Fairness in Online Bipartite Matching Problem

Wenhao Zhu, Chung-Yu Wang

April 28, 2024

Abstract

This paper explores the transformation of offline bipartite matching problems into maximum flow problems, addressing the effectiveness of algorithms like Edmonds-Karp and Push-relabel. It also examines the efficacy of online matching algorithms such as the greedy and ranking algorithms. Furthermore, we introduce the "Priority-Updated-Match" algorithm which aims to ensure fairness and optimize the matching weight across different agent groups. Our experiments demonstrate its advantages over conventional greedy algorithms, especially in settings that reflect real-world resource constraints. Our code is available online on `https://github.com/wangjohn5507/Fairness_Online_Bipartite_Matching`.

1 Introduction

The bipartite matching problem is a fundamental concept in computer science, with extensive applications in practical scenarios like resource allocation and job assignments. This problem involves pairing elements from two distinct sets in a way that matches are optimal based on certain criteria. Such matchings are not only crucial in theoretical research but also have direct implications in everyday applications.

There are two main versions of this problem: offline and online bipartite matching. The offline version allows for a complete overview of the elements in both sets before making any matching decisions. In contrast, the online version requires that decisions be made sequentially and immediately as elements from one of the sets are revealed over time, without prior knowledge of future elements. This temporal element adds complexity and has spurred much recent research into effective algorithms.

A particularly challenging aspect of the online bipartite matching problem is ensuring fairness in the matching process. Fairness here refers to creating a system where the outcomes are equitable across various groups or conditions, even when decisions must be made under uncertainty. This is especially correlated in settings like job recruitment or school admissions, where biases in algorithmic decisions could have long-term negative impacts.

The pursuit of both efficiency and fairness poses unique challenges and opportunities for innovation in algorithm design. This study aims to explore these challenges in-depth, focusing on the necessity for algorithms that not only perform efficiently under a variety of conditions but also uphold principles of fairness. By examining these issues, we seek to contribute to the development of more robust and equitable solutions in the field of online bipartite matching.

1.1 Previous works

In order to better understand the bipartite matching problem and various fairness notions, we have reviewed a few papers. These papers cover topics such as offline and online algorithms for bipartite matching, different fairness notions, algorithms for the weighted-edge version of online bipartite matching, and algorithms for the non-weighted version of the online bipartite matching problem. In the following sections, we will provide a review of these topics.

1.1.1 Offline Bipartite Matching

We intend to begin our investigation by focusing on the offline version of the bipartite matching problem to enhance our understanding. We found that the transformation of an offline bipartite matching problem into a maximum flow problem is a classic technique in combinatorial optimization, we will discuss some maximum flow algorithms in the session.

The offline version of the bipartite matching problem can be defined as the problem being set in a bipartite graph $G = (U, V, E)$, where U and V are disjoint sets of vertices and E is the set of edges connecting vertices from U to V. The goal is to find a maximum matching, which is a set of edges such that no two edges share a common vertex and the number of edges is maximized. For its transformation into a maximum flow problem, the bipartite graph with two additional vertices added: a source vertex s and a sink vertex t. Edges are drawn from the source s to every vertex in set U and from every vertex in set V to the sink t. This step is the beginning of the transformation where the bipartite graph is converted into a flow network. The transformed network is analyzed using algorithms that can find the maximum flow from source s to sink t. In the context of the bipartite graph, the flow value corresponds to the size of the matching in the original bipartite graph. Algorithms mentioned for solving the maximum flow problem include Edmonds-Karp [1], Push-relabel [2], and an algorithm by Chen et al. [3].

The **Edmonds-Karp algorithm** begins by setting the flow through all edges in the network to zero, establishing a starting point where no flow has yet been sent from the source to the sink. The core of the algorithm revolves around repeatedly searching for augmenting paths—routes from the source to the sink with available capacity. Unlike the broader Ford-Fulkerson method, which can use any search technique, Edmonds-Karp specifically employs breadth-first search to ensure that the augmenting path found is the shortest by edge count, not capacity. Upon identifying an augmenting path, the algorithm calculates the minimum residual capacity, which is the maximum additional flow that can be pushed through the path without violating capacity constraints. The flow is then augmented along this path by that minimum amount, and the network is updated to reflect the new flows and reduced capacities. This process is iterative and continues until no further augmenting paths can be found, at which point the flow value is maximized, and the algorithm terminates. The algorithm ensures that the length of the augmenting path increases monotonically, which leads to a polynomial time complexity of $O(V \times E^2)$, where V is the number of vertices and E is the number of edges in the graph.

