Matching Policy Design for Gig Platforms with "Priority" Features

Evan Yifan Xu[1, 2] **and Pan Xu**[3]

[1]School of Cyber Science and Engineering, Southeast University
[2]School of Computing and Information Technology, University of Wollongong
[3]Department of Computer Science, New Jersey Institute of Technology
xyf@seu.edu.cn, pxu@njit.edu

Abstract

In recent years, gig platforms like Uber and DoorDash have implemented strategies to boost gig drivers' earnings during peak hours. Uber's 'back-to-back' feature allows drivers to accept new trips while still on route, and Uber Eats' 'Batch Order Route' initiative allows drivers to pick up multiple deliveries from different locations, which may result in multiple tops before one order is delivered. Despite revenue gains, these features lead to user complaints about extended waiting times. In response, platforms introduce features like Uber Eats' 'Priority Delivery' and Uber's 'Priority', where customers pay an extra subscription fee for guaranteed reduced waiting times. This paper focuses on designing matching policies to enhance system revenue while limiting customer waiting times. We present a hybrid model combining online matching and queue theory for quantitative analysis of users' waiting times. Additionally, we introduce an LP-based sampling framework and a unified queue-theory-based method for evaluating online performance. Comprehensive experiments on real datasets validate our theoretical findings, highlighting the efficiency of our matching framework in promoting profit and meeting committed waiting times.

1 Introduction

We have seen a flourishing gig economy during the last few decades, from ride-hailing apps like Uber and Lyft to online food ordering and delivery platforms like DoorDash and Grubhub. Gig economy features connecting short-term freelancers to online orders through powerful digital platforms. To promote the revenue especially during peak hours, a common strategy used is to assign multiple online orders to one single gig drivers. For example, Uber Eats has launched a feature, called *Batch Order Route,* which allows an Uber driver to collect up to four different deliveries from different locations, which can lead up to eight different stops before the food gets delivered (HghSsociety 2020). In the ride-hailing business, Uber has tested a "Back-to-Back" feature that allows drivers to accept a new trip before dropping off their current passenger (Huet 2015). While these strategies create more revenue for gig platforms, they also bring about many complaints from users. Among all these complaints, *excessive waiting time* is perhaps the most prevalent (Jandiss 2021; Matarese 2021; Kiwidrew 2017). In response to this, gig companies have taken a few initiatives to combat the issue of prolonged waiting time. In June 2020, Uber Eats has introduced a feature, called *Priority Delivery*, which requires customers to pay an extra fee in exchange for a guarantee that they receive orders within the estimated delivery time or even less (Uber 2020). Meanwhile, Uber has been testing a new service option, called *Priority*, that requires riders to pay more for a guaranteed reduced waiting time for cabs (Sarwar 2021). These examples underscore the need to minimize users' waiting time in online matching.

Online (Bipartite) Matching has been a popular framework widely used to study matching-policy design for gig platforms. The overall picture is to use a bipartite graph to model the network between gig drivers (called offline agents) and online orders (called online agents) (Manshadi and Rodilitz 2022; Ho and Vaughan 2012; Dickerson et al. 2024, 2021). A key feature there, called *real-time decision-making requirement*, states that upon the arrival of an online order, the decision-maker has to assign it to a then *free* driver; otherwise the order is gone. This assumption, inherited directly from the original model introduced by Karp, Vazirani, and Vazirani (1990), ignores the practices recently implemented by gig companies of assigning multiple orders to one single driver for revenue-promotion purposes. As a result, it is nearly impossible to quantitatively analyze request waiting times caused by preceding service delays in the current online-matching framework.

In this paper, we propose a hybrid model combining features from online matching and queue theory. We aim to design matching policies to promote *profit* in gig platforms, while curbing users' waiting time to guaranteed levels. We formally state our model as follows.

Online Matching with Guaranteed Waiting Time (OM-GWT). We are given an edge-weighted bipartite graph $G = (I, J, E)$, called *compatible graph*, where I and J denote sets of types of offline servers and online requests. There are specific spatial attributes associated with each server type (*e.g.,* preferred working areas) and each request type (*e.g.,* starting and ending locations). Each online request (of type) j arrives following an independent Poisson process of rate λ_j over a time horizon $[0, T]$. Each server i is designated with a warranty of an expected waiting time ω_i, which means every request selecting service from i enjoys a guarantee of an expected waiting time no more than ω_i be-

fore served by i. An edge $e = (i, j) \in E$ suggests that server i meets the waiting-time expectation from request j, in addition to spatial constraints such as j's starting and ending locations fall in i's working areas. Each edge $e = (i, j)$ is associated with a weight $v_e > 0$ representing the profit flowing to the system from the match of e, which incurs a random service time C_e denoting the time that i needs to complete request j. [1] Note that all the information above is known as part of the input, including the graph $G = (I, J, E)$, $\{\lambda_j, \omega_i\}$, $\{v_e\}$, and distributions of $\{C_e\}$. By default, we assume $\{C_e | e \in E\}$ are all independent from each other.

The online process goes as follows. Upon the arrival of an online request j, we should make an instant and irrevocable decision: either reject it or assign it to a feasible server i with $(i, j) \in E$. In the latter case, j will join a "virtual" queue $\mathcal{Q}_i$ maintained by server i. Note that (1) once a request j is assigned to i and joins the queue $\mathcal{Q}_i$, it cannot be kicked out; and (2) each server i will process all requests on queue $\mathcal{Q}_i$ following the first-come-first-serve (FCFS) policy to maintain the fairness.[2] By default, we assume every server has a unit serving capacity, *i.e.,* it can serve only one request at any time. A policy ALG is called *feasible* if the queue maintained by each server i meets the designated waiting-time warranty ω_i. In other words, any request j that is assigned to i is guaranteed an expected waiting time no more than ω_i on queue $\mathcal{Q}_i$ before getting the service. Note that we consider a *robust* version of expected-waiting-time guarantee: The expectation of waiting time is taken only on the randomness in the service time of requests before j on $\mathcal{Q}_i$, and it is irrespective of randomness in the policy or the arriving time of j. We aim to design an online policy, denoted by ALG, such that it is feasible and achieves a total expected profit as large as possible. The total profit is captured by $\max \sum_{e \in E} v_e \cdot \mathsf{E}[X_e]$, where X_e is the (random) number of matches of e made by ALG over the time horizon $[0, T]$.

