DISPATCH: An Optimally-Competitive Algorithm for Maximum Online Perfect Bipartite Matching with i.i.d. Arrivals

Minjun Chang\textsuperscript{1}[0000−0002−2251−4163], Dorit S. Hochbaum\textsuperscript{1}[0000−0002−2498−0512], Quico Spaen\textsuperscript{1}[0000−0003−2788−1904], and Mark Velednitsky\textsuperscript{1}[0000−0003−1176−5159]

University of California, Berkeley
\{minjun.lynn,dhochbaum,qspaen,marvel\}@berkeley.edu

Abstract. This work presents an optimally-competitive algorithm for the problem of maximum weighted online perfect bipartite matching with i.i.d. arrivals. In this problem, we are given a known set of workers, a distribution over job types, and non-negative utility weights for each pair of worker and job types. At each time step, a job is drawn i.i.d. from the distribution over job types. Upon arrival, the job must be irrevocably assigned to a worker and cannot be dropped. The goal is to maximize the expected sum of utilities after all jobs are assigned.

We introduce Dispatch, a 0.5-competitive, randomized algorithm. We also prove that 0.5-competitive is the best possible. Dispatch first selects a “preferred worker” and assigns the job to this worker if it is available. The preferred worker is determined based on an optimal solution to a fractional transportation problem. If the preferred worker is not available, Dispatch randomly selects a worker from the available workers. We show that Dispatch maintains a uniform distribution over the workers even when the distribution over the job types is non-uniform.

Keywords: Perfect matching, i.i.d. arrivals, competitive ratio

1 Introduction

We consider the problem of maximum online perfect bipartite matching. Suppose that we have a set of jobs and a set of workers. At every time step, a single job arrives to be served by one of the workers. Upon a job’s arrival, we observe the utility of assigning the job to each of the workers. We must immediately decide which worker will serve the job. Once a worker is assigned a job, it is busy and cannot be assigned to another job. Jobs continue to arrive until all workers are busy.

In the natural bipartite graph that arises, there is an edge between each worker and job with a non-negative utility of assigning that worker to that job. The assignment of workers to jobs will form a perfect matching in this bipartite graph. Our goal is to design a dispatching algorithm that maximizes the expected sum of utilities of the perfect matching.
In this work, we consider the maximum online perfect bipartite matching problem with independent and identically distributed (i.i.d.) arrivals. This means that, at each time step, a job is drawn i.i.d. from a known distribution over job types.

Examples of online bipartite matching include matching doctors to patients in hospitals, matching operators to callers in call centers, matching drivers to passengers in ride-sharing, and matching impressions to customers in online ad auctions [17].

We introduce the randomized algorithm Dispatch for the problem of online weighted perfect bipartite matching with i.i.d. arrivals. Dispatch is 0.5-competitive algorithm: the total expected utility of the perfect matching produced by Dispatch is at least half of the total expected utility of an optimal algorithm that knows the job arrival sequence in advance. We also describe a family of problem instances for which 0.5 is the best-possible competitive ratio. The Dispatch algorithm, thus, achieves the best-possible competitive ratio. In contrast, the same problem with adversarial job arrivals cannot be bounded, as observed by Feldman et al. [6].

To assign workers to jobs, Dispatch first selects a preferred worker. This preferred worker is determined based on an optimal solution to a fractional transportation problem. If the preferred worker is available, then job is assigned to this worker. Otherwise, Dispatch randomly selects a worker from the available workers.

1.1 Related Work

Our work resides in the space of online matching problems, including the Maximum (Imperfect) Bipartite Matching problem and the Minimum (Perfect) Bipartite Matching problem. Another closely related problem is the $k$-Server problem. For each of these problems, several arrival models are considered. Arrival models including adversarial, where the adversary chooses jobs and their arrival order; random order, where the adversary chooses jobs but not their arrival order; and i.i.d., where the adversary specifies a probability distribution over job types and each arrival is sampled independently from the distribution. We briefly describe each of these problems and present best-known results, contrasting it to the setting considered here. A summary is in Table [1]

**Maximum Online (Imperfect) Bipartite Matching** The maximum online (imperfect) bipartite matching problem is defined on a bipartite graph with $n$ known workers and $n$ jobs that arrive one at a time. Jobs either get assigned to a worker or are discarded. The goal is to maximize the cardinality (or sum of weights) of the resulting matching. In contrast to our problem, jobs may be the discarded and the resulting matching may be imperfect.

For the unweighted problem with adversarial arrivals, Karp, Vazirani, and Vazirani [10] showed a best-possible algorithm that achieves a competitive ratio of $1 - \frac{1}{e} \approx 0.632$. Variations of the problem have been proposed: addition of edge
or vertex weights, the use of budgets, different arrival models, etc. Mehta [17] provides an excellent overview of this literature. When the arrivals are in a random order, it is possible to do better than $1 - \frac{1}{e}$. Mahdian and Yan [14], in 2011, achieved a competitive ratio of 0.696. Manshadi et al. [16] showed that you cannot do better than 0.823. If the problem also has weights, then the best-possible competitive ratio is 0.368 by a reduction from the secretary problem as shown by Kesselheim et al. [11]. They also give an algorithm that attains this competitive ratio.

