The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

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

Abstract

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.

1 Introduction

Starting from the seminal work by Karp, Vazirani, and Vazirani (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 arrive online and depart after a sojourn time. This paper studies this general setting. 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. 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. The goal is to maximize the total reward of successful matches. We name this general problem a fully online matching problem.

Fully online matching generalizes online bipartite matching in several dimensions (e.g., from a bipartite graph to a general graph, all agents arrive online) and is hence much more complicated. There is 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 (Huang, Shu, and Yan 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. In addition, we consider maximizing the edge-weighted reward of successful matches. 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 is 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.

Fully online matching has potential applications in various domains besides ride-sharing. For example, in a chess game platform, players join the platform in an online manner and will wait for an opponent to match for only a limited 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. Another example is kidney exchange. Donors and recipients arrive in the market sequentially and stochastically. The lifetime for recipients and kidneys is limited. They must be matched within their lifetime, otherwise, they will be abandoned. The goal is to maximize the total matching quality.

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.

We design a *distributionally free* LP-based algorithm, and investigate its performance measured by a competitive ratio, see Theorem 4.6. Under mild assumptions, we prove that the competitive ratio of our algorithm is at least 0.155, see Corollary 4.7. Moreover, for some specific distributions, we can achieve better competitive ratios, see Corollary 4.8, 4.9 and 4.10.

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 5.1. 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 5.2.

We conduct extensive numerical studies to evaluate the performance of our algorithms. Our algorithms can significantly outperform the baseline algorithms from related works in most parameter settings.

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, Vazirani, and Vazirani 1990) considered maximizing the number of matches when agents arrive in an adversary setting. Many works further consider generalizing the objective to maximizing vertex-weighted (or edge-weighted) matching, and the arrival process to a stochastic process (Feldman et al. 2009; Aggarwal et al. 2011; Huang and Shu 2021; Huang, Shu, and Yan 2022). To the best of our knowledge, the best bound under a stochastic arrival model is achieved by (Huang, Shu, and Yan 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.

Recently, fully online matching has attracted increasing attention, 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. (2021) proposed a 0.125-competitive algorithm for an edge-weighted setting, where the goal is to maximize total weights defined on edges. Aouad and Saritac (2019) studied a dynamic stochastic matching with the same arrival and departure process. They model the problem as an infinite-horizon continuous-time Markov decision process and provide 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 stream of literature models the delay in matching by incorporating the delay cost in the total cost function and makes the matching decision to minimize cost (Ashlagi et al. 2016; Emek, Kutten, and Wattenhofer 2016; Azar and Fanani 2020; Azar, Chiplunkar, and Kaplan 2017; Wang and Bei 2022). It is in contrast to the modeling perspective in fully online matching literature, where a hard constraint for the match to be restricted in a time interval is imposed.

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 $t \in \{1, 2, \ldots, T\}$, 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 define the sojourn time by the number of future agents that an agent wants to wait for. Hence the sojourn time follows a discrete distribution.

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 $\text{ALG}(I)$ 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 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.

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

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

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

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

In LP (1), each variable n_{xy} denotes the expected number of times that an online agent of type y is matched to an unmatched online agent of type x. Constraints (1a) upper bound each type x's total expected number of matches by its expected number of occurrences. Constraints (1b) restrict n_{xy} by the total number of occurrences of the event that an online agent of type y is in the sojourn time of x.

We then show in Lemma 3.1 that LP (1) is a relaxation of the offline optimal. The intuition behind the proof is as follows. We use n^*_{xy} to denote the optimal solution to OPT. We then show such $\{n^*_{xy}\}$ is a feasible solution to LP (1). Due to the space limit, we move all the lemmas' proof to the technical appendix but include the proof sketch.

Lemma 3.1. *For any instance I, the optimal value of LP* (1) *is an upper bound of OPT(I).*

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.

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

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

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

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

4 Approximation Algorithm

Inspired by the algorithm used in Collina et al. (2021), 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 $v \in V$.

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**

We next 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.

Analysis

Note that the total weight generated by OPT cannot be greater than the optimal value of LP (2) from Lemma 3.1, 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.

The first event E_1 is defined as an agent j of type x arrives at time t'. From the known i.i.d. arrival setting we can easily derive Lemma 4.1.