In contrast, the **Push-relabel algorithm** operates on the concept of maintaining a preflow, where the inflow to a node can temporarily exceed the outflow, and the nodes are assigned heights to guide the flow direction. Initially, the algorithm saturates all outgoing edges from the source, sets the source's height to the total number of nodes, and initializes other nodes' heights to zero. The central mechanics of the algorithm involve two operations: push and relabel. A push operation is applicable when a node has excess preflow and an adjacent node has a lower height; in such cases, the

algorithm pushes as much flow as possible to the neighbor. If no push is possible and the node still has excess preflow, the algorithm increases the node's height through a relabel operation, making future pushes possible. These operations are applied repeatedly across the network. The algorithm terminates when equilibrium is reached, meaning no further pushes or relabels can be performed and all excess preflow has been distributed, ensuring that the preflow becomes a valid flow from source to sink. At this stage, the maximum flow through the network has been achieved. The algorithm often performs better in practice than the Edmonds-Karp algorithm, though its worst-case time complexity is $O(V^2 \times E)$.

1.1.2 On-line Bipartite Matching Made Simple [4]

When it comes to the online bipartite matching problem, two simple but effective algorithms can be applied to solve it. The first one is the greedy algorithm, which attempts to match any vertex on the given side if possible. Although this algorithm is straightforward, it achieves a competitive ratio of 2 for deterministic algorithms for the online bipartite matching problem.

Another algorithm that applies randomization to improve the competitive ratio of the greedy algorithm is the ranking algorithm, firstly described by Karp et al. [5]. In this algorithm, vertices given in advance are first assigned a random permutation. Then, when a new vertex on the other side appears, it is matched with the eligible vertex having the highest rank. This algorithm achieves the best competitive ratio among other algorithms for online bipartite matching, which is $1 - \frac{1}{e}$. The authors also proved that the ranking algorithm is the best possible online bipartite matching algorithm.

However, the original proof is a little bit complicated to understand. Moreover, Birnbaum and Mathieu [4] stated that there was a mistake in the lemma of the original paper in 1990. Furthermore, they proved that the ranking algorithm has a competitive ratio of at least $1 - (1 - \frac{1}{n+1})^n$, which is asymptotically $1 - \frac{1}{e}$. Their algorithm is a little bit different from the original algorithm, instead of matching the possible vertex with the highest rank, they match the vertex with the one with the lowest rank.

In this section, we will briefly review their paper and focus on the intuition of their proof. The ranking algorithm in their paper is similar to the normal bipartite matching problem, the vertices $v \in V$ on the right side are given a random permutation in advance, while the vertices $u \in U$ on the left side arrive in an online manner from the top side to the downside and they will match a possible vertex in V with the lowest rank that is adjacent to it. (fig. 1 illustrates the model).

From their Lemma 2, they demonstrated that by removing one vertex x from V, the new matching is either identical to the original matching or differs by at most one single alternating path (as shown in fig. 2).

Then, they applied the idea that a perfect match can be achieved by removing all the vertices that are not matched from a maximum matching. Assuming there exists a perfect matching m^* in the original graph. Let $u = m^*(v)$ for $u \in U$, they proved that if v is not matched by the ranking algorithm, u must be matched by the ranking algorithm to another vertex whose rank is lower than u, which is presented in their Lemma 3.

Subsequently, in Lemma 4, they first assume $u = m^*(v)$ for $u \in U$. Then, they proved that if we moved a vertex v from a permutation setting σ' and put v back so that the rank of v would be exactly i, thus creating a new permutation σ_i (The rank of vertices in σ_i will change at most 1). Based on Lemma 3, they argued that if v is not matched by the ranking algorithm in permutation

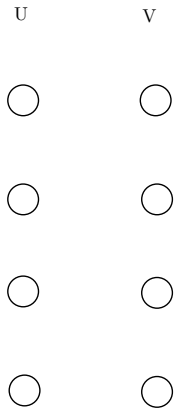

Figure 1: The model used for the ranking algorithm

σ', u must be matched to a vertex v' with a rank at most $\sigma'(v)$ in permutation σ_i, for every i.

The permutation σ' is generated from permutation σ by Lemma 4 so that they could make u independent of σ, which can be applied to prove their Lemma 5.

In Lemma 5, they first define x_t as the probability of the vertex with the rank of t over σ to be matched by the ranking algorithm. Then, based on the Lemma 4, they proved that $1 - x_t \leq (1/n) \sum_{1 \leq s \leq t} x_s$ and extended that $S_t = \sum_{s=1}^{t} (1 - 1/(n+1))^s$ for all t, where $S_t = \sum_{1 \leq s \leq t} x_s$.

