Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

Authors are encouraged to submit new papers to INFORMS journals by means of a style file template, which includes the journal title. However, use of a template does not certify that the paper has been accepted for publication in the named journal. INFORMS journal templates are for the exclusive purpose of submitting to an INFORMS journal and should not be used to distribute the papers in print or online or to submit the papers to another publication.

A Distribution-free Algorithm for Fully Online Matching with Stochastic Arrivals and Departures

Zihao Li, Hao Wang, Zhenzhen Yan
Nanyang Technological University,
zihao004@e.ntu.edu.sg, hao_wang@ntu.edu.sg, yanzz@ntu.edu.sg

We study a fully online matching problem with stochastic arrivals and departures. In this model, each online arrival follows a known identical and independent distribution over a fixed set of agent types. Its sojourn time is unknown in advance and follows type-specific distributions with known expectations. The goal is to maximize the weighted reward from successful matches. To solve this problem, we first propose a linear program (LP)-based algorithm whose competitive ratio is lower bounded by 0.155 under mild conditions. We further achieve better ratios in some special cases. To demonstrate the challenges of the problem, we further establish several hardness results. In particular, we show that no online algorithm can achieve a competitive ratio better than $\frac{2}{3}$ in this model and there is no LP-based algorithm (with respect to our proposed LP) with a competitive ratio better than $\frac{1}{3}$. Finally, we demonstrate the effectiveness and efficiency of our algorithm numerically.

Key words: Fully Online Matching, Distribution-free Bound, Competitive Ratio

1. Introduction

Starting from the seminal work by Karp et al. (1990), online matching has been a fundamental research topic in online resource allocation. Many online matching studies focus on online bipartite matching, where vertices on one side are assumed to be known upfront, and those on the other side arrive online. However, this setting fails to model some modern applications, such as ride-sharing, where all vertices including riders and drivers arrive (become available) in the matching platform in an online manner and depart after a stochastic sojourn time. This paper studies a general model that can capture these properties. In particular, all vertices arrive in the system in an online manner. When a vertex arrives, the edges with the previously arrived vertices are revealed, and it stays in the platform for a random duration. A vertex will be matched to another unmatched *neighboring vertex* (linked to the vertex by an incident edge) before its departure or be left unmatched and depart. If a matching is made between two vertices, a reward determined by the matching pair is generated. The goal is to maximize the total reward of successful matches. We name this general problem a fully online matching problem.

This model has applications in various domains besides ride-sharing. Examples include kidney donation and player matching online games.

Kidney Donation In kidney exchange schemes, incompatible pairs of donors and recipients are seeking mutual exchanges. Donors and recipients arrive in the market randomly in an online manner. They have limited sojourn time in the market. They must be matched within their lifetime, otherwise, they will be abandoned. Their sojourn time in the system is generally uncertain. Different matches might be of different qualities influenced by various effects such as patient-specific mortality rates and the availability of outside options for transplants, hence generate different rewards (Ashlagi and Roth 2021). The goal is to maximize the total matching quality.

Player Matching in Online Games In an one-to-one online game such as chess game, the platform needs to match a player to an opponent to start a game. Players join the platform online and randomly. They may get inpatient and leave the platform without playing the game if the platform fails to find him/her an opponent after a random waiting time. We can measure the quality of a match by the rating difference between the two matched players. The platform's goal is to maximize the total quality of successful matches.

This fully online matching is in general challenging since it generalizes online bipartite matching in several dimensions. First, it is built on a general graph structure, which is not necessarily equipped with some nice properties shared in a bipartite graph. Second, all agents arrive online in contrast to online-to-offline matching in the online bipartite matching. Finally, all the agents will stay in the system for a limited and random duration. Hence it brings another dimension of decision making, that is to determine the matching time besides the matching pair.

There is so far limited literature on the related study. Huang et al. (2020a) and Huang et al. (2020b) are inspiring ones. They assume that agents arrive and depart in an adversary manner. Their goal is to maximize the number of matches. In contrast, in this paper we assume arrivals follow an identical and independent (i.i.d.) probability distribution, which is a common assumption in online matching literature (cf. Huang et al. (2022), Huang and Shu (2021), Jaillet and Lu (2014), Feldman et al. (2009)). Upon arrival, each agent will stay in the system for a sojourn time before leaving the system. We do not specify the exact distribution of the sojourn time but assume it follows a type-specific distribution with known expectations. Finally, we consider maximizing the edge-weighted reward of successful matches, which is more general than the number of successful matches as studied in Huang et al. (2020a) and Huang et al. (2020b). We claim the model settings considered in this paper are more applicable in ride-sharing. The arrival distribution can be easily estimated using customers' arrival data. However, the data on sojourn time are often less available. Hence we consider a distributionally free setting, without assuming a specific distribution but require information on the mean sojourn time. Finally, the rewards from different paired agents are often different. Our model captures this feature by maximizing the total weight of matched pairs.

Another related stream of literature is the dynamic stochastic matching (cf. Aouad and Saritaç (2020) and Collina et al. (2020)). In the dynamic stochastic matching, a type-specific known Poisson process is often assumed for both arrivals and departures. As a result, one can conduct a steady state analysis for the associated Markov decision process using the property of Poisson processes. In contrast, this paper studies a more general setting. Specifically, we do not assume a particular distribution for agents' sojourn time. Instead, we only need the mean sojourn time in our algorithm and analysis. We derive a distribution free bound for the fully online matching problem, which demonstrates a potentially good performance of our algorithm in various problems.

1.1. Our Contributions

We summarize the main contributions as follows. We study a fully online matching model with stochastic arrivals and departures. In particular, the arrivals follow a known i.i.d. distribution, and the sojourn time before agents depart can follow a large family of distributions with a known expectation. The goal is to maximize the total weight (defined on edges) of successful matches. The model settings are applicable in a wide family of applications, including ride-sharing.

1. We design a *distributionally free* LP-based algorithm, and investigate its theoretical performance measured by the competitive ratio, see Theorems 1 and 4. Under mild assumptions, we prove that the competitive ratio of our algorithm is at least 0.155 if the sojourn time is discrete distributed, see Corollary 1. For some specific discrete-type distributions, we can achieve better competitive ratios, see Corollary 2, 3 and 4. Moreover, in the special case where the arrivals and departures both follow Poisson processes (the setting in Collina et al. (2020)), our analysis can generate the same competitive ratio (0.125) as established in Collina et al. (2020).

2. We further establish two hardness results to foreground the technical challenges of the problem we study. We first show no online algorithm can achieve a competitive ratio of more than $\frac{2}{3}$, see Theorem 2. But if we restrict the algorithms to LP-based algorithms with respect to the LP we derive, we show that there exists no such online algorithm with a competitive ratio larger than $\frac{1}{3}$, see Theorem 3.

3. Extensive numerical studies are conducted to evaluate the performance of our algorithms, including experiments based on synthetic data and the New York City taxi data. Our algorithms can significantly outperform the baseline algorithms from related works in most parameter settings.

1.2. Related Work

There is extensive literature on online bipartite matching, where there exists a set of offline vertices, and each online vertex will be matched to an offline vertex *immediately* upon its arrival or be rejected. A seminar work by Karp et al. (1990) considered maximizing the number of matches when agents arrive in an adversary setting. A follow-up work by Manshadi et al. (2012) considered a stochastic arrival process and proposed a 0.702-competitive algorithm, which improves the competitive ratio of $1-1/e$ provided in Karp et al. (1990). Many other works further consider generalizing the objective to maximizing the vertex-weighted (or edge-weighted) matching under a stochastic arrival model (Feldman et al. 2009, Aggarwal et al. 2011, Huang and Shu 2021, Huang et al. 2022). To the best of our knowledge, the best bound under a stochastic arrival model is achieved by Huang et al. (2022). They provided a 0.716-competitive algorithm in a vertex-weighted setting and a 0.706-competitive algorithm in an edge-weighted with free disposal setting, i.e., each offline vertex can update its matching vertex upon new arrivals. Chen and Wang (2015) further generalized the linear objective function to a concave reward function and proposed a near-optimal dynamic learning algorithm.

Another flow of works on online bipartite matching allows randomness in the success of each matching edge (Mehta and Panigrahi 2012, Mehta et al. 2014, Golrezaei et al. 2014, Ma and Simchi-Levi 2020, Goyal and Udwani 2022). To the best of our knowledge, Ma and Simchi-Levi (2020) considered the most generalized model, where each edge is associated with multiple feasible prices and the success probability of each matching edge depends on the chosen price of this edge. They then provided the algorithms with the best-possible weight-dependent competitive ratios.

