throughput experiments. We compare the *total weight* of all the assignments made by each of these algorithms. The first column in Figure 2 describes the details. It is clear that our main algorithm NADAP performs the best. Among the baselines Greedy is better than the other algorithms. Despite SCALED having the information of the optimal solution from the LP, it is not able to perform as good as Greedy. This shows the inherent power of *adaptive* algorithms.

Fairness experiments. We run further additional experiments to study the *fairness* of these algorithms. Fairness is a broad topic and our goal in this paper is to show that despite not explicitly optimizing for it, NADAP performs well compared to the baselines. We discuss two notions of fairness which are inspired from the *max-sum* and *max-min* fairness of Kash et al. [36]. Our setting differs from theirs and hence we cannot directly use their definitions. However we define two notions, namely *drop-sum* and *drop-max* fairness. For a give type j, let ds_j be defined as the expected difference between the number of times this type appears in an arrival sequence and the number of times an algorithm assigns it successfully. *Drop-sum* metric calculates the sum of ds_j for every type j while *drop-max* calculates the maximum over all types j of ds_j. Intuitively, higher the value of either of these metrics, the more *unfair* the algorithm is. In the second column of Figure 2 we show how the baselines and our algorithm performs on the *drop-sum* metric. Our algorithm once again comes on top with almost no difference among the other three baselines. On the *drop-max* metric in the third column of Figure 2, however, we see an interesting result where our algorithm is slightly worse than the three baselines. However the difference is not too significant, suggesting that alongside maximizing throughput, NADAP is also inherently *fair.*

Does graph sparsity matter? [4] We further study if either throughput or the two fairness metrics significantly change for the algorithms when the graph sparsity is varied. We use the setup of 10 machines, 20 jobs types and 1000 arrivals as the underlying parameters. Graph sparsity is varied by controlling the number of neighbors each job-arrival can be assigned to. Figure 3 shows the results for this experiments. As evident, there is some variation in the absolute numbers, but these are not significant enough in

[4]Term sparsity is overloaded. Here it refers to the number of edges, which is different from sparsity of resources used throughout the paper.