The problem has also been studied when the jobs are drawn i.i.d. from a known distribution. This problem is also referred to as Online Stochastic Matching. The first result to break the $1 - \frac{1}{e}$ barrier for the unweighted case was the 0.67-competitive algorithm of Feldman et al. [7] in 2009. To date, the best-known competitive ratio of 0.730 is due to Brubach et al [2]. This is close the best-known bound of 0.745 by Correa et al. [4].

**Online Minimum (Perfect) Bipartite Matching** The online minimum (perfect) bipartite matching addresses the question of finding a minimum cost perfect matching on a bipartite graph with $n$ workers and $n$ jobs. Given any arbitrary sequence of jobs arriving one by one, each job needs to be irrevocably assigned to worker on arrival. This problem is the minimization version of the problem considered in this work. However, the obtained competitive ratios do not transfer.

The problem was first considered by Khuller, Mitchell, and Vazirani [12] and independently by Kalyanasundaram and Pruhs [9]. If the weights are arbitrary, then the competitive ratio cannot be bounded. To address this, both papers considered the restriction where the edge weights are distances in some metric on the set of vertices. They give a $2n-1$ competitive algorithm, which is the best-possible for deterministic algorithms. When randomized algorithms are allowed, the best-known competitive ratio is $O(\log^2(n))$ by Bansal et al. [1]. If the arrival order is also randomized, then Raghvendra [19] shows that $2 \log(n)$ is attainable. He also shows that this is the best possible.

**k-Server Problem** In the $k$-server problem, $k$ workers are distributed at initial positions in a metric space. Jobs are elements of the same metric space and arrive one at a time. When a job arrives, it must be assigned to a worker which moves to the job’s location. The goal in the $k$-server problem is to minimize the total distance traveled by all workers to serve the sequence of jobs. After an assignment, the worker remains available for assignment to new jobs. This reassignment distinguishes the $k$-server problem from ours, where workers are fixed to a job once assigned.

The $k$-server problem was introduced by Manasse, McGeoch, and Sleater [15]. A review of the $k$-server problem literature was written by Koutsoupias [13]. For randomized algorithms in discrete metrics, the competitive ratio $O(\log^2(k) \log(n))$ was attained by Bubeck et. al. [3], where $n$ is the number of points in the discrete metric space. On the other hand, $\Omega(\log(k))$ is a known lower bound. In the
Table 1: Best-known competitive ratios and impossibility bounds for various online bipartite matching problems. ★: Results presented in this paper.

| Sense        | Matching | Arrivals | Restrictions | Best Known | Best Possible |
|--------------|----------|----------|--------------|-------------|---------------|
| Max          | Imperfect| Advers.  | 0/1          | 0.632 [10]  | 0.632 [10]    |
| Max          | Imperfect| Rand. Ord.| 0/1          | 0.696 [11]  | 0.823 [16]    |
| Max          | Imperfect| Rand. Ord.| None         | 0.368 [11]  | 0.368 [11]    |
| Max          | Imperfect| i.i.d.   | None         | 0.730 [2]   | 0.745 [4]     |
| Min          | Perfect  | Advers.  | Metric       | $O(\log^2(n))$ [1] | $\Omega(\log(n))$ [18] |
| Min          | Perfect  | Rand. Ord.| Metric       | $2 \log(n)$ [19] | $2 \log(n)$ [19] |
| Max          | Perfect  | i.i.d.   | None         | -           | 0 [6]         |
| Max          | Perfect  | i.i.d.   | None         | $\frac{1}{2}$ ★ | $\frac{1}{2}$ ★ |

i.i.d. setting, Dehghani et. al. [5] consider a different kind of competitive ratio: they give an online algorithm with a cost no worse than $O(\log(n))$ times the cost of the optimal online algorithm.

1.2 Structure of this Work

This paper is organized as follows. Section 2 formally introduces the problem of online perfect bipartite matching with i.i.d. arrivals and defines the concept of competitive ratio. Section 3 describes Dispatch, presents an example to demonstrate the algorithm, and provides the proof that Dispatch is 0.5-competitive. Section 4 introduces a family of instances of the online perfect bipartite matching problem for which no online algorithm performs better than $\frac{1}{2}$ in terms of competitive ratio. Finally, section 5 summarizes the results and suggests directions for future research.

2 Preliminaries

The set of workers is denoted by $W$ with size $n = |W|$. The set $J$ denotes the set of job types with size $k = |J|$. For every worker $w \in W$ and job type $j \in J$ there is a utility of $u_{wj} \geq 0$ for assigning a job of type $j$ to worker $w$. Let $\mathcal{D}(J)$ be a known probability distribution over the job types.

At every time step $t = 1, \ldots, n$, a single job is drawn i.i.d. from $J$ according to $\mathcal{D}$. The job must be irrevocably assigned to a worker before the next job arrives. Workers are no longer available after they have been assigned a job. Let $r_j$ denote the expected number of jobs of type $j$ that arrive. After $n$ steps, each worker is assigned to one job and the resulting assignment forms a perfect matching. Our goal is to design a procedure such that the expected sum of the utilities of the resulting perfect matching is as high as possible.