Recently, fully online matching has attracted increasing attentions, where each vertex has its arrival and departure time and can be matched *anytime* before it departs. In other words, a delay in matching is allowed. Our paper lies in this steam of research. Starting from a non-weighted setting, Huang et al. (2020a,b, 2019), Eckl et al. (2021) studied fully online matching with adversarial arrivals and departures, and provided a 0.569-competitive algorithm and hardness results. Considering edge-weighted reward and assuming fixed and identical sojourn time, Ashlagi et al. (2019) proposed a 0.25-competitive algorithm when agents arrive in an adversary manner and a 0.279-competitive algorithm when the arrival sequence follows a random order model. Several papers focus on the setting where both arrival and departure follow a type-specific Poisson process. Collina et al. (2020) proposed a 0.125-competitive algorithm for an edge-weighted setting, where the goal is to maximize total weights defined on edges. Aouad and Saritaç (2020) studied a dynamic stochastic matching with the same arrival and departure process. They modeled the problem as an infinite-horizon continuous-time Markov decision process and provided an approximation policy that can achieve $\frac{e-1}{4e} \approx 0.158$ of the optimality, in sharp contrast with the competitive ratio for online matching problems. Our paper differs from those papers in the following perspectives. First, all online vertices arrive according to a known i.i.d. distribution. Second, we don't assume a specific type of distribution for agents' sojourn time. In other words, our algorithm works for a large family of distributions with a known expectation and is robust when the distribution varies.

Another related stream of research is the delayed matching (cf. Ashlagi et al. (2017), Emek et al. (2016), Azar and Fanani (2020), Azar et al. (2017), Wang and Bei (2022), Hu and Zhou (2022)). In the delayed matching, instead of imposing a hard constraint for the match to be restricted in a time interval, the model penalizes the delay by adding a delay cost in the total cost function. Compared to this alternative modeling perspective, our modeling perspective by incorporating agents' limited duration in the system is more practically relevant in the ride-sharing application.

2. Preliminaries

We consider the following online matching problem. Given an edge-weighted graph $G = (V, E)$, each vertex $v \in V$ represents one agent type and each edge $e = (x, y) \in E$ connects two vertices x and y with a weight $w_e \in \mathbb{R}_{\geq 0}$. Self-loops are allowed.

For the online process, we consider a given time horizon of T. For each time $t \in \{1, 2, \dots, T\}$, one agent arrives and is represented by (x, d), where x is the agent type in V and d is the sojourn time of this agent. We will start our analysis for discrete-type sojourn time by defining the sojourn time as the number of future agents that an agent is willing to wait for. Later in Section 6, we will extend our analysis to the setting with continuous sojourn time, where the sojourn time is defined as the duration for an agent staying in the system before its departure.

At each time, an online agent (x, d) is determined in the following way. x is chosen from a known i.i.d. distribution $\{p_v\}$ where $\sum_{v \in V} p_v = 1$ and $\Pr[x = v] = p_v$ for all $v \in V$. d is chosen from a discrete distribution $\mathbb{D}_x$ and unknown to us until it departs. For each $v \in V$, we only know the expectation D_v but not the specific distribution of $\mathbb{D}_v$.

After an arrival, we can match some available vertex pair(s) irrevocably. Here an available pair is a pair of vertices (i, j) that have not departed or been matched, and are connected, i.e., $e = (x^i, y^j) \in E$, where x^i and y^j are their corresponding types. For each pair matched, we can gain a reward w_e where e is its corresponding edge. Our goal is to maximize the total reward over the whole time horizon.

Note that for each $x, y \in V$ we can always add one edge $e = (x, y)$ with weight 0 in E and only retain the one with the largest weight for reward maximization. Hence we assume G is a complete graph in the following analysis for simplicity, i.e., $\forall x, y \in V$, there exists *exactly* one edge (x, y) in E. We use w_{xy} and w_e for edge $e = (x, y)$ interchangeably in the following analysis.

Competitive ratio. We use competitive ratio to measure the performance of online algorithms. For an online algorithm ALG and an instance I of our problem, we use ALG(I) to represent the expected total reward output by ALG on I. Here, the expectation is taken over random arrival sequences of online agents, the random sojourn time of each online agent and the randomized (if needed) algorithm. Similarly, we can define OPT(I) as the expected total reward output by a clairvoyant optimal algorithm OPT, where this algorithm holds the information of all the subsequent agents (x,d). We also call OPT(I) the offline optimal, and we will drop I when there is no ambiguity. The competitive ratio of ALG is defined as the minimum ratio of ALG(I) over OPT(I) among all instances I of our problems.

3. Linear Programming Benchmark

To bound the competitive ratio, we first provide a linear program to bound the OPT. We define a variable n_{xy} for each ordered pair (x,y) where $x,y \in V$. We then define a benchmark LP (1) as follows.

$$\mathbf{max} \sum_{x,y\in V} w_{xy} n_{xy} \tag{1}$$

$$\mathbf{s.t.} \sum_{y\in V} n_{xy} + \sum_{y\in V} n_{yx} \le p_x T, \qquad \forall x \in V, \tag{1a}$$

$$n_{xy} \le p_x T p_y D_x, \qquad \forall x,y \in V, \tag{1b}$$

$$n_{xy} \ge 0, \qquad \forall x,y \in V, \tag{1c}$$

We show in Lemma 1 that LP (1) is a relaxation of the offline optimal. The intuition behind the proof is as follows. We denote each n^*_{xy} as the expected number of times that an online agent of type y is matched a previous unmatched online agent of type x according to OPT. We then show such $\{n^*_{xy}\}$ is a feasible solution to LP (1).

Lemma 1. *For any instance I, the optimal value of LP* (1) *is an upper bound of OPT(I).*

Proof Let r denote a realization of instance I as one possible input sequence of agent types with the corresponding sojourn time after randomization. We further define $n_{r,x,y}$ as the number of matches between one agent of type y and one previous agent of type x under realization r given by OPT and P_r as the probability of the realization r.

We can define $n^*_{xy} = \sum_r P_r n_{r,x,y}$. Since $\text{OPT}(I) = \sum_r P_r \sum_{x,y\in V} n_{r,x,y} w_{xy} = \sum_{x,y\in V} w_{xy} \left(\sum_r P_r n_{r,x,y}\right)$, which is equal to $\sum_{x,y\in V} w_{xy} n^*_{xy}$. It suffices to show $\{n^*_{xy}\}$ is one feasible solution to LP (1).

The first is to check the feasibility of Constraints (1a). $\forall x \in V$, $\sum_{y\in V} n^*_{xy} + \sum_{y\in V} n^*_{yx} = \sum_r P_r (\sum_{y\in V} n_{r,x,y} + n_{r,y,x}) \le \sum_r P_r(\text{number of type } x \text{ in realization } r) = p_x T$. The inequality is because the number of matched agents of type x cannot exceed the number of appearing agents of type x in any realization. The last equality is from the linearity of expectation.

Since Constraints (1c) are obviously satisfied, the remaining is to show Constraints (1b) are satisfied. $\forall x, y \in V$, we have $n^*_{xy} = \sum_r P_r n_{r,x,y} \le \sum_r P_r$(number of the occurrence of event E in r), where event E is that one agent of type x can see one following agent of type y. The last inequality has the same reason as that of the above inequality. By linearity of expectation, we can transform it into the sum of expected times of one agent of type x at time t seeing one following agent of type y over all t from 1 to T, which is upper bounded by the multiple of Tp_x and the expected times of one following agent of type y existing in the sojourn time of one agent of type x. We assume the support set of $\mathbb{D}_x$ is S_x where the probability of support $s \in S_x$ is q_s. So n^*_{xy} is upper bounded by $Tp_x \sum_{s\in S_x} q_s \cdot p_y s$ from linearity of expectation, which is equal to $p_x T p_y D_x$. □

For analysis convenience, we let α_{xy} be $\frac{n_{xy}}{p_y T}$ for all $x, y \in V$. $\alpha_{xy} \le 1$ according to Constraints (1a). Then we can reformulate LP (1) as LP (2), and we will use LP (2) in the following analysis.

$$\mathbf{max} \sum_{x,y\in V} w_{xy}\alpha_{xy}p_y T \tag{2}$$

$$\mathbf{s.t.} \sum_{y\in V}\alpha_{xy}p_y + \sum_{y\in V}\alpha_{yx}p_x \le p_x, \qquad \forall x \in V, \tag{2a}$$

$$\alpha_{xy} \le p_x D_x, \qquad \forall x, y \in V, \tag{2b}$$

$$\alpha_{xy} \in [0,1], \qquad \forall x, y \in V, \tag{2c}$$

4. Approximation Algorithms

Inspired by the algorithm used in Collina et al. (2020), we propose our LP-based Algorithm 1. In the algorithm, we set the matching probability according to the optimal solution $\{\alpha_{xy}\}$ to LP (2). Specifically, the matching probability between an arriving agent of type y and an existing agent of type x is set to $\gamma \cdot \alpha_{x,y}/(p_x D_x)$, where γ is a scaling parameter and the term $1/(p_x D_x)$ is designed to increase the matching probability appropriately. The matching probability is not greater than 1 according to Constraints (2b) and $\gamma \le 1$. We use J to denote the *multiset* of types of all existing unmatched agents when an agent i of type $y \in V$ arrives. We enumerate all elements x in J in a uniformly random order and match agent i with the specific agent j of type x with the above probability (Lines 5-6 in Algorithm 1). When an agent j is matched with i successfully, no further enumeration is needed. Algorithm 1 is *solvable in polynomial time* since LP (2) can be solved in polynomial time and the number of computations per arrival is $O(|J|)$ where the size $|J|$ of the set J defined in Line 3 of Algorithm 1 can be bounded by the maximum support among all $\mathbb{D}_v$s of $v \in V$.