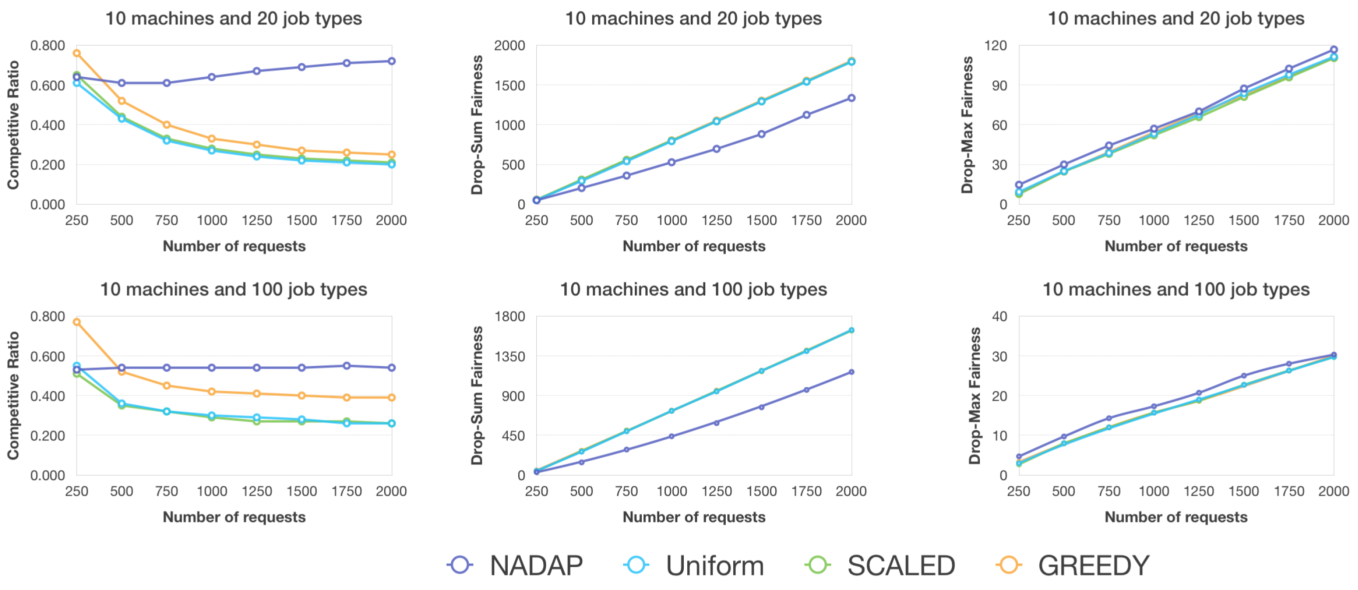


Figure 2: Allocation experiments on the Google cluster trace dataset. The upper row corresponds to 10 machines and 20 job types, while the lower row corresponds to 10 machines and 100 job types. First column is the results for Throughput experiments, second column is the results for the *drop-sum* fairness experiments and the third column is the results for the *drop-max* fairness experiments.

absolute terms. Additionally, these numbers do not change the relative ordering for the performance of the algorithms in either of the three metrics.

Discussion. The main experiments suggest that NADAP performs well in practice, alongside having good theoretical guarantees. In fact, the throughput experiments indicate that the performance guarantee is far better than the theoretical prediction of around 0.3. We also show that for a reasonable definition of fairness metrics, our algorithm performs as good or better than the baselines. This suggests that our model and algorithm is suitable for scenarios where we want to maximize both throughput and fairness properties. It is also an interesting future direction to explicitly account for fairness to further improve the performance of NADAP.

6.1 Additional Experiments for the Public Safety Application

We use a large-scale policing dataset [44] for the public safety application.

Dataset and assumptions. The policing dataset [44] contains records from the Texas state since 2010. Every record in this dataset contains the following: County, Latitude and Longitude, Time, Violation, Officer id, whether there was a search, stop outcome and driver details such as gender, age, race (which we do not require for our purposes). We use this information to create offline and online vertices as well as the graph as follows. For every county we create one offline vertex and pick one of the locations within this county as the station (we assume that dispatch for this county is from this station). For the online side, we create one online vertex for every pair of (location, offense).[5] For each vertex or type (offline and online), the set of neighbors is the set of 5 counties that are within a radius $r = 2$ (in terms of differences in latitude and longitude) of the location associated with this vertex. The dataset provides the time rounded to the nearest minute. We regard every minute as a time-step and set the time horizon $T = 24 * 60 = 1440$ (24-hour period). We sample 20 frequently appearing counties for the offline side, 120 frequently appearing (location, offense) pairs for the online types. To learn p_{jt}'s, we average the arrival frequencies in a randomly chosen 90-day period and use another randomly chosen 14-day period for testing. We consider the unweighted case and are only interested in maximizing the total number of unlawful activities handled.

We assume that there are 7-types of resources, namely, total travel distance, officers, patrol vehicles and equipments for speeding, search, citation and arrest. The first 3 resources are owned by each offline vertex (station) while the last four are shared across vertices (therefore a total of $3 * |I| + 4$ resources). The first resource captures the total working hours each station can continuously be engaged in and account for it as the total travel distance travelled. The last four represent the resources involved for detecting and issuing a speeding violation, for conducting a search on the vehicle, for issuing a citation and for making an arrest on the spot respectively. Each match $e = (i, j)$ (assignment of online offense type j to station i) will consume 7 resources; the consumption of the first three resources is jointly determined by the pair (i, j) (denoted by $\mathbf{a}_{e,1}$) and the consumption of the last four resources is determined by type j (denoted by $\mathbf{a}_{e,2}$). By averaging over all records from the dataset, we compute and normalize such that $\mathbf{a}_{e,1} \in [0, 1]^3$ and $\mathbf{a}_{e,2} \in [0, 1]^4$. To make $\mathbf{a}_{e,2}$ depend on station i, we uniformly sample a number λ_i from $[0, 1]$ and set the final cost $\mathbf{a}_e$ as $(\mathbf{a}_{e,1}, \lambda_i * \mathbf{a}_{e,2})$.

[5]Location is a latitude longitude pair rounded to the nearest integer.

