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

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 propose a linear program (LP)-based algorithm whose competitive ratio is lower bounded by 0.192 under mild conditions. 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 $\frac{1}{3}$. We next extend our model to the setting with general sojourn time under Poisson arrivals and provide the same competitive ratio guarantee for our LP-based algorithm. 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 car-pooling, also known as ride-sharing, where individuals travel together in a single car to a shared/similar

destination. In the ride-sharing application, individuals arrive in the system in an online manner. They wait to be matched to another one who shares a similar destination and can be assigned to the same car. But they are inpatient. After a random waiting time in the platform, they will leave the platform even without being matched. In this application, there are no clear sides. Each arrival arrives 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 is generated. The reward can vary for different matching pairs. The goal is to maximize the total reward of successful matches. We name this general model a fully online matching model.

The fully online matching model has applications in various domains besides ride-sharing. Examples include kidney exchange, player matching in online games and individual matching in online dating platforms.

Kidney Exchange In kidney exchange schemes, incompatible pairs of donors and recipients are seeking mutual exchanges. Pairs of donors and recipients arrive in the market randomly in an online manner. A pair needs to find another incompatible pair to finish a successfully mutual exchange. 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 a 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.

Individual Matching in Online Dating The online matching platform needs to match individuals based on their preferences, interests, and other factors. Individuals join the platform sequentially and randomly. After a random waiting time, they will leave the platform even without being matched to anyone. The platform's goal is to improve the chances of finding compatible partners and also help individuals save time and effort in the dating process.

Despite potential wide applications, the 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.

Limited research has been conducted on the topic, with Huang et al. (2020a) and Huang et al. (2020b) being notable ones. These studies assume that agents arrive and depart in an adversary manner, aiming to maximize the number of matches. In contrast, our paper assumes that arrivals follow an identical and independent (i.i.d.) probability distribution, 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 stays in the system for a sojourn time before leaving. We do not need to specify the exact distribution of the sojourn time but assume it follows a type-specific distribution with known expectations. Our approach aims to maximize the edge-weighted reward of successful matches, which is more general than the number of successful matches studied in Huang et al. (2020a) and Huang et al. (2020b). We argue that our model is more applicable to ride-sharing. This is because the distributionally free algorithm has reduced the dependency on data availability. Although the arrival distribution can be estimated through customers' arrival data, the data on sojourn time may not be as readily available. This makes it challenging to obtain the distributional information of sojourn time. However, our approach overcomes this challenge

by utilizing only the mean sojourn time. Furthermore, in the context of ride-sharing, rewards for paired agents can vary. Our model addresses this 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 of the associated Markov decision process using the properties of Poisson processes. Our developed algorithm can be easily extended to this setting. More generally, it can solve the dynamic stochastic matching problem in the presence of Poisson arrivals and a general continuous sojourn time. Specifically, our algorithm applies to any continuous distribution with a known mean sojourn time. We establish a distributionally free bound for this general setting, indicating that our algorithm has the potential to perform well 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. Specifically, the arrivals are based on a known i.i.d. distribution, and the sojourn time before agents depart is determined by a broad range of distributions, all of which have a known expectation. The goal is to maximize the total weight of successful matches, where the weight is defined on the edges. The model settings are relevant for a variety 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 0.192 if the sojourn time is defined as the number of future agents that an agent is willing to wait for (i.e., distributed discretely) and arrivals are i.i.d, see Corollary 1. We further prove that the same competitive ratio can be achieved if the sojourn time is continuously distributed and the arrivals follow a Poisson process, see Corollary 2, which demonstrates that our algorithm is applicable in a variety of applications. Moreover, in the special case where both the arrivals and departures follow Poisson processes (as

assumed in Collina et al. (2020) and Aouad and Saritaç (2020)), our analysis can achieve a superior competitive ratio of 0.192 compared to the ratios of 0.125 and 0.158 established in Collina et al. (2020) and Aouad and Saritaç (2020).

2. We 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. If we further restrict the algorithms to LP-based algorithms with respect to the LP we derive, we demonstrate that it is impossible to design such LP-based online algorithms 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 exhibit superior performance compared to the baseline algorithms from related works in a majority of parameter settings.

1.2. Related Work

There is extensive literature on online bipartite matching, where a set of offline vertices is available, and each online vertex is matched to an offline vertex either *immediately* upon its arrival or is 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) studied a stochastic arrival process and proposed an algorithm that achieves a competitive ratio of 0.702, improving upon the $1 - 1/e$ competitive ratio provided in Karp et al. (1990). Numerous other works have further generalized 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, where 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) studied the most generalized model, where each edge is associated with multiple prices and the success probability of each matching edge depends on the chosen price of this edge. They provided the algorithms with the best-possible weight-dependent competitive ratios.

Recently, there has been growing interest in fully online matching, where each vertex has its arrival and departure time, and can be matched at *any time* before departure. In other words, a delay in matching is allowed. Our paper falls within this research stream. 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 in an adversary setting and a 0.279-competitive algorithm under a random order model. Several papers focus on the setting where both arrival and departure follow a type-specific Poisson process. For instance, Collina et al. (2020) proposed a 0.125-competitive algorithm for maximizing 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 optimal, in sharp contrast to 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 do not make assumptions about the specific distribution of agents' sojourn time. Rather, our algorithm is designed to function effectively for a wide range of distributions with known expectations, and can remain 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, rather than enforcing a strict time constraint on the match, the model penalizes the delay by adding a delay cost in the total cost function. Compared to this alternative modeling perspective, our model is more relevant for practical applications in ride-sharing because we explicitly model agents' limited duration in the system.

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.

Suppose the time horizon for online process is T. For each time $t \in \{1, 2, \ldots, T\}$, 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 start our analysis from 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 extend our analysis to the continuous sojourn time, where the sojourn time is defined as the duration that an agent will stay in the system before 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 expected sojourn time, denoted by D_v instead of the specific distribution of $\mathbb{D}_v$.

