Competitive Ratios for Online Multi-capacity Ridesharing

Meghna Lowalekar
School of Information Systems,
Singapore Management University
meghnal.2015@phdcs.smu.edu.sg

Pradeep Varakantham
School of Information Systems,
Singapore Management University
pradeepv@smu.edu.sg

Patrick Jaillet
Dept. of Electrical Engineering and
Computer Science, Massachusetts
Institute of Technology, USA
jaillet@mit.edu

ABSTRACT

In multi-capacity ridesharing, multiple requests (e.g., customers, food items, parcels) with different origin and destination pairs travel in one resource. In recent years, online multi-capacity ridesharing services (i.e., where assignments are made online) like Uber-pool, foodpanda, and on-demand shuttles have become hugely popular in transportation, food delivery, logistics and other domains. This is because multi-capacity ridesharing services benefit all parties involved – the customers (due to lower costs), the drivers (due to higher revenues) and the matching platforms (due to higher revenues per vehicle/resource). Most importantly these services can also help reduce carbon emissions (due to fewer vehicles on roads).

Online multi-capacity ridesharing is extremely challenging as the underlying matching graph is no longer bipartite (as in the unit-capacity case) but a tripartite graph with resources (e.g., taxis, cars), requests and request groups (combinations of requests that can travel together). The desired matching between resources and request groups is constrained by the edges between requests and request groups in this tripartite graph (i.e., a request can be part of at most one request group in the final assignment). While there have been myopic heuristic approaches employed for solving the online multi-capacity ridesharing problem, they do not provide any guarantees on the solution quality.

To that end, this paper presents the first approach with bounds on the competitive ratio for online multi-capacity ridesharing (when resources rejoin the system at their initial location/depot after serving a group of requests). The competitive ratio is : (i) 0.31767 for capacity 2; and (ii) γ for any general capacity κ, where γ is a solution to the equation $\gamma = (1-\gamma)^{\kappa+1}$.

ACM Reference Format:
Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2020. Competitive Ratios for Online Multi-capacity Ridesharing. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Motivated by multiple online to offline services including point-to-point transportation, food delivery, logistics, etc., online matching problems have received tremendous interest in the recent years. Specifically, on-demand unit-capacity (e.g., UberX, Lyft) and multi-capacity (e.g., Uberpool, Lyftline, Deliveroo, Food Panda) ridesharing services have become hugely popular in many cities around the world. In these platforms, resources have to be matched online (in real-time) to either one request (unit-capacity) or a group of requests (multi-capacity) so as to maximize the weight of the matching (e.g., revenue, number of requests served).

Given the win-win properties of multi-capacity ridesharing to all the concerned parties (customers, drivers, matching platform) and the environment (through reduced carbon emissions), we are interested in developing a performance guaranteed approach for multi-capacity ridesharing.

There are two major threads of relevant research. The *first thread* is on online unit-capacity ridesharing where the underlying problem is an online bipartite matching problem. The standard online bipartite matching problem involves matching known (i.e., available offline) disposable resources [1] on one side to the online arriving vertices/requests on the other side, over multiple timesteps. Many approaches provide performance guarantees under different arrival assumptions for incoming vertices [6, 11, 12]. Mehta [16] provides a detailed survey of the same. One popular arrival assumption is the known identical independent distribution (KIID) [11, 15], where online vertices arrive over T rounds and their arrival distributions are assumed to be identically distributed and independent over T rounds. This distribution is also known to the online algorithm in advance. The existing literature provide bounds of at least $1 - \frac{1}{e}$ on the expected competitive ratio (ratio of the expected value obtained by the algorithm to the expected value obtained by an offline optimal algorithm) for online bipartite matching problems under KIID.

In case of unit-capacity ridesharing, the offline available resources (i.e., vehicles) are reusable. Dickerson *et.al.* [8] were able to provide a $\frac{1}{2}$ bound for the unit-capacity ridesharing in which resources are reusable and they join the system after serving the requests at the same location. Instead of KIID, they consider that arrival distributions of online vertices can change from time to time (i.e., it is not iid) but this distribution is also known to the algorithm. They refer to this distribution as the Known Adversarial Distribution (KAD).

Unfortunately, this thread of work is only applicable for unit-capacity resources and cannot be directly adapted to consider multi-capacity resources because the underlying problem is no longer an online bipartite matching problem (see below). Another limitation is that the existing work for unit-capacity ridesharing has primarily focused on requests arriving sequentially (i.e., one by one) and not in batches which is a desirable property when considering multi-capacity ridesharing problems (for instance, last mile services at train stations need to consider that the large number of passengers

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

[1] Once a resource is assigned, it can not be used by any other incoming vertex/request.

will arrive and request for last mile transportation to their home at the same time.).

The *second thread* of relevant research is on approaches to solve online multi-capacity (capacity > 1) ridesharing problems. There have been multiple heuristic approaches [2, 14] provided for solving the ridesharing problem for multi-capacity resources in batch arrival model. However, none of these approaches provide any bounds on the performance and are typically myopic (i.e., they do not consider any future information) due to the challenging nature of the problem.

The multi-capacity resources (capacity > 1) make the problem challenging because resources have to be matched to groups of requests and not just to individual requests. This results in a significant change in the structure of the underlying matching graph. Unlike unit-capacity ridesharing, where the underlying graph is bipartite, the multi-capacity ridesharing has a tripartite graph [5] with reusable resources (vehicles), request groups (i.e., combinations of passenger requests) and online vertices (corresponding to passenger requests). The desired matching between the resources and request groups (combination of requests) is constrained by the edges between requests and request groups (i.e., a request can be part of at most one request group in final assignment) in this tripartite graph. It should be noted that this matching problem in tripartite graph is not equivalent to any variant of bipartite matching problem [1, 9, 10, 13] studied in the literature. This is because the weight of a match and the time after which resource becomes available again is dependent on the requests which are paired together in the group assigned to the resource.