By proving that, the competitive ratio of the ranking algorithm is at least $(1/n) \sum_{s=1}^{n} (1 - 1/(n+1))^s = 1 - (1 - 1/(n+1))^n$.

As n approaches infinity, the competitive ratio will approach $1 - \frac{1}{e}$.

Therefore, they have proven that the competitive ratio of the ranking algorithm is $1 - \frac{1}{e}$.

1.1.3 Approximating Maximin Share Allocations

To gain a better understanding of fairness notions, we explored several papers. However, there is a scarcity of literature applying maximin share fairness to online bipartite matching problems. Thus, we aim to delve further into the maximin share fairness notion. In this section, we aim to review the paper titled "Approximating maximin share allocations", which seeks to find an approximate 2/3 maximin share allocation. Although the allocation problem differs from the bipartite matching problem, there are some similarities between the two, such as items need to be assigned to different agents. In their paper, the authors aim to find a fair allocation of M indivisible items among N agents, using maximin share fairness as their measure of fairness.

They begin by explaining different kinds of fairness notions, such as envy-free fairness, proportional fairness, and maximin share fairness. Similar to the statement from Hosseini et al. [6], Garg et al. [7] assert the idea that achieving envy-free fairness for indivisible items is impossible. We intend to delve deeper into this impossibility in the following section. Thus, we conclude that without any further relaxations, envy-free fairness is unattainable for indivisible matching or allocation. Meanwhile, proportional fairness implies that every agent will receive a bundle of items with a value of at least $1/N$, but it is not guaranteed all the time, especially for indivisible items. Hence, they opt for the idea of maximin share fairness.

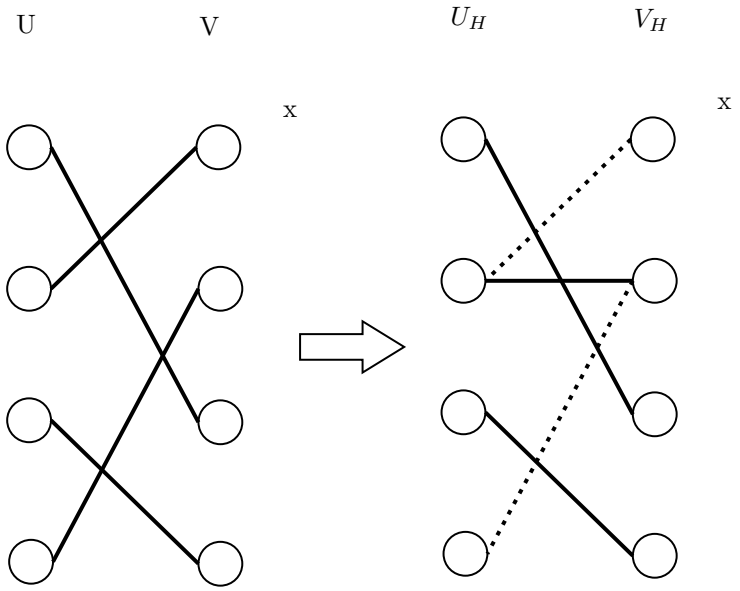

Figure 2: The new matching will differ at most one single path

Firstly, we introduce the notion of maximin share fairness. The idea is similar to the cake-cutting problem. Assuming agent i can partition M items into N bundles, with other $N-1$ agents choosing $N-1$ bundles before he. In the worst-case scenario, agent i receives his least preferred bundle. Thus, agent i must partition the items to maximize the value of his least preferred bundle, known as their maximin share (MMS). Since the other $N-1$ agents may receive bundles with at least the same valuation as MMS, the MMS represents the most agent i can guarantee to receive.

Next, we briefly introduce their model from the paper: There are M = $\{1,\ldots,m\}$ items and N = $\{1,\ldots,n\}$ agents, and they seek a fair allocation among these agents. An allocation $A = \{A_1,\ldots,A_n\}$ is a partition of total m items into n bundles, with each bundle allocated to an agent. The maximin share of agent i, denoted as μ_i, is the maximum value he can guarantee to receive. Let $\mathcal{A}$ be the set of all possible allocations. The maximin share of agent i is defined as: $\mu_i = \underset{A\in\mathcal{A}}{Max}\ \underset{A_k\in A}{Min}\ v_i(A_k)$.

However, ensuring that each agent receives his maximin share simultaneously is challenging. Therefore, the notion is relaxed to α MMS. An allocation is α approximate MMS if each agent i receives a bundle of items A_i with a value of at least $\alpha\mu_i$, expressed as $v_i(A_i) \geq \alpha\mu_i$, for some $\alpha \in (0,1)$. Moreover, the problem instance is denoted as $I = (N, M, V)$.