After an arrival, we can match some available vertex pair(s) irrevocably. Here an available pair is defined as a pair of connected vertices (i, j) that have not departed or been matched. Specifically, a vertex i of type x and a vertex j of type y can be matched if they are connected (i.e., $e = (x, y) \in E$) and both are in the system without being matched upon the time of match. For each matched pair, we can gain a reward w_e where e denotes its corresponding edge. Our goal is to maximize the total reward over the whole time horizon.

Without loss of generality, we assume G to be a complete graph, i.e., $\forall x, y \in V$, there exists *exactly* one edge (x, y) in E. Note that for each pair of vertices $x, y \in V$ we can always add an edge $e = (x, y)$ between them with a weight of 0. This will not affect the optimal allocation in a reward maximization problem since only the ones with the largest weights can be retained. 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 $\text{ALG}(I)$ 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 $\text{OPT}(I)$ 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 $\text{OPT}(I)$ 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 $\text{ALG}(I)$ over $\text{OPT}(I)$ 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. Define a variable n_{xy} for each ordered pair (x, y) where $x, y \in V$ and consider a linear program as follows.

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

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

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

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

Here, Constraint (1a) is to upper bound the expected number of matches for each vertex type x by the expected number of arrivals. Constraint (1b) is to further upper bound the expected number of matches between two types of vertices in a pair by the expected frequency of encounters between

an online agent of type x and a subsequent agent of type y, which can be calculated by the product of Tp_x, the total number of type-x agents, and $D_x p_y$, the expected number of occurrences of a subsequent agent of type y appearing within the sojourn time of each type-x's agent. The objective is to maximize the weighted sum of successful matches. We show in Lemma 1 that this LP (1) is a relaxation of the offline optimal. The intuition behind the proof is as follows. We show the optimal solution from the OPT denoted by $\{n^*_{xy}\}$ is a feasible solution to LP (1), where n^*_{xy} represents the expected number of times that an online agent of type y is matched to an earlier-arriving agent of type x that remains unmatched according to the OPT. The details of the proof are deferred to the appendix due to the space limit.

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

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

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

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

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

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

4. Approximation Algorithms

Inspired by the algorithm provided 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 $\gamma \cdot \alpha_{xy}/(p_x D_x)$, 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 $\gamma \leq 1$. We use J to denote the *multiset* of types of all existing unmatched agents upon the arrival of an agent i of type $y \in V$. We enumerate all elements x in J in a uniformly random order and choose a specific agent j of type x to match agent i according to the aforementioned probability (see Lines 5-6 in Algorithm 1). When agent i is matched to an agent j 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 $|J|$ denotes the cardinality of the set J defined in Line 3 of Algorithm 1 and can be bounded by the largest support over all $\mathbb{D}_v$s for $v \in V$.

It is worthwhile to mention that although our algorithm is motivated from Collina et al. (2020), it has significantly improved their algorithms in adaption to our general model settings. In addition, we manage to conduct a comprehensive analysis of the algorithm to achieve a better performance guarantee than theirs. Next, we will elaborate in detail how we analyze the competitive ratio of Algorithm 1. In the subsequent analysis in this section, we assume the maximal value in the support of $\mathbb{D}_v$ is much lower than T for each $v \in V$. The assumption is mild in ride-sharing applications since the time horizon is often much larger than the possible sojourn time of every agent.

4.1. Analysis

Note that the total weight generated by OPT cannot be greater than the optimal value of LP (2) according to Lemma 1. Hence we can compare the performance of Algorithm 1 with the value of LP (2) to get a lower bound of the competitive ratio. Specifically, we can calculate 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) with $x, y \in V$ to derive a lower bound of the competitive ratio. Here, we only consider the pair (x, y) with a positive sojourn time of x, i.e., $D_x > 0$ since otherwise, the agents (of type x) are not available when an agent of type y arrives at a later time.

We assume an agent i of type $y \in V$ arrives at time t. We next calculate the probability of matching this agent to an existing agent j of type $x \in V$ who arrives at time $t' < t$. To start the analysis, we first define four events as follows.

Algorithm 1 SAM(γ)

Input: Online arrivals of agents

Output: Matching pairs of vertices

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 with probability $\gamma \cdot \alpha_{xy}/(p_x D_x)$;
7: **end for**
8: **end for**

- E_1: An agent j of type x arrives at time t'.
- E_2: Conditioning on E_1, this agent j 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 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 records the number of type z in the multiset J defined in Line 3 of Algorithm 1. By conditioning on the probability distribution over $\vec{b}$, we upper bound the probability of agent j being matched to an agent of type $z \in V$. Using the union

bound, we get Lemma 3. Here, we use C_0 and C_1 to denote $\frac{\sum_{z\in V}\alpha_{xz}p_z}{p_x}$ and $\sum_{z\in V}\alpha_{zx}$, respectively. According to Constraint (2a), we have $C_0, C_1 \geq 0$ and $C_0 + C_1 \leq 1$.

LEMMA 3. *The probability of E_2 is at least $1 - C_1\gamma$.*

Proof We use a vector $\vec{b}$ to store the information of unmatched agents upon the arrival of agent j that is of type x, where each entry b_x records the number of x in set J in Line 3. According to Algorithm 1, we can calculate the probability for an agent j (of type x) arriving at time t' to be unmatched as $\Pi_{z\in V}\left(1-\frac{\gamma\alpha_{zx}}{p_z D_z}\right)^{b_z}$ for any given $\vec{b}$. In other words, none of the existing unmatched agents will be matched to the agent j. Thus,

$$\begin{aligned}
\Pr[E_2] &= \sum_{\vec{b}} \Pr[\vec{b}]\Pi_{z\in V}\left(1-\frac{\gamma\alpha_{zx}}{p_z D_z}\right)^{b_z} \\
&\geq \sum_{\vec{b}} \Pr[\vec{b}]\left(1-\gamma\sum_{z\in V}\frac{b_z\alpha_{zx}}{p_z D_z}\right) \\
&= 1-\gamma\sum_{z\in V}\frac{\alpha_{zx}}{p_z D_z}\sum_{\vec{b}}\Pr[\vec{b}]b_z \\
&\geq 1-\gamma\sum_{z\in V}\frac{\alpha_{zx}}{p_z D_z}\cdot p_z D_z \\
&= 1-\gamma\sum_{z\in V}\alpha_{zx} = 1-C_1\gamma.
\end{aligned}$$

The first equality is according to the law of total probability and our earlier analysis. The first inequality is from the union bound. The second inequality holds because at any time the total number of unmatched agent of any type is not more than the total number of agents of that type in the system. Note that the expected total number of agents of any type z in the system at any time can be calculated as dp_z given the sojourn time is d. Hence, we can calculate the expected total number of agents of any type z in the system at any time as $\sum_{d\in S_z} p_d dp_z = p_z D_z$. □

Next, we bound the probability of E_3 by making use of the independence between different t'' from $t'+1$ to $t-1$.