To the best of our knowledge, there has been no research on providing performance guaranteed algorithms for such tripartite graphs. There has been some work on solving a part of this matching problem which focused on finding the requests which can be grouped together over time by considering the sequential arrival of requests [3, 4] in the adversarial and random order arrival. However, these works ignore the main component of matching the resources to the request groups.

1.1 Contributions

Our *first* contribution is in designing a performance guaranteed online algorithm that provides a competitive ratio of $\frac{1}{2}$ for the unit-capacity ridesharing problem that considers batch arrival of online vertices [2] under the known arrival distribution. Due to the change in the value obtained by optimal algorithm (more details in Section 3), it is not obvious whether the competitive ratio will increase or decrease or remain the same as compared to the sequential arrival case [8]. Therefore, this is an important result where in we are able to show that the same competitive ratio can be achieved even when the vertices arrive in batches.

Our *second and the main* contribution is to provide a performance guaranteed online algorithm that provides a non-zero competitive ratio for the online multi-capacity ridesharing problems considering batch arrival of online vertices under the known arrival distribution. The competitive ratio is:

- 0.31767 for capacity 2
- γ for any arbitrary capacity κ, where γ is solution to the the expression $(1-\gamma)^{\kappa+1} = \gamma$.

Even though we require groups of vertices in this online algorithm, these groups can be generated offline and hence does not add to the run-time complexity. These general bounds for arbitrary capacity ridesharing are applicable under the assumption that the type of the resources/vehicles (i.e., their location) rejoining the system (after serving a group of vertices) does not change [8].

Finally, we provide simple heuristics (based on the offline optimal LP) which work well in practice (as demonstrated in our experimental results).

Due to space constraints, we are unable to include complete proofs in the paper. Omitted proofs and other specific details with regards to algorithms can be found at this link: https://tinyurl.com/rjs524p

2 BACKGROUND

In this section, we provide the formal definition of expected competitive ratio and the research relevant [8] to the work in this paper.

2.1 Expected Competitive Ratio

The performance of any online algorithm is measured using a metric called competitive ratio. An online algorithm with a competitive ratio of γ is called γ-competitive algorithm. In case of known distribution models, the expected value of the competitive ratio is employed. The expected competitive ratio of any algorithm ALG is defined [16] as $\min_{I,D} \frac{E[ALG(I,D)]}{E[OPT(I)]}$, where I denotes the input and D denotes the arrival distribution and $E[OPT(I)]$ denotes the expected value of the offline optimal algorithm. In general, an upper bound on the value of $E[OPT(I)]$ is provided by using a benchmark linear program. This results in providing a valid lower bound on the resulting competitive ratio.

Since we only employ expected competitive ratio in this paper, we henceforth just refer to it as competitive ratio.

2.2 OM-RR-KAD

We now describe the Online Matching with (Offline) Reusable Resources under Known Adversarial Distributions (OM-RR-KAD) model [8] for ridesharing in which the vehicle capacity is restricted to 1. OM-RR-KAD is a bipartite matching problem between offline reusable resources (e.g., vehicles), $\mathcal{U}$, and vertices that arrive online, $\mathcal{V}$ (e.g., user requests), over T rounds [3]. Online vertices arrive according to a *Known Adversarial Distribution (KAD)* represented by a set of arrival probabilities, $\{p_v^t\}$ ($\sum_v p_v^t = 1, \forall t$). Once an online vertex of type v arrives (i.e., sampled from p_v^t), an irrevocable decision needs to be taken immediately to match it to one of the offline resources, for which a weight, $w_{u,v}^t$ is received, or to reject it. The offline resource becomes unavailable for a few rounds after it is matched and the number of rounds of unavailability, $c_{u,v}^t$, is characterized by an integral distribution, $c_{u,v}^t \in \{1, 2, \ldots, T\}$. The offline resource rejoins the system after $c_{u,v}^t$ rounds. The goal is to design an online assignment policy that will maximize the weight.

There are two key steps in obtaining a performance guaranteed online assignment policy:

[2] The online arriving vertices correspond to the requests. Throughout the paper we use vertices and requests interchangeably.

[3] We use round and timestep interchangeably in the paper

LPSequential:

$$\max \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} w_{u,v}^{t} \cdot x_{u,v}^{t}$$

$$s.t. \quad \sum_{u \in \mathcal{U}} x_{u,v}^{t} \leq p_{v}^{t} ::: \forall v \in \mathcal{V}, 0 \leq t < T \quad (1)$$

$$\sum_{t'=0}^{t} \sum_{v' \in \mathcal{V}} x_{u,v'}^{t'} \cdot Pr[c_{u,v'}^{t'} > t - t'] + \sum_{v \in \mathcal{V}} x_{u,v}^{t} \leq 1 ::: \forall u \in \mathcal{U}, 0 \leq t < T \quad (2)$$

$$0 \leq x_{u,v}^{t} \leq 1 ::: \forall u \in \mathcal{U}, v \in \mathcal{V}, 0 \leq t < T \quad (3)$$

Table 1: Unit Capacity Sequential Arrival

First, an upper bound on the offline optimal, $\mathbf{x}^*$ is computed using the linear program (LP) of Table 1. $x_{u,v}^{*,t}$ denotes the probability of assigning resource u to online vertex of type v in round t. Constraint (1) ensures that the expected number of times a vertex of type v is matched is less than or equal to the expected number of times the vertex is available. Constraint (2) ensures that the resource u is assigned in round t if and only if it is available in round t. It should be noted that this LP provides a solution over all realizations of online vertices and hence that solution may not be applicable to a specific instantiation of online vertex (as the corresponding u may not be available).

Second, an assignment rule is provided to compute the online probability of assigning a resource u for a specific instantiation of online vertex (of type v in round t) and is given by:

$$\frac{x_{u,v}^{*,t} \cdot \gamma}{p_{v}^{t} \cdot \beta_{u}^{t}} \quad (4)$$

where $x_{u,v}^{*,t}$ is a solution to the LP in Table 1 and γ is the desired competitive ratio of the online assignment; and β_u^t is the probability that resource u is safe for assignment in round t. By simulating the current strategy up to t, β_u^t can be estimated with a small error.

The following theorem characterizes the $\frac{1}{2}$ bound on the expected competitive ratio.

