A Unified Model for the Two-stage Offline-then-Online Resource Allocation

Yifan Xu[1,2,4] , **Pan Xu**[3] , **Jianping Pan**[4] and **Jun Tao**[1,2]
[1]Key Lab of CNII, MOE, Southeast University, Nanjing, China
[2]School of Cyber Science and Engineering, Southeast University, Nanjing, China
[3]Department of Computer Science, New Jersey Institute of Technology, Newark, USA
[4]Department of Computer Science, University of Victoria, Victoria, Canada
xyf@seu.edu.cn, pxu@njit.edu, pan@uvic.ca, juntao@seu.edu.cn

Abstract

With the popularity of the Internet, traditional offline resource allocation has evolved into a new form, called *online resource allocation*. It features the online arrivals of agents in the system and the real-time decision-making requirement upon the arrival of each online agent. Both offline and online resource allocation have wide applications in various real-world matching markets ranging from ridesharing to crowdsourcing. There are some emerging applications such as *rebalancing in bike sharing* and *trip-vehicle dispatching in ridesharing*, which involve a two-stage resource allocation process. The process consists of an offline phase and another sequential online phase, and both phases compete for the same set of resources. In this paper, we propose a unified model which incorporates both offline and online resource allocation into a single framework. Our model assumes *non-uniform and known* arrival distributions for online agents in the second online phase, which can be learned from historical data. We propose a parameterized linear programming (LP)-based algorithm, which is shown to be at most a constant factor of 1/4 from the optimal. Experimental results on the real dataset show that our LP-based approaches outperform the LP-agnostic heuristics in terms of robustness and effectiveness.

1 Introduction

Matching markets involve heterogeneous agents (typically from two parties) who are paired for mutual benefits. During the last decade, matching markets have emerged and grown rapidly through the medium of the Internet. They have evolved into a new style, called *Online Matching Markets* (OMMs). Typical examples include ridesharing (riders and drivers), crowdsourcing (workers and tasks), and Internet advertising (impressions and advertisers). In OMMs, agents of at least one type join the market in an online fashion, and are referred to as online agents (*e.g.,* riders, workers, and impressions). Furthermore, upon the arrival of any online agent, we have to decide quickly and irrevocably which offline agent(s) to match it with. That is mainly due to the low "patience" of the online agents. These features—online arrivals and the real-time decision-making requirement—distinguish OMMs from traditional matching markets where the information of all agents is fully disclosed in advance.

OMMs have received significant interest in both computer science and operations research communities. There is a large body of research work who studied matching policy design for the profit maximization in ridesharing [Ashlagi *et al.*, 2019; Lowalekar *et al.*, 2018; Bei and Zhang, 2018; Zhao *et al.*, 2019; Dickerson *et al.*, 2018a], crowdsourcing [Assadi *et al.*, 2015; Ho and Vaughan, 2012; Dickerson *et al.*, 2018b], admission scheduling and online recommendations [Ma and Simchi-Levi, 2017; Chen *et al.*, 2016]. Recently, [Dickerson *et al.*, 2019] presented a general online resource allocation model, called Multi-Budgeted Online Assignment (MBOA), to address the matching policy design in various real-world OMMs featuring that each assignment could potentially consumes multiple resources. The basic model is as follows. We are given a bipartite graph $G = (I, J, E)$ where I and J represent the respective sets of offline and online agents. There is a set $\mathcal{K}$ of K resources and each resource $k \in \mathcal{K}$ has a given budget B_k. Each edge $e = (i, j)$ or assignment[1] of j to i is associated with a profit and a vector cost of dimension K. A natural question is how to arrange an assignment for each online agent upon its arrival such that the total expected profit is maximized subject to the budget constraints.

MBOA [Dickerson *et al.*, 2019] has greatly generalized the assignment problems in OMMs. Consider Amazon Mechanical Turk for example, which automatically crowdsources online workers for offline tasks. In this case, there are typically two kinds of resources, budget (deposited by a task manager to pay workers) and tasks (typically each task has a limited number of copies). In addition to OMMs, MBOA has captured a wide range of online resource allocation problems in datacenters [Ghodsi *et al.*, 2012; Ghodsi *et al.*, 2013], public safety [Lee *et al.*, 1979; Guedes *et al.*, 2014] and candidates recruitment [Yi *et al.*, 2007], for example. Note that MBOA considers resource allocation only for online arrival agents. In some emerging applications, however, there are *both offline and online agents* who are competing for the same set of resources. Consider the two motivating examples below.

[1]Throughout this paper, we use the two terms "edge" and "assignment" interchangeably.