Lemma 4.1. *The probability of E_1 is p_x.*

The second event is defined to calculate the probability of the agent j who arrives earlier (at $t' < t$) and is of type x being unmatched. We denote this event as E_2. In our proof, we use 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 4.2.

Lemma 4.2. *The probability of E_2 is at least $1-\gamma$.*

Next, we use E_3 to represent the event that no arriving agent between time $t'+1$ and $t-1$ matches agent j given the occurrence of events E_1, E_2.

Lemma 4.3. *If $D_x \geq 1$, the probability of E_3 is at least $(1-\gamma/D_x)^{t-t'-1}$.*

The sketch of this proof is to first provide an upper bound of the event that an arriving agent at time $t'' \in [t'+1, t-1]$ matches agent j. Using the independence between different t'', we prove Lemma 4.3.

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. We denote this event as E_4.

Lemma 4.4. *The probability of E_4 is at least $\frac{\gamma\alpha_{xy}}{p_x D_x}\left(1-\frac{\gamma}{2}\right)$.*

The intuition behind the proof is that 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. The first can be easily calculated, while the second needs to utilize the uniformly random order of elements in J and apply similar arguments as in the proof of Lemma 4.2.

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 4.5. $(1-x)^d \leq 1-dx+\frac{d(d-1)}{2}x^2$*, for all $x \in [0,1]$ and all non-negative integer d.*

Now we are ready to formally present our main result.

Theorem 4.6. *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 $C = \min_{v\in V}\frac{D_v - Var_v}{D_v^2}$.*

Proof. We assume the support set S of the distribution 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

$$\sum_{s\in S} q_s \sum_{t'=t-s}^{t-1} (1-\gamma/D_x)^{t-t'-1}$$

$$= \sum_{s\in S} q_s \frac{1-(1-\gamma/D_x)^s}{\gamma/D_x}$$

$$\geq \sum_{s\in S} q_s \frac{s\cdot\gamma/D_x - \frac{1}{2}s(s-1)\cdot(\gamma/D_x)^2}{\gamma/D_x}$$

$$\geq D_x - \frac{\gamma}{2D_x}\sum_{s\in S} q_s(s^2-s)$$

$$= D_x + \frac{\gamma}{2} - \frac{\gamma}{2D_x}\left(Var_x + D_x^2\right)$$

$$= D_x\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}\frac{D_x - Var_x}{D_x^2}\right)$$

Here, the inequality is from Lemma 4.5. 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 pairs (x, y) should be at least

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

$$= Tp_y\alpha_{xy}\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C\right)$$

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

Theorem 4.6 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 4.7. *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 4.7 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 the definition).

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 4.8. *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 4.6. We replace the inequality from Lemma 4.5 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

$$Tp_y p_x(1-\gamma)\frac{\gamma\alpha_{xy}}{p_x D_x}\left(1-\frac{\gamma}{2}\right)\frac{1-e^{-\gamma}}{\gamma/D_x}$$

$$= Tp_y\alpha_{xy}(1-\gamma)\left(1-\frac{\gamma}{2}\right)(1-e^{-\gamma}).$$

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

Second, we consider three common discrete distributions: Poisson distribution $\text{Poi}(\lambda^P)$, geometric distribution $\text{Geo}(p^G)$ and binomial distribution $\text{B}(n^B, p^B)$. 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 $\text{Poi}(\lambda^P)$ are both λ^P, we get the same competitive ratio as in Corollary 4.7 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 4.6 to $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)(1-(1-p^G)\gamma)$ for $\text{Geo}(p^G)$ and $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}(1-\frac{1}{n^B})\right)$ for $\text{B}(n^B, p^B)$ to get the respective competitive ratios (see Corollaries 4.9 and 4.10, respectively).

Corollary 4.9. *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 4.10. *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 4.7, 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.

Remark. Algorithm 1 can obtain a similar performance guarantee if the arrivals of agents in each type $v \in V$ is driven by an independent Poisson arrival process (see Huang and Shu (2021) for details) and the corresponding $\mathbb{D}_v$ is an arbitrary (continuous or discrete) distribution with non-negative support. Specifically, after modifying the statements of some used lemmas such as Lemma 4.3, we can show the competitive ratio of Algorithm 1 is at least $\gamma(1-\gamma)\left(1-\frac{\gamma}{2}\right)\left(1-\frac{\gamma}{2}+\frac{\gamma}{2}C'\right)$, where $C' = \min_{v\in V}\frac{-Var_v}{D_v^2}$.