Throughout this work, we will repeatedly use two bipartite graphs; the expectation graph $G$ and the realization graph $\hat{G}$. The expectation graph $G = (W, J, E)$ is a complete bipartite graph defined over the set of workers $W$ and the set of
job types $J$. An edge $[w, j] \in E$ has associated utility $u_{wj} \geq 0$, for $w \in W$ and $j \in J$. The realization graph $\hat{G} = (W, \hat{J}, \hat{E})$ is the random bipartite graph obtained after all $n$ jobs have arrived. $\hat{J}$ denotes the set of $n$ jobs that arrived. We use $j_t \in \hat{J}$ to denote the job that arrives at time $t$ and $j_t \in J$ to denote its job type. The edge set $\hat{E}$ consists of all worker-job pairs, such that $\hat{G}$ is a complete bipartite graph defined over $W$ and $\hat{J}$. Every edge $[w, j] \in \hat{E}$ has utility $u_{wj}$, where $j$ is the job type of job $\hat{j}$. It is important to remember that the expectation graph $G$ is deterministic and known in advance whereas the realization graph $\hat{G}$ is a random graph representing a realization of the job arrival process and is revealed over time.

An instance of the online perfect bipartite matching problem with i.i.d. arrivals is defined by the set of workers $W$, the job types $J$, non-negative utilities $u_{wj}$, and a distribution over the job types $\mathcal{D}(J)$. Equivalently, the expectation graph $G$ and the distribution $\mathcal{D}(J)$ defines an instance of this problem. Here we analyze the family of potentially randomized algorithms that return a perfect matching $\hat{M}$ on $\hat{G}$. The performance of an algorithm $ALG$ for a single realization $\hat{G}$ is given by:

$$ALG(\hat{G}) = \mathbb{E} \left[ \sum_{[w,j] \in E} u_{wj} I_{wj} \right],$$

where $I_{wj}$ is a random indicator variable that equals 1 if $ALG$ assigned a job of type $j$ to worker $w$ and equals 0 otherwise. For a given problem instance defined by expectation graph $G$ and distribution $\mathcal{D}(J)$, $\mathbb{E}[ALG(\hat{G})]$ measures the algorithm’s expected performance over samples of $\hat{G}$ from $G$ according to $\mathcal{D}(J)$.

The worst-case performance across instances is measured by the competitive ratio. Let $OPT(\hat{G})$ be the maximum weight perfect matching in the realization graph $\hat{G}$ and let $\mathbb{E}[OPT(\hat{G})]$ be its expectation across different realizations for a given expectation graph $G$ and distribution $\mathcal{D}(J)$. $\mathbb{E}[OPT(\hat{G})]$ measures the performance of an optimal algorithm that has full information about the arrival sequence. This is known as an adaptive online adversary. The ratio $\frac{\mathbb{E}[ALG(\hat{G})]}{\mathbb{E}[OPT(\hat{G})]}$ measures the performance of $ALG$ relative to the optimal algorithm for a given instance of the problem. The competitive ratio is the worst-case, i.e. lowest, ratio among all possible instances of the expectation graph $G$ and distributions $\mathcal{D}(J)$:

**Definition 1 (Competitive Ratio).** An algorithm $ALG$ is said to have a competitive ratio of $\alpha$ when for all instances of the expectation graph $G$ and distribution $\mathcal{D}(J)$:

$$\alpha \leq \frac{\mathbb{E}[ALG(\hat{G})]}{\mathbb{E}[OPT(\hat{G})]}.$$
2.1 Bounding the Performance of OPT

It is difficult to compute $E\left[OPT(\hat{G})\right]$ directly. We show that the randomness in $\hat{G}$ reduces the expected value of the optimal perfect matching compared to the value of the optimal transportation problem where the number of jobs of each type is equal to its expectation. This offline transportation problem is then used to guide the online assignment.

A similar approach was used in the context of unweighted online imperfect bipartite matching by Feldman et al. [7] and Haepler et al. [8]. Here, we use a transportation problem instead of a maximum weight matching. We also bound the performance of OPT differently.

Recall that, in expectation, $r_j$ jobs of job type $j \in J$ will arrive in $\hat{G}$. An optimal fractional matching of these jobs is obtained by solving a fractional transportation problem on the expectation graph $G$, where each job type has a demand of $r_j$ and each worker has a supply of 1 and the sum of utilities is maximized.

Formally, let $f_{wj} \geq 0$ be the flow from worker $w \in W$ to job type $j \in J$. This can be interpreted as a fractional assignment of worker $w$ to jobs of job type $j$. We define the transportation problem $TPP$:

$$TPP(G) = \max_{f_{wj} \geq 0} \sum_{w \in W} \sum_{j \in J} u_{wj} f_{wj},$$

subject to:

$$\sum_{w \in W} f_{wj} = r_j \quad \forall j \in J,$$

$$\sum_{j \in J} f_{wj} = 1 \quad \forall w \in W.$$

Let $f_{wj}^*$ be an optimal flow on edge $[w, j] \in E$.