Theorem 1. *Dickerson* et.al.*[[8]] The optimal value of LPSequential in Table 1 provides a valid upper bound on the offline optimal value for OM-RR-KAD. The online assignment rule of Equation 4 based on the LP achieves an online competitive ratio of $\frac{1}{2} - \epsilon$ for any given $\epsilon > 0$.*

The ϵ factor comes in the competitive ratio due to the error in the estimation of β_u^t. For a clean presentation, throughout the paper, we assume that these values can be estimated correctly and ignore the estimation error.

3 BATCH ARRIVAL OF VERTICES

In ridesharing problems, user requests typically arrive in batches instead of arriving sequentially (e.g., users coming out of a train, theatre or mall looking for shared rides). So, we extend the OM-RR-KAD model to consider batch arrival of online vertices and also provide an online algorithm that achieves the same competitive ratio of $\frac{1}{2}$ as in the sequential arrival case. Batch arrival is different from sequential arrival because multiple online vertices (more information at each step) have to be matched to multiple offline resources at each round.

Since there are more vertices available in each round, online algorithms can potentially make better assignments in the batch case as compared to the sequential case. Due to this, it seems that the competitive ratio in the batch arrival case will be higher than the sequential arrival case. However,

- As the assignment for any vertex should be made in the same round of its arrival, in batch case where each round has multiple vertices, optimal algorithm (denominator of competitive ratio) also considers a greater number of vertices in each round and hence optimal value can also improve (as compared to the optimal value for sequential case).
- Compared to the sequential case, more time is spent deliberating (since we must wait until end of batch to make assignments) and during that time no assignment will happen and hence the number of vertices assigned by the optimal algorithm can be lower.

Therefore, the relationship between the competitive ratio for the sequential and batch cases is non trivial. We now provide an algorithm which ensures that the competitive ratio in batch arrival case is equal to the sequential arrival case.

We first mention the changes required in OM-RR-KAD model for the batch arrival case and then provide the performance guaranteed online algorithm for the unit-capacity case.

Changes to OM-RR-KAD for Batch Case: In the OM-RR-KAD model, at each round t, a single vertex is sampled using the probability $\{p_v^t\}$. However, in the batch extension, b^t vertices arrive at each round and each of these b^t vertices is sampled using the same probabilities $\{p_v^t\}$. The expected number of vertices of type v arriving in round t is q_v^t and is given by:

$$q_{v}^{t} = b^{t} \cdot p_{v}^{t}$$

LP for Upper Bound on Offline Batch Optimal, LPBatch: The optimization formulation for the batch case is same as the LP in Table 1, except for the constraint in Equation (1). Given that there are q_v^t (and not p_v^t) expected arrivals of vertex of type v at each round, the modified constraint is:

$$\sum_{u \in \mathcal{U}} x_{u,v}^{t} \leq q_{v}^{t} ::: \forall v \in \mathcal{V}, 0 \leq t < T \quad (5)$$

We will refer to the modified LP as LPBatch.

Proposition 1. *The optimal value of LPBatch provides a valid upper bound on the offline optimal value* [4].

ADAPBatch The online algorithm presented in Algorithm 1 is used to make an online assignment of the resources to the incoming vertices that are arriving in batches. We use an adaptive algorithm [5]

[4] Proof is omitted due to space constraints.

[5] For an LP-based algorithm, we say that the algorithm is adaptive if for a given LP solution, the computation of strategy in each round t depends on the strategies in the previous rounds [7].

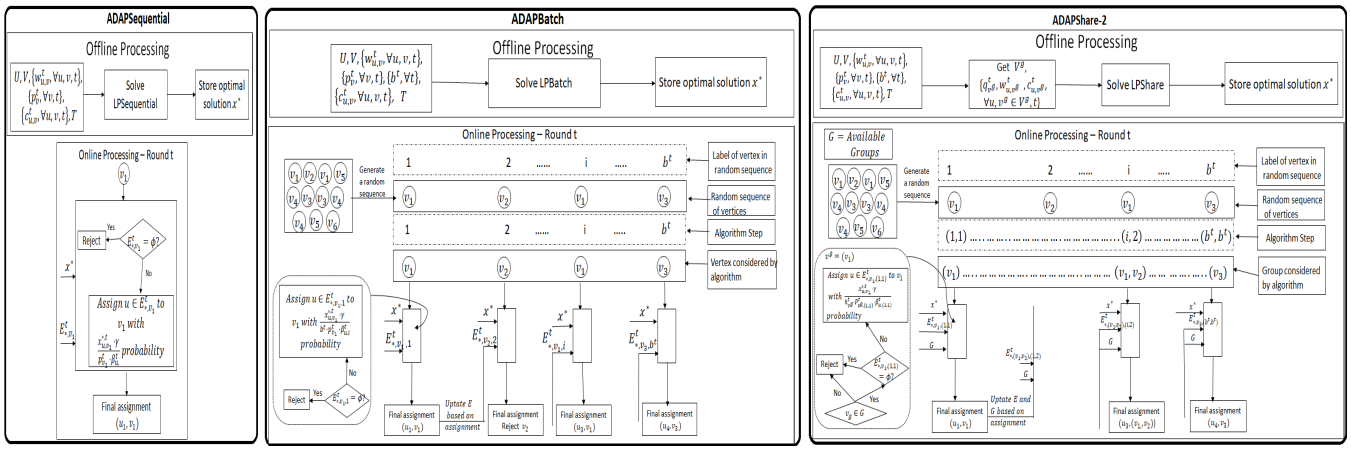


Figure 1: The Figure depicts the difference in the processing of algorithms in unit-capacity sequential, unit-capacity batch and multi-capacity share case. The online component in each of the algorithms corresponds to the processing in round t for a single instance of arrival of vertices. We only show the detailed flow diagram in the first block for each of the algorithms, rest of the blocks will have similar flow.