Connections to Existing Models. A model closely relevant to OM-GWT is called *Online Matching with (offline) Reusable Resources* (OM-RR) (Dickerson et al. 2021). It shares a setting similar to OM-GWT except that every request accepts no waiting time, *i.e.,* requests will leave system if not served immediately. Thus, OM-RR can be viewed as a special case of OM-GWT when there is one single priority level with the waiting-time warranty being zero. Note that our model assumes an independent Poisson arrival process of homogeneous rate for each request type, a common arrival setting in online-matching models (Huang, Shu, and Yan 2022; Huang and Shu 2021; Ma, Xu, and Xu 2022). Another discrete counterpart, called *Known Independent Identical Distributions* (KIID), is widely studied as well (Manshadi, Gharan, and Saberi 2012; Feldman et al. 2009). Note that these two arrival settings are asymptotic equivalent (Huang and Shu 2021).

Other Related Work. There are quite a few follow-up studies on the model of Online Matching with (offline) Reusable Resources (OM-RR) especially in the Operations Research community (Feng, Niazadeh, and Saberi 2020, 2019; Delong et al. 2022; Gong et al. 2021; Goyal, Iyengar, and Udwani 2020, 2021; Liu and Hajiesmaili 2025), among which many have considered online assortment optimization under the large-capacity assumption. In that context, each server is an offline resource type with a large number of copies (called inventory), and each request can occupy multiple resource types simultaneously for a random time before those resources become available again. However, all of them assumes no patience from requests, *i.e.,* requester's demand should be accommodated upon her arrival.

We list a few recent studies on matching policy design in gig platforms as follows. Better matching on gig platforms can reduce revenue by revealing demand to workers, affecting their participation (Liu et al. 2024). Nair et al. (2022) aim to achieve that income guarantees while maintain a low system cost. Ulmer et al. (2021) focus on minimizing the order delivery time, while (Joshi et al. 2022) manage to map the vehicle assignment problem to that of minimum weight perfect matching on a bipartite graph. Most of them adopt an offline setting where all agents are static. In contrast, our model considers an online setting with requests join the system dynamically. Additionally, there are many research works applying queuing related models to study mechanism design via pricing and/or manipulation of the waiting-list in scenarios when non-linear utility function (*e.g.,* submodular) exists between offline and online agents and when waiting time of online agent is associated with some cost (Bloch and Cantala 2017; Thakral 2019; Ashlagi et al. 2020; Baccara, Lee, and Yariv 2020; Ashlagi, Monachou, and Nikzad 2021; Leshno 2022). In our paper, we consider linear utilities and impose a waiting-time constraint on each server.

G	Input network graph $G = (I, J, E)$
$I(J)$	Set of server(request) types
$E_i(E_j)$	Set of edges incident to i (j)
λ_j	Arrival rate of request type $j \in J$
T	Length of the time horizon
ω_i	Expected-waiting-time warranty on server i
v_e	Profit flowing to the system by the match $e \in E$
C_e	(Random) service time taken by $e = (i, j)$
π^*_{off}	An *offline* optimal policy for OM-GWT
$\mathbf{x}^*$	An optimal solution to **LP-P** with Objective (1)

Table 1: A glossary of notations used throughout this paper.

Preliminaries and Main Contributions

There is a large body of existing results related to queue theory and related topics. Here we list only a few relevant to this paper; see the book for more details (Gallager 2011).

Properties Relevant to Poisson Process. Let $\{N_j(t) : t \in [0, T] | j \in J\}$ be $|J|$ independent Poisson processes such that each $N_j(t)$ has a homogeneous rate of $\lambda_j > 0$. Consider a given subset $S \subseteq J$, and define $N_S(t) = \sum_{j \in S} N_j(t)$. **(P1)** $N_S(t)$ is a Poisson process of rate $\lambda_S := \sum_{j \in S} \lambda_j$. **(P2)** For each arrival in $N_S(t)$, it belongs to $N_j(t)$ with prob-

[1] Our setting of random service time is inspired by the work of (Dickerson et al. 2021).

[2] These two observations are inspired and consistent with most real-world scenarios in gig economy.

ability λ_j/λ_S for each $j \in S$.

Properties Relevant to an M/G/1/1 queue (Bose 2013). Consider an M/G/1/1 queue $\mathcal{Q}$, where it admits a Poisson arrival process of rate λ, a random service time C following a general distribution, one single server, and a unit queue capacity. Note that in this case, any arriving request has to leave the system if the server is busy. **(P3)** $\mathcal{Q}$ has a mean load $\rho := \lambda \cdot \mathsf{E}[C]$, and it will be busy and free in the steady state with probability $\rho/(1+\rho)$ and $1/(1+\rho)$, respectively.

Properties Relevant to an M/M/1/L queue (Smith 2003). Consider an M/M/1/L queue $\mathcal{Q}$, where it admits a Poisson arrival process of rate λ, an exponentially distributed service time C, one single server, and a queue capacity of L. **(P4)** $\mathcal{Q}$ has a mean load $\rho := \lambda \cdot \mathsf{E}[C]$, and it has a equilibrium non-blocking probability equal to $\gamma := \frac{1}{1+1/(\rho^{-L}+\rho^{-(L-1)}+\cdots+\rho^{-1})}$, where γ can be interpreted as the expected ratio of time when $\mathcal{Q}$ has a capacity less than L (*i.e.,* it can accept new arrivals) to the whole horizon.

Competitive Ratio (CR). Competitive ratio is a common metric to evaluate the performance of online algorithms. Consider an instance $G = (I, J, E)$ of OM-GWT and a given feasible policy ALG. Let π^*_{off} be an offline optimal policy, which is also known as *a clairvoyant optimal or an optimal with hindsight*. Here are a few similarities and differences between ALG and π^*_{off}. First, π^*_{off} has full access to the random number of arrivals of online requests for every type in advance, while ALG does not. Second, neither ALG nor π^*_{off} has access to the random realization of service time C_e before matching e for every $e \in E$. Third, π^*_{off} assumes all online requests arrive at time $t = 0$, and each of them has an unbounded patience such that it will never leave the system until matched. In contrast, ALG has to guarantee an expected waiting time capped by ω_i for every request assigned to server i. Let $\mathsf{E}[\mathsf{ALG}]$ and $\mathsf{E}[\pi^*_{\mathsf{off}}]$ denote the total expected profit achieved by ALG and π^*_{off}, respectively. We say ALG achieves a competitive ratio of $\alpha \in [0,1]$ if $\mathsf{E}[\mathsf{ALG}] \geq \alpha \cdot \mathsf{E}[\pi^*_{\mathsf{off}}]$ for any instance of OM-GWT.