5 Hardness Results

In this section, we will present hardness results to demonstrate the challenge of the problem considered in the paper. 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 5.1. *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. □

To capture the hardness of our problem based on the LP (2), we conclude the following theorem. The full proof is shown in the technical appendix.

Theorem 5.2. *No LP-based online algorithm with respect to LP* (2) *can reach a competitive ratio better than* $\frac{1}{3}$.

6 Experiments

In this section, we compare our algorithms to several baseline algorithms over synthetic datasets to demonstrate the effectiveness and efficiency of our algorithms.

Synthetic Datasets

We denote $U(a, b)$ as the uniform distribution that samples value from a to b uniformly, and $U_{int}[a, b]$ as the integer uniform distribution that samples integer value from a to b (a and b are included) uniformly.

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 $x \leq y$, we generate a value w'_{xy} from $U(0, 1)$. If the value $w'_{xy} \geq 1 - \frac{2q}{m+1}$, 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 of the 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 $\sum_{v \in V} p_v = 1$.

Three types of sojourn time distributions are tested:

- Geometric distribution Geo(p^G): $p^G \sim U(P^G, 1)$ where $P^G \in (0, 1)$ is a hyperparameter.
- Binomial distribution 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 Poi(λ^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 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 $T = 3000$ which is much larger than the sojourn time of any vertex under any tested distribution.

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)

Baseline Algorithms

- RCP: This is the randomized compatibility policy from Appendix B.3 of Aouad and Saritac (2019). 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 $\gamma = 0.36$.

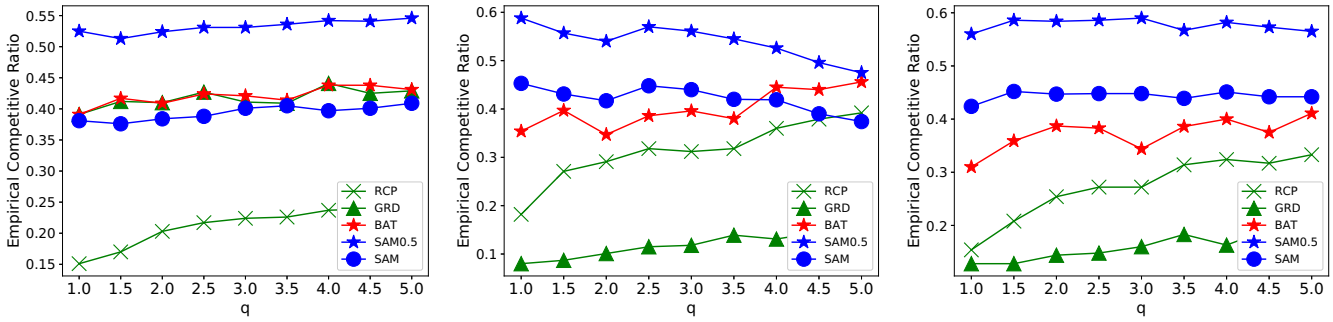
Here we test two different γs for Algorithm 1. SAM uses $\gamma = 0.36$ which is suggested by Corollary 4.7 for theoretical analysis. However, we note that SAM may be too conservative in practice. Hence, we would like to test a larger value. $\gamma = 0.5$ is selected since its associated lower bound for the competitive ratio is 0.141 according to Corollary 4.7, 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: ECR $= \frac{\sum_{r \in R} \text{ALG}(r)}{\sum_{r \in R} \text{OPT}(r)}$ where ALG(r) is the reward if we run ALG for r and OPT(r) is the hindsight optimal for r. For each parameter setting, we test $|R| = 50$ realized sequences.

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.