Rebalancing in Bike Sharing Systems (BSSs). BSSs have become an important alternative for addressing the last mile problem in city Intelligent Transportation Systems. Users varied travel patterns lead to uneven distributions of bikes among docking stations. A common scenario is during afternoon peak hours, "demanding" stations (*e.g.,* near shopping malls or metro stations) have a high demand but a low supply, while "supplying" stations (*e.g.,* suburb areas far away from downtown) have a high supply but a low demand. Thus, BSSs have to rebalance distributions by moving bikes from "supplying" to "demanding" stations such that they can address as many user requests as possible. Two common rebalancing strategies are proposed. One is the Truck-Based Rebalancing (TBR), which directly utilizes trucks or trailers to move bikes from one station to the other. The other is Crowdsourcing-Based Rebalancing (CBR), which incentives online bike users to participate in rebalancing tasks. TBR has a strong controllability, high efficiency during off-peak hours, and a high labor cost. In contrast, CBR has a relatively low controllability, high efficiency during peak hours, and a low labor cost. The two approaches each has received considerable interest, see, *e.g.,* [Raviv *et al.*, 2013; O'Mahony and Shmoys, 2015; Liu *et al.*, 2016; Li *et al.*, 2018] for TBR, and [Singla *et al.*, 2015; Haider *et al.*, 2018; Pan *et al.*, 2019; Duan and Wu, 2019] for CBR. However, very few of them have ever considered both TBR and CBR in the same framework, though the two are proposed to address the same issue. For TBR, it is typically deployed during off-peak hours and involves renting *offline* trucks and labors. For CBR, it is operated during peak hours instead, and crowdsources *online* users as potential workers. Note that both TBR and CBR compete for one resource, the global budget used for renting truck and hiring labors in TBR, and paying online workers in CBR. Additionally, the two compete for the same set of tasks, *i.e.,* moving bikes from supplying to demanding stations. In fact, we can view each supplying and demanding station as a "resource" with the capacity being the number of bikes in supply and demand. A natural question arises: How to optimally allocate the "resources" to the two approaches such that we can complete as many tasks as possible?

Trip-vehicle dispatching with multi-type requests. In recent years, ridesharing apps such as DiDi launch services that allow riders to reserve a trip in advance, which is called *scheduled requests* [Huang *et al.*, 2019]. Consider a short time window $\mathcal{W}$ during peak hours. Let C_1 be the set of scheduled requests received with starting time in $\mathcal{W}$ and C_2 be the (random) set of real-time requests which arrive in $\mathcal{W}$. Note that C_1 is received well before the start of $\mathcal{W}$ and thus can be viewed as *offline* requests. Both C_1 and C_2 are competing for the same set of "resources": the available drivers in $\mathcal{W}$. A natural question is: How to allocate drivers to requests such that the total expected profit obtained is maximized?

The above two examples both feature a two-stage resource allocation: the first stage involves a set of *static* offline agents while the second stage involves a set of *dynamic* online agents, and they are competing for the same set of resources.

Main contributions. Our contributions are summarized as follows. First, we propose a unified framework, called *Two-Stage Resource Allocation* (TS-RA), by generalizing the model in [Dickerson *et al.*, 2019]. Overall, TS-RA incorporates the offline and online resource allocation into one single framework, which can potentially capture a wider range of applications and optimize the allocation jointly. Second, we propose a parameterized linear programming (LP)-based algorithm, which is shown to be at most a constant factor of $1/4$ from the optimal. One highlight is the performance of our LP-based approaches depends only on the *sparsity*, *i.e.,* the maximum number of different resources requested by an assignment, regardless of the total number of resources involved which can be potentially large. Third, we present several greedy-based heuristics and compare our LP-based algorithms against those LP-agnostic heuristics. Experimental results on the real dataset show our LP-based approaches are robust and effective in a wide range of settings and they can universally dominate all LP-agnostic heuristics.

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 $[k] \doteq \{1, 2, \ldots, k\}$ for any positive integer k.

Two-Stage Resource Allocation (TS-RA). We have a bipartite graph G of vertex sets $(I, H \uplus J)$ where I and H represent the sets of *types* of offline vertices, and J disjointed from H represents the set of *types* of online vertices. Let E^1 and E^2 be the respective sets of edges in the subgraphs G^1 of (I, H) and G^2 of (I, J). There are K offline resources and each resource k has a budget $B_k \in \mathbb{R}^+$. Each edge or assignment e yields a positive reward (weight) $w_e > 0$ and incurs a vector-valued cost $\mathbf{a}_e = (a_{e,k}) \in [0, 1]^K$, where $a_{e,k}$ denotes the consumption of resource k by e. The assignment process consists of two *sequential* stages, (offline) Phase I and (online) Phase II. We first select an arbitrary set of edges E_1 (can be a multiset of E^1) in Phase I and then in Phase II, we conduct an online assignment process on the subgraph G^2 with I and J be the respective sets of offline and online vertices and output a set of edges E_2 (can be a multiset of E^2). Our overall goal is to design an allocation policy to maximize the expected total weights of all edges in $E_1 \cup E_2$ while the total cost of all edges in $E_1 \cup E_2$ does not exceed the budget $\mathbf{B} = (B_k)$ (the budget constraint is enforced throughout Phase I and Phase II).

Note that the full graph G of $(I, H \uplus J)$ is known as part of the input[2]. The only stochastic part to the algorithm is the arrival sequence of online vertices in Phase II, which is specified as follows. We have a finite horizon T (known in advance). For each time (or round) $t \in [T] \doteq \{1, 2, \ldots, T\}$, a vertex of type[3] $j \in J$ will be sampled (or j arrives) from a known distribution $\{p_{jt}\}$ such that $\sum_{j \in J} p_{jt} = 1$, and the sampling procedures are independent across the T rounds. Upon the arrival of an online vertex j, an *immediate and irrevocable* decision is required: either reject j, or select an edge $e \in E^2_j$,

[2]A future direction is to explore how to incorporate online learning into our framework to learn the input dynamically.

[3]The two terms "a vertex of type j" and "a vertex j" will be used interchangeably when the context is clear.

where $E_j^2 \subseteq E^2$ is the set of incident edges to j in G^2. Observe that there could be multiple arrivals for each given type of online vertices, we can possibly select multiple copies of a single edge $e \in E^2$ in Phase II.