Main Contributions. Throughout this paper, we assume by default $T \gg 1$, and part of our results are obtained when $T \to \infty$.[3] In this paper, we present a general LP-based sampling framework featuring two parameters: one is a feasible solution to a benchmark LP; the other is a vector specifying the queue capacity maintained by each server, which is jointly determined by the service time distribution and the waiting-time warranty claimed by the server. To analyze the performance of our framework under different settings, we propose a unified queue-theory-based approach for competitive analysis. Our main results are stated as follows.

Theorem 1. *[Section 3] There exists a $1/2$-competitive policy when $\omega_i = 0$ for all $i \in I$ (*i.e., *every request accepts no waiting time if assigned). Our competitive analysis is tight.*

Remarks on Theorem 1. (1) The assumption of $\omega_i = 0$ for all $i \in I$ implies that each server admits a new request only when it is free such that it can maintain the warranty of zero waiting time. Thus, for each arriving request, we can either assign it to a free neighboring server or reject it otherwise. Under this assumption, OM-GWT gets reduced to the model in (Dickerson et al. 2021). Compared with the $1/2$-competitive policy there, our policy features its simplicity: It is non-adaptive and remove the time-consuming procedure of simulation-based attenuations (Dickerson et al. 2021; Feng, Niazadeh, and Saberi 2019). (2) Dickerson et al. (2021) offered a hardness instance showing that no non-adaptive policy can achieve a CR better than $1/2$ when $T \to \infty$, showing our policy's optimality among non-adaptive ones and the tightness of the analysis.[4]

Theorem 2. *[Section 4] There exists a policy achieving a competitive ratio (CR) of at least $1 - 1/(1+\underline{L})$ when all service time is IID exponentially distributed, where $\underline{L} := 1 + \lfloor \min_{i \in I} \omega_i / \mathsf{E}[C_e] \rfloor$.*

Remarks on Theorem 2. (1) When $\omega_i = 0$ for all $i \in I$, the result of Theorem 2 matches that of Theorem 1. (2) The settings considered in Theorems 1 and 2 are incomparable: The former assumes general edge-dependent service-time distributions but a strict warranty of zero waiting time, while the latter considers IID exponential service-time distributions but general guarantees of expected waiting time. (3) One motivating example of IID service-time distributions can be seen in the work by Dickerson et al. (2021), where they analyzed real datasets from NYC Yellow Cabs and found the majority trips shared a similar pattern and were completed within 10 to 15 mins. (4) Exponential service-time distributions are widely used in practice; see, *e.g.,* (Krishnamoorthi and Wood 1966; Chakraborty, Muthuraman, and Lawley 2010; Matyushenko and Ermolayeva 2021).

For the general case, we focus on the asymptotical CR when the min waiting-time warranty goes to infinity. Intuitively, when $\underline{\omega} := \min_i \omega_i \to \infty$, the performance of an optimal *online* policy can match that of an offline optimal, and thus, the resulting CR should approach 1. In light of this, we investigate the asymptotical CR of our LP-based sampling framework when $\underline{\omega} \to \infty$, given all service time has a constant bounded memory as defined in (6). For each $e \in E$, let

$$\beta_e := \mathsf{E}[C_e],\ s_e^2 := \mathsf{Var}[C_e]/\mathsf{E}^2[C_e],$$

where β_e denotes the expected service time of the assignment e, and s_e^2 is called the squared coefficient of variation of C_e.

Theorem 3. *[Section 5] There exists a policy achieving a competitive ratio (CR) of at least $1 - (1/\underline{\eta})(1 + o(1))$, where $\underline{\eta} = \min_i \eta_i$ with $\eta_i = 2\omega_i / \big(\max_{e \in E_i} \beta_e \big(1 + \max(1, s_e^2)\big)\big)$, and $o(1)$ is vanishing when $\underline{\eta} \to \infty$.*

Remarks. Note that when all service time are IID exponentially distributed, we have $s_e^2 = 1$ for all $e \in E$.

[3] It is a common practice to assume the time horizon $T \gg 1$ in analyzing online algorithms under known distributions; see *e.g.,* (Brubach et al. 2020; Jaillet and Lu 2013; Manshadi, Gharan, and Saberi 2012; Haeupler, Mirrokni, and Zadimoghaddam 2011; Feldman et al. 2009).

[4] Note that Dickerson et al. (2021) also gave an upper bound of $1/2$ for adaptive policies but for the case when each online request is allowed to have heterogeneous arriving rates during the online phase.

In this case, the result of Theorem 3 can be simplified as $1-(1/\min_i \omega_i/\mathsf{E}[C_e])(1+o(1))$, where $o(1)$ vanishes when $\underline{\eta} = \min_i \omega_i/\mathsf{E}[C_e] \to \infty$. This is consistent with the result of Theorem 2. The parameter η serves a role analogous to L in the IID case, representing expected queue capacity.

We implement our algorithms on a real dataset against a few heuristics and some state-of-the-art algorithms such as reinforcement-learning-based approaches (Xu et al. 2018; Feng, Gluzman, and Dai 2021; Shi et al. 2021; Tong et al. 2021). Experimental results demonstrate the efficiency and effectiveness of our proposed policies in promoting revenue while curbing users' waiting time to pre-specified guaranteed levels. All details can be seen in Section 6.

2 An LP-Based Sampling Framework and a Meta-Competitive Analysis

Consider an offline optimal policy π^*_{off}. For each edge $e \in E$, let x_e be the expected number of assignments of e *averaged on a unit time* over $[0,T]$ in π^*_{off}. Consider the LP below.

$$\textbf{LP-P}: \quad \max \sum_{e\in E} v_e \cdot x_e \tag{1}$$

$$\sum_{e\in E_j} x_e \le \lambda_j, \qquad \forall j \in J; \tag{2}$$

$$\sum_{e\in E_i} x_e \cdot \mathsf{E}[C_e] \le 1, \qquad \forall i \in I; \tag{3}$$

$$0 \le x_e, \qquad \forall e \in E. \tag{4}$$

Throughout this paper, we use **LP-P** to denote the linear program above with Objective (1) and Constraints (2)-(4). Let OPT_P be the optimal values of **LP-P**.