It is worthwhile to mention that although our algorithm is motivated from Collina et al. (2021), it has improved their algorithms in adaption to our general model settings. In addition, we manage to conduct a comprehensive and non-trivial analysis of the algorithm to achieve a better performance guarantee as established in their paper. In the subsequent parts, we will elaborate in detail how we analyze the competitive ratio of Algorithm 1. In the following analysis in this section, we assume the maximal value in the support of the distribution $\mathbb{D}_v$ is much lower than T for each $v \in V$. The assumption is mild in ride-sharing applications since the time horizon is much larger than the possible sojourn time of every agent.

Algorithm 1 Sam(γ)

```
Input: Online arrivals of agents
Parameter: Scaling parameter $\gamma \in (0,1]$
1: $\{\alpha_{xy}\}$ := Solution to LP (2);
2: for each arriving agent $i$ whose type is $y \in V$ do
3:     $J$ := The multiset of types of unmatched agents;
4:     for each type $x \in J$ in a uniformly random order do
5:         $j$ := The corresponding unmatched agent of type $x$;
6:         Match $i$ and $j$ w.p. $\gamma \cdot \alpha_{xy}/(p_x D_x)$;
7:     end for
8: end for
```

4.1. Analysis

Note that the total weight generated by OPT cannot be greater than the optimal value of LP (2) from Lemma 1, we can compare the performance of Algorithm 1 with the value of LP (2) to get a lower bound of the competitive ratio. Thus, the strategy of calculating the competitive ratio is to lower bound the ratio between the expected number of successful matches and the term $\alpha_{xy} p_y T$ in the objective function of LP (2), for each ordered pair (x,y), where $x, y \in V$. Here, we only consider the pair (x,y) such that $D_x > 0$ since agents of type x will not wait otherwise.

We assume an agent i of type $y \in V$ arrives at time t. We will calculate the probability of matching this agent to an existing agent j of type $x \in V$ who arrives at time $t' < t$ by considering four events separately. We define these four events as follows.

- E_1: An agent i of type y arrives at time t.
- E_2: An agent j who arrives earlier (at $t' < t$) and is of type x is unmatched at time t'
- E_3: No arriving agent between time $t'+1$ and $t-1$ matches agent j given the occurrence of events E_1, E_2.

- E_4: An agent i of type y is matched to the agent j of type x given the fact that agent j is unmatched before time t.

We first analyze the probability of the first event E_1. From the assumption of i.i.d. arrivals we can easily derive Lemma 2.

LEMMA 2. *The probability of E_1 is p_x.*

To calculate the probability of E_2, we first introduce a vector $\vec{b}$ to store the information of unmatched agents at time t', where each element b_z equals the number of type z in set J defined in Line 3 of Algorithm 1. By conditioning on the probability distribution over $\vec{b}$, we upper bound the probability that there is one agent of type $z \in V$ matching to the agent j. By union bound, we get Lemma 3.

LEMMA 3. *The probability of E_2 is at least $1-\gamma$.*

Proof We use a vector $\vec{b}$ to store the information of unmatched agents, i.e. the element b_x is equal to the number of x in set J in Line 3. Thus,

$$\begin{aligned}
\Pr[E_2] &= \sum_{\vec{b}} \Pr[\vec{b}] \Pi_{z \in V} \left(1 - \frac{\gamma \alpha_{zx}}{p_z D_z}\right)^{b_z} \\
&\geq \sum_{\vec{b}} \Pr[\vec{b}] \left(1 - \gamma \sum_{z \in V} \frac{b_z \alpha_{zx}}{p_z D_z}\right) \\
&= 1 - \gamma \sum_{z \in V} \frac{\alpha_{zx}}{p_z D_z} \sum_{\vec{b}} \Pr[\vec{b}] b_z \\
&\geq 1 - \gamma \sum_{z \in V} \frac{\alpha_{zx}}{p_z D_z} \cdot p_z D_z \\
&= 1 - \gamma \sum_{z \in V} \alpha_{zx} \geq 1 - \gamma
\end{aligned}$$

The first equality is from the description of algorithm 1. The first inequality is from the union bound and the second inequality is because at any fixed time, the total number of remaining unmatched agent of one fixed type is not greater than the total number of all existing agent of this type. The last inequality is from Constraints (2a) after ignoring the first term in left hand side. □

Next, we bound the probability of E_3 by providing an upper bound of the event that an arriving agent at time $t'' \in [t'+1, t-1]$ matches agent j. Then, using the independence between different t'', we establish Lemma 4.

LEMMA 4. *If $D_x \geq 1$, the probability of E_3 is at least $(1 - \gamma/D_x)^{t-t'-1}$.*

Proof

$$\Pr[E_3] \geq \left(1 - \sum_{z \in V} p_z \frac{\gamma \alpha_{xz}}{p_x D_x}\right)^{t-t'-1}$$

$$= \left(1 - \frac{\gamma}{p_x D_x}\sum_{z\in V} p_z \alpha_{xz}\right)^{t-t'-1}$$
$$\geq \left(1 - \frac{\gamma}{D_x}\right)^{t-t'-1}$$

The first inequality is because at each time between $t'+1$ and $t-1$, the matching probability of the arriving agent of one possible type z at this time with agent j is upper bounded by $\frac{\gamma\alpha_{xz}}{p_x D_x}$ by the description of our algorithm. The second inequality is from Constraint (2a) after ignoring the second term in left hand side. □

The remaining is to bound the probability of the agent i of type y matching the agent j of type x given the fact that agent j is unmatched before time t, i.e., E_4. Intuitively, there are two parts needed to match i to j successfully. The first is that j can be matched to i in Line 6 of Algorithm 1, and the second is that all unmatched type $z \in J$ that joins J earlier than the agent j's type x cannot match i successfully. We will bound these two terms respectively in the proof.

Lemma 5. *The probability of E_4 is at least $\frac{\gamma\alpha_{xy}}{p_x D_x}\left(1-\frac{\gamma}{2}\right)$.*

Proof We define $\vec{b}$ in the same way as above, but here one difference from the above $\vec{b}$ in the proof of Lemma 3 is that the corresponding x of agent j is not counted in $\vec{b}$.

$$\Pr[E_4] \geq \frac{\gamma\alpha_{xy}}{p_x D_x}\left(1 - \sum_{\vec{b}} \Pr[\vec{b}] \sum_{z\in V} \frac{\gamma\alpha_{zy} b_z}{2p_z D_z}\right)$$
$$= \frac{\gamma\alpha_{xy}}{p_x D_x}\left(1 - \frac{\gamma}{2}\sum_{z\in V} \frac{\alpha_{zy}}{p_z D_z} \sum_{\vec{b}} \Pr[\vec{b}] b_z\right)$$
$$\geq \frac{\gamma\alpha_{xy}}{p_x D_x}\left(1 - \frac{\gamma}{2}\sum_{z\in V} \frac{\alpha_{zy}}{p_z D_z} p_z D_z\right)$$
$$= \frac{\gamma\alpha_{xy}}{p_x D_x}\left(1 - \frac{\gamma}{2}\sum_{z\in V} \alpha_{zy}\right)$$
$$\geq \frac{\gamma\alpha_{xy}}{p_x D_x}\left(1 - \frac{\gamma}{2}\right)$$

The reason of the first inequality is that if we want to match i and j successfully, besides that i can match j in Line 6 corresponding to the first term, all $z \in J$ before the corresponding x of j cannot match i successfully. Because the probability of one $z \in J$ before the corresponding x of j is $\frac{1}{2}$ from the uniformly random order and the matching probability is $\frac{\gamma\alpha_{zy} b_z}{p_z D_z}$, we can calculate one upper bound of the probability of the event that there exists one $z \in J$ before the corresponding x of j matching i successfully by union bound.