Remarks. First, the algorithms presented in this paper are applicable directly to a more general setting where each assignment e yields a *random* reward W_e (independent from others). What we need is just the expectation $w_e \doteq \mathbb{E}[W_e]$ for each e. Second, we cannot simply merge Phase I to part of Phase II. Recall that in Phase II, we have to make an irrevocable matching decision upon the arrival of each online vertex. Suppose we try to treat Phase I as part of Phase II by viewing each arrival $h \in H$ with probability 1 during the first $|H|$ rounds. This reduction will essentially restrict the power of algorithm design in Phase I in the way that we should process all assignments in E^1 following the arrival order of vertices of H. In fact, we can process all assignments in E^1 simultaneously since the graph G^1 is fully known in advance.

Integral and non-integral resources. For an integral resource k, we have that for any edge $e \in E^1 \cup E^2$, $a_{e,k} \in \{0, 1\}$ 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 edge e, let $\mathcal{S}_e$ be the set of resources requested by e, *i.e.,* $\mathcal{S}_e = \{k \in \mathcal{K} : a_{e,k} > 0\}$. In this paper, we assume that $|\mathcal{S}_e \cap \mathcal{K}_1| \le \ell_1$ and $|\mathcal{S}_e \cap \mathcal{K}_2| \le \ell_2$, where ℓ_1 and ℓ_2 are the integral and non-integral sparsities (*i.e.,* for types of resources), respectively.

Let us briefly show how *rebalancing in BSSs* can be cast under TS-RA (similarly for trip-vehicle dispatching). Suppose we have two sets, C_1 and C_2, which denote the respective sets of supplying and demanding bike stations. Let each $a \in C_1$ have a supply of $c_a \in \mathbb{Z}^+$ and each $b \in C_2$ have a demand of $c_b \in \mathbb{Z}^+$. Define $I = C_1 \times C_2$, $H = \{h\}$ includes one dummy node, and J is the set of all online worker types. Consider a given $i = (a, b)$ with $a \in C_1, b \in C_2$. Let the edge $e = (i, h)$ denote the assignment of the task of moving one bike from a to b via TBR and $e = (i, j)$ denote the assignment of the same task to a worker of type j via CBR. In this way, we have $|C_1|+|C_2|$ kinds of integral resources and each $a \in C_1$ and $b \in C_2$ have a budget of c_a and c_b, respectively. We have one non-integral resource, which is the global budget used to pay rented trucks, labor fees, and online workers. Though we potentially have a huge number of resources ($|C_1| + |C_2| + 1$), each assignment requests at most two integral resources (the supplying and demanding stations) and one non-integral resource (the global budget), *i.e.,* $\ell_1 = 2, \ell_2 = 1$. Later we will show that the performance of our allocation policy depends only on the total number of sparsity $\ell_1 + \ell_2$, regardless of the total number of resources.

Extension of competitive ratio. Competitive ratio (CR) is a commonly-used metric to evaluate the performance of online algorithms. In our case, we have two stages, offline Phase I and online Phase II. We can extend CR to our setting as follows. Recall that our overall goal is to maximize the total weights of all assignments made in the two phases. Let ALG denote an assignment policy for Phases I and II. Consider an input $\mathcal{I}$ of the problem. Let $\mathbb{E}[\text{ALG}(\mathcal{I})]$ denote the expected profit obtained by ALG for this instance, where the expectation is over the randomness in the arrival sequence in Phase II and any internal randomness wired in the algorithm. Similarly, let $\mathbb{E}[\text{OPT}(\mathcal{I})]$ denote the expected value of the optimal offline solution (*i.e.,* the expected value of optimal assignments for Phases I and II after observing the entire arrival sequence in Phase II). The competitive ratio is defined as $\inf_{\mathcal{I}} \mathbb{E}[\text{ALG}(\mathcal{I})]/\mathbb{E}[\text{OPT}(\mathcal{I})]$.

For any maximization problem such as the one studied here, we say ALG achieves a ratio at least $\alpha \in (0, 1)$ if for any input 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 $\mathbb{E}[\text{OPT}(\mathcal{I})]$ directly is hard. A common method to bypass this is to construct a linear program (called *benchmark* LP) whose optimal value is an upper bound on $\mathbb{E}[\text{OPT}(\mathcal{I})]$ due to the integrality gap. Hence comparing $\mathbb{E}[\text{ALG}(\mathcal{I})]$ 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.

Benchmark LP. Recall that $E_j^2 \subseteq E^2$ is the set of incident edges to j in G^2. Consider an offline optimal algorithm. For each $e \in E^1$, let x_e be the expected number of copies that e is selected during Phase I and for each $e \in E^2$ and $t \in [T]$, let $y_{e,t}$ be the probability that edge e is selected at t. Consider the following benchmark LP.

$$\max \textstyle\sum_{e \in E^1} w_e x_e + \sum_{e \in E^2, t \in [T]} w_e y_{e,t} \tag{1}$$

$$\text{s.t.} \textstyle\sum_{e \in E_j^2} y_{e,t} \le p_{jt}, \qquad \forall j \in J, t \in [T] \tag{2}$$

$$\textstyle\sum_{e \in E^1} x_e a_{e,k} + \sum_{e \in E^2, t \in [T]} y_{e,t} a_{e,k} \le B_k, \quad \forall k \in [K] \tag{3}$$

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

This LP can be interpreted as follows. Constraint (2) : for any given vertex of type j and time t during Phase II, the probability that we select an edge incident to j is at most the probability that j arrives at time t. Constraint (3) : for any resource k, the expected consumption cannot be larger than its budget (B_k). The last constraint (4) is due to the fact that all $\{y_{e,t}\}$ are probability values and hence should lie in the interval $[0, 1]$. Also x_e should be non-negative since it denotes the number of copies that e is selected. The above analysis suggests that any offline optimal algorithm, $\{x_e, y_{e,t}\}$ should be feasible for the above LP. Formally, we have Lemma 1 which claims that the optimal solution of this LP is an upper bound on the expected offline optimal value. We defer the proof of Lemma 1 to the full version [4].