Lemma 1. *The total expected profit achieved by an offline optimal policy over $[0,T]$ is upper bounded by $T \cdot \mathrm{OPT}_P$.*

Proof. We can verify that the total expected profit achieved by π^*_{off} over $[0,T]$ is equal to $T \cdot \sum_{e\in E} x_e \cdot v_e$. Thus, it suffices to show that $\{x_e | e \in E\}$ is feasible to Constraints (2)-(4) above. Constraints (2) are valid since the expected number of served requests (of type) j in a unit time should be no larger than that of arrivals, which is equal to λ_j. Constraints (3) follow from the fact that the expected accumulative service time on server i in a unit time should be no larger than one. □

Our main sampling framework is formally stated in Algorithm 1, which takes as input two parameters. The first is a feasible solution to Constraints (2), (3), and (4). Generally, we will feed SAMP with an optimal solution to the benchmark **LP-P**. The second is a capacity vector $\mathbf{L} = (L_i)_{i\in I}$, where L_i denotes the queue capacity imposed on server i, *i.e.,* a cap on the number of requests admissible to i (including the one receiving the service). Intuitively, each L_i will be determined based on the input waiting-time warranty ω_i designated on server i such that the resulting policy is feasible, *i.e.,* each request assigned to server i should enjoy a guarantee of an expected waiting time of no more than ω_i.

Algorithm 1 $\mathsf{SAMP}(\mathbf{x}, \mathbf{L})$

Two parameters: $(\mathbf{x}, \mathbf{L})$, where $\mathbf{x} = \{x_e\}$ is a feasible solution to Constraints (2), (3), and (4), and where $\mathbf{L} = \{L_i\}$ with L_i denoting the queue capacity imposed on server $i \in I$.

1: Let an online request of type j arrive at time $t \in [0,T]$.
2: Sample an edge $e = (i,j) \in E_j$ with probability x_e/λ_j and reject j otherwise. (This is a valid sampling distribution due to Constraints (2)).
3: Assign j to i if the queue on server i is not blocked, *i.e.,* there are no more than $L_i - 1$ requests assigned to i, including the one receiving the service then. Reject j otherwise.

*Throughout this paper, we use $\mathbf{x}^* = (x^*_e)$ to denote an optimal solution to the benchmark **LP-P**.* Suppose we feed SAMP with $\mathbf{x}^*$ and a capacity vector $\mathbf{L}$ such that $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ is feasible.[5] Let γ_i be the equilibrium non-blocking probability of queue on server $i \in I$, which can be viewed as the expected fractional of time when i accepts a new request over $[0,T]$ when $T \to \infty$. The lemma below suggests that we can reduce the job of lower bounding the competitive ratio achieved by SAMP to that of the equilibrium non-blocking probability over all servers.

Lemma 2. $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ *achieves a competitive ratio of at least $\min_{i\in I} \gamma_i$, where γ_i is the equilibrium non-blocking probability of queue on server $i \in I$.*

Proof. Let $\gamma := \min_{i\in I} \gamma_i$. Consider a given edge $e = (i,j) \in E_i$, and let M_e be the random number of assignments of e made in $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$. Let $\mathsf{SF}_i(t) = 1$ indicate that server i is not blocked at time $t \in [0,T]$, *i.e.,* it can accept new requests.

$$\begin{aligned}\mathsf{E}[M_e] &= \mathsf{E}\Big[\int_0^T \lambda_j \cdot (x^*_e/\lambda_j) \cdot \mathsf{SF}_i(t)\, dt\Big] \\ &= x^*_e \cdot T \cdot \frac{\mathsf{E}\big[\int_0^T \mathsf{SF}_i(t) dt\big]}{T} \\ &= x^*_e \cdot T \cdot \gamma_i \ge x^*_e \cdot T \cdot \gamma.\end{aligned}$$

Let $\mathsf{E}[\mathsf{SAMP}(\mathbf{x}, \mathbf{L})]$ and $\mathsf{E}[\pi^*_{\mathsf{off}}]$ denote the total expected profit achieved by $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ and π^*_{off} (an offline optimal), respectively.

$$\begin{aligned}\mathsf{E}[\mathsf{SAMP}(\mathbf{x}, \mathbf{L})] &= \sum_{e\in E} v_e \cdot \mathsf{E}[M_e] \ge \Big(\sum_{e\in E} v_e \cdot x^*_e\Big) \cdot T \cdot \gamma \\ &= \mathrm{OPT}_P \cdot T \cdot \gamma \ge \mathsf{E}[\pi^*_{\mathsf{off}}] \cdot \gamma,\end{aligned}$$

where the last inequality is due to Lemma 1. □

[5] Note that there always exists some choice of $\mathbf{L}$ to ensure the feasibility of $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ in terms of fulfilling the expected-waiting-guarantee of ω_i on each server $i \in I$. For example, when $\mathbf{L} = \mathbf{1}$ (a vector of all ones), $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ will reject many requests to guarantee that every assigned request has a zero waiting time.

3 The First Special Case: Serve-or-Go Requests

We consider a special case when every request will leave the system unless being served upon the arrival. In this case, we essentially have one single priority level with $\omega_i = 0$ for all $i \in I$. Thus, we need to assign every arriving request j to a then *free* server i with $(i, j) \in E$ upon her arrival; otherwise, j will leave the system. In light of this, we feed SAMP with $\mathbf{L} = (L_i) = \mathbf{1}$, a vector of length $|I|$ with all entries being one. This ensures that every request be assigned to a then *free* server if getting served. Recall that $\mathbf{x}^*$ is an optimal solution to the benchmark **LP-P**. Next, we prove the main theorem 1 by showing that $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L} = \mathbf{1})$ achieves a competitiveness of at least $1/2$.

Proof of Theorem 1. Consider a given server $i \in I$, and let $\mathcal{Q}$ be the virtual queue maintained by i in the policy $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L} = \mathbf{1})$. For the ease of notation, we drop subscriptions i when the context is clear. The queue maintained by i, denoted by $\mathcal{Q}$, is an M/G/1/1 queue: (1) it admits a Poisson arrival process of rate $\lambda := \sum_{e\in E_i}(x^*_e/\lambda_j)\cdot\lambda_j = \sum_{e\in E_i} x^*_e$, which is due to Property **P1** in Preliminaries; (2) it has a service time $C := \sum_{e\in E_i} C_e \cdot \mathbf{I}_e$, where $\mathbf{I}_e$ is a Bernoulli random variable of mean x^*_e/λ for each $e \in E_i$ (due to **P2**); (3) it has a unit system capacity and accepts a new request only when it is free. We can verify that the mean load on server i satisfies $\rho := \lambda\cdot\mathsf{E}[C] = \sum_{e\in E_i}\mathsf{E}[C_e]\cdot x^*_e \le 1$, which follows from Constraints (3). Thus, when $T \gg 1$, $\mathcal{Q}$ will enter a steady state, and it will be free with probability $\gamma := 1/(1+\rho)$ (due to **P3**). As $\rho \le 1$, $\gamma = 1/(1+\rho) \ge 1/2$, and the result follows by Lemma 2. □

4 The Second Special Case: IID Exponentially Distributed Service Time

Consider a special case when $\{C_e\}$ are all IID exponentially distributed. Set

$$\beta = \mathsf{E}[C_e],\ \ \mathbf{L} = \{L_i := \lfloor \omega_i/\beta \rfloor + 1 | i \in I\};\ \ \underline{L} := \min_{i\in I} L_i. \tag{5}$$

Recall that ω_i is the expected-waiting-time warranty designated on server i. Thus, $\underline{L} - 1 = \min_{i\in I}\lfloor\omega_i/\beta\rfloor \approx (\min_i \omega_i)/\beta$, which can be viewed as the ratio of the min expected-waiting-time warranty over all servers to the expected service time.