We claim that $E\left[OPT(\hat{G})\right] \leq TPP(G)$. The reason is that the weighted average of perfect matchings $OPT(\hat{G})$ forms a feasible solution to the transportation problem above.

**Lemma 1.** Given any expectation graph $G$ and distribution over job types $D(J)$,

$$E\left[OPT(\hat{G})\right] \leq TPP(G).$$

**Proof.** Assign each edge in $G$ an indicator variable $I_{wj}$, which takes on the value 1 if $OPT$ assigns worker $w$ to a job of type $j$ in $\hat{G}$ and 0 otherwise. We claim that $f_{wj} = E[I_{wj}]$ forms a feasible solution to the transportation problem in $G$.

Indeed,

$$\sum_{w \in W} E[I_{wj}] = E\left[\sum_{w \in J} I_{wj}\right] = r_j,$$

$$\sum_{j \in J} E[I_{wj}] = E\left[\sum_{j \in W} I_{wj}\right] = 1.$$
Since $E[I_{w,j}]$ is feasible for the transportation problem, it must have objective smaller than $TPP(G)$:

$$E\left[\text{OPT}(\hat{G})\right] = E\left[\sum_{(w,j) \in E} u_{w,j}I_{w,j}\right] = \sum_{(w,j) \in E} u_{ij}E[I_{w,j}] \leq TPP(G).$$

This implies that we can bound the performance of an algorithm with respect to $TPP(G)$. We apply this technique in section 3.3.

### 3 A 1/2-Competitive Algorithm

#### 3.1 The Dispatch Algorithm

Before any jobs arrive, Dispatch solves the offline transportation problem $TPP$ on the expectation graph $G$. We find an optimal flow $f^*_w$ from workers to jobs. Throughout the online stage, the algorithm reconstruct this flow between job types and workers as much as possible. For each arriving job, a preferred worker $w^P$ is randomly selected with a probability proportional to the optimal flow $f^*$ between the corresponding job type and the worker in the transportation problem. If the preferred worker is no longer available, then the job is assigned to a worker selected randomly from the set of available workers $AW$. We refer to this worker as the assigned worker $w^A$. The resulting assignment forms a perfect matching on $\hat{G}$ since each worker is assigned at most once and each job is assigned to a worker.

In the context of online bipartite matching, the idea of using an offline solution to guide the online algorithm was used in the “Suggested Matching” algorithm [7] and subsequent work, e.g. [8]. Our algorithm differs in two ways. First, the offline solution is a transportation problem instead of a maximum weight matching problem. Second, the job is randomly assigned instead of discarded when the preferred worker is no longer available. This random selection ensures that we obtain a perfect matching and is crucial for lemma 3. The analysis of the competitive performance of Dispatch is also novel except for lemma 2.

The algorithm is formally defined in Algorithm 1. We prove the following result:

**Theorem 1.** Dispatch achieves a competitive ratio of $\frac{1}{2}$ for the online perfect bipartite matching problem with i.i.d. arrivals.

#### 3.2 Example

To illustrate Dispatch, we consider the example shown in Figure 1. The example has five workers ($n = 5$) and three job types ($k = 3$). The expectation graph is shown in Figure 1a. Note that the distribution over job types, $D(J)$, is fully specified by $r_j$. An instance of the realization graph is shown in Figure 1c.
Algorithm 1 Dispatch

**Input:** Expectation graph $G$.

**Output:** Perfect matching $\hat{M}$ on $\hat{G}$.

**Initialization:**
Solve the transportation problem $TPP$ on $G$ to obtain the optimal flow $f^*$.

$\hat{M} \leftarrow \emptyset$

$AW \leftarrow W$

**Online stage:**
for $t = 1, \ldots, n$ do

# Job $j_t$ arrives with job type $j_t$.
Randomly draw preferred worker $w^P$ with probability $p(w) = \frac{f^*_w}{\sum_{w \in W} f^*_w}$ for $w \in W$.

# Use preferred worker ($w^P$) as assigned worker ($w^A$) if possible.
if $w^P \in AW$ then

$w^A \leftarrow w^P$

else

Randomly draw $w^A \in AW$ with equal probability.

end if

$\hat{M} \leftarrow \hat{M} \cup [w^A, j_t]$

$AW \leftarrow AW \setminus \{w^A\}$

end for

Figure 1b shows $f^*$, the solution to the transportation problem on $G$ that is used by Dispatch. The corresponding objective value is $TPP(G) = 8$. Figures 1d to 1h show the arrival of the jobs and the corresponding assignment made by Dispatch. Figure 1h illustrates an instance where the preferred worker selected by Dispatch is not available, and a different worker is assigned. For this particular realization $\hat{G}$, the perfect matching constructed by Dispatch has a total utility 6, while the optimal perfect matching on $\hat{G}$ has a total utility 8. Note that these values are for this particular realization of $\hat{G}$. The performance guarantee is with respect to the expectation over all realizations of $\hat{G}$.