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

3 Main Algorithm and Results

In this section, we describe our main algorithm SAMP, which is an LP-guided sampling policy.

Let $\{x_e^*, y_{e,t}^*\}$ be an optimal solution to the LP (1). The main idea behind SAMP (described in Algorithm 1) is as follows.

[4]https://tinyurl.com/ybfyg9cx

Algorithm 1 SAMP(η, α)

```
    Offline Phase I:
1   Independently sample $\mathrm{RD}(\eta x^*_e)$ copies of edges for each
      $e \in E^1$.
2   Remove all copies of $e$ if $e$ is not safe (i.e., $e$ requests
      some resource on which there is a budget overrun).
    Online Phase II:
3   Assume a vertex (of type) $j$ arrives at time $t$.
4   Sample an edge $e$ from $E^2_j$ with probability $\alpha y^*_{e,t}/p_{jt}$.
5   If $e$ is safe (i.e., adding $e$ will not break any budget
      constraint at $t$), then select it; otherwise reject it.
```

For a given non-negative value $x \geq 0$, let $\mathrm{RD}(x)$ be such an integral random variable that $\mathrm{RD}(x) = \lceil x_e \rceil$ with probability $x - \lfloor x \rfloor$ and $\mathrm{RD}(x) = \lfloor x \rfloor$ otherwise. Essentially $\mathrm{RD}(x)$ takes values between $\lfloor x \rfloor$ and $\lceil x \rceil$ with $\mathbb{E}[\mathrm{RD}(x)] = x$. In the offline Phase I, we independently sample $\mathrm{RD}(\eta x^*_e)$ copies of edges for each $e \in E^1$, where $\eta \in (0, 1]$ is a scaling factor to optimize later. Let $\mathcal{E}$ be the random multiset of all edges sampled in Phase I. Notice that the total consumption of $\mathcal{E}$ can potentially violate budget constraints for some resources. If so, then remove all copies of e if any number of e contributes to the violation[5]. In the online Phase II, suppose a vertex (of type) j arrives at time t. We say an edge $e \in E^2_j$ is *safe* iff adding e will not violate any budget constraint (*i.e.,* the remaining budget at t is enough to cover all resources requested by e). Sample an edge e from E^2_j with probability $\alpha y^*_{e,t}/p_{j,t}$ and select it iff e is safe. Here $\alpha \in (0, 1]$ is scaling factor to optimize later. Note that Step 4 of SAMP is well defined since we have $\sum_{e \in E^2_j} \alpha y^*_{e,t}/p_{jt} \leq \sum_{e \in E^2_j} y^*_{e,t}/p_{jt} \leq 1$, due to Constraint (2) in LP (1).

3.1 Main Theoretical Results

Theorem 1 (Performance of SAMP for the integral case). *For* TS-RA *when all resources are integral with a sparsity of ℓ,* SAMP *with $\eta = \alpha = \frac{1}{2\ell}$ achieves a competitive ratio of at least $\frac{1}{4\ell}$ using* LP (1) *as the benchmark.*

Theorem 2 (Performance of SAMP for the general case). *For* TS-RA *when both integral and non-integral resources are involved with respective sparsities of ℓ_1 and ℓ_2,* SAMP *with $\eta = \alpha = \frac{1}{2\ell}$ achieves a competitive ratio of at least $\frac{1}{4\ell}(1 - \epsilon)$ using* LP (1) *as the benchmark, where $\ell = \ell_1 + \ell_2$ and $\epsilon = 2 \max\left(1/(B-2), \ell_2 \exp(-B/12 + 1/6)\right)$ with B being the minimum budget over all non-integral resources.*

The main technique used here is a combination of randomized rounding with alterations introduced to solve packing integer programs [Bansal *et al.*, 2012] and randomized sampling for online resource allocation [Dickerson *et al.*, 2019]. Generally, we cannot beat the ratio of $1/(\ell - 1 + 1/\ell)$ for the integral case with a sparsity of ℓ using LP (1) as the benchmark, due to the hardness result derived from the set packing problem [Füredi *et al.*, 1993]. This suggests that the performance of SAMP is almost a constant factor (1/4) from the optimal for the integral case. For the general case, Theorem 2 implies that we can simply treat non-integral resources as integral and SAMP will achieve nearly the same ratio when the budgets among all non-integral resources are moderately large $\Omega(1/\epsilon)$. Note that in the absence of a lower bound on the budgets of all non-integral resources, LP-based randomized samplings can have an arbitrarily bad performance for online resource allocation, as shown in [Dickerson *et al.*, 2019]. We will present the proof of Theorem 1 in Section 4 and defer the proof of Theorem 2 to the full version [4].

[5]Theoretically, we can act in this way, though we can implement the removal of copies sequentially until no violation of budget.