Lemma 3. *The policy $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ with $\mathbf{L}$ defined in (5) is feasible, ensuring that every request assigned to server $i \in I$ is guaranteed an expected waiting time no more than ω_i.*

Proof. Consider a given server $i \in I$, and $\mathcal{Q}$ be the virtual queue maintained by i in $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$. Observe that $\mathcal{Q}$ has an exponential service time $C := C_e$ and a finite system capacity L_i. Thus, for each request j assigned to i at any time, there will be no more than $L_i - 1$ requests before j (including the one served by i then) on $\mathcal{Q}$. As for the request, say j', served by i upon the assignment of j to i, we know the expected *remaining* service time should be β due to the memory-less property of exponential service time. Thus, the total expected waiting time of j before getting the service should be no more than $(L_i - 1)\cdot\beta = \lfloor\omega_i/\beta\rfloor\cdot\beta \le \omega_i$. □

The proof of Lemma 3 suggests that any policy is *feasible* if it maintains a queue capacity no more than L_i on each server $i \in I$, where $\mathbf{L} = (L_i)$ is defined in (5). In the following, we prove the policy $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L})$ with $\mathbf{L}$ defined in (5) achieves a performance as stated in Theorem 2.

Proof of Theorem 2. Consider a given server i, and let $\mathcal{Q}$ be the virtual queue maintained by i. For the ease of notation, we drop subscriptions i when the context is clear. Observe that $\mathcal{Q}$ is an M/M/1/L queue such that (1) it admits a Poisson arrival process of rate $\lambda := \sum_{e\in E_i}(x^*_e/\lambda_j)\cdot\lambda_j = \sum_{e\in E_i} x^*_e$; (2) it has an exponential service time $C := C_e$; and (3) it has a system capacity of L_i. We can verify that the mean load on server i satisfies $\rho := \lambda\cdot\mathsf{E}[C] = \sum_{e\in E_i}\mathsf{E}[C_e]\cdot x^*_e \le 1$, which is due to Constraints (3). Thus, when $T \gg 1$, $\mathcal{Q}$ will enter a steady state, and it will have a non-blocking equilibrium probability $\gamma_i := \frac{1}{1+1/(\rho^{-L_i}+\rho^{-(L_i-1)}+\cdots+\rho^{-1})}$. Since $\rho \le 1$, we have $\gamma_i \ge L_i/(L_i+1) = 1 - 1/(1+L_i)$. By the setting of L_i in Equation (5) and Lemma 2, we establish the result. □

5 The General Case

Consider the general case. To make our model tractable, we assume every service time has a constant bounded memory, which is defined as follows:

$$\max_{t\ge 0:\Pr[C_e\ge t]>0} \big(\mathsf{E}[C_e | C_e \ge t] - t\big) \le \bar{c}_i \cdot \mathsf{E}[C_e], \quad \forall e \in E_i, \text{ with } \bar{c}_i \le \bar{c}, \forall i \in I, \tag{6}$$

where $\bar{c}$ is a constant.[6] Recall that $\mathbf{x}^* = \{x^*_e | e \in E\}$ is an optimal solution to **LP-P** with $x^*_i := \sum_{e\in E_i} x^*_e$. Let $\bar{\beta}_i := \max_{e\in E_i}\mathsf{E}[C_e]$ for $i \in I$. Set $\mathbf{L}(\mathbf{x}^*) = \{L_i(\mathbf{x}^*) | i \in I\}$, where

$$L_i(\mathbf{x}^*) = \begin{cases} 1 & \text{If } \omega_i < \bar{c}_i\cdot\bar{\beta}_i; \\ 2 + \left\lfloor \dfrac{\omega_i - \bar{c}_i\cdot\bar{\beta}_i}{\sum_{e\in E_i}\mathsf{E}[C_e]\cdot x^*_e/x^*_i} \right\rfloor & \text{If } \omega_i \ge \bar{c}_i\cdot\bar{\beta}_i. \end{cases} \tag{7}$$

We split the proof of Theorem 3 into the following two lemmas. Recall that $\mathbf{x}^* = (x^*_e)$ is an optimal solution to the benchmark **LP-P**, and $x^*_i := \sum_{e\in E_i} x^*_e$ for each $i \in I$. Due to the space limit, we defer the proof of Lemma 4 and Lemma 5 to the full version.

Lemma 4. *$\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L}(\mathbf{x}^*))$ is a feasible policy such that every request assigned to any server $i \in I$ will have an expected waiting time no more than ω_i, where $\mathbf{L}(\mathbf{x}^*)$ is defined in Equation (7).*

[6] The constant-bounded-memory property is a natural generalization of the classical memoryless property. We can verify that most practical service distributions satisfy it. For example, Uniform and Exponential distributions both conform to it with $\bar{c} = 1$.

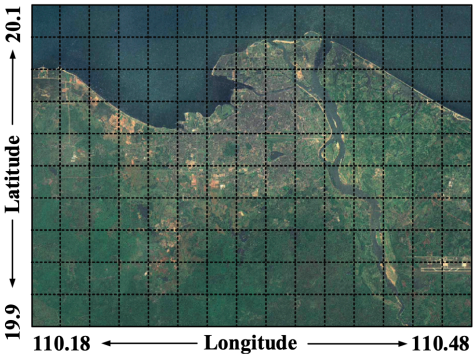


Figure 1: Testing Zone for Didi Dataset in Haikou, China.