LEMMA 4. *If $D_x \geq 1$, the probability of E_3 is at least $\left(1-{}^{C_0\gamma}/_{D_x}\right)^{t-t'-1}$.*

Proof Note that at each time, an arriving agent is of type z with a probability of p_z according to the i.i.d. assumption, and the agent j (of type x) can be matched to the arriving agent of type z with a probability no larger than $\frac{\gamma\alpha_{xz}}{p_x D_x}$ according to Algorithm 1. Hence, the probability for the agent j to be unmatched at each time is at least $1-\sum_{z\in V} p_z \frac{\gamma\alpha_{xz}}{p_x D_x}$. Thus, we have

$$\begin{aligned}\Pr[E_3] &\geq \left(1-\sum_{z\in V} p_z \frac{\gamma\alpha_{xz}}{p_x D_x}\right)^{t-t'-1} \\ &= \left(1-\frac{\gamma}{p_x D_x}\sum_{z\in V} p_z\alpha_{xz}\right)^{t-t'-1} \\ &= \left(1-\frac{C_0\gamma}{D_x}\right)^{t-t'-1}.\end{aligned}$$

The last equality is according to the definition of C_0. □

The remaining is to bound the probability of the agent i of type y to be matched to the agent j of type x given the fact that agent j is unmatched before time t, i.e., the probability of E_4. Intuitively, it requires two conditions to match i to j successfully. The first is that j can be matched to i,and the second is that all unmatched type $z\in J$ that joins J earlier than the agent j (of type x) cannot match i successfully. We will bound these two terms respectively in the proof, where the second needs to utilize the uniformly random order of elements in J and apply similar arguments as in the proof of Lemma 3. The detailed proof can be referred to the appendix.

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

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

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

This lemma can be proved by a simple induction and the details can be found in the appendix.

Now we are ready to formally present our main result.

THEOREM 1. *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 $\gamma\left(1-\frac{\gamma}{2}\right)\min\left\{1-\gamma, 1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right\}$, 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 Let S denote the support of the random sojourn time $\mathbb{D}_x$ and q_s denote the probability mass for each $s \in S$. For an agent i of type y at time t, the probability of matching this agent to an existing agent j of type $x \in V$ who arrives at $t' < t$ can be calculated as $\Pr[E_1]\Pr[E_2]\Pr[E_3]\Pr[E_4]$ according to our earlier analysis. Note that such a match can happen if $t - t'$ denoted by Δt is no more than agent j's sojourn time, say s. Hence, the expected number of matches can be calculated as $\sum_{\Delta t=1}^{M} \sum_{s\in S, s\geq \Delta t} q_s \Pr[E_1]\Pr[E_2]\Pr[E_3]\Pr[E_4]$, where M denotes the maximal value in S. It can be reorganized to $\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \Pr[E_1]\Pr[E_2]\Pr[E_3]\Pr[E_4]$ under the assumption that $T \gg M$, which allows us to directly start t' from $t-s$ instead of $\max\{1, t-s\}$.

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 at least $\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \left(1 - {C_0\gamma}/{D_x}\right)^{t-t'-1}$ according to Lemma 4. Note

$$
\begin{aligned}
&\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} \left(1 - {C_0\gamma}/{D_x}\right)^{t-t'-1} \\
&= \sum_{s\in S} q_s \frac{1 - \left(1 - {C_0\gamma}/{D_x}\right)^{s}}{{C_0\gamma}/{D_x}} \\
&\geq \sum_{s\in S} q_s \frac{s \cdot {C_0\gamma}/{D_x} - \frac{1}{2}s(s-1)\cdot \left({C_0\gamma}/{D_x}\right)^2}{{C_0\gamma}/{D_x}} \\
&= D_x - \frac{C_0\gamma}{2D_x} \sum_{s\in S} q_s (s^2 - s) \\
&= D_x + \frac{C_0\gamma}{2} - \frac{C_0\gamma}{2D_x}\left(Var_x + D_x^2\right) \\
&= D_x \left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}\frac{D_x - Var_x}{D_x^2}\right).
\end{aligned}
$$

Here, the inequality is from Lemma 6. Since t can vary from 1 to T and agent i of type y will arrive with a probability of p_y, the total expected number of matches for ordered pair (x, y) should be at least $Tp_yp_x(1 - C_1\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1 - \frac{\gamma}{2}\right) D_x \left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}C\right)$ with $C = \min_{v\in V} \frac{D_v - Var_v}{D_v^2}$ according to Lemmas 2, 3, 5. Note

$$
\begin{aligned}
&Tp_yp_x(1 - C_1\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1 - \frac{\gamma}{2}\right) D_x \left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}C\right) \\
&= Tp_y\alpha_{xy}\gamma\left(1 - \frac{\gamma}{2}\right)(1 - C_1\gamma)\left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}C\right) \\
&\geq Tp_y\alpha_{xy}\gamma\left(1 - \frac{\gamma}{2}\right)\min\left\{1 - \gamma, 1 - \frac{\gamma}{2} + \frac{\gamma}{2}C\right\}.
\end{aligned}
$$

We can easily verify that the last inequality holds when $C \geq 1$ since the term $1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}C \geq 1$ and $0 \leq C_0, C_1 \leq 1$. We next consider the case when $C < 1$. From $C_0 + C_1 \leq 1$, we can rewrite the term $(1 - C_1\gamma)\left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}C\right)$ in the form $(1 - Dx)(1 - D'(1 - x))$, where $x \in [0, 1]$ represents C_1 and $D, D' > 0$ represent the corresponding coefficients of the terms C_1 and C_0. By calculus, the minimum value can be only obtained in the cases that $x = 0$ or $x = 1$, which finishes the proof. □

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. We next provide a specific bound under some assumptions.