4 Performance Analysis of SAMP

For each $e \in E^1$, let X_e be the random number of copies of e selected in Phase I and for each $e \in E^2$ and $t \in [T]$, let $Y_{e,t}$ indicate if e is selected at t in Phase II by SAMP. Suppose for a given target constant $\gamma \in (0, 1)$, we can show that $\mathbb{E}[X_e] \geq \gamma x^*_e$ for all $e \in E^1$ and $\mathbb{E}[Y_{e,t}] \geq \gamma y^*_{e,t}$ for all $e \in E^2$ and $t \in [T]$. Then by linearity of expectation, the expected total rewards should be at least $\gamma\left(\sum_{e \in E^1} w_e x^*_e + \sum_{e \in E^2} w_e y^*_{e,t}\right)$, which is followed by that SAMP achieves an online ratio at least γ from Lemma 1.

For Phase I, we have the following lemma.

Lemma 2. *For each $e \in E^1$, we have $\mathbb{E}[X_e] \geq \eta(1 - \eta)^\ell x^*_e$.*

Proof. According to SAMP, we first sample $\mathrm{RD}(\eta x^*_e)$ copies of edge e and then include all of them iff e is *safe*, *i.e.,* all resources requested by e have no budget overrun. Consider a given $e \in E^1$ and recall that $\mathcal{S}_e \subseteq \mathcal{K}_1$ is the set of all integral resources requested by e. By the sparsity assumption, we have $|\mathcal{S}_e| \leq \ell$.

Consider a given $k \in \mathcal{S}_e$ and thus $a_{e,k} = 1$. For each $e' \in E^1$ including $e' = e$, let $Z_{e'} = \mathrm{RD}(\eta x^*_{e'})$. Notice that

$$\Pr\left[\sum_{e' \in E^1} Z_{e'} a_{e',k} > B_k \mid Z_e = \lceil \eta x^*_e \rceil\right] \tag{5}$$

$$= \Pr\left[\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k} \geq B_k + 1 - \lceil \eta x^*_e \rceil\right] \tag{6}$$

$$\leq \Pr\left[\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k} \geq B_k - x^*_e\right] \tag{7}$$

$$\leq \mathbb{E}\left[\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k}\right] / (B_k - x^*_e) \leq \eta \tag{8}$$

Inequality (7) is due to $\lceil \eta x^*_e \rceil \leq \lceil x^*_e \rceil \leq x^*_e + 1$. Inequality (8) is due to Markov's inequality and the fact that $\mathbb{E}\left[\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k}\right] = \sum_{e' \in E^1, e' \neq e} \eta x^*_{e'} a_{e',k} \leq \eta(B_k - x^*_e)$ from Constraint (3) in LP (1). We can verify that in the other case when Z_e gets rounded down to $\lfloor \eta x^*_e \rfloor$, the probability that resource k has a budget overrun should be no larger than that when Z_e is rounded up to $\lceil \eta x^*_e \rceil$. Therefore, we claim that regardless the outcomes of $Z_e = z_e \in \{\lceil \eta x^*_e \rceil, \lfloor \eta x^*_e \rfloor\}$, there is a budget overrun on resource k with probability at most η.

Observe that $\{Z_{e'} | e' \in E^1\}$ are all independent random variables and events $\{(\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k} \geq B_k - z_e a_{e,k}) | k \in \mathcal{S}_e\}$ are decreasingly likely events for each given z_e. Thus applying the FKG inequality [Fortuin *et al.*, 1971], the probability that e is safe should be

$$\Pr\left[\bigwedge_{k \in \mathcal{S}_e}\left(\sum_{e' \in E^1, e' \neq e} Z_{e'} a_{e',k} \geq B_k - z_e a_{e,k}\right)\right] \geq (1 - \eta)^\ell$$

Note that the above inequality is valid regardless the choices of $z_e \in \{\lceil \eta x_e^* \rceil, \lfloor \eta x_e^* \rfloor\}$. Therefore, $\mathbb{E}[X_e] \geq \mathbb{E}[Z_e] \cdot (1-\eta)^\ell = \eta(1-\eta)^\ell x_e^*$. □

For Phase II, we have the following lemma.

Lemma 3. *For each $e \in E^2$ and $t \in [T]$, we have $\mathbb{E}[Y_{e,t}] \geq \alpha\big(1 - \ell \cdot \max(\eta, \alpha)\big) y_{e,t}^*$.*

Proof. Consider a given $t \in [T]$, $e = (i,j) \in E^2$ and $k \in \mathcal{S}_e$. Let $\tau_1 = \sum_{e' \in E^1} x_{e'}^* a_{e',k}$ and $\tau_2 = \sum_{e' \in E^2, t' < t} y_{e',t'}^* a_{e',k}$. From Constraint (3) in LP (1), $\tau_1 + \tau_2 \leq B_k$.

Assume j arrives at time t, which occurs with probability p_{jt}. Let U_1 and $U_{2,t}$ be the respective (random) consumptions of resource k during Phase I and at the beginning of time t during Phase II. Notice that the event occurs that e is safe at t with respect to resource k, denoted by $\mathsf{SF}_{e,t,k}$, iff $U_1 + U_{2,t} \leq B_k - 1$ (since $a_{e,k} = 1$).

$$\Pr[\mathsf{SF}_{e,t,k}] = 1 - \Pr[U_1 + U_{2,t} \geq B_k] \geq 1 - \tfrac{\mathbb{E}[U_1 + U_{2,t}]}{B_k} \quad (9)$$