In their paper, the authors present an algorithm that provides approximate 2/3 MMS allocations. They use several propositions to aid in the proof. We will now briefly go through some of these propositions and illustrate their algorithm.

Firstly, in Propostion 1, they prove that by scaling the valuation of agent i, the α approximate MMS of agent i will remain the same, Hence, they extend it to Propostion 2, stating that the α approximate MMS of agent i remains unchanged if the valuation of agent i is normalized to the size of N, denoted as $|N|$, and $\mu_i \leq 1$ for all agents i.

In Proposition 3, they demonstrate that removing one agent and one item at the same time

from the original problem instance I does not decrease the α approximate MMS of other agents. Consequently, they extend to Corollary 4, asserting that the α approximate MMS of other agents does not decrease if we remove the same number of agents and items from the original problem instance I.

Moreover, in Proposition 5, they prove that if all agents' valuations are normalized and no item is worth more than δ for all the remaining agents, where $\delta \in (0, 1/2)$, they can utilize a simple bag filling algorithm to achieve a $1 - \delta$ MMS allocation.

In order to achieve 2/3 approximate MMS allocations, they decide to categorize the items into three sets:

- High-valued item (S_H): These are items valued by at least one agent at a minimum of $2/3\mu_i$, which is $S_H = \{j \in M : \exists i \in N \ s.t. \ v_{ij} \geq 2/3\}$.
- Medium-valued item (S_M): These are items valued by at least one agent at a minimum of $1/3\mu_i$, while no other agent values them higher than $2/3\mu_i$, which is $S_M = \{j \in M : \exists i \in N \ s.t. \ v_{ij} \geq 1/3, v_{ij} < 2/3, \forall i \in N\}$.
- Low-valued item (S_L): These are items valued by no agent higher than $1/3\mu_i$,, which is $S_L = \{j \in M : \forall i \in N \ s.t. \ v_{ij} < 1/3\}$

The authors propose a greedy allocation of high-valued items to agents who value them more than 2/3 without decreasing the MMS of other agents. Low-valued items can be allocated to agents with at least a 2/3 MMS by Proposition 5. The challenge lies in handling the medium-valued items.

To address this, the authors prove their Proposition 6: given any problem instance I, the ordered instance of it I' can be found in polynomial time. In I', the valuations are sorted for each agent using selection sort. They further demonstrate that their Theorem 7: if an allocation A' is the α approximate MMS allocation for I', it can be used to find an allocation A that achieves an α approximate MMS allocation for I. This algorithm simply traces all the items allocated by A' and incorporates them into the allocation A.

Next, we will discuss how they achieve a 2/3 MMS approximate allocation by the propositions they had before.

The first step is **Matching Procedure**, which allocates high-valued items (S_H) through maximum matching. Initially, they normalize all item valuation to ensure $\mu_i \leq 1$ for all agents(by proposition 2). Then, they construct a bipartite graph with agents on one side and all items from S_H on the other. Subsequently, they solve this maximum matching and remove all matched items and agents from the instance. This process continues until all items in S_H are allocated.

The second step is **Greedy Assignment from** S_M. After completing the first step, all items in S_H are allocated. When $|S_M| > |N|$, each agent i must create a bundle containing at least two items from S_M when calculating their MMS μ_i, according to the pigeonhole principle. Moreover, Proposition 6 ensures the existence of a lest preferred item j^* in any set of items. Therefore, two items from S_M can be greedily assigned together, denoted as $S = \{j^* - 1, j^*\}$, to an arbitrary agent i with valuation $v_i(S) \geq 2/3$. This ensures that allocating S to agent i guarantees they receive at least $2/3\mu_i$. Then, they prove that the new instance $I' = (N \backslash i, M \backslash S, V)$ will satisfy $\mu'_k \geq \mu_k$ for all remaining agents k. Therefore, they can keep normalizing the agents' valuations again and repeat this step until $|S_M| \leq |N|$.

The last step is **Modified Bag Filling**. When $|S_M| \leq |N|$, they initialize a bag S by using one arbitrary item from S_M and fill this bag S with items from S_L until there exists an agent valuing S

at least $v_i(S) \geq 2/3$. Once $|S_M| = 0$, the standard bag filling algorithm will be applied to achieve at least 2/3 MMS allocation for the remaining agents.

This paper demonstrates that their algorithm can achieve a 2/3 MMS allocation. Later, Garg and Taki [8] provided an improved approach to achieve 3/4 MMS allocation.