Algorithm 1: ADAPBatch(γ)

1: **for** $t < T$ **do**
2: Generate a random shuffling of the incoming b^t vertices. Label the vertices from 1 to b^t.
3: **for** $i = 1$ to b^t **do**
4: v = type of vertex with label i
5: **If** $E^t_{*,v,i} = \phi$, then reject the vertex with label i;
6: **Else** choose $u \in E^t_{*,v,i}$ with probability $\frac{x^{*,t}_{u,v} \cdot \gamma}{q^t_v \cdot \beta^t_{u,i}}$
7: Update the sets $E^t_{*,v,j}$ for all $j > i$ based on the assignment.

that employs the probability of a resource being safe (available for assignment) while making assignments. The assignment rule to compute the online probability of assigning a resource u for the vertex of type v with label i in round t is:

$$\frac{x^{*,t}_{u,v} \cdot \gamma}{b^t \cdot p^t_v \cdot \beta^t_{u,i}}$$

where $\beta^t_{u,i}$ denotes the probability that resource u is safe in round t when the vertex with label i is being considered; and $E^t_{*,v,i} \subset \mathcal{U}$ is used to denote the set of safe neighbours for a vertex of type v in round t when the vertex with label i is being considered.

In the algorithm, we process the vertices that have arrived in a batch one by one by considering a uniform random shuffling of incoming vertices. The intuition behind the assignment rule is to divide the optimal assignment for round t uniformly into b^t steps ($\frac{x^{*,t}_{u,v}}{b^t}$) and then to make sure that the vertex of type v is matched to resource u at any step with probability $\frac{x^{*,t}_{u,v} \cdot \gamma}{b^t}$ unconditionally. Another key change in the algorithm from the sequential case is the last step where the availability of offline resources is updated based on assignments made in the same round. Figure 1 highlights the difference in the way the algorithms process online information in the sequential and batch case.

Proposition 2. *The online algorithm ADAPBatch is $\frac{1}{2}$ competitive.*

Proof Sketch: The maximum value of γ for which the algorithm ADAPBatch is valid [6] is $\gamma = \frac{1}{2}$. The proof involves showing that the minimum possible value of $\beta^t_{u,i}$ is $\frac{1}{2}$, for which we use mathematical induction. Finally, we show that ADAPBatch is γ competitive and since the maximum value of γ for which the assignment rule is valid is $\frac{1}{2}$, the algorithm is $\frac{1}{2}$ competitive. ■

4 MULTI-CAPACITY REUSABLE RESOURCES

In this section, we provide a model, an online algorithm and competitive ratio analysis for the online multi-capacity ridesharing problem with reusable resources.

4.1 Model: OPERA

To address the challenges associated with multi-capacity resources, we propose a new model called OPERA (**O**nline matching with offline multi-ca**P**acity r**E**usable **R**esources in b**A**tch Arrival Model). In OPERA, online vertices arrive in batches according to a *Known Adversarial Distribution (KAD)*. Once the online vertices arrive, there has to be an irrevocable decision made immediately on matching each offline resource u to a group of online vertices v^g. The groups chosen for all vehicles should be such that each online vertex appears in at most one group. For each assignment of an offline resource u to a group of online vertices of type v^g in round t, a weight w^t_{u,v^g} is received. After the assignment, the offline resource u is unavailable for c^t_{u,v^g} rounds before joining the system again [7]. The goal is to design an online assignment policy for assigning offline reusable resources to the groups of online vertices that will maximize the weight received over all time steps.

[6] Algorithm is valid when the assignment rule probability lies between 0 and 1.
[7] In the context of last mile ridesharing – after serving the group of passengers, vehicle comes back to its initial location

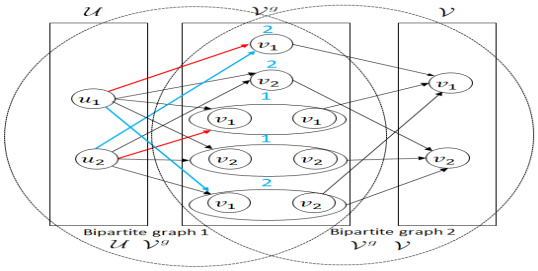


Figure 2: The Figure depicts the tripartite graph used in the OPERA model. It is a combination of 2 bipartite graphs. The goal is to find the matching in the first bipartite graph subject to the constraints enforced due to the edges present in the second bipartite graph. The blue numbers in $\mathcal{V}$ indicate the number of vertices of each type available and blue numbers in $\mathcal{V}^g$ denote the number of groups of each type which can be formed using available vertices in $\mathcal{V}$. The blue lines indicate a valid assignment of resources in $\mathcal{U}$ to groups in $\mathcal{V}^g$. Red lines indicate an invalid assignment as the vertex of type v_1 is used 3 times in this assignment but there are only 2 vertices of type v_1 available.

Unlike in OM-RR-KAD, the underlying problem in OPERA is no longer a bipartite matching problem but a matching in a tripartite graph containing offline resources, $\mathcal{U}$ groups of online vertices, $\mathcal{V}^g$ and online vertices, $\mathcal{V}$.Figure 2 shows the tripartite graph formed in the case of OPERA.

Here are other key differences between OPERA and OM-RR-KAD:

$\mathcal{U}$: Each offline resource, $u \in \mathcal{U}$ in OPERA has a fixed capacity κ.