Observe that $\mathbb{E}[U_1] \leq \sum_{e' \in E^1} \mathbb{E}[\mathrm{RD}(\eta x_{e'}^*) \cdot a_{e',k}] \leq \eta \tau_1$. For each $t' < t$ and $j \in J$, let $\mathbf{1}_{j,t'}$ indicate if j arrives at t'. For each $e' \in E_j^2$, let $\mathbf{1}_{e',t'}$ indicate if e' is sampled when j comes at t'. Thus,

$$\begin{aligned}\mathbb{E}[U_{2,t}] &\leq \textstyle\sum_{t'<t, j \in J, e' \in E_j^2} \mathbb{E}[\mathbf{1}_{j,t'} \cdot \mathbf{1}_{e',t'} \cdot a_{e',k}] \\ &\leq \textstyle\sum_{t'<t, j \in J, e' \in E_j^2} p_{j,t'} \cdot (\alpha y_{e',t'}^* / p_{j,t'}) \cdot a_{e',k} \leq \alpha \tau_2\end{aligned}$$

Substituting results on $\mathbb{E}[U_1]$ and $\mathbb{E}[U_{2,t}]$ back to Equation (9), we have

$$\Pr[\mathsf{SF}_{e,t,k}] \geq 1 - \tfrac{\eta\tau_1 + \alpha\tau_2}{B_k} \geq 1 - \max(\eta, \alpha)$$

The last inequality is due to $\tau_1 + \tau_2 \leq B_k$. Therefore, for the edge $e = (i,j)$,

$$\begin{aligned}\mathbb{E}[Y_{e,t}] &= \Pr[(\mathbf{1}_{j,t} = 1) \wedge (\mathbf{1}_{e,t} = 1) \wedge (\wedge_{k \in \mathcal{S}_e} \mathsf{SF}_{e,t,k})] \\ &\geq p_{jt} \cdot (\alpha y_{e,t}^* / p_{jt}) \cdot \big(1 - \ell \cdot \max(\eta, \alpha)\big)\end{aligned}$$

Thus, we get our claim. □

Proof of main Theorem 1.

Proof. From Lemmas 2 and 3, the final online competitive ratio achieved by $\mathsf{SAMP}(\eta, \alpha)$ should be $\min\big(\eta(1-\eta)^\ell, \alpha(1 - \ell \cdot \max(\eta, \alpha))\big)$. By choosing $\eta = \alpha = 1/(2\ell)$, we see that the final ratio is $1/(4\ell)$. □

5 Experiments

5.1 The Bike Sharing Dataset and Preprocessing

We use the Citibike dataset in New York City,[6] which contains the trip histories for trips and real-time data for hundreds of stations. Each trip record includes the starting station, the destination station, and the starting and ending time of each trip. In our experiments, we focus on the city of Manhattan.

[6]https://www.citibikenyc.com/system-data

We partition all stations into 50 sites by applying K-medians clustering method, based on the geographical location and Manhattan distance. Note that the clustering method may converge to a local optimum. Basically, we have two kinds of sites, *supplying sites* with a large check-in amount and thus abundant bikes available, and *demanding sites* with a high check-out demand. Since not all sites are highly imbalanced, we focus on the rebalancing among the top 10 *supplying sites* and the top 10 *demanding sites*, which are denoted by C_1 and C_2, respectively. The total set of task types $I = C_1 \times C_2$ and each $i = (a,b) \in I$ represents a task type of moving one bike from $a \in C_1$ to $b \in C_2$. Focus on the morning rush hour of 8 AM to 9 AM from Monday to Thursday in September and October of 2017 (34 days in total). For each $a \in C_1$ and $b \in C_2$, we first compute the average supply (s_a) at site a and the average demand (d_b) at b and then set the final supply $c_a = \gamma \cdot s_a$ for a and the final demand $c_b = \gamma \cdot d_b$ for b, where γ is a parameter adjusting the scale of supply and demand overall. We assume that for TBR, one unit task of each type incurs a uniform cost $\lambda \geq 1$[7], which captures the labor cost of loading one bike up and off a truck and the amortized cost of renting and operating a truck.

Now we discuss the setting for CBR. Let $\mathcal{L}$ be the collection of 50 sites. For each pair $(c,d) \in \mathcal{L} \times \mathcal{L}$, we create a worker type $j = (c,d)$ such that j has the starting and ending sites of c and d, respectively. For a given worker of type $j = (c,d)$, we set that it is qualified for task $i = (a,b)$ if $D(c,a) + D(b,d) \leq \tau$, where $D(\cdot)$ refers to the Manhattan distance between two sites and τ is a given threshold for the additional walking distance. For each assignment $e = (i,j)$, the cost of payment for worker j is set as $\mathsf{Cost}(e) = \rho + (1-\rho)\frac{D(c,a)+D(b,d)}{\tau}$, which consists of a basic rate $\rho \in [0,1]$ (a parameter) and an incentive part proportional to the travel distance due to the rebalancing work. Here we scale down such that the largest cost of payment is 1 among all assignments in CBR.

For each worker of type j, we first compute the average number of trips r_j' during the rush hour over the 34 days. Then we introduce a parameter β to adjust the degree of peak hour and set the final arrival rate $r_j = \beta \cdot r_j'$ and the total arrivals $T = \sum_{j \in J} r_j$. We consider a special arrival setting of KIID, where arrival distributions are assumed identical and independent throughout the online phase. This is mainly due to the short time window considered here (*i.e.,* the rush hour of 8 AM to 9 AM). The same setting is adopted by [Zhao *et al.*, 2019; Dickerson *et al.*, 2018b] for ridesharing and crowdsourcing. After splitting the online phase into T rounds, the arrival distributions are set as $p_{jt} = p_j = r_j/T$ for every $t \in [T]$ and $j \in J$ such that $\sum_{j \in J} p_j = 1$. For each task i, we assign it with a uniformly random weight $w_i \in [0,1]$ and assume all assignments with respect to i return a profit/weight of w_i regardless of TBR or CBR. Set the default total budget as $\kappa \cdot A$, where $\kappa = 0.5$ and A is the expected minimum cost of payment to complete all tasks. Since the average cost of CBR is less than that of TBR for each task, we set A as the product of the average cost of all assignments in CBR and the total num-

[7]Later we will set the maximum cost of assignments in CBR is 1; thus, λ can be viewed as the relative cost of TBR to CBR.