1.1.4 Edge-Weighted Online Bipartite Matching [9]

Since our algorithm assigns weights to the relationships between items and agents, we explored the paper to gain insight into previous research on weighted matching problems. The paper discusses a significant advancement in edge-weighted online bipartite matching. Traditionally, this problem has been challenging, with the best-known algorithms only achieving a 1/2 competitive ratio. This paper introduces a novel algorithm that surpasses this ratio, achieving a competitive ratio of at least 0.5086, which is a breakthrough given the problem's complexity and previous results.

The algorithm's success hinges on an innovative approach called "online correlated selection" (OCS), which allows for decisions across different pairs of vertices to be negatively correlated, enhancing the overall selection process. This method contrasts with previous strategies that relied on independent random choices, thereby improving the competitive ratio.

Imagine a job fair with multiple companies and job seekers attending. Each company rates each job seeker, and these ratings represent how well-suited the job seeker is for the company. These ratings can be seen as the "weights of the edges". In this scenario, each connection between a company and a job seeker is an edge in a bipartite graph.

The goal of this problem is to create the best possible match-ups between job seekers and companies so that the total ratings (i.e., the sum of the edge weights) are maximized. This is referred to as "maximum weight matching".

The new algorithm discussed in the paper employs a method called "online correlated selection" (OCS). This method allows the algorithm to make matching decisions where different decisions are negatively correlated. This means that if one decision matches a job seeker to a company, subsequent decisions are more likely to match other job seekers to different companies.

Here's a simple example to illustrate:

- Suppose there are three companies (A, B, C) and three job seekers (1, 2, 3).
- Each job seeker has random ratings for each company, for instance, job seeker 1 has a high rating for company A but lower ratings for companies B and C.
- Using traditional algorithms, you might randomly decide to match job seeker 1 with company A, and then consider other job seekers and companies independently.
- Using the OCS algorithm, once the decision to match job seeker 1 with company A is made, it influences subsequent decisions, making it more likely for job seekers 2 and 3 to be matched to companies B or C, thereby achieving a higher overall total score.

The benefit of this method is that it allows the algorithm to more comprehensively consider all possible pairings, leading to higher overall utility. This is why the algorithm in the paper can achieve a competitive ratio better than previous algorithms.

The paper also mentions that this algorithm provides strong evidence that edge-weighted bipartite matching, even though a generalized problem, might be strictly easier to solve in an online setting compared to monotone submodular welfare maximization. This is due to the specific characteristics of the edge-weighted model that the novel algorithm effectively exploits.

1.1.5 Class Fairness in Online Matching [6]

The paper addressed the issue of class fairness in online matching scenarios, where items that arrive in real time need to be allocated to agents. These agents are organized into predefined classes, and the objective is to ensure equitable treatment across these classes. Given that our paper also explored fairness in online matching issues, we investigated the annotation and definition of class fairness as presented in the document.

Initially, the paper establishes the concept of envy-freeness, which, when applied to individual agents, requires that no agent should feel envious of the resources allocated to another agent. **CEF1** stands for **Class Envy-freeness up to One Item**, which is the class level of EF1 (Envy-freeness up to One Item) [10]. It is a fairness criterion adapted from the fair division literature to the context of online matching involving classes of agents. In this setting, CEF1 demands that no class of agents envies another class after the possible removal of one item from the envied class's allocation. This condition tries to ensure that each class feels its allocation is fair, relative to what others have received, even if one item was hypothetically removed from the others' allocations.

Despite CEF1 seems reasonable, it is **impossible to apply CEF1 in online matching problems**. For instance, imagine two classes, each with two agents, and two items arrive that all four agents desire. The algorithm might allocate both items to agents within a single class, causing the other class to feel envy, even if one item is subsequently removed. This issue is straightforward to rectify with a "class-aware" algorithm that considers class distinctions: simply give the second item to an agent from the class that did not get the first item. Nonetheless, even algorithms aware of class distinctions cannot consistently achieve CEF1 in a slightly more complex scenario. Therefore, here are several criteria for evaluating the matching algorithm besides using CEF1.