### 3.3 Proof of $\frac{1}{2}$-competitiveness

To prove that the perfect matching produced by Dispatch has a competitive ratio of $\frac{1}{2}$, we rely on a key feature of Dispatch: It maintains the invariant, lemma 4, that workers are equally likely to be available even though the distribution over job types may not be uniform. To prove this invariant, we first show that both the preferred and the assigned worker are selected uniformly across workers. Recall that the preferred worker may be different than the assigned worker. In fact, the preferred worker does not have to be available and could have been assigned to another job already. Lemma 2 states this formally for the selection of the preferred worker. The observation underlying this lemma is that each worker is selected with a probability proportional to the total flow $f^*$ originating at the worker, which is equal to one for each worker.
DISPATCH: Online Perfect Bipartite Matching with i.i.d. Arrivals

Fig. 1: An example of the Dispatch algorithm on the realization graph shown in Figure 1c. The underlying expectation graph $G$ with $n = 5$ and $k = 3$ is shown in Figure 1a. In Figures 1b up to 1i the numbers in parenthesis denote the probability of selecting that worker as the preferred worker. Red edges represent the assignment made by the algorithm, thick black edges are previous assignments, and blue edges mark unavailable preferred workers. Figure 1h shows an instance where the preferred worker is busy.
Throughout this section we use additional notation. Let the random variable \( W^p_t \) represent the preferred worker for the job arriving at time \( t \), and let the random variable \( W^a_t \) be the assigned worker. Furthermore, let the random set \( AW_t \) consist of the available workers when the job at time \( t \) arrives. We make no further assumptions on the expectation graph \( G \) and/or distribution \( D(J) \) other than those outlined in section \( 2 \). Lemmas and theorems in this section are therefore applicable to all problem instances.

**Lemma 2.** At each time \( t \), the **preferred** worker \( W^p_t \) is drawn uniformly from all workers:

\[
P (W^p_t = w) = \frac{1}{n} \quad \text{for all } w \in W \text{ and } t = 1, \ldots, n.
\]

**Proof.** By conditioning on the job type \( j_t \) at stage \( t \) and using the law of total probability, we can rewrite the probability of selecting worker \( w \) as:

\[
P (W^p_t = w) = \sum_{j \in J} P (W^p_t = w | j_t = j) P (j_t = j).
\]

Since the jobs are drawn i.i.d., a job of type \( j \) is selected with probability \( P (j_t = j) = \frac{r_j}{n} \), by definition of \( r_j \). Given a job of type \( j \), the algorithm selects a worker \( w \) as the preferred worker with probability \( P (W^p_t = w | j_t = j) = \frac{f^*_w}{r_j} \).

Thus,

\[
P (W^p_t = w) = \sum_{j \in J} \frac{f^*_w}{r_j} \frac{r_j}{n} = \sum_{j \in J} \frac{f^*_w}{n}.
\]

Finally, recall that every worker supplies a unit of flow in the offline transportation problem, equivalent to the expected number of jobs it serves. The edges adjacent to worker \( w \) must thus transport a unit of flow, so \( \sum_j f^*_w = 1 \). Thus,

\[
P (W^p_t = w) = \frac{1}{n}.
\]

Next we show that the assigned worker is selected uniformly at random from the set of available workers. For this lemma to hold, it is crucial that the draw of the assigned worker is done uniformly at random when the preferred worker is not available. Recall that \( W^a_t \) is the assigned worker for the job arriving at time \( t \) and that \( AW_t \) are the available workers before the job arrives.

**Lemma 3.** At each time step \( t \), the **assigned** worker \( W^p_t \) is drawn uniformly from the available workers:

\[
P (W^a_t = w | w \in AW_t) = \frac{1}{n - (t - 1)}.
\]

**Proof.** Assume that \( w \) is fixed and that \( w \in AW_t \). There are two ways for \( w \) to be the assigned worker. Either \( w \) is the preferred worker or the preferred worker
is not available and \( w \) is randomly selected. We express this as:

\[
\mathbb{P}(W_t^A = w | w \in AW_t) = \mathbb{P}(W_t^P = w | w \in AW_t) \\
+ \mathbb{P}(W_t^A = w | W_t^P \notin AW_t, w \in AW_t) \times \mathbb{P}(W_t^P \notin AW_t | w \in AW_t)
\]

The selection of \( W_t^P \) is independent of whether \( w \in AW_t \). Therefore,

\[
\mathbb{P}(W_t^A = w | w \in AW_t) = \mathbb{P}(W_t^P = w) \\
+ \mathbb{P}(W_t^A = w | W_t^P \notin AW_t, w \in AW_t) \mathbb{P}(W_t^P \notin AW_t)
\]