$\mathcal{V}^g$: As $\kappa > 1$, unlike in OM-RR-KAD model, resources can be assigned to more than one vertex at a round, i.e., resources can be assigned to groups of vertices where group sizes vary from 1 to κ. For ease of analysis, we consider that all the vertices can be paired together, and the constraints on the feasibility of pairing of vertices are handled through the weights received. Types of groups of vertices are obtained by generating all possible combinations (with repetitions) of size 1 to κ of the set $\mathcal{V}$[8]. The resulting set is denoted by $\mathcal{V}^g$. Therefore,

$$|\mathcal{V}^g| = \sum_{k=1}^{\kappa} \left(\!\!\binom{|\mathcal{V}|}{k}\!\!\right) = \sum_{k=1}^{\kappa} \binom{|\mathcal{V}| + k - 1}{k}$$

For each group of type v^g, n_{v,v^g} denotes the number of times vertex of type $v \in \mathcal{V}$ is present in group of type v^g (From the example Figure 2, for $v^g = (v_1, v_1)$, n_{v_1,v^g} will be 2 and for $v^g = (v_1, v_2)$, n_{v_1,v^g} will be 1.)

q_v^t: We consider batch arrival of vertices. Therefore, similar to the extension in Section 3, b^t vertices arrive at each round and each of these b^t vertices is sampled using the same probabilities $\{p_v^t\}$. The expected number of vertices of type v arriving in round t is q_v^t and is given by:

$$q_v^t = b^t \cdot p_v^t$$

[8] $\left(\!\!\binom{n}{k}\!\!\right)$ denotes the number of multisets of cardinality k, with elements taken from a finite set of cardinality n.

w_{u,v^g}^t: Weight received is now based on the type of group assigned to the resource.

c_{u,v^g}^t: Rounds of unavailability after an assignment is now based on the type of the group assigned to the resource.

Apart from the model differences, there are also differences with respect to the online assignments that can be made. The irrevocable assignment of resources in $\mathcal{U}$ to $\mathcal{V}^g$ should satisfy the following constraints:

C1: Each resource $u \in \mathcal{U}$ is assigned at most once in each round.

C2: The total number of vertices of each type $v \in \mathcal{V}$ used in the assigned groups is less than or equal to the number of vertices available.

C3: The number of groups of type $v^g \in \mathcal{V}^g$ assigned in round t is less than or equal to the number of available groups of type v^g.

In order to enforce constraint [C3] above in expectation (i.e., over all possible instantiations of arrivals), we need to compute $q_{v^g}^t$ — the expected number of times group of type v^g can be formed in round t. It is given by [9]:

$$q_{v^g}^t = h_{v^g}^t \prod_{v \in v^g} (p_v^t)^{n_{v,v^g}} \text{ where } h_{v^g}^t = \frac{\prod_{i=0}^{i=|v^g|} (b^t - i)}{\prod_{v \in \mathcal{V}} (n_{v,v^g})!} \quad (6)$$

We make the following assumptions in the model: (1) Once a resource u is assigned to a group of type v^g at t it becomes unavailable for further matches for c_{u,v^g}^t rounds irrespective of the size of v^g, i.e., insertion is not allowed. (2) The vertices can be grouped together iff they are arriving in same round. (3) For ease of explanation, we assume that $b^t > \kappa, \forall t$. However, this can be relaxed easily.

4.2 Online Algorithm

We first provide an LP for computing the upper bound on the offline optimal and then provide an adaptive assignment method based on the offline optimal solution.

LP for Upper Bound on Offline Batch Optimal with Multi-Capacity Resources: The optimization formulation [10] is provided in Table 2. We refer to this LP as LPShare. Since LP is for the offline case over all possible instantiations on arrival vertices, the constraints hold in expectation. Constraints (8), (9) and (10) refer respectively to **C1**, **C2** and **C3** constraints (described in Section 4.1) in expectation (i.e., over all possible instantiations of arrivals). Constraint (8) ensures that the resource u is assigned in round t iff u is available in round t.

Proposition 3. *The optimal value of LPShare provides a valid upper bound on the offline optimal value* [11].

ADAPShare-κ: For ease of explanation, we first present the online algorithm and competitive analysis for $\kappa = 2$.

[9] It corresponds to drawing n_{v,v^g} vertices of each type $v \in v^g$ out of total b^t trials for a multinomial distribution. Please refer to https://tinyurl.com/rjs524p for details on deriving the expression.

[10] LP is based on satisfying the flow constraints in the graph shown in Figure 2.

[11] Proof is omitted due to space constraints.

LPShare:

$$\max \sum_{t=0}^{T} \sum_{u \in \mathcal{U}} \sum_{v^g \in \mathcal{V}^g} w^t_{u,v^g} \cdot x^t_{u,v^g} \quad (7)$$

$$s.t. \quad \sum_{t'<t} \sum_{v^{g'} \in \mathcal{V}^g} x^{t'}_{u,v^{g'}} \cdot Pr[c^{t'}_{u,v^{g'}} > t - t'] + \sum_{v^g \in \mathcal{V}^g} x^t_{u,v^g} \leq 1 ::: \forall u \in \mathcal{U}, 0 \leq t < T \quad (8)$$

$$\sum_{v^g; v \in v^g} \sum_{u \in \mathcal{U}} n_{v,v^g} \cdot x^t_{u,v^g} \leq q^t_v ::: \forall v \in \mathcal{V}, 0 \leq t < T \quad (9)$$

$$\sum_{u \in \mathcal{U}} x^t_{u,v^g} \leq q^t_{v^g} ::: \forall v^g \in \mathcal{V}^g, 0 \leq t < T \quad (10)$$

$$0 \leq x^t_{u,v^g} \leq 1 ::: \forall u \in \mathcal{U}, v^g \in \mathcal{V}^g, 0 \leq t < T \quad (11)$$

Table 2: Optimization Formulation - Multi-capacity

Algorithm 2: ADAPShare-2(γ)

1: **for** $t < T$ **do**
2: Generate a random shuffling of the incoming b^t vertices. Label the vertices from 1 to b^t.
3: **for** $i = 1$ to b^t **do**
4: **for** $j = 1$ to b^t **do**
5: v^g = type of group formed at step (i, j) based on the labels assigned to the vertices.
6: **if** v^g is available for assignment at step (i, j) **then**
7: **If** $E^t_{*,v^g,(i,j)} == \phi$, reject v^g
8: **Else** choose $(u, v^g) \in E^t_{*,v^g,(i,j)}$ with probability p where $p = \frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g} \cdot P^t_{v^g,(i,j)} \cdot \beta^t_{u,(i,j)}}$
9: Update $E^t_{*,*,(i,j)}$, available groups based on the assignment.