COROLLARY 1. *Under the assumptions that $D_v + D_v^2 \geq Var_v$ and $D_v \geq 1$ for all $v \in V$, the competitive ratio of Algorithm 1 is at least $\gamma(1-\gamma)\left(1 - \frac{\gamma}{2}\right)$. By setting $\gamma^* = 1 - \frac{1}{\sqrt{3}} \approx 0.42$, the competitive ratio is at least* 0.192.

We claim the assumptions in Corollary 1 are mild since they hold for many classic discrete distributions, such as binomial distribution, Poisson distribution, geometric distribution, and hypergeometric distribution (see technical appendix for their definitions).

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. From the representation of A_t, it is equal to $1 + 1 + \varepsilon g(t-2) + \varepsilon(1-\varepsilon) g(t-3) + (1-\varepsilon)^2 f(t-2)$. 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 $f(t-1) \geq B_{t-1} = p_1 \cdot (w_{(1,2)} + g(t-3)) + p_2 \cdot f(t-2)$. 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 $p_2p_1Tw_{(1,2)} + o(T)$. 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 $(len-1)/2 \geq (len-2)/2$. 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 $p_2p_1Tw_{(1,2)} + \frac{p_2-2p_1}{2}T + o(T)$.

Then, in the strategy which only matches agents between type 2 and type 1, the expected value is exactly $p_2p_1Tw_{(1,2)} + o(T)$. So the expected value output by the online optimal algorithm is $p_2p_1Tw_{(1,2)} + o(T)$.

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. □

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

Up until now, we have made the assumption that the sojourn time is discrete. However, it is worth noting that the developed algorithm and its analysis can be easily extended to the scenario where the sojourn time is continuous. In this section, we will elaborate this extension. In particular, we define the sojourn time as the duration that an agent will stay in the system prior to departing. We

assume that the sojourn time of agents in type v is drawn from a general (continuous or discrete) distribution $\mathbb{D}_v$ with a non-negative support. Continuing with our previous assumption that only the expectation D_v is known, we again do not require 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, all events (arrivals and departures) occur in the time interval $[0, T)$.

We follow a similar analysis procedure as in sections 3 and 4. Specifically, we use LP (1) to upper bound the optimal expected total reward and its reformulation LP (2) to design the algorithm and analyze its performance. In particular, 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.

For an agent i of type $y \in V$ that arrives at time t, we again 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. We will define those events in detail later when analyzing their probabilities. It is notable that we only need to consider the type x with $D_x > 0$, since otherwise it is not available at a later time when agent i arrives.

The first event E_1 is defined as follows: the agent j of type x who arrives at time $t' < t$ is not matched to any agent that arrives earlier. Here, we also use C_0 and C_1 to denote $\frac{\sum_{z \in V} \alpha_{xz} p_z}{p_x}$ and $\sum_{z \in V} \alpha_{zx}$, respectively. We have $C_0, C_1 \geq 0$ and $C_0 + C_1 \leq 1$ from Constraint (2a).

LEMMA 7. *The probability of E_1 is at least $1 - C_1\gamma$.*

The proof of this lemma follows the same logic as in Lemma 3, by introducing a vector $\vec{b}$ to store the information of unmatched agents at time t' and calculating the probability of event E_1 by conditioning on the probability distribution over $\vec{b}$. Interested readers are referred to the appendix for the proof details.

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 $e^{\frac{-C_0\gamma(t-t')}{D_x}}$.*

Proof We can consider a different process in which agents of type z who arrive between time t' and t can only be matched with agent j with a probability of $\frac{\gamma\alpha_{xz}}{p_x D_x}$, which is determined according to Line 6 of Algorithm 1. Since some agents arriving between time t' and t can match to other agents in our algorithm, the probability of event E_2 occurring is weakly greater than the probability for agent j to remain unmatched under this process.

Under this process, as the arrival of type-z agents follows a Poisson process with parameter p_z, we can treat the matching event between agents of type z and agent j as a compound Poisson process, where each random variable follows a binomial distribution determined by the matching probability. Note that the probability of matching an arriving agent of type z with agent j is $\frac{\gamma\alpha_{xz}}{p_x D_x}$, in the definition of this process. Thus, the matching event between agents of type z and agent j can be considered a Poisson process with a parameter of $\frac{\gamma\alpha_{xz}p_z}{p_x D_x}$.

From the independence of different possible types of z, we can calculate the probability of the event that there is no match with agent j. That is,

$$\begin{aligned}\Pr[E_2] &\geq e^{\sum_{z\in V}\frac{-\gamma\alpha_{xz}p_z(t-t')}{p_x D_x}} \\ &= e^{\frac{-\gamma(t-t')}{p_x D_x}\sum_{z\in V}\alpha_{xz}p_z} \\ &= e^{\frac{-C_0\gamma(t-t')}{D_x}}.\end{aligned}$$

□

The next step is to determine the probability of the agent i of type y being matched with the agent j of type x given that agent j remains unmatched up to time t. We use E_3 to denote this event. We can utilize the same proof as in the proof of Lemma 5 to induce the similar result in Lemma 9. The proof details are deferred to the appendix.

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 the

case where each $\mathbb{D}_v$ is a continuous distribution. For the discrete case, we can adopt the similar arguments as in Theorem 1 to prove it.