Now we use three observations to complete the proof. First, lemma 2 implies that \( \mathbb{P}(W_t^P = w) = \frac{1}{n} \). Second, since there are \( t - 1 \) busy workers, lemma 2 implies that \( \mathbb{P}(W_t^P \notin AW_t) = \frac{(t-1)}{n} \). Third, the fact that the assigned worker is drawn uniformly at random when the preferred worker is not available implies that \( \mathbb{P}(W_t^A = w | W_t^P \notin AW_t, w \in AW_t) = \frac{1}{n-(t-1)} \). Thus,

\[
\mathbb{P}(W_t^A = w | w \in AW_t) = \frac{1}{n} + \frac{1}{n-(t-1)} \frac{(t-1)}{n} = \frac{1}{n-(t-1)}
\]

Lemma 3 specifies each available worker is equally likely to be assigned to the next job. As a consequence, we can derive the probability that a worker is still available after \( t-1 \) jobs have arrived:

**Lemma 4.** Dispatch maintains the following invariant throughout the online stage:

\[
\mathbb{P}(w \in AW_t) = \frac{n-(t-1)}{n} \quad \text{for all} \quad w \in W \quad \text{and} \quad t = 1, \ldots, n.
\]

**Proof.** At every time step, a worker is chosen randomly from the remaining available workers, as shown in lemma 3. The probability that an available worker in time step \( t \) is still available in time step \( t + 1 \) is:

\[
\mathbb{P}(w \in AW_{t+1} | w \in AW_t) = 1 - \mathbb{P}(W_t^A = w | w \in AW_t)
\]

\[
= 1 - \frac{1}{n-(t-1)} = \frac{n-t}{n-(t-1)}
\]

Thus, the probability of being available for the \( t^{th} \) job is equal to:

\[
\mathbb{P}(w \in AW_t) = \prod_{i=1}^{t} \mathbb{P}(w \in AW_t | w \in AW_{t-1}) \\
= \frac{n-(t-1)}{n-(t-2)} \frac{n-(t-2)}{n-(t-3)} \cdots \frac{n-1}{n} = \frac{n-(t-1)}{n}.
\]
From lemma 4, we know the probability that a worker is available at each
time step. We use this to bound the probability that a worker \( w \) is assigned
to a job with job type \( j \) by DISPATCH. We use the indicator random variable
\( I_{wj} = 1 \) when the DISPATCH assigns worker \( w \) to a job with job type \( j \), and
\( I_{wj} = 0 \) otherwise. We bound the probability with respect to \( f^*_wj \) in \( TPP(G) \). By
bounding the algorithm’s performance with respect to \( TPP(G) \) we can bound
the competitive ratio of DISPATCH. See section 2.1 for more details.

**Lemma 5.** Given a perfect matching \( \hat{M} \) constructed by DISPATCH, the proba-
bility that worker \( w \) is assigned to a job of type \( j \) is bounded by:

\[
\mathbb{P}(I_{wj} = 1) \geq \frac{1}{2} f^*_wj.
\]

**Proof.** If \( I_{wj} = 1 \), then worker \( w \) must have been assigned to a job of type \( j \) in
one of the time steps. Thus, \( I_{wj} = \sum_{t=1}^{n} I_{wj}^t \) where \( I_{wj}^t \) is indicator for whether
worker \( w \) is assigned to a job of type \( j \) at time step \( t \):

\[
\mathbb{P}(I_{wj} = 1) = \sum_{t=1}^{n} \mathbb{P}(I_{wj}^t = 1).
\]

Let us bound the probability \( \mathbb{P}(I_{wj}^t = 1) \) for all \( t = 1, \ldots, n \). First, we condition
on the job type arriving at time \( t \). Note that \( j_t \) must equal \( j \):

\[
\mathbb{P}(I_{wj}^t = 1) = \mathbb{P}(I_{wj}^t = 1|j_t = j) \mathbb{P}(j_t = j).
\]

Recall that there are two ways for worker \( w \) to be assigned after a job of type \( j \) arrives. Either \( w \) is the preferred worker and is assigned the job, or another
worker \( w' \) is selected as the preferred worker but is not available. \( w \) is then
selected as the assigned worker. We lower bound the probability that worker \( w \)
is assigned for the job of type \( j \) by considering only the case where \( w \) is the
preferred worker.

\[
\mathbb{P}(I_{wj}^t = 1) \geq \mathbb{P}(w \in AW_t, W_j^P = w|j_t = j) \mathbb{P}(j_t = j)
\]

\[
= \mathbb{P}(w \in AW_t) \mathbb{P}(W_j^P = w|j_t = j) \mathbb{P}(j_t = j)
\]

\[
= \frac{n - (t - 1)}{n} \frac{f^*_wj}{r_j} \frac{n}{n}
\]

\[
= \frac{1}{n} \frac{n - (t - 1)}{n} f^*_wj.
\]

For the first equality, we use that the job type at time \( t \) and the selection of
the preferred worker are independent from whether \( w \) is available at time \( t \). The
second equality follows from lemma 4, the weighted random selection of the
preferred worker, and the job arrival process.