Let $x^{*,t}_{u,v^g}$ denotes the optimal probability of assigning a resource u to a group of type v^g in round t (computed from offline optimal LP). We use Algorithm 2 to make online assignment of resources to the groups of vertices based on $\{x^{*,t}_{u,v^g}\}$ values from the offline optimal LP. As shown in the algorithm, we perform a random shuffling of the b^t vertices (that arrive in a batch in round t) and label the vertices from 1 to b^t. The assignment of resources to groups is performed across $b^t \cdot b^t$ steps (as we consider groups of size 2). Step (i, j) corresponds to a step where we compute the probability for assignment of a group formed by vertices with labels i and j. It should be noted that when $i = j$, (i, j) corresponds to a group of size 1 with only vertex with label i.

The assignment rule to compute the online assignment probability of assigning resource u to a group of type v^g at step (i, j) of the algorithm is defined by

$$\frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g} \cdot P^t_{v^g,(i,j)} \cdot \beta^t_{u,(i,j)}} \quad (12)$$

where $\beta^t_{u,(i,j)}$ denotes the probability that resource u is available for assignment in round t at step (i, j) over all arrival sequences. Similarly $P^t_{v^g,(i,j)}$ denotes the probability that group of type v^g can be considered for assignment in round t at step (i, j) over all arrival sequences. $h^t_{v^g}$ was defined in Equation (6). We use $E^t_{*,v^g,(i,j)} \subset \mathcal{U}$ to denote the set of safe resources for group of type v^g at step (i, j).

Similarities and Differences to ADAPBatch: The intuition behind the assignment rule for a step is similar to the one in ADAPBatch. Assignment for a group of type v^g in a step is obtained by dividing the optimal assignment of round t for group of type v^g by the total number of steps where group of type v^g can be considered [12].

The key differences in assignment rule of ADAPShare-κ and ADAPBatch:

- For $\kappa = 2$, since we can consider 2 vertices together (in a group) for assignment, we process the groups in $b^t \cdot b^t$ steps for ADAPShare-2. This is in comparison to b^t steps in ADAPBatch.
- In ADAPBatch, during online processing, vertex with label i in the batch will be considered for assignment only at one of b^t steps. In ADAPShare-κ, a vertex is part of multiple groups, so it will be considered at multiple steps. Therefore, at each step, the probability of vertex being available (and as a result a group being available) needs to be recomputed based on the groups assigned at previous steps in the same round.

Figure 1 highlights the difference in the way the algorithms ADAPBatch and ADAPShare-κ process the online information.

Competitive Ratio for ADAPShare-2

In this section, we provide the analysis to compute the competitive ratio for ADAPShare-2. We first find the value of γ for which the assignment rule in Equation (12) is valid, i.e., it corresponds to a valid probability value between 0 and 1.

Proposition 4. *The maximum value of γ for which assignment rule in Equation* (12) *is valid is 0.31767.*

Proof: Since the assignment rule always generates a positive value, the condition to be satisfied for the assignment rule to be valid is

$$\frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g} \cdot P^t_{v^g,(i,j)} \cdot \beta^t_{u,(i,j)}} \leq 1 \quad (13)$$

Using Equation (6) in Constraint (10) of optimization formulation in Table 2, we have

$$\sum_u x^{*,t}_{u,v^g} \leq h^t_{v^g} \cdot \prod_{v \in v^g} (p^t_v)^{n_{v,v^g}} \implies x^{*,t}_{u,v^g} \leq h^t_{v^g} \cdot \prod_{v \in v^g} (p^t_v)^{n_{v,v^g}}$$

[12] Each group of type v^g will be considered at $h^t_{v^g}$ steps out of the total $b^t \cdot b^t$ steps. For $\kappa = 2$, from Equation (6)

$$h^t_{v^g} = \begin{cases} b^t, \ if \ |v^g| = 1, \\ b^t \cdot (b^t - 1) \ if \ |v^g| = 2 \ and \ v^g = (v, v'), \\ \frac{b^t \cdot (b^t - 1)}{2} \ if \ |v^g| = 2 \ and \ v^g = (v, v). \end{cases}$$

This is because when both vertices are of same type in the group, for example if $v^g = (v, v)$, then v^g considered at step (i, j) means that the vertex with label i and the vertex with label j both are v and therefore steps (i, j) and (j, i) would be identical. On the other hand when both vertices are of different type, for example if $v^g = (v, v')$, then v^g considered at step (i, j) means that the vertex with label i is v and the vertex with label j is v' but v^g considered at step (j, i) means the opposite. Hence in this case the group of type v^g will be considered at $b^t \cdot (b^t - 1)$ steps across different online arrivals. Please refer to the example in the document at https://tinyurl.com/rjs524p for more clarity.

Substituting this in Equation (13) and rearranging terms, we get

$$\beta^t_{u,(i,j)} \geq \frac{\gamma \cdot \prod_{v \in v^g} (p^t_v)^{n_{v,v^g}}}{P^t_{v^g,(i,j)}} \ \forall t, i, j, v^g \quad (14)$$

By considering the probabilities with which each of the vertex of type $v \in v^g$ is available at step (i, j), we can show that [13],

$$\frac{\prod_{v \in v^g} (p^t_v)^{n_{v,v^g}}}{P^t_{v^g,(i,j)}} \leq \frac{1}{(1-\gamma)^2}, \forall t, i, j, v^g \quad (15)$$

Using Equations (14) and (15), for the assignment rule to be valid it is sufficient to show that $\beta^t_{u,(i,j)} \geq \frac{\gamma}{(1-\gamma)^2}$.
We can compute a lower bound on the value of $\beta^t_{u,(i,j)}$ based on assignments performed in previous steps and rounds. Specifically, using mathematical induction, we can show that $\beta^t_{u,(i,j)} \geq 1 - \gamma$.

So, to find the maximum value of γ for which the assignment rule is valid, we take γ such that $1 - \gamma = \frac{\gamma}{(1-\gamma)^2}$ Therefore, the possible value of γ is the solution to the equation $\gamma = (1 - \gamma)^3$, which is $\gamma = 0.31767$.