Parameters	No. of Experiments							
	1	2	3	4	5	6	7	8
ω_1 (sec.)	600	700	900	1100	1200	1300	1400	1500
ω_2 (sec.)	900	1050	1350	1650	1800	1950	2100	2350
α	1	0.9	0.8	0.7	0.6	0.54	0.48	0.45
$\underline{L}$	3	4	5	7	9	10	12	14
$\underline{\eta}$	1.48	1.92	2.78	3.88	4.94	5.95	7.2	8.23

Table 2: Parameter settings. In the case of IID exponentially distributed service time: $\underline{L}$ denotes the minimum queue capacity over all servers; In the general case: $\underline{\eta}$ denotes the minimum η over all servers.

Lemma 5. *Each server $i \in I$ in $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L}(\mathbf{x}^*))$ will have a non-blocking equilibrium probability at least $\gamma_i := 1 - (1/\eta_i)(1 + o(1))$, where $\eta_i = 2\omega_i / \big(\max_{e \in E_i} \beta_e \big(1 + \max(1, s_e^2)\big)\big)$, and $o(1)$ is a vanishing term when $\eta_i \to \infty$.*

Proof of Theorem 3. Observe that Lemma 4 justifies the feasibility of the policy $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L}(\mathbf{x}^*))$. Results of Lemmas 5 and 2 establish the competitiveness of $\mathsf{SAMP}(\mathbf{x}^*, \mathbf{L}(\mathbf{x}^*))$ as stated in Theorem 3. □

6 Experiments

Dataset Preprocessing. Our experiments use a real-world Didi dataset[7] containing millions of taxi trips collected in Haikou, China, from May to September 2017. Each record has 23 fields (e.g., pick-up/drop-off coordinates, timestamps, estimated fare). Following (Dickerson et al. 2021), we focus on trips between 8:00–22:00, most of which finish within 20 minutes. We restrict the study area to longitude/latitude ranges $(110.18, 110.48)$ and $(19.90, 20.10)$, partitioned into a 15×10 grid (see Figure 1).

We construct a bipartite graph $G = (I, J, E)$ as follows. Each grid defines a driver type i whose working area includes its neighboring grids; each origin–destination grid pair defines a rider type j, with arrival rate λ_j set to the average number of daily requests in May 2017. Motivated by Uber's practice, we assign two priority levels—Priority (P) and Normal (N)—and designate each driver and rider as P or N with probabilities 0.2 and 0.8 (Koch 1998). For serve-or-go requests (single priority level), we add an edge $e = (i, j)$ when j's OD pair lies in i's working area. For the IID-exponential and general models, an edge is added only if the spatial condition holds and i's priority level is no higher than j's.

For each rider type j, let d_j and τ_j denote the average trip distance and duration. For each edge e, we set the profit $v_e = 5d_j$ and assume the service time $C_e \sim \mathrm{Exp}(\alpha\tau_j)$ with $\alpha = 1$. For serve-or-go, we test $T \in \{3000, 6000, \ldots, 21000\}$ with fixed warranties $\sigma_1 = 1200$ and $\sigma_2 = 1800$. For the

[7] https://www.didiglobal.com/

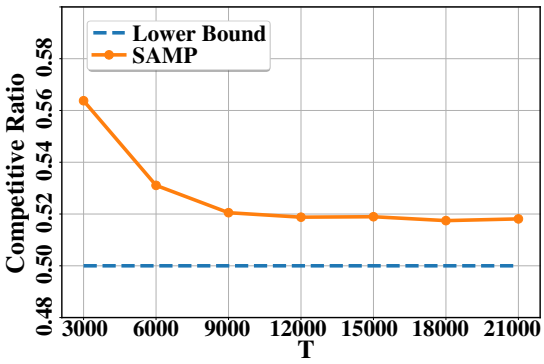
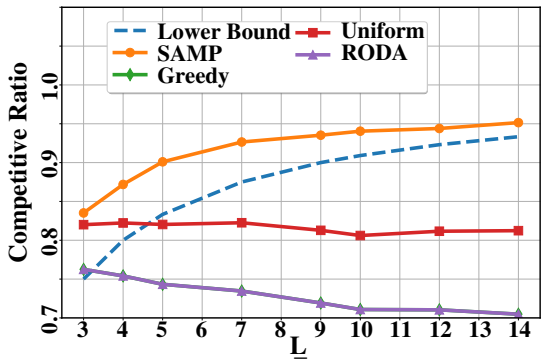
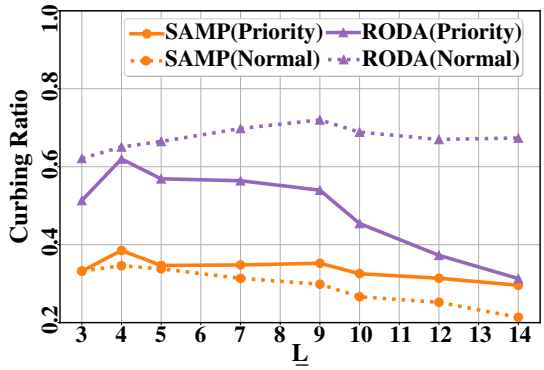


Figure 2: Results of the serve-or-go case.

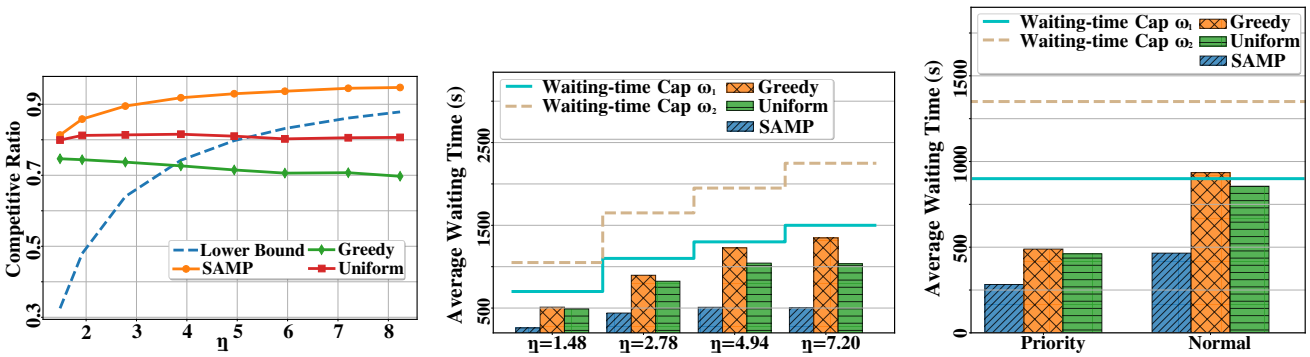


(a) Competitive ratios - varying $\underline{L}$.



(b) Curbing ratios - varying $\underline{L}$.

Figure 3: Special case of IID exponentially service time.