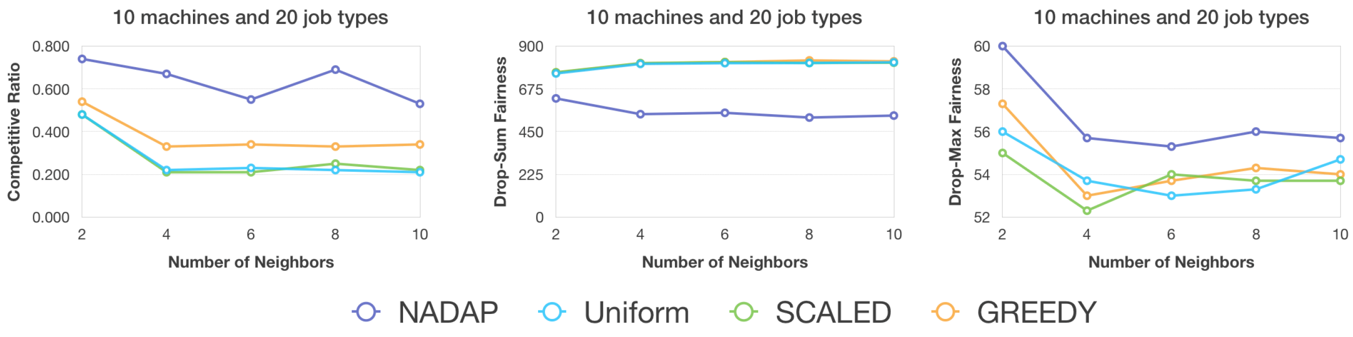
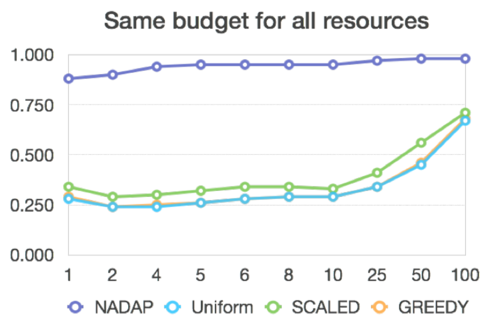
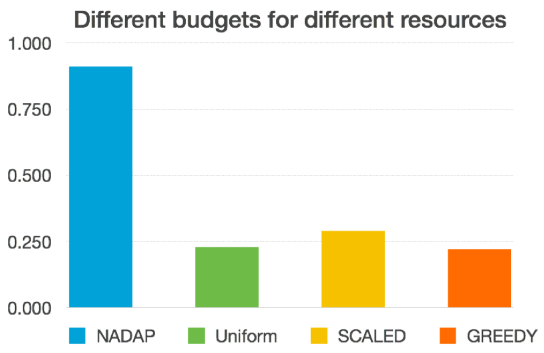


Figure 3: Varying the graph sparsity for instance with 10 machines, 20 job types and 100 arrivals of jobs. First, second and third column corresponds to throughput, drop-sum and drop-max metrics respectively.



(a) Comparing algorithms when all resources have a uniform budget. x and y-axises represent the budget value and competitive ratio respectively.



(b) Comparing algorithms when different resources have different budgets

Experimental parameters and baselines. First, we conduct experiments by choosing a unform budget for all resources. In particular we choose the budget value in the range $\{1, 2, 4, 5, 6, 8, 10, 25, 50,100\}$. We also run an experiment with different budgets for different resources. For every resource we choose a random integer between 1 and 100 as the budget. The value of these budgets are chosen to ensure that it accounts for both the small budget (*i.e.,* optimal solution is much smaller than T) and the large budget (*i.e.,* optimal is very close to T) case. For each given experiment and algorithm, we run 10 independent runs on each of the 14-day testing period and take the average. We use the following parameters for the algorithms. For NADAP, we set $\alpha = 1$ (which is higher than the theoretical value which was fine-tuned via standard cross-validation approach) and set $\gamma = 1$ for ADAP.

Results and discussion. Figures 4a and 4b describe the respective experimental results when the total budget takes a uiform value or different values among resources. From the results we can observe that our algorithms ADAP and NADAP significantly out-perform all our natural baselines. Among the baselines the LP-based baseline (SCALED) beats the LP-agnostic baselines (albeit by a small amount). Additionally, our algorithms almost approach the optimal on this dataset which is far better than the theoretical guarantee (around $1/8 = 0.125$). Hence experimentally we show that our algorithms are useful, beat the natural baselines comprehensively and achieve near optimal in practice.