- Non-wastefulness (NW): A matching is considered non-wasteful if the arriving item is matched to an agent who likes it whenever such an agent exists.
- Class Maximin Share Fairness (CMMS): This is a class level of MMS (Maximin Share Fairness) [11]. A matching is a class maximin share fair if for every class i in a set of classes k, the value $V_i(X)$ of the matching X is greater than $\alpha \times mms_i$ is the maximin share of class i. When $\alpha = 1$, it is simply referred to as class maximin share fairness (CMMS).
- Class Proportionality (CPROP): A matching X is said to be α-class proportional if for every class i in a set of classes k, the value $V_i(X)$ for the class from the matching exceeds α times the proportional share $prop_i$ of that class. The proportional share $prop_i$ is defined as the maximum value class i can derive from the worst bundle among all fractional partitions of items into k bundles. When $\alpha = 1$, the matching is simply referred to as class proportional (CPROP).
- Utilitarian Social Welfare (USW): The utilitarian social welfare of a matching X is defined as the sum of values that all agents derive from the items matched to them. A matching X is

Indivisible			Divisible		
Fairness	Algorithm	Upper Bound	Fairness	Algorithm	Upper Bound
α-CEF1 + NW	1/2	1/2	α-CEF1 + NW	$1 - 1/e$	3/4
α-CMMS	1/2	1/2	α-CPROP	$1 - 1/e$	$1 - 1/e$
α-USW	1/2	1/2	α-USW	1/2	$1 - 1/e$

Table 1: The summary of our results on deterministic algorithms for matching indivisible and divisible items[6].

α-USW if the utilitarian social welfare of X is greater than α times the utilitarian social welfare of the best possible matching X^*. When $\alpha = 1$, the matching is referred to as USW-optimal.

These definitions help in understanding how the concepts of efficiency and fairness are operationalized in the context of indivisible item matching.

The paper employs three algorithms to address indivisible, divisible, and randomized matching problems, and the outcomes of the evaluation criteria previously mentioned for these three algorithms are summarized in table 1.

- MATCH-AND-SHIFT Algorithm: This algorithm is designed for indivisible item allocation in an online setting. It operates by allocating items to classes based on a predetermined order. Once a class receives an item, it is moved to the end of the queue to ensure other classes get a chance to receive items, promoting a fair distribution. This procedure helps the algorithm achieve approximations for non-wastefulness, class envy-freeness up to one item (CEF1), class maximin share fairness (CMMS), and utilitarian social welfare (USW).
- EQUAL-FILLING Algorithm: For divisible item allocation, this algorithm uses a water-filling approach where each class receives an equal portion of each item. It then distributes a class's total share among its agents to maximize satisfaction within the class. This method achieves good approximations for class envy-freeness and class proportionality by ensuring that no class receives less than a nearly optimal fraction of their maximum possible valuation of the items.
- EQUAL-FILLING-OCS Algorithm: This algorithm extends the "EQUAL-FILLING" approach for indivisible items by incorporating a randomized method known as Online Correlated Selection [9]. The algorithm first computes a fractional solution using the "EQUAL-FILLING" method and then uses OCS to round this solution to an indivisible allocation. This rounding process allows the algorithm to exceed the typical approximation bounds achieved by deterministic methods, improving fairness metrics like class proportionality while maintaining efficiency in terms of USW.

2 Online Weighted-Edge Bipartite Matching Problem

After thoroughly examining relevant literature, we have gained valuable insights into designing an algorithm to address the model we have proposed. Among the approaches we explored, we have observed that many successful algorithms emphasize a balance between fairness and efficiency, often

employing greedy algorithms. Drawing inspiration from these findings, we have chosen to build upon the algorithm proposed by Hosseini et al. [6]. Our algorithm demonstrates similar performance when the edge weight is set to 1.

In subsequent sections, we will elucidate the problem we have investigated, provide a brief description of the algorithm we propose, and illustrate the experimental results of our algorithm. Through this, we aim to shed light on our methodology and demonstrate its effectiveness in tackling the challenges posed by our model.

2.1 Model

Similar to the model proposed by Hosseini et al. [6], we begin by considering a bipartite graph $G = (U, V, E)$, where the vertices $u \in U$ are predefined on the left side, representing agents, and a sequence of vertices $v \in V$ arrives in an online fashion on the right side, representing items.

The set of agents U can be divided into k groups denoted as $N_1, \ldots, N_k$. These groups are mutually exclusive ($N_i \cup N_j = \emptyset$, for all $i \neq j$) and collectively cover the entire set of agents ($\cup_{i=1}^{k} = U$).

The edges adjacent to vertex v become apparent upon its arrival, and these edges carry different weights corresponding to different groups. Specifically, if $u' \in U$ and $v' \in V$ are adjacent, the edge's $e_{u',v'} \in E$ weight $w_{u',v'}$ is non-zero. Otherwise, the edge weight $w_{u',v'}$ would be set to zero. In our model, we constrain the weight range to integers from 1 to 10 ($w \in [1, 10]$). Furthermore, if u' and v' are matched, $m_{u',v'} = w_{u',v'}$, and $m_{u',v'} \in [1, 10]$. Otherwise, $m_{u',v'} = 0$. We represent our matching as a matrix $M = (m_{u,v})_{u \in U, v \in V} \in [0, 10]^{U \times V}$.