(a) Competitive ratios achieved when varying $\underline{\eta}$.

(b) Average waiting time achieved when varying $\underline{\eta}$.

(c) Average waiting time on two priority levels when $\underline{\eta} = 2.78$.

Figure 4: Experimental results on the general case.

other models, we fix $T = 18000$ seconds and vary λ_L and λ_η by adjusting σ_1, σ_2, and α (see Table 2).

Algorithms. We implement SAMP, which is short for SAMP($\mathbf{x}$, $\mathbf{L}$) and compare it against three baselines: a greedy-based algorithm Greedy, a uniform-based algorithm Uniform, and a reinforcement-learning-based algorithm RODA (Xu et al. 2018; Feng, Gluzman, and Dai 2021; Shi et al. 2021; Tong et al. 2021). Both Greedy and Uniform share the same queue capacity as SAMP in the sense that each driver i (or queue on i) is called *unblocked* if there are no more than $L_i - 1$ requests assigned to i with L_i as specified in SAMP. The difference lies in the sampling phase: Greedy always assigns an arriving rider to an *unblocked* driver with the maximum profit, while Uniform-P assigns an arriving rider to an *unblocked* driver uniformly at random. Since RODA is not designed to tackle the same problem as ours, we make the following adjustments: (1) For each driver's state, we add a parameter l to define the driver's real-time serving capacity. As long as the driver accepts an order, his capacity l gets decreased by one and the state will transfer. (2) For each edge $e = (i, j)$, we denote the reward as v_e (following our notation here), *i.e.,* the profit of assigning rider (type) j to driver (type) i. (3) We use a replay buffer to store the experiment data (s, a, r, s'), where s is the current state of driver; a denotes the action; r is the reward; s' denotes the state of driver after the execution of action a. We record the experiment data by running Greedy for 100 times, then put them into replay buffer, which is used to get the value function iteratively.

We run all algorithms 100 times for each group of parameters and take the average as the final performance. We use $T \cdot \mathrm{OPT}_P$ as the performance of an offline optimal with regard to profit. We compute the ratio of the performance of each algorithm to the optimal value as the final competitive ratio achieved. Additionally, we report another metric, called *curbing ratio*, which refers to the ratio of the average waiting time of riders of a given priority level to the designated waiting-time warranty. Thus, *a lower curbing ratio suggests a higher effectiveness in reducing riders' waiting time to the guaranteed levels*. Note that the curbing-ratio metric does not apply to serve-or-go requests because both the average riders' waiting time and the waiting-time warranty are equal to zero. All experiments are conducted on a PC with 3.2GHz 12th Gen Intel Core i9 processor and 128GB main memory.

Results and Discussions. Figure 2 shows that in the serve-or-go setting, SAMP achieves decreasing CRs as T increases, remaining consistently above the 1/2 lower bound. When $T > 9000$, SAMP nearly reaches this bound, confirming the tightness of our CR analysis. In the IID service-time case (Figure 3a), SAMP 's CRs always exceed the lower bound in Theorem 2, and its advantage over other baselines grows with larger $\underline{L}$, demonstrating its superiority in longer-queue scenarios common in *online food-ordering platforms* during peak hours.[8] Greedy performs well with small queue capacities but saturates as $\underline{L}$ increases, since it prioritizes immediate matches over future demand, resulting in less efficient queue use. Figure 3b shows that SAMP significantly reduces riders' waiting times across priorities under profit maximization, while all baselines stay below a curbing ratio of 1. Similar trends appear in the general case (Figure 4), where SAMP maintains CRs above the bounds in Theorem 3, outperforms all baselines for any η, and reduces average waiting time by over 65% (Figure 4b).

7 Conclusion and Future Directions

In this paper, we proposed a hybrid model combining features from both online matching and queue theory. We presented an LP-based sampling parameterized policies that can effectively promote profit in gig platforms, while reduce users' waiting time to pre-selected guaranteed levels. Our research opens a few directions. The first is to generalize the current setting to heterogeneous arrival rates for online requests. The second is to study the potential tradeoff between profit and fairness within the current framework. We believe strategies such as hybrid samplings from the LP solutions for profit and fairness can lead to compromised but provable performance on both objectives simultaneously.

[8] Gig platforms often relax waiting-time guarantees during peak hours, leading to longer queues; see Equation (5) for their proportional relation.

Acknowledgements

Evan Yifan Xu was partially supported by the China Postdoctoral Science Foundation Funded Project under Grant 2025M771567. Pan Xu was partially supported by the NSF CRII Award IIS-1948157 and the BGU-NJIT seed grant.

References

Ashlagi, I.; Leshno, J.; Qian, P.; and Saberi, A. 2020. Queue lengths as constantly adapting prices: Allocative efficiency under random dynamics. In *Proceedings of the 21st ACM Conference on Economics and Computation*, 317–318.

Ashlagi, I.; Monachou, F.; and Nikzad, A. 2021. Optimal dynamic allocation: Simplicity through information design. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, 101–102.

Baccara, M.; Lee, S.; and Yariv, L. 2020. Optimal dynamic matching. *Theoretical Economics*, 15(3): 1221–1278.

Bloch, F.; and Cantala, D. 2017. Dynamic assignment of objects to queuing agents. *American Economic Journal: Microeconomics*, 9(1): 88–122.

Bose, S. K. 2013. *An introduction to queueing systems*. Springer Science & Business Media.

Brubach, B.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2020. Online stochastic matching: New algorithms and bounds. *Algorithmica*, 82(10): 2737–2783.

Chakraborty, S.; Muthuraman, K.; and Lawley, M. A. 2010. Sequential clinical scheduling with patient no-shows and general service time distributions. *IIE Transactions*, 42: 354 – 366.

Delong, S.; Farhadi, A.; Niazadeh, R.; and Sivan, B. 2022. Online bipartite matching with reusable resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, 962–963.

Dickerson, J. P.; Sankararaman, K. A.; Srinivasan, A.; and Xu, P. 2021. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, 9(3): 1–17.

Dickerson, J. P.; Sankararaman, K. A.; Srinivasan, A.; Xu, P.; and Xu, Y. 2024. Matching Tasks and Workers under Known Arrival Distributions: Online Task Assignment with Two-sided Arrivals. *ACM Transactions on Economics and Computation*.

Feldman, J.; Mehta, A.; Mirrokni, V.; and Muthukrishnan, S. 2009. Online stochastic matching: Beating 1-1/e. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, 117–126. IEEE.