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 $\gamma\left(1-\frac{\gamma}{2}\right)\min\left\{1-\gamma, 1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right\}$, 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 Denote the support of $\mathbb{D}_x$ as S and the probability density for each $s \in S$ as q_s. For an agent i of type $y \in V$ at time t, the probability of matching this agent to an existing agent j of type $x \in V$ who arrives at time $t' < t$ can be calculated as $\Pr[E_1]\Pr[E_2]\Pr[E_3]$ according to our earlier analysis. Notice that such a match can be made if the difference $t - t'$ denoted by Δt is no more than agent j's sojourn time, say s. Thus, if we denote M as the supremum of the set S, the expected number of matches can be calculated as $\int_{\Delta t=0}^{M}\int_{s\in S\cap\{s:s>=\Delta t\}} q_s p_x \Pr[E_1]\Pr[E_2]\Pr[E_3] ds d(\Delta t)$. It can be reorganized to $\int_{s\in S} q_s \int_{t-s}^{t} p_x \Pr[E_1]\Pr[E_2]\Pr[E_3] dt' ds$ under the assumption that $T \gg M$, which allows us to directly start t' from $t-s$ instead of $\max\{0, t-s\}$.

We observe that the lower bound of $\Pr[E_1]$ and $\Pr[E_3]$ don't contain the term t'. Hence, we can move these terms outside and only focus on the value of $\int_{s\in S} q_s \int_{t-s}^{t} \Pr[E_2] dt' ds$. According to Lemma 8, it is at least $\int_{s\in S} q_s \int_{t-s}^{t} e^{\frac{-C_0\gamma(t-t')}{D_x}} dt' ds$. Note

$$
\begin{aligned}
&\int_{s\in S} q_s \int_{t-s}^{t} e^{\frac{-C_0\gamma(t-t')}{D_x}} dt' ds \\
&= \int_{s\in S} q_s \frac{D_x}{C_0\gamma}(1 - e^{-\frac{C_0\gamma s}{D_x}}) ds \\
&\geq \int_{s\in S} q_s \frac{D_x}{C_0\gamma}\left(\frac{C_0\gamma s}{D_x} - \frac{C_0^2\gamma^2 s^2}{2D_x^2}\right) ds \\
&= \int_{s\in S} q_s s ds - \frac{C_0\gamma D_x}{2}\int_{s\in S} q_s \frac{s^2}{D_x^2} ds \\
&= D_x\left(1 - \frac{C_0\gamma}{2}\frac{D_x^2 + Var_x}{D_x^2}\right) \\
&= D_x\left(1 - \frac{C_0\gamma}{2} + \frac{C_0\gamma}{2}\frac{-Var_x}{D_x^2}\right)
\end{aligned}
$$

The inequality is from the inequality $e^{-x} \leq 1 - x + \frac{x^2}{2}$ for $x \geq 0$. Since t can vary from 0 to T and the arrival of type-y agents follows a Poisson process with a parameter p_y,

the total expected number of matches for ordered pair (x, y) should be at least $Tp_yp_x(1 - C_1\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\right)D_x\left(1-\frac{C_0\gamma}{2}+\frac{C_0\gamma}{2}C\right)$ according to Lemmas 7 and 9, where $C = \min_{v\in V}\frac{-Var_v}{D_v^2}$. Note

$$\begin{aligned}
&Tp_yp_x(1-C_1\gamma)\frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\right)D_x\left(1-\frac{C_0\gamma}{2}+\frac{C_0\gamma}{2}C\right)\\
=&Tp_y\alpha_{xy}\gamma\left(1-\frac{\gamma}{2}\right)(1-C_1\gamma)\left(1-\frac{C_0\gamma}{2}+\frac{C_0\gamma}{2}C\right)\\
\geq&Tp_y\alpha_{xy}\gamma\left(1-\frac{\gamma}{2}\right)\min\left\{1-\gamma,1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right\}
\end{aligned}$$

The last inequality is because the minimum value of the term $(1-C_1\gamma)\left(1-\frac{C_0\gamma}{2}+\frac{C_0\gamma}{2}C\right)$ is obtained when either $C_0 = 0, C_1 = 1$ or $C_0 = 1, C_1 = 0$, from the similar arguments as in the proof of Theorem 1. □

Under slightly more restricted assumptions, we can provide a constant competitive ratio in Corollary 2.

COROLLARY 2. *Under the assumptions that $D_v^2 \geq Var_v$ and $D_v \geq 1$ for all $v \in V$, the competitive ratio of Algorithm 1 is at least $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)$. By setting $\gamma^* = 1-\frac{1}{\sqrt{3}} \approx 0.42$, the competitive ratio is at least* 0.192.

It is notable that the additional assumption $D_v^2 \geq Var_v$ holds in a special case where agents leave the system according to a Poisson process. In other words, our algorithm can guarantee a competitive ratio of at least 0.192 when agents' arrival and departure follow Poisson processes. This is better than the state-of-the-art ratios of 0.125 and 0.158 achieved by Collina et al. (2020) and Aouad and Saritaç (2020), respectively.

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 dataset 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 was 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 $p_v^G \sim U(0.5, 1)$ 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 Saritaç (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.
- SAM0.6: Algorithm 1 with $\gamma = 0.6$.
- SAM: Algorithm 1 with $\gamma = 0.42$.

Here we test two different γs for Algorithm 1. SAM uses $\gamma = 0.42$ which is suggested by Corollary 1 for theoretical analysis. However, we note that SAM may be a bit conservative in practice. Hence, we would like to test a larger value. $\gamma = 0.6$ is selected since its associated lower bound for the competitive ratio is 0.168 according to Corollary 1, which is not bad in the theoretical bound (larger than the state-of-art ratio 0.158 provided in Aouad and Saritaç (2020)) 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.