In our model, we exclusively consider indivisible matching, meaning that items cannot be divided into fractions.

Our objective is to maximize the total matching weight by assigning items to agents, while also ensuring fairness among different groups.

2.2 Priority-Updated-Match Algorithm

In this section, we introduce our algorithm 1, designed to provide an effective and fair solution to the model we mentioned before.

We begin by maintaining all groups in a priority queue. Upon the arrival of a new vertex v, we inspect the groups with the lowest priority value. Subsequently, we arrange all possible vertices $u' \in U$ in descending order of edge weight. We then examine these vertices in sequence, starting from the one with the highest edge weight. If a possible match is found, two vertices will be matched by our algorithm, and the priority value of the corresponding group is incremented by the weight of the edge. If no available match is found, we proceed to check the groups with higher priority values. If a match cannot be made even after examining all groups, the incoming vertex v is dropped. Finally, the priority queue is updated accordingly.

Algorithm 1: Priority-Updated-Match

```
Push all groups N = (N_1, ..., N_k) into a priority queue Q
while item v ∈ V arrives do
    while Q ≠ ∅ do
        Pop all groups N_low with the lowest priority value from the priority queue Q
        Let U_possible be all the possible agents from N_low
        if U_possible ≠ ∅ then
            Order u ∈ U_possible by the ascending order of the weight w_{u,v}
            Match u_highest to the item v
            Update the weight of N_j, where u_highest ∈ N_j
            N_checked ← N_low
            N_low ← ∅
            break
        end
        N_checked ← N_low
        N_low ← ∅
    end
    if No match found then
        Drop v
    end
    Push N_checked into Q
end
```

In this project, we aim to utilize three metrics proposed Hosseini et al. [6] to evaluate the performance of our algorithm.

- Non-wastefulness
- Utilitarian social welfare
- Group level fairness

Non-wastefulness This notion refers to the absence of any unmatched agent who has an adjacent unmatched item.

In our algorithm, we prioritize making a match between an unmatched agent and an unmatched item when it satisfies group-level fairness. However, when maintaining fairness among different groups is impossible, the algorithm will match the potential match which will ensure the absence of any unmatched agent who has an adjacent unmatched item. Hence, our algorithm is non-wastefulness.

Utilitarian Social Welfare This notion refers to the summation of the total weight of a matching.

Mathematically, it can be represented as: $\sum_{u \in U} \sum_{v \in V} m_{u,v}$. Here, $M_{u,v}$ represents the weight of the match between agent u and item v.

Assuming there exists an optimal matching M^* that maximizes the utilitarian social welfare. We define an α utilitarian social welfare matching as follows: if for a matching M, we have $USW(M) \geq \alpha \times USW(M^*)$. Particularly, when $\alpha = 1$, this matching represents the optimal utilitarian social welfare matching.

Proposition 1. *All non-wastefulness matching for a weighted-edge bipartite graph, where the maximum weight of an edge is w_{max}, is $\frac{1}{2\times w_{max}} - USW$*

Proof. Inspired by the proof provided by Hosseini et al. [6] for the non-weighted version, we extend it to our weighted-edge model.

Let M^* be the optimal matching which maximizes the utilitarian social welfare. Then, for every match $m^* \in M^*$, the optimal matching will have:

$\sum_{(u,v):(m^*_{u,v}\geq 1)} m^*_{u,v} \leq \sum_{(u,v):(m^*_{u,v}\geq 1)} w_{max}$

This is because the weight of every match will be at most w_{max} and at least 1.

Let M be a non-wastefulness matching. Based on the definition of non-wastefulness matching, for every $m_{u,v} \in M$, if u, v are not matched, either $u \in U$ will match another item $v' \in V\backslash\{v\}$ or $v \in V$ will match another agent $u' \in U\backslash\{u\}$.

Hence, we have:

$\sum_{u'\in U} m_{u',v} \geq 1$ or $\sum_{v'\in V} m_{u,v'} \geq 1$.

Therefore, $w_{max} \times \sum_{u'\in U} m_{u',v} \geq w_{max}$ or $w_{max} \times \sum_{v'\in V} m_{u,v'} \geq w_{max}$.