Proposition 5. *The online algorithm ADAPShare-2 is 0.31767 competitive.*

Proof: The proof involves first showing that the ADAPShare-2 is γ competitive. Now, as from Proposition 4, the maximum value of γ for which assignment rule is valid is 0.31767, therefore the algorithm is 0.31767 competitive.
To show that the ADAPShare-2 is γ competitive, we compute with respect to the optimal, the fraction of times any resource u is assigned to any group of type v^g. The probability that the resource u is assigned to a group of type v^g in round t in step (i, j) is given by

$$\frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g} \cdot P^t_{v^g,(i,j)} \cdot \beta^t_{u,(i,j)}} \cdot \beta^t_{u,(i,j)} \cdot P^t_{v^g,(i,j)} = \frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g}}$$

where first term in the product is the assignment rule, second term is the probability that u is available and the last term is the probability that v^g is available in round t at step (i, j).
As mentioned before, each group of type v^g will be considered for assignment at a total of $h^t_{v^g}$ steps. Therefore, the expected number of times a resource u is assigned to a group of type v^g in round t is given by $h^t_{v^g} \cdot \frac{x^{*,t}_{u,v^g} \cdot \gamma}{h^t_{v^g}} = x^{*,t}_{u,v^g} \cdot \gamma$, i.e., in online case each resource u is matched to group of type v^g with probability equal to $x^{*,t}_{u,v^g} \cdot \gamma$. Therefore, ADAPShare-2 is γ competitive. ■

Corollary 1. *The online algorithm ADAPShare-κ (generalization of ADAPShare-2 for any value of κ) is γ competitive where the value of γ is the solution to the equation $\gamma = (1 - \gamma)^{\kappa+1}$.*

Proof Sketch: The proof is along the same lines as the proof for Proposition 5. In the Equation (15), instead of $(1 - \gamma)^2$, we will have $(1 - \gamma)^\kappa$. Therefore, the value of γ for which assignment rule is valid is the solution to the Equation $\gamma = (1 - \gamma)^{\kappa+1}$. ■

Hardness Result for Non-Adaptive Algorithms:

[13]Please refer to https://tinyurl.com/rjs524p for the detailed proof.

Dickerson *et.al.* [8] prove that no non-adaptive algorithm based on LPSequential can achieve a competitive ratio of more than $\frac{1}{2} + o(1)$ in OM-RR-KAD model. The analysis can be easily extended for the batch arrival case when $\kappa = 1$. As unit-capacity batch arrival is a special case of multi-capacity OPERA model with all $w^t_{u,v^g} = 0$, $if\ |v^g| \geq 2$, therefore, no non-adaptive algorithm based on LPShare can achieve a competitive ratio of more than $\frac{1}{2} + o(1)$ for OPERA model.

Discussion: We now provide the justifications for the choices made in the modelling and analysis in section 3 and 4.1. (1) We assume that there are b^t arrivals in round t and b^t is known in advance. However, this is not at all a strong assumption because by considering a null type vertex in $\mathcal{V}$ and p^t_ϕ as the probability of null vertex, b^t can be used to denote the maximum number of arrivals in round t. (2) For theoretical analysis of the solution quality, we ignore the computational complexity of generating exponential number of groups in OPERA model. For practical purposes, the algorithms provided in [2] can be used to heuristically prune the exponential set and generate the feasible groups efficiently. The pruned set of groups is used by both offline and online algorithms. This is because, if the offline optimal algorithm can generate the groups, as the type of vertices are known in advance (through the known distribution), the online algorithm can also use those groups.

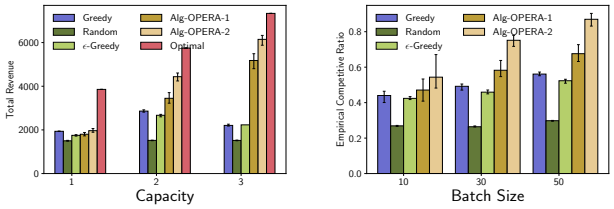
5 EXPERIMENTS

In this section, we compare the following five approaches on the empirical competitive ratio metric:

- **Greedy** - Runs an integer optimization at each timestep (based on the current information) to assign the requests/groups to the available offline resources [14].
- **Random** - Shuffles available requests/groups randomly and then assigns each request/group randomly to an available offline resource.
- **Alg-OPERA-1** - Algorithm based on the offline optimal LP where match for any available resource u to a vertex or group is performed by looking at the value of $\frac{x^{*,t}_{u,v^g}}{q^t_{v^g}}$ [15].
- **Alg-OPERA-2** - Another algorithm based on the offline optimal LP where match for any available resource u to a vertex or a group is performed by looking at the value of $\frac{x^{*,t}_{u,v^g}}{\sum_u x^{*,t}_{u,v^g}}$.
- **ϵ-Greedy** - With probability ϵ, greedy algorithm is executed and with probability $1 - \epsilon$, Alg-OPERA-1 algorithm is executed.

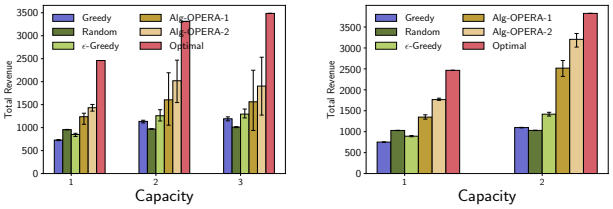
The goal of the experiments is to show that the algorithms which use guidance from the offline optimal LP, outperform the myopic approaches [16], which do not consider future information. All the values in the results are computed by taking an average over 10 instances and each instance is run 100 times.