7 CONCLUSIONS & FUTURE RESEARCH

In this paper, we studied the multi-budgeted allocation problem in the context of matching markets such as crowdsourcing, candidate hiring, etc. In this context, we proposed a novel model and provided efficient LP-based algorithms with improved competitive ratios. In particular, we showed two algorithms and analyzed their performance formally. In the theoretical analysis of these algorithms, we used novel ideas which can potentially be of independent interest in both the analysis of online matching and allocation algorithms. Finally, our algorithms were compared against several heuristic baselines experimentally on a real-world dataset to validate our theoretical results. We also explored properties of our algorithms in the context of tradeoffs in economic efficiency and fairness.

In addition to further exploration of the interplay between various fairness objectives and the traditional efficiency-maximizing objective used in many settings, we believe that the inclusion of incentives in our models would be of future interest. For example, Kash et al. [36] explore various design desiderata in their dynamic fair divison (of divisible tasks) model; similar qualitative steps could be taken in our model (which focuses on *in*divisible allocation). As another example, resource allocation in the *security games* setting [38], limited resources are deployed to prevent a strategic adversary from attacking targets. Recent work has explored security games under various forms of dynamism [8, 53, 55].

REFERENCES

[1] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. 2015. Improved approximation algorithms for stochastic matching. In *ESA*.

[2] Shipra Agrawal and Nikhil Devanur. 2016. Linear contextual bandits with knapsacks. In *NIPS*.

[3] Shipra Agrawal and Nikhil R Devanur. 2014. Bandits with concave rewards and convex knapsacks. In *EC*.

[4] Shipra Agrawal and Nikhil R Devanur. 2014. Fast algorithms for online stochastic convex programming. In *SODA*.

[5] Shipra Agrawal, Nikhil R Devanur, and Lihong Li. 2016. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *COLT*.

[6] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. 2014. A dynamic near-optimal algorithm for online linear programming. *Operations Research* 62, 4 (2014).

[7] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. 2013. The online stochastic generalized assignment problem. In *APPROX-RANDOM*.

[8] Tansu Alpcan and Sonja Buchegger. 2011. Security games for vehicular networks. *IEEE Transactions on Mobile Computing* 10, 2 (2011), 280–290.

[9] Sepehr Assadi, Justin Hsu, and Shahin Jabbari. 2015. Online Assignment of Heterogeneous Tasks in Crowdsourcing Markets. In *AAAI-HComp*.

[10] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. 2013. Bandits with knapsacks. In *FOCS*.

[11] Ashwinkumar Badanidiyuru, John Langford, and Aleksandrs Slivkins. 2014. Resourceful contextual bandits. In *COLT*.

[12] Omar Besbes and Assaf Zeevi. 2012. Blind network revenue management. *Operations research* 60, 6 (2012), 1537–1550.

[13] Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*.

[14] Niv Buchbinder and Joseph Naor. 2009. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research* 34, 2 (2009), 270–286.

[15] Niv Buchbinder, Joseph Seffi Naor, et al. 2009. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science* 3, 2–3 (2009), 93–263.

[16] Jan M. Chaiken and Peter Dormont. 1978. A Patrol Car Allocation Model: Background. *Management Science* 24, 12 (1978).

[17] Xi Chen, Will Ma, David Simchi-Levi, and Linwei Xin. 2016. Dynamic recommendation at checkout under inventory constraint. *http://dx.doi.org/10.2139/ssrn.2853093* (2016).

[18] Yanpei Chen, Archana Sulochana Ganapathi, Rean Griffith, and Randy H Katz. 2010. Analysis and lessons from a publicly available google cluster trace. (2010).

[19] V.E. Chenthamarakshan, N. Kambhatla, R.C. Kanjiranthinkal, A.K.R. Singh, and K. Visweswariah. 2012. Systems and methods for matching candidates with positions based on historical assignment data. (May 17 2012). https://www.google.com/patents/US20120123956 US Patent App. 12/944,868.

[20] Alan Demers, Srinivasan Keshav, and Scott Shenker. 1989. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM*.

[21] Arnoud V den Boer. 2014. Dynamic pricing with multiple products and partially specified demand distribution. *Mathematics of operations research* 39, 3 (2014), 863–888.

[22] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*.

[23] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources. In *AAAI*.

[24] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Assigning Tasks to Workers Based on Historical Data: Online Task Assignment with Two-sided Arrivals. In *AAMAS*. 318–326.

[25] Jon Feldman, Nitish Korula, Vahab Mirrokni, S Muthukrishnan, and Martin Pál. 2009. Online ad assignment with free disposal. In *WINE*.