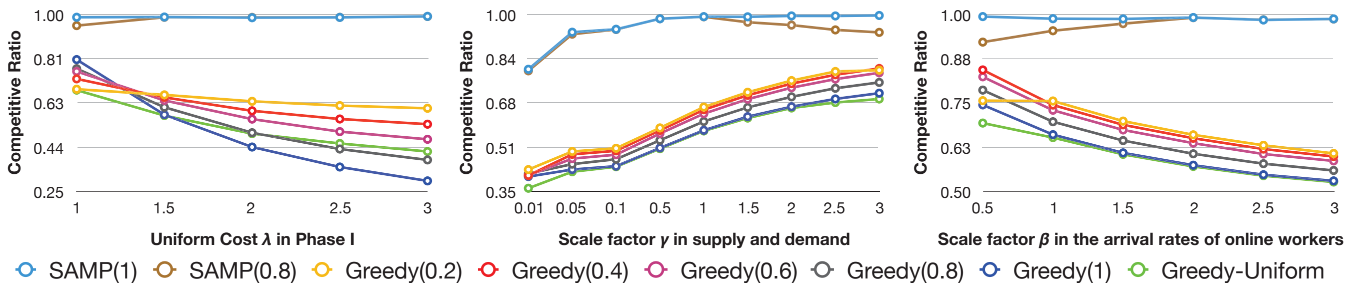

Figure 1: Real dataset: the change of competitive ratios when varying uniform cost in Phase I (Left), scale factor in supply and demand (Middle), and scale factor in the arrival rates of online workers (Right). Default setting is $\lambda = 1.5, \gamma = 2, \beta = 1$.

ber of all potential tasks (equal to $\min(\sum_{a \in C_1} c_a, \sum_{b \in C_2} c_b)$).

Algorithms. By default, we set $\alpha = 1$ and test SAMP(η) (which is short for SAMP($\eta, 1$)) with $\eta = 1$ and $\eta = 0.8$, respectively, against several heuristic baselines, namely Greedy and Greedy-Uniform. Note that for SAMP(1) and SAMP(0.8), they independently sample RD(x^*_e) and RD($0.8x^*_e$) copies of edges for each $e \in E^1$. For each heuristic Greedy(δ) with $\delta \in \{0.2, 0.4, 0.6, 0.8, 1\}$, it first allocates at most a fraction δ of the total budget ($\delta \cdot \mathbf{B}$) to Phase I and all the rest goes to Phase II. In Phase I, Greedy(δ) greedily chooses edges e sequentially in a decreasing order of their weights until allocated budgets are exhausted. In Phase II when a worker of type j arrives, Greedy(δ) will always assign j with an assignment $e \in E^2_j$ which has the maximum weight among all safe choices. Greedy-Uniform can be viewed as a randomized version of Greedy: it first samples a uniform value $q \in [0, 1]$ and then runs Greedy(q). For each given setting, we run all algorithms for 1000 times and take the average as the final performance. We compare the performance of each algorithm against the optimal value to the benchmark LP (1) and use that ratio as the final competitive ratio achieved. Note that the algorithms designed for the MBOA model [Dickerson *et al.*, 2019] are applicable in Phase II only, a future direction is to construct a new LP for these algorithms which can capture the features of the TS-RA model.

5.2 Results on the Real Dataset

Figure 1 (Left) shows the effect of λ, which captures the relative ratio of cost in Phase I to that in Phase II. The results show that for each given $\lambda \geq 1.5$, the performances of Greedy(δ) decreases as δ increases. What is more, the neighboring gaps widen when λ increases. Observe that a larger value of λ implies more costly TBR is and thus, it will be more profitable to allocate more budgets to Phase II. This explains why Greedy(δ) achieves a higher ratio when δ is smaller (thus a higher fraction of budget goes to Phase II). However, SAMP(1) and SAMP(0.8) universally beat all heuristics with a constant gap around 0.3 in the competitive ratios. Figure 1 (Middle) shows the effect of γ, which captures the scale of supply and demand. Note that when γ increases, we increase our default budget value A proportionally. Results show that all heuristics have an increasing performance though SAMP(1) and SAMP(0.8) strictly dominate all of them. This is due to the fact that the more resources we have (with a larger γ), the less planning we need. Thus, the superiority of our LP-based policies is reduced when resources are more abundant. Figure 1 (Right) shows the effect of β, which captures the different scale of arrival rates of worker types. When β increases, we have more arrivals of low-cost workers in Phase II. Thus, we expect that the optimal policy will allocate more budget to Phase II. However, each Greedy(δ) sticks with a fixed fraction of budget on each phase. This can explain why each Greedy(δ) has a decreasing performance. Again, our LP-based policies dominate all the greedy-based heuristics.