The second inequality is because the total number of unmatched agents of each type z is not greater than the total number of existing agents of type z at time t, which is further upper bounded by $p_z D_z$ because of the difference of the existence of agent j at time t'. The last inequality is again from Constraints (2a) after ignoring the first term in left hand side. $\square$

Besides the analysis of the four events, we need to utilize another lemma to calculate the total ratio between the expected matching number of (x,y) and the term $\alpha_{xy} p_y T$, which can be proved by induction.

LEMMA 6. $(1-x)^d \le 1 - dx + \frac{d(d-1)}{2}x^2$, *for all* $x \in [0,1]$ *and all non-negative integer* d.

Proof We can prove this by a simple induction. When $d = 0$, this is satisfied obviously. It suffices to show it's satisfied for $d+1$ when it's true for d.

$$\begin{aligned}(1-x)^{d+1} &= (1-x)^d(1-x)\\ &\le \left(1 - dx + \frac{d(d-1)}{2}x^2\right)(1-x)\\ &= 1-(d+1)x + \frac{d(d-1)}{2}x^2 + dx^2 - \frac{d(d-1)}{2}x^3\\ &\le 1-(d+1)x + \frac{d(d+1)}{2}x^2\end{aligned}$$

$\square$

Now we are ready to formally present our main result.

THEOREM 1. *Under the assumption that* $D_v \ge 1$ *for all* $v \in V$*, the competitive ratio of Algorithm 1 with parameter* γ *is at least* $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right)$*, where we define* Var_v *as the variance of the distribution* $\mathbb{D}_v$ *and* $C = \min_{v\in V} \frac{D_v - Var_v}{D_v^2}$.

Proof We assume the support set S of the distribution $\mathbb{D}_x$ with probability q_s of support $s \in S$. Fix agent i of type y at time t, we want to calculate the expected matching number of agent j of type x, which should be equal to $\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \Pr[E_1]\Pr[E_2]\Pr[E_3]\Pr[E_4]$. Here, because of the assumption that $T \gg s$ and we will also enumerate all possible ts, the case that $t - s < 1$ can be ignored, so we can directly start t' from $t-s$ but not $\max\{1, t-s\}$.

Then, by observation, the lower bound of $\Pr[E_1]$, $\Pr[E_2]$ and $\Pr[E_4]$ don't contain the term t', we can directly move these terms outside and only focus on the value $\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \Pr[E_3]$, which is equal to

$$\begin{aligned}&\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \left(1 - {}^{\gamma}\!/_{D_x}\right)^{t-t'-1}\\ &= \sum_{s\in S} q_s \frac{1-\left(1-{}^{\gamma}\!/_{D_x}\right)^s}{{}^{\gamma}\!/_{D_x}}\\ &\ge \sum_{s\in S} q_s \frac{s\cdot {}^{\gamma}\!/_{D_x} - \frac{1}{2}s(s-1)\cdot\left({}^{\gamma}\!/_{D_x}\right)^2}{{}^{\gamma}\!/_{D_x}}\end{aligned}$$

$$\begin{aligned}
&\geq D_x - \frac{\gamma}{2D_x}\sum_{s\in S} q_s(s^2-s) \\
&= D_x + \frac{\gamma}{2} - \frac{\gamma}{2D_x}\left(Var_x + D_x^2\right) \\
&= D_x\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}\frac{D_x - Var_x}{D_x^2}\right)
\end{aligned}$$

Here, the inequality is from Lemma 6. Since t can choose from 1 to T and agent i of type y will arrive w.p. p_y, the total expected number of ordered pair (x,y) should be at least

$$\begin{aligned}
&Tp_yp_x(1-\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\right)D_x\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right) \\
&= Tp_y\alpha_{xy}\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right)
\end{aligned}$$

where $C = \min_{v\in V}\frac{D_v - Var_v}{D_v^2}$. □

Theorem 1 provides a lower bound of competitive ratio with respect to γ and C. Note that C depends on the mean and variance of the online arrivals' sojourn time. In the remaining part, we will discuss several special types of distributions to get their bounds. For each type, we can tune γ to achieve the best bound.

Corollary 1. *Under the assumptions that $D_v \geq Var_v$ and $D_v \geq 1$ for all $v \in V$, the competitive ratio of algorithm 1 is at least $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)^2$. By setting $\gamma^* = \frac{7-\sqrt{17}}{8} \approx 0.360$, the competitive ratio is at least* 0.155*.*

We claim the assumptions in Corollary 1 are mild since they hold for many classic discrete distributions, such as binomial distribution, Poisson distribution, hypergeometric distribution, and geometric distribution with parameter $p^G \geq 0.5$ (refer to technical appendix for its definition).

4.2. Specific Distributions

In this part, we will discuss several special types of the distributions $\mathbb{D}_v$ on the achieved competitive ratios. First, we consider the case where the distribution $\mathbb{D}_v$ is a single-point distribution, i.e., each type $v \in V$ has fixed sojourn time D_v.

Corollary 2. *Under the assumption that each type $v \in V$ has fixed sojourn time D_v, the competitive ratio of algorithm 1 is at least* 0.159 *by setting $\gamma^* \approx 0.373$.*

Proof Most are similar to the proof of Theorem 1. We replace the inequality from Lemma 6 by the inequality $(1 - {}^{x}\!/_{d})^d \leq e^{-x}$ which is satisfied when $x \in [0,1]$ and d is a positive number, and we have $\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \Pr[E_3]$ is at least $\frac{1-e^{-\gamma}}{\gamma/D_x}$. Thus, the total expected number of ordered pair (x,y) is at least

$$\begin{aligned}
&Tp_yp_x(1-\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\right)\frac{1-e^{-\gamma}}{\gamma/D_x} \\
&= Tp_y\alpha_{xy}(1-\gamma)\left(1-\frac{\gamma}{2}\right)(1-e^{-\gamma}).
\end{aligned}$$

When $\gamma \approx 0.373$, the competitive ratio is the largest, which is ≈ 0.159. □

Second, we consider three common discrete distributions: Poisson distribution Poi(λ^P), geometric distribution Geo(p^G) and binomial distribution B(n^B, p^B). The detailed definitions of these distributions can be found in the technical appendix. The performance over these three distributions will also be further evaluated in the following experiments section. Since the expectation and the variance of Poi(λ^P) are both λ^P, we get the same competitive ratio as in Corollary 1 if $\lambda^P \geq 1$. For the remaining two distributions, from the simple expressions of the expectation and variance, we can transform the formula in Theorem 1 to $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)(1-(1-p^G)\gamma)$ for Geo(p^G) and $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}(1-\frac{1}{n^B})\right)$ for B(n^B, p^B) to get the respective competitive ratios (see Corollaries 3 and 4, respectively).

Corollary 3. *Under the assumption that each distribution $\mathbb{D}_v$ is a geometric distribution Geo(p_v^G), by setting $\gamma^* = 0.35$, the competitive ratio of Algorithm 1 is at least $0.122 + 0.066p^G$, where p^G is the smallest p_v^G for all $v \in V$.*

Corollary 4. *Under the assumption that each distribution $\mathbb{D}_v$ is a binomial distribution B(n_v^B, p_v^B) satisfying $D_v = n_v^B p_v^B \geq 1$, by setting $\gamma^* = 0.40$, the competitive ratio of Algorithm 1 is at least $0.154 + \frac{0.038}{n^B}$, where n^B is the largest n_v^B for all $v \in V$.*

Compared to the results in Corollary 1, we can get a better guarantee when $p^G > 0.5$ and $n^B < 38$, respectively. Moreover, the above choice of γ^* is from the consideration of being able to achieve relatively good performance over all possible values of p^G and n^B. If more refined ranges of p^G and n^B are given, we can adjust the value of γ^* to reach a better performance.

5. Hardness Results

In this section, we will present hardness results to foresee the complexity of the studied problem and demonstrate the quality of our proposed algorithm. We will first show that no online algorithm can reach a competitive ratio better than $\frac{2}{3}$. Next, by restricting the algorithms to LP-based online algorithms with respect to our LP (2), we further show that no LP-based online algorithm with respect to LP (2) can obtain a competitive ratio better than $\frac{1}{3}$.