[26] Ali Ghodsi, Vyas Sekar, Matei Zaharia, and Ion Stoica. 2012. Multi-resource fair queueing for packet processing. *ACM SIGCOMM* (2012).

[27] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. 2013. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *ACM ECCS*.

[28] R. Guedes, V. Furtado, and T. Pequeno. 2014. Multiagent models for police resource allocation and dispatch. In *2014 IEEE Joint Intelligence and Security Informatics Conference*.

[29] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. 2011. Online stochastic weighted matching: Improved approximation algorithms. In *WINE*.

[30] Elad Hazan, Shmuel Safra, and Oded Schwartz. 2006. On the complexity of approximating k-set packing. *computational complexity* 15, 1 (2006).

[31] J. L. Hellerstein. 2010. Google Cloud Trace Dataset. (2010). https://github.com/google/cluster-data

[32] Chien-Ju Ho and Jennifer Wortman Vaughan. 2012. Online Task Assignment in Crowdsourcing Markets.. In *AAAI*.

[33] Yan Huang, Favyen Bastani, Ruoming Jin, and Xiaoyang Sean Wang. 2014. Large Scale Real-time Ridesharing with Service Guarantee on Road Networks. *VLDB Endow.* 7, 14 (2014).

[34] Patrick Jaillet and Xin Lu. 2013. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* 39, 3 (2013).

[35] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. 2013. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM TON* (2013).

[36] Ian Kash, Ariel D Procaccia, and Nisarg Shah. 2014. No agent left behind: Dynamic fair division of multiple resources. *JAIR* (2014).

[37] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. 2014. Primal beats dual on online packing LPs in the random-order model. In *STOC*.

[38] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. 2009. Computing optimal randomized resource allocations for massive security games. In *AAMAS*.

[39] Sang M. Lee, Lori Sharp Franz, and A. James Wynne. 1979. Optimizing State Patrol Manpower Allocation. *Journal of the Operational Research Society* 30, 10 (01 Oct 1979).

[40] S. Ma, Y. Zheng, and O. Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE*.

[41] Will Ma. 2014. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In *SODA*.

[42] Will Ma and David Simchi-Levi. 2017. Online resource allocation under arbitrary arrivals: Optimal algorithms and tight competitive ratios. *http://dx.doi.org/10.2139/ssrn.2989332* (2017).

[43] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *MOR* (2012).

[44] Emma Pierson, Camelia Simoiu, Jan Overgoor, Sam Corbett-Davies, Vignesh Ramachandran, Cheryl Phillips, and Sharad Goel. 2017. A large-scale analysis of racial disparities in police stops across the United States. *arXiv preprint arXiv:1706.05678* (2017).

[45] Karthik Abinav Sankararaman and Aleksandrs Slivkins. 2018. Combinatorial Semi-Bandits with Knapsacks. In *AIStats*.

[46] Robert P Shumate and Richard F Crowther. 1966. Quantitative methods for optimizing the allocation of police resources. *J. Crim. L. Criminology & Police Sci.* 57 (1966).

[47] Yaron Singer and Manas Mittal. 2013. Pricing mechanisms for crowdsourcing markets. In *WWW*.

[48] Adish Singla and Andreas Krause. 2013. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *WWW*.

[49] Aleksandrs Slivkins and Jennifer Wortman Vaughan. 2014. Online decision making in crowdsourcing markets: Theoretical challenges. *ACM SIGecom Exchanges* 12, 2 (2014).

[50] Ashwin Subramanian, G Sai Kanth, Sharayu Moharir, and Rahul Vaze. 2015. Online incentive mechanism design for smartphone crowd-sourcing. In *WiOPT*.

[51] Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. 2017. Flexible Online Task Assignment in Real-time Spatial Data. *Proc. VLDB Endow.* (2017).

[52] Xinshang Wang, Van-Anh Truong, and David Bank. 2018. Online advance admission scheduling for services with customer preferences. *arXiv preprint arXiv:1805.10412* (2018).

[53] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. 2014. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*.

[54] Xing Yi, James Allan, and W Bruce Croft. 2007. Matching resumes and jobs based on relevance models. In *SIGIR*.

[55] Yue Yin, Haifeng Xu, Jiarui Gan, Bo An, and Albert Xin Jiang. 2015. Computing Optimal Mixed Strategies for Security Games with Dynamic Payoffs.. In *IJCAI*.

[56] Dong Zhao, Xiang-Yang Li, and Huadong Ma. 2014. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In *INFOCOM*.