We use \( \mathbb{P}(I_{wj}^t = 1) = \frac{1}{n} \frac{n - (t - 1)}{n} f^*_wj \) to bound the total probability of assign-
ing worker \( w \) for a job of type \( j \):

\[
\mathbb{P}(I_{wj} = 1) = \sum_{t=1}^{n} \mathbb{P}(I_{wj}^t = 1) \geq \sum_{t=1}^{n} \frac{1}{n} \frac{n - (t - 1)}{n} f^*_wj
\]

\[
= \frac{1}{n} \frac{1 + n}{2} f^*_wj \geq \frac{1}{2} f^*_wj.
\]
Lemma 5 bounds the probability that worker \( w \) is matched to a job of type \( j \). By linearity of expectation, Theorem 1 and the \( \frac{1}{2} \) competitive ratio follow almost immediately from lemma 5.

Proof (Proof of Theorem 1). The expected utility returned by the algorithm is a weighted sum of indicators whether worker \( w \) is assigned to a job of type \( j \). Note that each worker is assigned to at most one job (type). We can then apply lemma 5 to bound the probability \( P(I_{wj} = 1) \) and the expected utility of the algorithm:

\[
\mathbb{E}\left[ \text{Dispatch}(\hat{G}) \right] = \mathbb{E}\left[ \sum_{w \in W, j \in J} u_{wj}I_{wj} \right] \\
= \sum_{w \in W, j \in J} u_{wj} \mathbb{E}[I_{wj}] \\
= \sum_{w \in W, j \in J} u_{wj} P(I_{wj} = 1) \\
\geq \frac{1}{2} \sum_{w \in W, j \in J} u_{wj} f^*_w = \frac{1}{2} TPP(G).
\]

Note that the inequality requires that the utility weights are non-negative.

Finally, we apply Lemma 4 to obtain a bound on the competitive ratio attained by Dispatch for any expectation graph \( G \) and distribution \( D(J) \):

\[
\mathbb{E}\left[ \text{Dispatch}(\hat{G}) \right] \geq \frac{1}{2} TPP(G) \geq \frac{1}{2} \mathbb{E}\left[ \text{OPT}(\hat{G}) \right]
\]

4 Best-Possible Competitive Ratio

We present here a family of instances for which any online algorithm attains a competitive ratio of at most \( \frac{1}{2} \). The Dispatch algorithm guarantees a competitive ratio of \( \frac{1}{2} \) and is thus optimal with respect to competitive ratio.

Theorem 2. For the online perfect bipartite matching problem with an i.i.d. arrival process, no online algorithm can achieve a competitive ratio \( \frac{\mathbb{E}[\text{ALG}(\hat{G})]}{\mathbb{E}[\text{OPT}(\hat{G})]} \) better than \( \frac{1}{2} \).

Proof. Consider an instance \( G \) with the number of job types \( k = n + 1 \). Let the job types be indexed from 1 to \( n + 1 \) and the workers from 1 to \( n \). Job types 1 to \( n \) each arrive with probability \( p/n \) and job type \( n + 1 \) arrives with probability \( 1 - p \). For this graph, we set \( u_{wj} = 1 \) if \( w = j \) and to 0 otherwise. This implies \( u_{w,n+1} = 0 \) for all \( w \in W \).

Note that OPT gains a utility of one per unique job type in \( \{1, \ldots, n\} \) that arrives. The expected number of unique job types is computed by considering
each job type as a geometric random variable with a success probability of \( \frac{p}{n} \). Thus, \( \mathbb{E}[\text{OPT}(\hat{G})] = n \left( 1 - \left( \frac{p}{n} \right)^n \right) \).

For any online algorithm \( \text{ALG}^* \), \( t-1 \) workers are no longer available at time step \( t \) regardless of the strategy. Thus, with probability \( (1-p) + \frac{p}{n} \), the increase in utility is zero. Thus, the total expected utility increases by at most \( \frac{p}{n} \) in time step \( t \). The total expected utility obtained by \( \text{ALG}^* \) is then:

\[
\mathbb{E}[\text{ALG}^*(\hat{G})] \leq \frac{p}{n} + \frac{p}{n} - \frac{1}{n} + \frac{p}{n} - \frac{2}{n} + \cdots + \frac{1}{n} = \frac{1}{2} p(n + 1)
\]

We compute the relevant ratio and then take the limit as \( n \) goes to infinity:

\[
\lim_{n \to \infty} \frac{\mathbb{E}[\text{ALG}^*(\hat{G})]}{\mathbb{E}[\text{OPT}(\hat{G})]} = \lim_{n \to \infty} \frac{\frac{1}{2} p(n + 1)}{n \left( 1 - \left( \frac{p}{n} \right)^n \right)} = \frac{1}{2} \cdot \frac{1}{1 - e^{-p}}
\]

Since \( p \) can take on any value in the interval \( (0, 1) \), we consider the limit as \( p \) goes to zero:

\[
\lim_{p \to 0^+} \frac{1/2 \cdot p}{1 - e^{-p}} = \lim_{p \to 0^+} \frac{1/2}{e^{-p}} = \frac{1}{2}.
\]

**Corollary 1.** Dispatch achieves the best-possible competitive ratio of \( \frac{1}{2} \) for the Online Perfect Bipartite Matching problem.