[14]Equivalent to the myopic approaches used in practice [2, 14]
[15]We provide heuristics, which are close to ADAPShare-κ, as computing β exactly is not always simple and may require large number of simulations. We observed that even though these heuristics are non-adaptive, they can achieve empirical competitive ratio higher than the theoretical competitive ratio of ADAPShare-κ.
[16]Currently used in practice for multi-capacity resources [2, 14]



(a) (b)

Figure 3: $|\mathcal{U}|$ = **10** , $|\mathcal{V}|$ =**10,** $T = 200$ **(a) Varying** κ **(b)** $\kappa = 2$



(a) (b)

Figure 4: $|\mathcal{U}|$ = **30** , $|\mathcal{V}|$ =**121,** $T = 240$**, Real Dataset (a) 12am (b) 8am**

Synthetic Dataset: We first present the results on a synthetic dataset. We use 200 timesteps/rounds and generate the unavailability (or time occupied serving requests) time ($c^t_{u,v}$ or c^t_{u,v^g}) for each resource and vertex/group pair randomly between 1 and 60. Weights received (revenue) are generated based on revenue model used by taxi companies – base revenue + $0.5 \cdot c^t_{u,v}$ *or* c^t_{u,v^g}. The probability of arrival of each vertex type at each round (p^t_v) is also generated randomly. The test instances are generated by sampling the online vertices from the generated p^t_v values. We vary the batch size and capacity and present the representative results.

Figure 3a shows the total revenue obtained by different algorithms for different values of capacity κ. The key observations are:

(1) Our online approaches (Alg-OPERA-1 and Alg-OPERA-2) outperform other algorithms, with Alg-OPERA-2 performing better than Alg-OPERA-1 on all the instances.

(2) The performance of greedy algorithm decreases with the increase in capacity. Higher capacity provides more opportunity to serve requests at each timestep. Due to its myopic nature, greedy algorithm serves more requests initially, keeping the resources occupied for a longer time. On the other hand Alg-OPERA-1 and Alg-OPERA-2, based on the guidance provided by the offline optimal LP, ignore some requests/groups which have higher c^t_{u,v^g} value, to serve more requests at future timesteps.

(3) Figure 3b shows the empirical value of competitive ratio for different batch sizes. For these experiments, we take the identical value of batch size for all the timesteps. Higher batch size for multi-capacity resources provides an opportunity to group more requests. Therefore, as the batch size increases Alg-OPERA-1 and Alg-OPERA-2 show an improvement in performance.

Real World Dataset: We used the New York Yellow Taxi dataset which contains the records of trips in Manhattan city. We divided the map of the city into a grid of squares, each 4 by 4 km, which resulted in a total of 11 squares. Therefore, there can be 121 different types of requests, i.e., $|\mathcal{V}| = 121$ (origin-destination pairs). We experimented by taking real trips from the taxi dataset. We take the data across 10 days to compute the p^t_v values and the average number of requests at each round/timestep, i.e., average value of b^t. We run the offline optimal LP with these values and get a solution. The online algorithms are tested on actual instances (10 days) which are different from the ones we used for computing the parameter values. Therefore, the actual batch size b^t can be different from the value used by an offline optimal solution. The taxis are initialized at random locations and since we are testing the last mile scenario after serving the trips, they come back to their starting location. We observe a high variance in the performance of our algorithms on this dataset during night time (Figure 4a). This is because the distribution of requests during night have high variance across days. During the day, the variance in distribution of requests is low, and as a result our algorithms also show low variance. On an average , Alg-OPERA-1 and Alg-OPERA-2 outperform other algorithms on this dataset as well. These results indicate that the algorithms which use the guidance from offline optimal solution can consider the future effects of current matches and as a result provide better performance.

We would like to highlight that, to ensure that the theoretical bound on the competitive ratio holds empirically, correct estimates of probability values (p^t_v, β) are required, which requires running multiple simulations. It is possible to create scenarios, where a high number of simulations are required to get the correct estimates (e.g., when all the p^t_v values are very small and $\mathcal{V}$ is large.). In such cases, empirical competitive ratio measured over low number of simulations, will be a wrong indicator. We would also like to mention that, it is possible to synthetically create unrealistic scenarios where Greedy algorithm can achieve close to optimal value (essentially having a revenue model such that the difference between one long trip and multiple short trips is almost negligible, so myopic decisions do not hurt) and can perform better than the LP based approaches.

6 CONCLUSION

In this paper, we make a fundamental contribution of providing competitive ratios for the challenging online multi-capacity ridesharing problems – where resources or vehicles retain their type after serving the requests and rejoining the system – with batch arrival of requests. We demonstrate empirically on real and synthetic datasets that our online heuristics based on offline optimal LP perform well in practice, as compared to the myopic approaches.

7 ACKNOWLEDGEMENTS

This work was partially supported by the Singapore National Research Foundation through the Singapore-MIT Alliance for Research and Technology (SMART) Centre for Future Urban Mobility (FM). We thank Sanket Shah, Susobhan Ghosh and Tanvi Verma for providing valuable comments which greatly improved the paper.

REFERENCES

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. 2011. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1253–1264.

[2] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* 114, 3 (2017), 462–467.

[3] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. 2018. Maximum Weight Online Matching with Deadlines. *arXiv preprint arXiv:1808.03526* (2018).

[4] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. 2019. Edge Weighted Online Windowed Matching. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. ACM, 729–742.

[5] Lowell W Beineke. 1980. The Four Color Problem: Assaults and Conquest (Thomas Saaty and Paul Kainen). *SIAM Rev.* 22, 2 (1980), 241–243.

[6] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. 2013. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 101–107.

[7] 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 *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1681–1689.

[8] John P Dickerson, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. 2017. Allocation Problems in Ride-Sharing Platforms: Online Matching with Offline Reusable Resources. *arXiv preprint arXiv:1711.08345* (2017).

[9] Jon Feldman, Nitish Korula, Vahab Mirrokni, S Muthukrishnan, and Martin Pál. 2009. Online ad assignment with free disposal. In *International workshop on internet and network economics*. Springer, 374–385.

[10] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. 2018. How to match when all vertices arrive online. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 17–29.

[11] Patrick Jaillet and Xin Lu. 2013. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* 39, 3 (2013), 624–646.

[12] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 352–358.

[13] Euiwoong Lee and Sahil Singla. 2017. Maximum matching in the online batch-arrival model. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 355–367.

[14] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2019. ZAC: A Zone Path Construction Approach for Effective Real-Time Ridesharing. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*. 528–538.

[15] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research* 37, 4 (2012), 559–573.

[16] Aranyak Mehta et al. 2013. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science* 8, 4 (2013), 265–368.