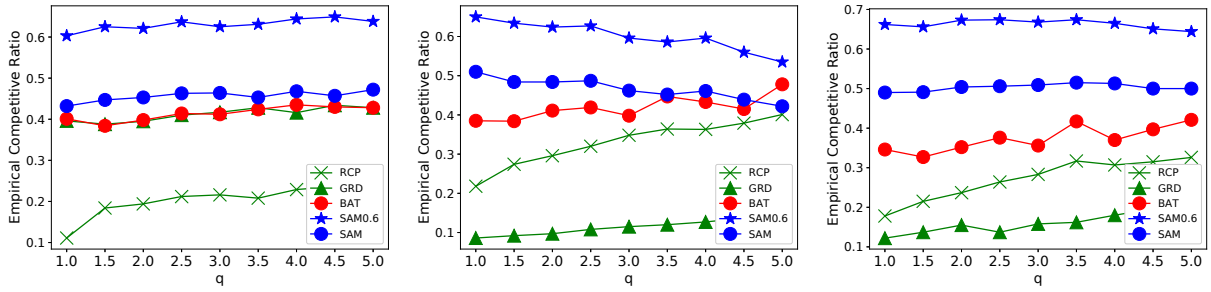
Performance Criterion. Let r denote a realization of our generated instance and R as the set of r that we test. We use *empirical competitive ratio* (ECR) as our performance criterion for an algorithm ALG: $\text{ECR} = \frac{\sum_{r \in R} \text{ALG}(r)}{\sum_{r \in R} \text{OPT}(r)}$ where $\text{ALG}(r)$ 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	SAM0.6	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 (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.

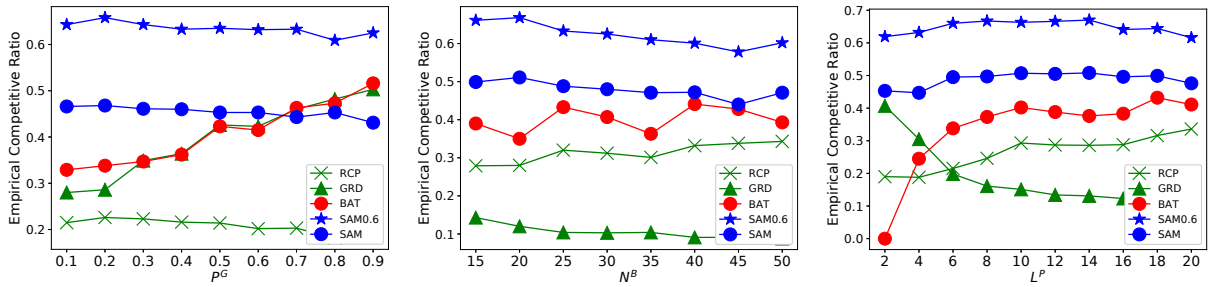
7.3. Results



(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, $q = 1.0, 1.5, \ldots, 5.0$

7.3.1. Results Based on Synthetic Data The results based on synthetic data are shown in Figures 1 and 2. In general, SAM0.6 outperforms other baselines in all cases and the gap between SAM0.6 and other baselines is at least 10% in most parameter settings. SAM can dominate other baselines (except SAM0.6) in around $\frac{5}{6}$ test settings, which demonstrates the superior and robust performance of the proposed algorithm.



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

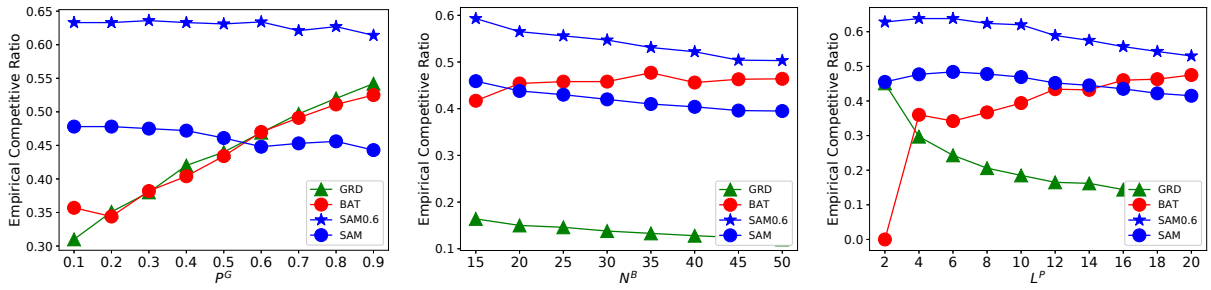
Figure 2 Performance of different algorithms w.r.t. different distributions, synthetic data, $q = 2.5$

Sparsity. Figure 1 compares the performance under different distributions and density with fixed hyperparameters. We can see that our algorithms' performance is stable when the density changes, SAM0.6 consistently outperforms all the other tested algorithms in all cases and SAM outperforms other baselines except binomial distribution with $q = 5.0$. It suggests that our algorithm performs particularly well when q is low. According to the definition of the density q in Section 7.1, the density q indicates the sparsity of a graph and the lower value the q, the more sparse the graph. Hence Figure 1 indicates that the advantage of our algorithms becomes more significant in a sparse graph. 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. Therefore, our algorithm is practically relevant.

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 SAM0.6 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 0.05 under any parameter settings. For the geometric distribution, the performance is similar to that tested over the synthetic data. For binomial distribution, SAM0.6 can outperform other baselines. For Poisson distribution, SAM0.6 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 SAM0.6 and SAM) are very effective, especially when the parameter P^G for geometric distribution is small, corresponding to a high diversity.



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

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 algorithm based on a LP and prove that the algorithm can achieve a competitive ratio of at least 0.192 under mild assumptions if the sojourn time is defined as the number of future agents that an agent is willing to wait for (i.e., distributed discretely) and the arrivals are i.i.d. We also generalize the algorithm to the scenario where the sojourn time is continuously

distributed and derive a performance guarantee for our algorithm under this setting assuming arrivals follow a Poisson process. In a special case where both arrivals and departures follow Poisson processes, our algorithm can also achieve a competitive ratio of 0.192, which improves the state-of-the-art ratios of 0.125 and 0.158 established in Collina et al. (2020) and Aouad and Saritaç (2020), respectively. To demonstrate the challenge of this 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 G, Goel G, Karande C, Mehta A (2011) Online vertex-weighted bipartite matching and single-bid budgeted allocations. *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, 1253–1264 (SIAM).

Aouad A, Saritaç Ö (2020) Dynamic stochastic matching under limited time. *Proceedings of the 21st ACM Conference on Economics and Computation*, 789–790.