Theorem 2. *No online algorithm can reach a competitive ratio better than $\frac{2}{3}$.*

Proof We consider such an instance:

- $T \to \infty$ and $V = \{1, 2\}$;
- $p_1 = \varepsilon$ and $p_2 = 1 - \varepsilon$ where ε is significantly small;
- $\mathbb{D}_1$ and $\mathbb{D}_2$ are both single-point distributions where $D_1 = 0$ and $D_2 = 1$;
- $w_{(1,2)} = \frac{1}{\varepsilon(1-\varepsilon)}$, $w_{(1,1)} = 0$ and $w_{(2,2)} = 1$;

We define $f(t)$ as the expected value of t rounds output by the online optimal algorithm given the first two agents are of type 2 and define $g(t)$ as the expected value of T rounds output by the online optimal algorithm given the first agent is of type 1. Our decision is needed only for each $f(t)$ with $t \geq 2$.

For $f(2)$, the optimal decision is to match the existing two agents of type 2, which means $f(2) = 1$. For $f(3)$, the value is the maximum of $q_3 \cdot w_{(2,2)} + (1 - q_3) \cdot (p_1 \cdot w_{(1,2)} + p_2 \cdot f(2))$, where $q_3 \in [0,1]$ is the decision parameter such that we match the existing two agents of type 2 with probability q_3. Since $p_1 \cdot w_{(1,2)} = \frac{1}{1-\varepsilon} > 1 = w_{(2,2)}$, $q_3 = 0$ is the optimal strategy.

We next consider $f(t)$ with $t \geq 4$. We again denote $q_t \in [0,1]$ as the decision parameter such that we match the first two agents of type 2 with probability q_t. If we match the first two agents, we get the expected value $1 + p_1 \cdot (0 + g(t-2)) + p_2 p_1 \cdot (w_{(1,2)} + g(t-3)) + p_2 p_2 \cdot f(t-2)$, and we denote it by A_t, where the three terms except the first one are corresponding to the following arrival sequence of type (1), $(2,1)$ and $(2,2)$, respectively. If we don't match the first two agents, we get the expected value $p_1 \cdot (w_{(1,2)} + g(t-2)) + p_2 \cdot f(t-1)$, and we denote it by B_t, where these two terms corresponding to the following arrival type 1 and 2, respectively. The value with respect to q_t is equal to $q_t A_t + (1 - q_t) B_t$. $f(t)$ is the optimum among them.

We then compare A_t and B_t. Since the representation of B_t also holds for the case when $t = 3$, we replace $f(t-1)$ in the representation of B_t by $f(t-1) \geq B_{t-1}$. So we have $B_t \geq 1 + \frac{1}{1-\varepsilon} + \varepsilon g(t-2) + \varepsilon(1-\varepsilon) g(t-3) + (1-\varepsilon)^2 f(t-2) > A_t$. Thus, $q_t = 0$ is the optimal strategy again.

To sum up, since $f(2) = 1$, the expected value output by the online optimal algorithm is not greater than the sum of the expected value output by the strategy which only matches agents between type 2 and type 1 and one.

We now compare the expected values output by the offline and the online optimal algorithm.

The offline optimal algorithm will match every pair of the type sequence $(2,1)$, which is equal to $p_2 p_1 T w_{(1,2)} + o(T)$. Considering the expected matching number of type sequence $(2,2)$, for every consecutive sequence of agents of type 2, if the total number len is even, the matching number is at least $(len - 2)/2$, while if the total number len is odd, the matching number is $(len - 1)/2$. With the fact that the total number of consecutive sequence is at most the number of agents of type 1 plus 1, the expected matching number of type sequence $(2,2)$ is lower bounded by $\frac{p_2 - 2p_1}{2} T + o(T)$. Thus, the expected value output by the offline optimal algorithm is $p_2 p_1 T w_{(1,2)} + \frac{p_2 - 2p_1}{2} T + o(T)$.

Then, in the strategy which only matches agents between type 2 and type 1, the expected value is exactly $p_2 p_1 T w_{(1,2)} + o(T)$. So the expected value output by the online optimal algorithm is $p_2 p_1 T w_{(1,2)} + o(T)$.

Replacing all the variables by ε and T, the competitive ratio is $\frac{2}{3(1-\varepsilon)}$, which is $\frac{2}{3}$ when ε is significantly small. $\square$

THEOREM 3. *No LP-based online algorithm with respect to LP (2) can reach a competitive ratio better than $\frac{1}{3}$.*

Proof We consider such an instance:

- $T = 3$ and $V = \{1, 2\}$;
- $p_1 = p_2 = 0.5$;
- $\mathbb{D}_1$ and $\mathbb{D}_2$ are both single-point distributions where $D_1 = 2, D_2 = 0$;
- $w_{(1,2)} = 1$ and $w_{(1,1)} = w_{(2,2)} = 0$;

In this instance, the optimal value of LP (2) is $\frac{3}{2}$, while the offline optimal value is $\frac{1}{2}$. □

6. Extending the Analysis to Continuous Sojourn time

Note that we have assumed discrete-type sojourn time so far. In fact, the developed algorithm and its analysis can be easily extended to the setting with continuous sojourn time. In this section, we will elaborate this extension. In particular, we define the sojourn time as the duration for an agent staying in the system before its departure. We assume that the sojourn time of agents in type v is chosen from a general (continuous or discrete) distribution $\mathbb{D}_v$ with a non-negative support. Similar to the settings in previous sections, we again assume only information on its expectation D_v without the knowledge on its specific distribution $\mathbb{D}_v$ for each $v \in V$. For the sake of analysis, we further assume that the arrivals of agents in each type $v \in V$ follow an independent type-specific Poisson point process. Specifically, agents of type v arrives at a rate $p_v > 0$. Given a time horizon T, we consider all events (arrivals and departures) which occur in the time interval $[0, T]$.

We follow a similar analysis procedure as in sections 3 and 4. Specifically, we first use LP (1) to upper bound the optimal expected total reward and use the reformulated LP (2) to design the algorithm and analyze its performance. Specifically, the same Algorithm 1 is applied in this continuous-time setting. We again assume the maximal value in the support of the distribution $\mathbb{D}_v$ is much lower than T for each $v \in V$ below.

We assume an agent i of type $y \in V$ arrives at time t and calculate the probability of matching this agent to an existing agent j of type $x \in V$ who arrives at time $t' < t$ by considering several events, which will be defined when analyzing their probabilities. It is notable that we only need to consider the type x with $D_x > 0$, since otherwise it will not wait to match other agents.

The first event E_1 is defined as the case that the agent j of type x who arrives at time $t' < t$ is not matched to one agent before her.

LEMMA 7. *The probability of E_1 is at least $1 - \gamma$.*

The proof of this lemma follows the same process as in Lemma 3, by using a vector $\vec{b}$ to store the information of unmatched agents at time t' and calculating the probability by conditioning on the probability distribution over $\vec{b}$.

Next, we use E_2 to represent the event that no arriving agents between time t' and t matches agent j given the occurrence of event E_1.

LEMMA 8. *The probability of E_2 is at least $e^{\frac{-\gamma(t-t')}{D_x}}$.*

Proof Since the probability of the arriving agent of one possible type z with agent j is upper bounded by $\frac{\gamma\alpha_{xz}}{p_x D_x}$, the matching event between agents of type z and agent j follows an exponential distribution with a parameter weakly less than $\frac{\gamma\alpha_{xz}p_z}{p_x D_x}$.

From the independency of different possible types of z, we have:

$$\begin{aligned}\Pr[E_2] &\geq e^{\sum_{z\in V}\frac{-\gamma\alpha_{xz}p_z(t-t')}{p_x D_x}}\\ &= e^{\frac{-\gamma(t-t')}{p_x D_x}\sum_{z\in V}\alpha_{xz}p_z}\\ &\geq e^{\frac{-\gamma(t-t')}{D_x}}\end{aligned}$$

The last inequality is from Constraint (2a) after ignoring the second term in the left hand side. □

The remaining is to measure the probability of the agent i of type y matching the agent j of type x given the fact that agent j is unmatched before time t, denoted as the event E_3. We can utilize the same proof as in the proof of Lemma 5 to induce the similar result in Lemma 9.

LEMMA 9. *The probability of E_3 is at least $\frac{\gamma\alpha_{xy}}{p_x D_x}\left(1-\frac{\gamma}{2}\right)$.*

Built on these lemmas, we are ready to present Theorem 4. Note that Theorem 4 is applicable for general sojourn time including continuous and discrete distributed time. But we only prove for the case where each $\mathbb{D}_v$ is a continuous distribution since we can adopt the similar arguments as in Theorem 1 for the discrete case. It is notable that in a special case where agents leave the system following a Poisson process, our Theorem 4 can generate the same competitive ratio (0.125) as established in Collina et al. (2020).