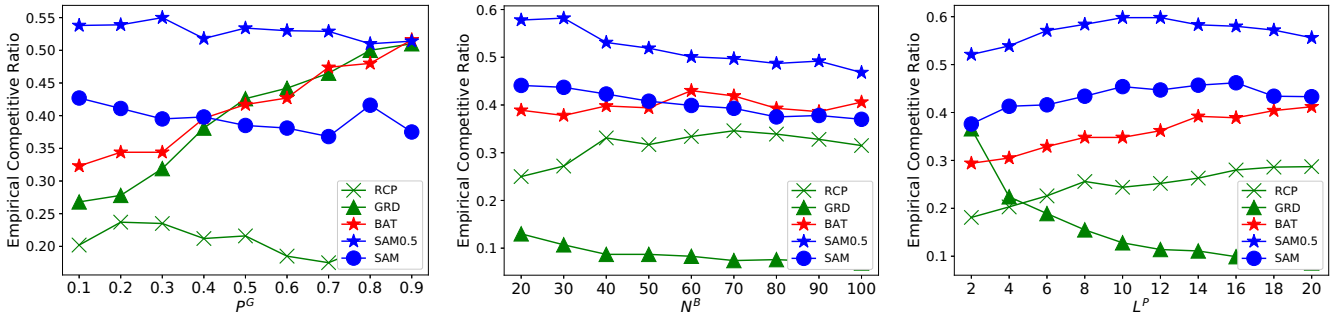
Results

Results 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.



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



(a) Geo. Dist., $P^G = 0.1, 0.2, \ldots, 0.9$ (b) Bin. Dist. $N^B = 20, 30, \ldots, 100$ (c) Poi. Dist., $L^P = 2, 4, \ldots, 20$

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

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 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 test 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 Conclusions

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. 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.

Acknowledgments

The research of Z. Yan was partly supported by Nanyang Technological University startup grant and MOE Academic Research Fund Tier 1 [Grant RG17/21] and Tier 2 [Grant MOE2019-T2-1-045] and NOL Fellowship grant [NOL21RP04].

References

Aggarwal, G.; Goel, G.; Karande, C.; and Mehta, A. 2011. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, 1253–1264. SIAM.

Aouad, A.; and Saritac, O. 2019. Dynamic Stochastic Matching Under Limited Time. SSRN Scholarly Paper ID 3497624, Social Science Research Network, Rochester, NY.

Ashlagi, I.; Azar, Y.; Charikar, M.; Chiplunkar, A.; Geri, O.; Kaplan, H.; Makhijani, R.; Wang, Y.; and Wattenhofer, R. 2016. Min-cost Bipartite Perfect Matching with Delays. 20.

Ashlagi, I.; Burq, M.; Dutta, C.; Jaillet, P.; Saberi, A.; and Sholley, C. 2019. Edge Weighted Online Windowed Matching. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 729–742. Phoenix AZ USA: ACM. ISBN 978-1-4503-6792-9.

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

Azar, Y.; and Fanani, A. J. 2020. Deterministic min-cost matching with delays. *Theory of Computing Systems*, 1–21.

Collina, N.; Immorlica, N.; Leyton-Brown, K.; Lucier, B.; and Newman, N. 2021. Dynamic Weighted Matching with Heterogeneous Arrival and Departure Rates. *arXiv:2012.00689 [cs]*. ArXiv: 2012.00689.

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

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

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

Gurobi Optimization, L. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com. Accessed: 2022-10-01.

Huang, Z.; Kang, N.; Tang, Z. G.; Wu, X.; Zhang, Y.; and Zhu, X. 2020a. Fully Online Matching. *Journal of the ACM*, 67(3): 17:1–17:25.

Huang, Z.; Peng, B.; Tang, Z. G.; Tao, R.; Wu, X.; and Zhang, Y. 2019. Tight Competitive Ratios of Classic Matching Algorithms in the Fully Online Model. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, 2875–2886. Society for Industrial and Applied Mathematics.

Huang, Z.; and Shu, X. 2021. Online Stochastic Matching, Poisson Arrivals, and the Natural Linear Program. *arXiv:2103.13024 [cs]*. ArXiv: 2103.13024.

Huang, Z.; Shu, X.; and Yan, S. 2022. The Power of Multiple Choices in Online Stochastic Matching. *arXiv:2203.02883 [cs]*. ArXiv: 2203.02883.

Huang, Z.; Tang, Z. G.; Wu, X.; and Zhang, Y. 2020b. Fully Online Matching II: Beating Ranking and Water-filling. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 1380–1391. ISSN: 2575-8454.

Jaillet, P.; and Lu, X. 2014. Online Stochastic Matching: New Algorithms with Better Bounds. *Mathematics of Operations Research*, 39(3): 624–646.

Karp, R. M.; Vazirani, U. V.; and Vazirani, V. V. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 352–358.

Wang, H.; and Bei, X. 2022. Real-Time Driver-Request Assignment in Ridesourcing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 3840–3849.