Ashlagi I, Azar Y, Charikar M, Chiplunkar A, Geri O, Kaplan H, Makhijani R, Wang Y, Wattenhofer R (2017) Min-cost bipartite perfect matching with delays. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)* 81:1.

Ashlagi I, Burq M, Dutta C, Jaillet P, Saberi A, Sholley C (2019) Edge Weighted Online Windowed Matching. *Proceedings of the 2019 ACM Conference on Economics and Computation*, 729–742 (Phoenix AZ USA: ACM), ISBN 978-1-4503-6792-9, URL `http://dx.doi.org/10.1145/3328526.3329573`.

Ashlagi I, Roth AE (2021) Kidney exchange: an operations perspective. *Management Science* 67(9):5455–5478.

Azar Y, Chiplunkar A, Kaplan H (2017) Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1051–1061 (SIAM).

Azar Y, Fanani AJ (2020) Deterministic min-cost matching with delays. *Theory of Computing Systems* 1–21.

Chen XA, Wang Z (2015) A dynamic learning algorithm for online matching problems with concave returns. *European Journal of Operational Research* 247(2):379–388.

Collina N, Immorlica N, Leyton-Brown K, Lucier B, Newman N (2020) Dynamic weighted matching with heterogeneous arrival and departure rates. *International Conference on Web and Internet Economics*, 17–30 (Springer).

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

Eckl A, Kirschbaum A, Leichter M, Schewior K (2021) A stronger impossibility for fully online matching. *Operations Research Letters* 49(5):802–808.

Emek Y, Kutten S, Wattenhofer R (2016) Online matching: haste makes waste! *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 333–344.

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

Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Science* 60(6):1532–1551.

Goyal V, Udwani R (2022) Online Matching with Stochastic Rewards: Optimal Competitive Ratio via Path-Based Formulation. *Operations Research* ISSN 0030-364X, URL `http://dx.doi.org/10.1287/opre.2022.2345`, publisher: INFORMS.

Gurobi Optimization L (2022) Gurobi optimizer reference manual. `https://www.gurobi.com`, accessed: 2022-10-01.

Hu M, Zhou Y (2022) Dynamic Type Matching. *Manufacturing & Service Operations Management* 24(1):125–142, ISSN 1523-4614, URL `http://dx.doi.org/10.1287/msom.2020.0952`, publisher: INFORMS.

Huang Z, Kang N, Tang ZG, Wu X, Zhang Y, Zhu X (2020a) Fully Online Matching. *Journal of the ACM* 67(3):17:1–17:25, ISSN 0004-5411, URL `http://dx.doi.org/10.1145/3390890`.

Huang Z, Peng B, Tang ZG, Tao R, Wu X, Zhang Y (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)*, 2875–2886, Proceedings (Society for Industrial and Applied Mathematics), URL `http://dx.doi.org/10.1137/1.9781611975482.178`.

Huang Z, Shu X (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 Z, Shu X, Yan S (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 Z, Tang ZG, Wu X, Zhang Y (2020b) Fully Online Matching II: Beating Ranking and Water-filling. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 1380–1391, URL `http://dx.doi.org/10.1109/FOCS46700.2020.00130`, iSSN: 2575-8454.

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

Karp RM, Vazirani UV, Vazirani VV (1990) An optimal algorithm for on-line bipartite matching. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 352–358.

Ma W, Simchi-Levi D (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 VH, Gharan SO, Saberi A (2012) Online Stochastic Matching: Online Actions Based on Offline Statistics. *Mathematics of Operations Research* 37(4):559–573, ISSN 0364-765X, URL `http://dx.doi.org/10.1287/moor.1120.0551`, publisher: INFORMS.

Mehta A, Panigrahi D (2012) Online matching with stochastic rewards. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, 728–737 (IEEE).

Mehta A, Waggoner B, Zadimoghaddam M (2014) Online stochastic matching with unequal probabilities. *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, 1388–1404 (SIAM).

Wang H, Bei X (2022) Real-time driver-request assignment in ridesourcing. *Proceedings of the AAAI Conference on Artificial Intelligence* 36(4):3840–3849, URL `http://dx.doi.org/10.1609/aaai.v36i4.20299`.

Yan C, Zhu H, Korolko N, Woodard D (2020) Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)* 67(8):705–724.

Appendix

A. Missing Proofs

A.1. Proof of Lemma 1

Proof Let r denote a realization of instance I which is a possible input sequence of agent types with the corresponding sojourn time. Define $n_{r,x,y}$ as the number of matches between one agent of type y and an earlier-arriving agent of type x calculated by OPT under realization r and P_r as the probability of the realization r. Hence we have $n^*_{xy} = \sum_r P_r n_{r,x,y}$. Note that $\mathrm{OPT}(I) = \sum_r P_r \sum_{x,y\in V} n_{r,x,y} w_{xy} = \sum_{x,y\in V} w_{xy} \left(\sum_r P_r n_{r,x,y}\right)$, which is equal to $\sum_{x,y\in V} w_{xy} n^*_{xy}$. Hence it suffices to show that $\{n^*_{xy}\}$ is a feasible solution to LP (1).

The first is to check the feasibility of Constraints (1a). $\forall x \in V$, $\sum_{y\in V} n^*_{xy} + \sum_{y\in V} n^*_{yx} = \sum_r P_r(\sum_{y\in V} n_{r,x,y} + n_{r,y,x}) \le \sum_r P_r N^r_x = p_x T$, where N^r_x denotes the number of type x in realization r. The inequality holds because the number of matched agents of type x cannot exceed the number of arriving 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 also satisfied. $\forall x, y \in V$, we have $n^*_{xy} = \sum_r P_r n_{r,x,y} \le \sum_r P_r N^r_E$, where N^r_E denotes the frequency of event E in realization r and an event E is defined as one agent of type x can see a subsequent agent of type y. The inequality holds since the number of successful matches can not exceed the number of available pairs in any realizations. By linearity of expectation, we can transform it into the total expected number of occurrences for an agent of type x that arrives at time t to encounter an agent of type y arriving after t, summed over all t from 1 to T. We can bound this term above by multiplying Tp_x (the total number agents of type x) with the expected number of occurrences of a subsequent agent of type y appearing within the sojourn time of an agent of type x, which can be calculated as $\sum_{s\in S_x} q_s \cdot p_y s$. Here S_x denotes the support set of agent x's sojourn time $\mathbb{D}_x$ and q_s denotes the probability mass. In summary, n^*_{xy} can be upper bounded by $Tp_x \sum_{s\in S_x} q_s \cdot p_y s$, which is equal to $p_x T p_y D_x$ from the linearity of expectation. □