Overall, Figure 1 suggests that our LP-based policies can optimize the budget allocation in Phases I and II to well respond to different costs in the two phases. Additionally, they have robust performances, which universally dominate all greedy-based heuristics under various settings. Furthermore, the ratios achieved by SAMP(1) and SAMP(0.8) are far larger than the lower bounds shown in Theorem 2, however. This is due to real-world settings are often faraway from the theoretical worst-case scenario.

6 Conclusions and Future Directions

In this paper, we proposed a unified model TS-RA, which incorporates both offline and online resource allocation into a single framework. Extensive experimental results on the real dataset show the robustness and effectiveness of our LP-based approaches in a wide range of settings. We observe that competitive ratios on experimental results, although guaranteed, are much above those theoretical lower bounds. This fact suggests that the hypothetical worst-case scenario has a structure faraway from those in the real world. A natural direction is to figure out the reasons behind the big gap between the actual performances and theoretical lower bounds. Can we improve the current ratio of $1/(4\ell)$ further? Another interesting question is whether we can remove the requirement of a lower bound of budgets on all non-integral resources.

Acknowledgments

Yifan Xu and Jun Tao's research was partially supported by the CERNET Southeastern China (North) Regional Network Center, CIGRI and the National Key Research and Development Program of China (No. 2018YFB1800205). Pan Xu was partially supported by NSF CRII Award IIS-1948157. Jianping Pan's research was partially supported by NSERC and CFI of Canada. The authors would like to thank the anonymous reviewers for their helpful feedback.

References

[Ashlagi *et al.*, 2019] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Chris Sholley, and Amin Saberi. Edge weighted online windowed matching. In *ACM EC*, pages 729–742, 2019.

[Assadi *et al.*, 2015] Sepehr Assadi, Justin Hsu, and Shahin Jabbari. Online assignment of heterogeneous tasks in crowdsourcing markets. In *HCOMP*, pages 12–21, 2015.

[Bansal *et al.*, 2012] Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1), 2012.

[Bei and Zhang, 2018] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *AAAI*, pages 3–9, 2018.

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

[Dickerson *et al.*, 2018a] John Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In *AAAI*, pages 1007–1014, 2018.

[Dickerson *et al.*, 2018b] John Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals. In *AAMAS*, pages 318–326, 2018.

[Dickerson *et al.*, 2019] John Dickerson, Karthik Abinav Sankararaman, Kanthi Kiran Sarpatwar, Aravind Srinivasan, Kun-Lung Wu, and Pan Xu. Online resource allocation with matching constraints. In *AAMAS*, pages 1681–1689, 2019.

[Duan and Wu, 2019] Yubin Duan and Jie Wu. Optimizing the crowdsourcing-based bike station rebalancing scheme. In *ICDCS*, pages 1559–1568, 2019.

[Fortuin *et al.*, 1971] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971.

[Füredi *et al.*, 1993] Zoltán Füredi, Jeff Kahn, and Paul D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2), 1993.

[Ghodsi *et al.*, 2012] Ali Ghodsi, Vyas Sekar, Matei Zaharia, and Ion Stoica. Multi-resource fair queueing for packet processing. In *ACM SIGCOMM*, pages 1–12, 2012.

[Ghodsi *et al.*, 2013] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *EuroSys*, pages 365–378, 2013.

[Guedes *et al.*, 2014] Ricardo Guedes, Vasco Furtado, and Tarcisio Pequeno. Multiagent models for police resource allocation and dispatch. In *JISIC*, pages 288–291, 2014.

[Haider *et al.*, 2018] Zulqarnain Haider, Alexander Nikolaev, Jee Eun Kang, and Changhyun Kwon. Inventory rebalancing through pricing in public bike sharing systems. *EJOR*, 270(1):103–117, 2018.

[Ho and Vaughan, 2012] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.

[Huang *et al.*, 2019] Taoan Huang, Bohui Fang, Hoon Oh, Xiaohui Bei, and Fei Fang. Optimal trip-vehicle dispatch with multi-type requests. In *AAMAS*, 2019.

[Lee *et al.*, 1979] Sang M. Lee, Lori Sharp Franz, and A. James Wynne. Optimizing state patrol manpower allocation. *Journal of the Operational Research Society*, 30(10), Oct 1979.

[Li *et al.*, 2018] Yexin Li, Yu Zheng, and Qiang Yang. Dynamic bike reposition: A spatio-temporal reinforcement learning approach. In *KDD*, pages 1724–1733, 2018.

[Liu *et al.*, 2016] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *KDD*, pages 1005–1014, 2016.

[Lowalekar *et al.*, 2018] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, 261:71 – 112, 2018.

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

[O'Mahony and Shmoys, 2015] Eoin O'Mahony and David B Shmoys. Data analysis and optimization for (Citi) bike sharing. In *AAAI*, pages 687–694, 2015.

[Pan *et al.*, 2019] Ling Pan, Qingpeng Cai, Zhixuan Fang, Pingzhong Tang, and Longbo Huang. A deep reinforcement learning framework for rebalancing dockless bike sharing systems. In *AAAI*, pages 1393–1400, 2019.

[Raviv *et al.*, 2013] Tal Raviv, Michal Tzur, and Iris A Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, 2013.

[Singla *et al.*, 2015] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. Incentivizing users for balancing bike sharing systems. In *AAAI*, pages 723–729, 2015.

[Yi *et al.*, 2007] Xing Yi, James Allan, and W Bruce Croft. Matching resumes and jobs based on relevance models. In *SIGIR*, pages 809–810, 2007.

[Zhao *et al.*, 2019] Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *AAAI*, pages 2245–2252, 2019.