THEOREM 4. *Under the assumption that $D_v \geq 1$ for all $v \in V$, the competitive ratio of Algorithm 1 with parameter γ is at least $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right)$, where we define Var_v as the variance of the distribution $\mathbb{D}_v$ and $C = \min_{v\in V}\frac{-Var_v}{D_v^2}$.*

Proof We assume the support set S of the distribution $\mathbb{D}_x$ with density q_s of $s \in S$. Fix agent i of type y at time t, we want to calculate the expected matching number of agent j of type x, which should be equal to $\int_{s\in S} q_s \int_{t-s}^{t} p_x \Pr[E_1]\Pr[E_2]\Pr[E_3]dt'ds$. Here, because of the assumption that $T \gg s$, we can ignore the case that $t-s<0$ and directly start t' from $t-s$ but not $\max\{1, t-s\}$.

By observation, the lower bound of $\Pr[E_1]$ and $\Pr[E_3]$ don't contain the term t', we can move these terms outside and only focus on the value $\int_{s\in S} q_s \int_{t-s}^{t} \Pr[E_2]dt'ds$, which is

$$
\begin{aligned}
&\int_{s\in S} q_s \int_{t-s}^{t} e^{\frac{-\gamma(t-t')}{D_x}} dt' ds \\
&= \int_{s\in S} q_s \frac{D_x}{\gamma}(1-e^{-\frac{\gamma s}{D_x}})ds \\
&\geq \int_{s\in S} q_s \frac{D_x}{\gamma}(\frac{\gamma s}{D_x} - \frac{\gamma^2 s^2}{2D_x^2})ds \\
&= \int_{s\in S} q_s s ds - \frac{\gamma D_x}{2}\int_{s\in S} q_s \frac{s^2}{D_x^2} ds \\
&= D_x\left(1-\frac{\gamma}{2}\frac{D_x^2+Var_x}{D_x^2}\right) \\
&= D_x\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}\frac{-Var_x}{D_x^2}\right)
\end{aligned}
$$

The inequality is from the inequality $e^{-x}\leq 1-x+\frac{x^2}{2}$ for $x\geq 0$. Since t can be chosen from 0 to T with density p_y, the total expected number of ordered pair (x,y) should be at least

$$
\begin{aligned}
&Tp_y p_x(1-\gamma)\frac{\gamma\alpha_{xy}}{p_x D_x}\left(1-\frac{\gamma}{2}\right)D_x\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right) \\
&= Tp_y\alpha_{xy}\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right)
\end{aligned}
$$

where $C=\min_{v\in V}\frac{-Var_v}{D_v^2}$. □

7. Experiments

In this section, we focus on the ride-sharing application to conduct numerical studies to compare our algorithms with several baseline algorithms over both synthetic dataset and the New York City taxi dataset (Donovan and Work (2014)) to demonstrate the effectiveness and efficiency of our algorithms.

7.1. Data Description

There are several parameters in our model: a graph $G=(V,E)$ specifying vertices $v\in V$ and edges $e\in E$ (together with its weight denoted by w_e), and each vertex v's arrival time determined by its arrival probability p_v and its sojourn time. In this section, we will illustrate how we obtain the data for these parameters in detail. In particular, we generate the data for $G=(V,E)$ and p_v using two ways: one is from the simulation and the other is from the pre-process of the New York City taxi dataset. We generate the sojourn time only using the simulation approach since the sojourn time information is not available in the New York City taxi dataset. For notation simplicity, we use $[L]$ to denote$\{1,2,\cdots,L\}$.

Synthetic Dataset. For the synthetic dataset, we generate a graph $G=(V,E)$ with $|V|=m=100$ and a parameter density q. Without loss of generality, we set $V=\{1,2,\ldots,m\}$. For each pair

$(x,y) \in V^2$ and $x \le y$, we generate a value w'_{xy} from $U(0,1)$, where $U(a,b)$ denotes a uniform distribution that samples value from a to b uniformly. If the value $w'_{xy} \ge 1 - \frac{2q}{m+1}$, we add two non-trivial (positive-weighted) edges $e=(x,y)$ and $e=(y,x)$ with a weight $w_e = w'_{xy}$ to the edge set E. For the rest cases, we add trivial edges with $w_{xy}=0$. It is straightforward to see that q is approximately the ratio between the number of non-trivial edges and the number of vertices (m). If a graph is sparse, q should be small compared to 1. The probability p_v of each type v is randomly generated from $U(0,1)$, and then we normalize it to satisfy $\sum_{v\in V} p_v = 1$.

New York City Taxi Dataset. We obtain the data from Donovan and Work (2014). The data set records taxis' trip information. Each trip record contains the pick-up and drop-off location and time. Our case study is based on the car-pooling problem that matches riders which is important in ride-sharing application. This problem is also studied by Aouad and Saritaç (2020) and Yan et al. (2020). We treat each trip as a rider, use 200,000 records of taxi trips and pre-process the data as below. We divide the map into an $L \times L$ grid graph denoted by $G=(V,E)$. Then we label each location by mapping its coordinate to the nearest grid cell center. For each trip record, we label its pick-up and drop-off location by $pu = (pu_x, pu_y)$ and $do = (do_x, do_y)$, respectively, where $pu, do \in [L]\times[L]$ represent the origin and destination of a vertex $v \in V$ in the grid graph. In other words, each trip can be labeled by a vertex $v \in V \subset [L]\times[L]\times[L]\times[L]$ in the grid graph. For trips with the same grid label for pick-up and drop-off locations, we treat them as the same vertex type. For each pair of vertices $(u,v)\in V$, we generate the edge weight $w_{(u,v)}$ as follows:

$$w_{(u,v)} = \begin{cases} 0, & dist(pu(u),pu(v)) > \delta \text{ or } dist(do(u),do(v)) > \delta, \\ route(u,v), & otherwise. \end{cases} \tag{3}$$

Here $dist(a,b)$ is the function that calculates the Manhattan distance between a and b, i.e., $dist(a,b) = |a_x - b_x| + |a_y - b_y|$. δ is a parameter that specifies the distance threshold. $route(u,v)$ is the function that calculates the shortest route between two vertices u and v. Its formal definition is provided below.

$$\begin{aligned} route(u,v) = \min\{ & dist(pu(u),pu(v)) + dist(pu(v),do(u)) + dist(do(u),do(v)), \\ & dist(pu(u),pu(v)) + dist(pu(v),do(v)) + dist(do(v),do(u)), \\ & dist(pu(v),pu(u)) + dist(pu(u),do(v)) + dist(do(v),do(u)), \\ & dist(pu(v),pu(u)) + dist(pu(u),do(u)) + dist(do(u),do(v))\}. \end{aligned}$$

In other words, we use the distance of this shortest route as our non-zero weight. Finally, we use the relative frequency of v in the 200,000 records to estimate its probability p_v, i.e., $p_v = \frac{\text{numbers of type } v}{200{,}000}$. In our experiments, we set $\delta = 2$ and $L = 20$.

Sojourn Time Distributions. We denote $U_{int}[a,b]$ as the integer uniform distribution that samples integer value from a to b (a and b are included) uniformly. Three types of sojourn time distributions are tested:

- Geometric distribution $\text{Geo}(p^G)$: $p^G \sim U(P^G, 1)$ where $P^G \in (0,1)$ is a hyperparameter.
- Binomial distribution $\text{B}(n^B, p^B)$: $n^B \sim U_{int}[10, N^B]$ and $p^B \sim U(0,1)$ where $N^B \geq 10$ is a hyperparameter.
- Poisson distribution $\text{Poi}(\lambda^P)$: $\lambda^P \sim U(0, L^P)$ where $L^P \geq 1$ is a hyperparameter.

We assume the sojourn time of all vertices in a graph follows the same type of distribution (geometric, binomial, or Poisson), and their distributions' parameters are randomly generated from a probability distribution. For example, if we assume vertices' sojourn time follow a geometric distribution with $P^G = 0.5$, then we will generate $p^G_v \sim U(0.5, 1)$ for each vertex $v \in V$.

In summary, a problem instance I is defined by a graph parametric by q (synthetic data) or generated from real-world data, and the type of distribution (geometric, binomial, or Poisson distribution) with its corresponding hyperparameter (P^G, N^B or L^P). For all experiments, we set $T = 5000$ which is much larger than the sojourn time of any vertex under any tested distribution.