A.2. Proof of Lemma 5

Proof We define $\vec{b^x}$ as a vector recording the count of each type in J (defined in Line 3 of Algorithm 1) except for the specific type x of agent j. Note that to match i to j successfully, in addition to matching i to j upon the arrival of agent i, we also need to ensure that all the other agent types $z \in J$ considered before the agent j's type x in Line 4 of Algorithm 1 cannot be matched to i. Note that the probability for an agent type $z \in J$ to be in front of x is $\frac{1}{2}$ since J is in a uniformly random order. Hence the probability for an agent type z that is before x in J being

matched to the agent i can be upper bounded by $\frac{\gamma\alpha_{zy}}{2p_zD_z}$, where $\frac{\gamma\alpha_{zy}}{p_zD_z}$ is the matching probability specified in Line 6 of Algorithm 1. Then we can bound the probability of the event that there exists one agent type z that is before the agent j's type x in J matching i successfully above by union bound. Specifically we have:

$$\begin{aligned}
\Pr[E_4] &\geq \frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\sum_{\vec{b}^x}\Pr[\vec{b}^x]\sum_{z\in V}\frac{\gamma\alpha_{zy}b_z^x}{2p_zD_z}\right)\\
&= \frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\sum_{z\in V}\frac{\alpha_{zy}}{p_zD_z}\sum_{\vec{b}^x}\Pr[\vec{b}^x]b_z^x\right)\\
&\geq \frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\sum_{z\in V}\frac{\alpha_{zy}}{p_zD_z}p_zD_z\right)\\
&= \frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\sum_{z\in V}\alpha_{zy}\right)\\
&\geq \frac{\gamma\alpha_{xy}}{p_xD_x}\left(1-\frac{\gamma}{2}\right).
\end{aligned}$$

The second inequality holds because the total number of unmatched agents for each type z is not more than the total number of existing agents of type z at time t. Note that the expected total number of agents of any type z in the system at any time can be calculated as dp_z given the sojourn time is d. Hence, the expected total number of agents of any type z in the system at any time as $\sum_{d\in S_z} p_d dp_z = p_zD_z$. The last inequality is from Constraints (2a) after ignoring the first term in the left-hand side. □

A.3. Proof of Lemma 6

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.

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

□

A.4. Proof of Lemma 7

Proof We adopt a similar technique used in the proof of Lemma 3. 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 of Algorithm 1. Thus,

$$\begin{aligned}
\Pr[E_1] &= \sum_{\vec{b}} \Pr[\vec{b}] \Pi_{z\in V} \left(1 - \frac{\gamma\alpha_{zx}}{p_z D_z}\right)^{b_z} \\
&\geq \sum_{\vec{b}} \Pr[\vec{b}] \left(1 - \gamma \sum_{z\in V} \frac{b_z \alpha_{zx}}{p_z D_z}\right) \\
&= 1 - \gamma \sum_{z\in V} \frac{\alpha_{zx}}{p_z D_z} \sum_{\vec{b}} \Pr[\vec{b}] b_z \\
&\geq 1 - \gamma \sum_{z\in V} \frac{\alpha_{zx}}{p_z D_z} \cdot p_z D_z \\
&= 1 - \gamma \sum_{z\in V} \alpha_{zx} = 1 - C_1\gamma
\end{aligned}$$

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, whose expectation is exactly equal to $p_z D_z$ from the properties of Poisson arrival process.

□

A.5. Proof of Lemma 9

Proof We follow the same proof of Lemma 5 to prove this argument. We define $\vec{b}^x$ similarly as the counting vectors of unmatched agents, excluding the corresponding x of agent j.

$$\begin{aligned}
\Pr[E_3] &\geq \frac{\gamma\alpha_{xy}}{p_x D_x} \left(1 - \sum_{\vec{b}^x} \Pr[\vec{b}^x] \sum_{z\in V} \frac{\gamma\alpha_{zy} b_z^x}{2 p_z D_z}\right) \\
&= \frac{\gamma\alpha_{xy}}{p_x D_x} \left(1 - \frac{\gamma}{2} \sum_{z\in V} \frac{\alpha_{zy}}{p_z D_z} \sum_{\vec{b}^x} \Pr[\vec{b}^x] b_z^x\right) \\
&\geq \frac{\gamma\alpha_{xy}}{p_x D_x} \left(1 - \frac{\gamma}{2} \sum_{z\in V} \frac{\alpha_{zy}}{p_z D_z} p_z D_z\right) \\
&= \frac{\gamma\alpha_{xy}}{p_x D_x} \left(1 - \frac{\gamma}{2} \sum_{z\in V} \alpha_{zy}\right) \\
&\geq \frac{\gamma\alpha_{xy}}{p_x D_x} \left(1 - \frac{\gamma}{2}\right)
\end{aligned}$$

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 of Algorithm 1 corresponding to the first term, all $z \in J$ before the corresponding x of agent 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 $\frac{\gamma\alpha_{zy}b_z}{p_zD_z}$, 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, whose expectation is further upper bounded by p_zD_z. The last inequality is again from Constraints (2a) in LP (2) after ignoring the first term in left hand side. □

B. 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. For all the following distributions, we can verify the additional assumption $\mathbb{E}[X] + (\mathbb{E}[X])^2 \geq Var(X)$ in Corollary 1 holds.

B.1. Geometric Distribution

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

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

B.2. Binomial Distribution

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

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

B.3. Poisson Distribution

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

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