Feng, J.; Gluzman, M.; and Dai, J. G. 2021. Scalable deep reinforcement learning for ride-hailing. In *2021 American Control Conference (ACC)*, 3743–3748. IEEE.

Feng, Y.; Niazadeh, R.; and Saberi, A. 2019. Linear programming based online policies for real-time assortment of reusable resources. *Chicago Booth Research Paper*, (20-25).

Feng, Y.; Niazadeh, R.; and Saberi, A. 2020. Near-optimal bayesian online assortment of reusable resources. *Chicago Booth Research Paper*, (20-40).

Gallager, R. G. 2011. Discrete stochastic processes. *OpenCourseWare: Massachusetts Institute of Technology*.

Gong, X.-Y.; Goyal, V.; Iyengar, G. N.; Simchi-Levi, D.; Udwani, R.; and Wang, S. 2021. Online assortment optimization with reusable resources. *Management Science*.

Goyal, V.; Iyengar, G.; and Udwani, R. 2020. Online allocation of reusable resources via algorithms guided by fluid approximations. *arXiv preprint arXiv:2010.03983*.

Goyal, V.; Iyengar, G.; and Udwani, R. 2021. Asymptotically Optimal Competitive Ratio for Online Allocation of Reusable Resources. In *Web and Internet Economics - 17th International Conference, WINE*, volume 13112, 543.

Gupta, A.; Yadav, R.; Nair, A.; Chakraborty, A.; Ranu, S.; and Bagchi, A. 2022. FairFoody: Bringing In Fairness in Food Delivery. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI*, 11900–11907.

Haeupler, B.; Mirrokni, V. S.; and Zadimoghaddam, M. 2011. Online Stochastic Weighted Matching: Improved Approximation Algorithms. In *Internet and Network Economics*, volume 7090 of *Lecture Notes in Computer Science*, 170–181. Springer Berlin Heidelberg. ISBN 978-3-642-25509-0.

HghSsociety. 2020. Uber Eats just started "Batch Order Routes" & it's terrible. https://forums.redflagdeals.com/uber-eats-just-started-batch-order-routes-its-terrible-2392687/. Accessed: 2025-04-01.

Ho, C.-J.; and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *Twenty-sixth AAAI conference on artificial intelligence*.

Huang, Z.; and Shu, X. 2021. Online stochastic matching, poisson arrivals, and the natural linear program. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 682–693.

Huang, Z.; Shu, X.; and Yan, S. 2022. The power of multiple choices in online stochastic matching. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, 91–103. ACM.

Huet, E. 2015. Uber Test Lets Drivers Accept New Ride Before Finishing Current One. https://www.forbes.com/sites/ellenhuet/2015/07/15/uber-test-back-to-back-rides-lets-drivers-pick-up-before-finishing-current-ride/?sh=f62f6a63c4d9. Accessed: 2025-04-01.

Jaillet, P.; and Lu, X. 2013. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3): 624–646.

Jandiss. 2021. https://www.reddit.com/r/UberEATS/comments/luybf9/i_understand_one_or_2_stops_along_the_way_but_why/. Accessed: 2025-04-01.

Joshi, M.; Singh, A.; Ranu, S.; Bagchi, A.; Karia, P.; and Kala, P. 2022. FoodMatch: Batching and Matching for Food Delivery in Dynamic Road Networks. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(1): 1–25.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An Optimal Algorithm for On-line Bipartite Matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, 352–358.

Kiwidrew. 2017. https://news.ycombinator.com/item?id=15548013. Accessed: 2025-04-01.

Koch, R. 1998. The 80/20 Principle: The Secret of Achieving More With Less.

Krishnamoorthi, B.; and Wood, R. C. 1966. Time-Shared Computer Operations With Both Interarrival and Service Times Exponential. *J. ACM*, 13: 317–338.

Leshno, J. D. 2022. Dynamic matching in overloaded waiting lists. *American Economic Review*, 112(12): 3876–3910.

Liu, Q.; and Hajiesmaili, M. 2025. Online fair allocation of reusable resources. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(2): 1–46.

Liu, Y.; Lou, B.; Zhao, X.; and Li, X. 2024. Unintended consequences of advances in matching technologies: Information revelation and strategic participation on gig-economy platforms. *Management Science*, 70(3): 1729–1754.

Ma, W.; Xu, P.; and Xu, Y. 2022. Group-level Fairness Maximization in Online Bipartite Matching. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 1687–1689.

Manshadi, V. H.; Gharan, S. O.; and Saberi, A. 2012. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4): 559–573.

Manshadi, V. H.; and Rodilitz, S. 2022. Online Policies for Efficient Volunteer Crowdsourcing. *Management Science*, 68(9): 6572–6590.

Matarese, J. 2021. Why your DoorDash, Grubhub or Uber Eats order may be late. https://www.wcpo.com/money/consumer/dont-waste-your-money/why-your-doordash-grubhub-or-uber-eats-order-may-be-late. Accessed: 2025-04-01.

Matyushenko, S.; and Ermolayeva, A. 2021. On stationary characteristics of a multiserver exponential queuing system with reordering of requests. *2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 98–103.

Nair, A.; Yadav, R.; Gupta, A.; Chakraborty, A.; Ranu, S.; and Bagchi, A. 2022. Gigs with Guarantees: Achieving Fair Wage for Food Delivery Workers. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 5122–5128.

Sarwar, N. 2021. Uber May Soon Force You To Pay More If You're In A Rush. https://screenrant.com/uber-priority-ride-fee-rate/. Accessed: 2025-04-01.

Shi, D.; Tong, Y.; Zhou, Z.; Song, B.; Lv, W.; and Yang, Q. 2021. Learning to assign: Towards fair task assignment in large-scale ride hailing. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3549–3557.

Smith, J. M. 2003. M/G/c/K blocking probability models and system performance. *Performance Evaluation*, 52(4): 237–267.

Thakral, N. 2019. Matching with stochastic arrival. In *AEA Papers and Proceedings*, volume 109, 209–12.

Tong, Y.; Shi, D.; Xu, Y.; Lv, W.; Qin, Z.; and Tang, X. 2021. Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale. *IEEE Transactions on Knowledge and Data Engineering*, 35(10): 9812–9823.

Uber. 2020. Introducing Priority Delivery and Restaurant Rewards Programs. https://www.uber.com/newsroom/introducing-priority-delivery-and-restaurant-rewards-programs/. Accessed: 2025-04-01.

Ulmer, M. W.; Thomas, B. W.; Campbell, A. M.; and Woyak, N. 2021. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, 55(1): 75–100.

Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; and Ye, J. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 905–913.