7.2. Baseline Algorithms

- Rcp: This is the randomized compatibility policy from Appendix B.3 of Aouad and Sarıtaç (2020). We adjust it to make it suitable for our model.
- Grd: Each arrival is matched to an available neighboring vertex with an incident edge whose weight is the largest.
- Bat: This is the batching algorithm described in Section 4.1 of Ashlagi et al. (2019). We set the batch size as $\lfloor \tilde{d} \rfloor + 1$ where $\tilde{d}$ is the expected sojourn time over all types.
- Sam0.5: Algorithm 1 with $\gamma = 0.5$.
- Sam: Algorithm 1 with $\gamma = 0.36$.

Here we test two different γs for Algorithm 1. Sam uses $\gamma = 0.36$ which is suggested by Corollary 1 for theoretical analysis. However, we note that Sam may be too conservative in practice. Hence, we would like to test a larger value. $\gamma = 0.5$ is selected since its associated lower bound for the competitive ratio is 0.141 according to Corollary 1, which is not bad in the theoretical bound but turns out to generate much better performance in expectation (the results will be discussed later). Note that we have tried many values for γ and obtained similar insights. These two values are chosen without loss of generality.

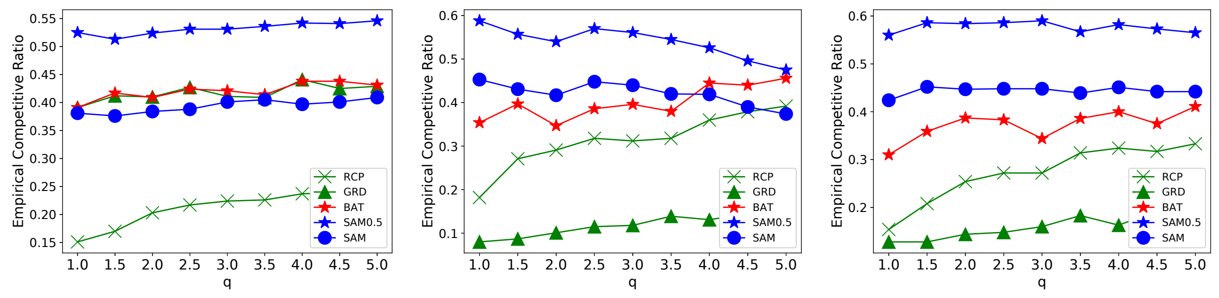
Performance Criterion. Let r denote a realization of our generated instance and R as the set of r that we test. We use *empirical competitive ratio* (ECR) as our performance criterion for an algorithm ALG: $\text{ECR} = \frac{\sum_{r \in R} \text{ALG}(r)}{\sum_{r \in R} \text{OPT}(r)}$ where $\text{ALG}(r)$ is the reward if we run ALG for r and $\text{OPT}(r)$ is the hindsight optimal for r. For each parameter setting, we test $|R| = 50$ realized sequences.

OPT	4.420	Bat	0.652
Rcp	1.131	Sam0.5	0.696
Grd	0.016	Sam	0.737

Table 1 Average runtimes of different algorithms (second)

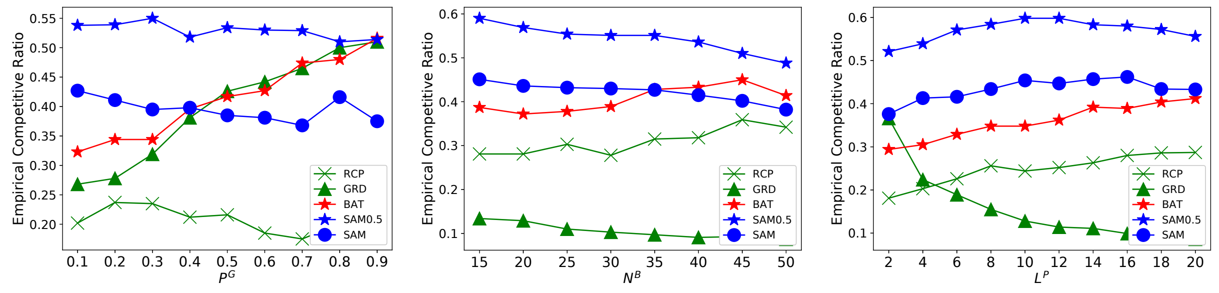
Runtime. We list the average runtimes of different algorithms in Table 1. The parameters are $q = 2.5$ and geometric distribution with $P^G = 0.5$. We use Gurobi Gurobi Optimization, LLC (2022) as our solver. We use a computer with 2.2 GHz Intel Core i7 processor, 16 GB 1600 MHz DDR3 memory and Intel Iris Pro 1536 MB Graphics to run all the experiments. In this parameter setting, the most time-consuming benchmark is OPT and the runtimes of our algorithm are comparable with other baselines except the simple Grd algorithm which shows that our algorithms are efficient. Other parameter settings obtain similar results.

7.3. Results



(a) Geo. Dist. $P^G = 0.5$ (b) Bin. Dist. $N^B = 30$ (c) Poi. Dist. $L^P = 10$

Figure 1 Performance of different algorithms w.r.t. different distributions and densities, synthetic data, $q = 1.0, 1.5, \ldots, 5.0$



(a) Geo. Dist., $P^G = 0.1, 0.2, \ldots, 0.9$ (b) Bin. Dist. $N^B = 15, 20, \ldots, 50$ (c) Poi. Dist., $L^P = 2, 4, \ldots, 20$

Figure 2 Performance of different algorithms w.r.t. different distributions, synthetic data, $q = 2.5$

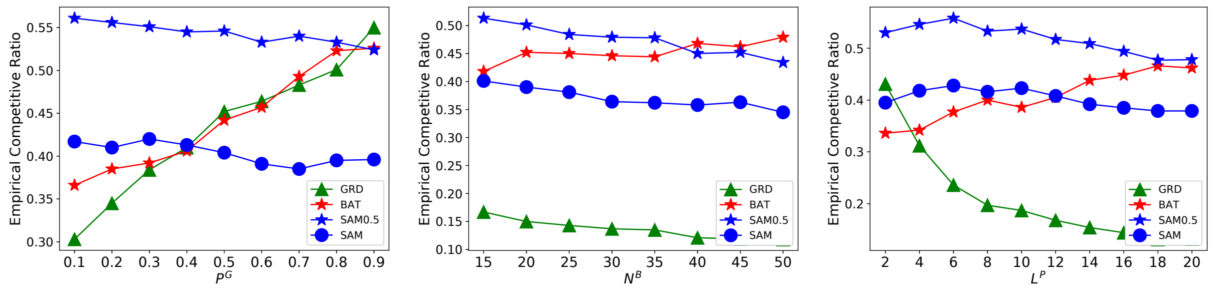
7.3.1. Results Based on Synthetic Data The results based on synthetic data are shown in Figures 1 and 2. In general, Sam0.5 outperforms other baselines by at least 10% in most parameter settings and Sam can dominate other baselines (except Sam0.5) in around $\frac{1}{3}$ test settings. As discussed earlier, Sam is too conservative to achieve a good performance in expectation, although it generates a good lower bound of the competitive ratio. Sam0.5 is good in both theoretical analysis and practice.

Sparsity. Figure 1 compares the performance under different distributions and densities with fixed hyperparameters. We can see that our algorithms' performance is stable when the density changes and Sam0.5 consistently outperforms all the other tested algorithms in all cases. In contrast, Bat, as the best baseline algorithm, does not perform as robust as ours. Its performance drops significantly when q decreases. However, in practice, the graph is often sparse. For instance, in ride-sharing, a non-trivial edge only exists between two vertices with close locations and arrival times. In other words, the advantage of our algorithms becomes more significant in applications with a sparse graph.

Diversity. Figure 2 compares the performance under different distributions and hyperparameters for synthetic data when fixing $q = 2.5$. Recall that the parameter of each vertex's distribution for sojourn time is uniformly generated from an interval defined by a hyperparameter (P^G, N^B, or L^P). The change of the hyperparameter will lead to different levels of diversity among agents (in terms of their sojourn time). For instance, for geometric distribution, when P^G decreases, the range to sample p^G for sojourn time's distribution gets larger, which leads to a higher level of diversity. In this case, Bat and Grd's performance drops significantly whereas our algorithms continue their good performance. This pattern is less significant for the other two distributions. But Sam0.5 consistently performs the best among all cases and outperforms the second-best algorithm by at least 10%.

In summary, our algorithms perform consistently well in all most cases and the advantage over the baseline algorithms is especially significant in a *sparse graph with heterogeneous agents*, which makes our algorithms practically relevant.