## 5 Conclusion

In this paper, we examine the problem of online perfect bipartite matching with i.i.d. arrivals from a known distribution. We present the Dispatch algorithm. It attains a competitive ratio of \( \frac{1}{2} \). We show that this is the best possible. Thus, the algorithm Dispatch is optimal in terms of competitive ratio.

There is an intriguing difference between online perfect bipartite matching algorithms for minimization and the Dispatch algorithm for maximization. Whereas the competitive ratio for minimization is bounded logarithmically, a constant bound was obtained for maximization with i.i.d. arrivals. This raises the question of whether a constant competitive ratio is possible for minimization with i.i.d. arrivals.

It may be possible to translate the analysis in this work to other contexts. Our analysis relied on two key ideas; the use of the expectation graph and proving that, regardless of how the jobs arrive, the Dispatch algorithm effectively translates the non-uniform sampling over jobs to a uniform sampling over workers.

## References

1. Bansal, N., Buchbinder, N., Gupta, A., Naor, J.S.: An \( O(\log^2 k) \)-competitive algorithm for metric bipartite matching. In: European Symposium on Algorithms. pp. 522–533. Springer (2007). [https://doi.org/10.1007/978-3-540-75520-3_47](https://doi.org/10.1007/978-3-540-75520-3_47)
2. Brubach, B., Sankararaman, K.A., Srinivasan, A., Xu, P.: New algorithms, better bounds, and a novel model for online stochastic matching. In: 24th Annual European Symposium on Algorithms. vol. 57, pp. 24:1–24:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.ESA.2016.24

3. Bubeck, S., Cohen, M.B., Lee, Y.T., Lee, J.R., Madry, A.: $k$-server via multiscale entropic regularization. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. pp. 3–16. ACM (2018). https://doi.org/10.1145/3188745.3188798

4. Correa, J., Foncea, P., Hoeksma, R., Oosterwijk, T., Vredeveld, T.: Posted price mechanisms for a random stream of customers. In: Proceedings of the 2017 ACM Conference on Economics and Computation. pp. 169–186. ACM (2017). https://doi.org/10.1145/3033274.3085137

5. Dehghani, S., Ehsani, S., Hajiaghayi, M., Liaghat, V., Seddighin, S.: Stochastic $k$-server: How should uber work? In: 44th International Colloquium on Automata, Languages, and Programming, vol. 80, pp. 126:1–126:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2017)

6. Feldman, J., Korula, N., Mirrokni, V., Muthukrishnan, S., Pál, M.: Online ad assignment with free disposal. In: International Workshop on Internet and Network Economics. pp. 374–385. Springer (2009). https://doi.org/10.1007/978-3-642-10841-9_34

7. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: 50th Annual IEEE Symposium on Foundations of Computer Science. pp. 117–126. IEEE (2009). https://doi.org/10.1109/FOCS.2009.72

8. Haeupler, B., Mirrokni, V.S., Zadimoghaddam, M.: Online stochastic weighted matching: Improved approximation algorithms. In: International Workshop on Internet and Network Economics. pp. 170–181. Springer (2011). https://doi.org/10.1007/978-3-642-25510-6_15

9. Kalyanasundaram, B., Pruhs, K.: Online weighted matching. Journal of Algorithms 14(3), 478–488 (1993). https://doi.org/10.1006/jagm.1993.1026

10. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An optimal algorithm for on-line bipartite matching. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing. pp. 352–358. ACM (1990). https://doi.org/10.1145/100216.100262

11. Kesselheim, T., Radke, K., Tönnis, A., Vöcking, B.: An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In: European Symposium on Algorithms. pp. 589–600. Springer (2013). https://doi.org/10.1007/978-3-642-40450-4_50

12. Khuller, S., Mitchell, S.G., Vazirani, V.V.: On-line algorithms for weighted bipartite matching and stable marriages. Theoretical Computer Science 127(2), 255–267 (1994). https://doi.org/10.1016/0304-3975(94)90042-6

13. Koutsoupias, E.: The $k$-server problem. Computer Science Review 3(2), 105–118 (2009). https://doi.org/10.1016/j.cosrev.2009.04.002

14. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In: Proceedings of the 43rd annual ACM symposium on Theory of computing. pp. 597–606. ACM (2011). https://doi.org/10.1145/1993636.1993716

15. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. Journal of Algorithms 11(2), 208–230 (1990). https://doi.org/10.1016/0196-6774(90)90003-W

16. Manshadi, V.H., Gharam, S.O., Saberi, A.: Online stochastic matching: Online actions based on offline statistics. Mathematics of Operations Research 37(4), 559–573 (2012). https://doi.org/10.1287/moor.1120.0551
17. Mehta, A., et al.: Online matching and ad allocation. Foundations and Trends in Theoretical Computer Science 8(4), 265–368 (2013). 
https://doi.org/10.1561/0400000057

18. Meyerson, A., Nanavati, A., Poplawski, L.: Randomized online algorithms for minimum metric bipartite matching. In: Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithms. pp. 954–959. Society for Industrial and Applied Mathematics (2006)

19. Raghvendra, S.: A robust and optimal online algorithm for minimum metric bipartite matching. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. vol. 60, pp. 18:1–18:16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016). 
https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.18