7.3.2. Results Based on the New York City Taxi Data We plot the results based on the New York City taxi data in Figure 3. Figure 3 tests the same parameter settings as in Figure 2 using the New York City taxi data. We do not report the performance of Rcp because its empirical competitive ratio does not exceed 5% under any parameter settings. For the geometric distribution, the performance is similar to that tested over the synthetic data. For binomial distribution, Sam0.5 can outperform other baselines when $N^B \leq 35$. For Poisson distribution, Sam0.5 still outperforms other baselines while the gap becomes smaller when L^P gets larger. From all these graphs, we can see that our algorithms (including Sam0.5 and Sam) are be very effective, especially when the expectations of sojourn time distributions are small, i.e., p^G, N^B and L^P are small.



(a) Geo. Dist., $P^G = 0.1, 0.2, \dots, 0.9$ (b) Bin. Dist. $N^B = 15, 20, \dots, 50$ (c) Poi. Dist., $L^P = 2, 4, \dots, 20$

Figure 3 Performance of different algorithms w.r.t. different distributions, the New York City taxi data

8. Conclusions and Future Work

In this paper, we study a general fully online matching model with stochastic arrivals and departures. We provide an LP benchmark for this problem and based on this LP, we design an algorithm with at least a 0.155 competitive ratio if the sojourn time is discrete distributed. If both arrivals and departures follow Poisson process, which is a special case of our general model, our algorithm can achieve the same competitive ratio (0.125) as established in Collina et al. (2020). Our algorithm applies to a large family of departure distributions with a performance guarantee. To demonstrate the challenge of the problem, we further provide several hardness results. Specifically, we show that no algorithm can achieve a competitive ratio better than $\frac{2}{3}$ and no algorithm based on our LP can achieve a ratio better than $\frac{1}{3}$. Finally, we demonstrate the effectiveness and efficiency of our algorithm by conducting extensive numerical studies over both synthetic data and the New York City taxi data.

In the future, we may improve our LP benchmark by capturing the structure of the optimal offline solution. An extension to a fully online k-way matching (a match needs k agents) is also interesting.

References

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

Aouad, Ali, Ömer Saritaç. 2020. Dynamic stochastic matching under limited time. *Proceedings of the 21st ACM Conference on Economics and Computation*. 789-790.

Ashlagi, Itai, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, Roger Wattenhofer. 2017. Min-cost bipartite perfect matching with delays. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, 81 1.

Ashlagi, Itai, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, Chris Sholley. 2019. Edge Weighted Online Windowed Matching. *Proceedings of the 2019 ACM Conference on Economics and Computation*. ACM, Phoenix AZ USA, 729-742. doi:10.1145/3328526.3329573. URL `https://dl.acm.org/doi/10.1145/3328526.3329573`.

Ashlagi, Itai, Alvin E Roth. 2021. Kidney exchange: an operations perspective. *Management Science*, 67 (9), 5455-5478.

Azar, Yossi, Ashish Chiplunkar, Haim Kaplan. 2017. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1051-1061.

Azar, Yossi, Amit Jacob Fanani. 2020. Deterministic min-cost matching with delays. *Theory of Computing Systems*, 1-21.

Chen, Xiao Alison, Zizhuo Wang. 2015. A dynamic learning algorithm for online matching problems with concave returns. *European Journal of Operational Research*, 247 (2), 379-388.

Collina, Natalie, Nicole Immorlica, Kevin Leyton-Brown, Brendan Lucier, Neil Newman. 2020. Dynamic weighted matching with heterogeneous arrival and departure rates. *International Conference on Web and Internet Economics*. Springer, 17-30.

Donovan, Brian, DB Work. 2014. New york city taxi trip data (2010-2013). *Univ. Illinois Urbana-Champaign, Champaign, IL, USA, Tech. Rep*, .

Eckl, Alexander, Anja Kirschbaum, Marilena Leichter, Kevin Schewior. 2021. A stronger impossibility for fully online matching. *Operations Research Letters*, 49 (5), 802-808.

Emek, Yuval, Shay Kutten, Roger Wattenhofer. 2016. Online matching: haste makes waste! *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 333-344.

Feldman, Jon, Aranyak Mehta, Vahab Mirrokni, Shan Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 117-126.

Golrezaei, Negin, Hamid Nazerzadeh, Paat Rusmevichientong. 2014. Real-time optimization of personalized assortments. *Management Science*, 60 (6), 1532-1551.

Goyal, Vineet, Rajan Udwani. 2022. Online Matching with Stochastic Rewards: Optimal Competitive Ratio via Path-Based Formulation. *Operations Research*, doi:10.1287/opre.2022.2345. URL `https://pubsonline.informs.org/doi/abs/10.1287/opre.2022.2345`. Publisher: INFORMS.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. URL `https://www.gurobi.com`.

Hu, Ming, Yun Zhou. 2022. Dynamic Type Matching. *Manufacturing & Service Operations Management*, 24 (1), 125-142. doi:10.1287/msom.2020.0952. URL `https://pubsonline.informs.org/doi/10.1287/msom.2020.0952`. Publisher: INFORMS.

Huang, Zhiyi, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, Xue Zhu. 2020a. Fully Online Matching. *Journal of the ACM*, 67 (3), 17:1-17:25. doi:10.1145/3390890. URL `https://doi.org/`

10.1145/3390890.

Huang, Zhiyi, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, Yuhao Zhang. 2019. Tight Competitive Ratios of Classic Matching Algorithms in the Fully Online Model. *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Proceedings, Society for Industrial and Applied Mathematics, 2875-2886. doi:10.1137/1.9781611975482.178. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.178.

Huang, Zhiyi, Xinkai Shu. 2021. Online stochastic matching, poisson arrivals, and the natural linear program. *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 682-693.

Huang, Zhiyi, Xinkai Shu, Shuyi Yan. 2022. The Power of Multiple Choices in Online Stochastic Matching. *arXiv:2203.02883 [cs]*, URL http://arxiv.org/abs/2203.02883. ArXiv: 2203.02883.

Huang, Zhiyi, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang. 2020b. Fully Online Matching II: Beating Ranking and Water-filling. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 1380-1391. doi:10.1109/FOCS46700.2020.00130. ISSN: 2575-8454.

Jaillet, Patrick, Xin Lu. 2014. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39 (3), 624-646.

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

Ma, Will, David Simchi-Levi. 2020. Algorithms for Online Matching, Assortment, and Pricing with Tight Weight-Dependent Competitive Ratios. *Operations Research*, 68 (6), 1787-1803. URL https://ideas.repec.org/a/inm/oropre/v68y2020i6p1787-1803.html. Publisher: INFORMS.

Manshadi, Vahideh H., Shayan Oveis Gharan, Amin Saberi. 2012. Online Stochastic Matching: Online Actions Based on Offline Statistics. *Mathematics of Operations Research*, 37 (4), 559-573. doi:10.1287/moor.1120.0551. URL https://pubsonline.informs.org/doi/abs/10.1287/moor.1120.0551. Publisher: INFORMS.

Mehta, Aranyak, Debmalya Panigrahi. 2012. Online matching with stochastic rewards. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 728-737.

Mehta, Aranyak, Bo Waggoner, Morteza Zadimoghaddam. 2014. Online stochastic matching with unequal probabilities. *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1388-1404.

Wang, Hao, Xiaohui Bei. 2022. Real-time driver-request assignment in ridesourcing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (4), 3840-3849. doi:10.1609/aaai.v36i4.20299. URL https://ojs.aaai.org/index.php/AAAI/article/view/20299.

Yan, Chiwei, Helin Zhu, Nikita Korolko, Dawn Woodard. 2020. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)*, 67 (8), 705-724.

Appendix

A. Classic Discrete Distributions

Let X denote a discrete random variable, $\Pr[X]$ is the probability mass function, $\mathbb{E}[X]$ is the expectation and $Var(X)$ is its variance.

A.1. Geometric Distribution

X follows a geometric distribution $\mathrm{Geo}(p)$ $(0 < p \leq 1)$.

- $\Pr[X = k] = (1-p)^{k-1} \cdot p$, $k = 1, 2, \cdots$.
- $\mathbb{E}[X] = \frac{1}{p}$.
- $Var(X) = \frac{(1-p)}{p^2}$.

A.2. Binomial Distribution

X follows a binomial distribution $\mathrm{B}(n, p)$ $(n = 0, 1, 2 \cdots,\ 0 \leq p \leq 1)$.

- $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ where $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, $k = 0, 1, \cdots, n$.
- $\mathbb{E}[X] = np$.
- $Var(X) = np(1-p)$.

A.3. Poisson Distribution

X follows a Poisson distribution $\mathrm{Poi}(\lambda)$ $(\lambda > 0)$.

- $\Pr[X = k] = \frac{\lambda^k e^{-\lambda}}{k!}$, $k = 0, 1, 2 \cdots$.
- $\mathbb{E}[X] = \lambda$.
- $Var(X) = \lambda$.