Thus, we will have:

$USW(X^*) \leq \sum_{(u,v):(m^*_{u,v}\geq 1)} w_{max} \leq w_{max} \times \sum_{v\in V}\sum_{u'\in U} m_{u',v} + w_{max} \times \sum_{u\in U}\sum_{v'\in V} m_{u,v'} = 2 \times w_{max} \times USW(X)$

Hence, $USW(X)$ with max weight to be w_{max} is $\frac{1}{2\times w_{max}} - USW$

□

Since our matching is non-wastefulness and the maximum weight is 10, our algorithm is $\frac{1}{20} - USW$

Group-level fairness Before defining group-level fairness, it is essential to establish how each group evaluates the value they and other groups obtain. Inspired by Hosseini et al. [6], we can adopt the following approach:

- When evaluating their own value, each group i sums the weight of all matches made to its agents. This can be represented as $V_i(X_i)$ for all group $i \in N$.
- When evaluating the value obtained by other groups, we employ the optimistic valuation. Specially, this involves estimating the optimal value of the items assigned to the agents of other groups. This can be represented as $V_i^*(M_j)$, where $i \in N$ and $j \in N\backslash\{i\}$.

However, as Hosseini et al. [6] declared in their paper, it is impossible to prove group-level fairness. Suppose we need to match one item to an agent i at the beginning, other $k-1$ groups must envy the group containing this agent i. Therefore, we need to relax this notion to group-level fairness by removing at most 10 items, denoted as GEF-10.

Proposition 2. *Priority-Updated-Match algorithm for a weighted-edge graph, where the maximum weight of an edge is w_{max}, is $\frac{1}{w^2_{max}+w_{max}} - GEF - w_{max}$*

Proof. Inspired by the proof provided by Hosseini et al. [6] for the non-weighted version, we extend it to our weighted-edge model.

Let M_i denote the set of items matched to group i, and let $t = |M_i|$ represent the number of items matched to the group i. We can infer that $t \leq V_i(M_i) \leq w_{max} \times t$ since the weight of every match will be at most w_{max} and at least 1.

Consider an arbitrary group j distinct from i, where $j \in N \backslash i$. Let M_j be the set of items matched to group j, and M_j^* be the subset of items in M_j that could potentially be matched by an unmatched agent in group i

When $M_j^* = \emptyset$. Since the match with the weight of w_{max} can be replaced by w_{max} matches with the weight of 1, the size of the optimal matching of group i towards the set of items M_j will be at most the number of the valuation of the total match, denoted as $V_i(M)$. By the definition of non-wastefulness,$V_i(M)$ will be bounded above by $w_{max} \times t$. Moreover, those w_{max} matches will have a weight at most w_{max}, we have $V_i^*(M_j) \leq w_{max} \times V_i(M) \leq w_{max}^2 \times t$.

When $M_j^* \neq \emptyset$, each item matched to group j from M_j^* implies the priority value of group j is less than that of group i. Since the weight of matches ranges from 1 to w_{max}, each item matched to group i allows for a maximum of w_{max} items to be matched to group j. Moreover, for each item matched to group i, at most w_{max} additional matches can be made to group j from M_j^*. Therefore, $|M_j^*| \leq w_{max} \times t + w_{max}$.

Let S be the set of items such that $s \subseteq M_j^*$ and $|S| = w_{max}$. Then, $|M_j^* \backslash S| \leq w_{max} \times t$. Consequently, $V_i^*(M_j \backslash S) \leq w_{max}^2 + w_{max} \times t$, implying that $\frac{1}{w_{max}^2 + w_{max}} \times V_i^*(M_j \backslash S) \leq t$.

Therefore, we have $\frac{1}{w_{max}^2 + w_{max}} \times V_i^*(M_j \backslash S) \leq V_i(M_i)$.

Thus, when the maximum weight is w_{max}, our algorithm is $\frac{1}{w_{max}^2 + w_{max}} - GEF$ by removing w_{max} items. □

Since the maximum weight is 10 in our matching, our algorithm is $\frac{1}{110} - GEF$ by removing 10 items, which is $\frac{1}{110} - GEF - 10$

2.3 Experiment

As we have reviewed several pieces of literature, it is evident that many of them have leveraged the concept of the greedy algorithm in designing their own algorithms. Consequently, we have opted to use a greedy algorithm as our baseline for comparison. Specifically, our baseline greedy algorithm will always match the agent to the potential item with the highest edge weight, ensuring maximum immediate utility, without considering broader fairness considerations. This approach allows us to evaluate the performance of our algorithm relative to a well-established benchmark in the field.

2.3.1 Experiment setting

For our experiment, we set the number of groups to be 4, with each group containing 25 agents, resulting in a total of 100 agents. We conduct two experiments: one with 200 items to ensure maximum matching opportunities, and another with only 20 items, simulating a scenario where resources are limited. This setup allows us to assess the performance of our algorithm under different scenarios, varying the number of items available for matching.

2.3.2 Experiment results

Based on our experimental results in fig. 3, it is evident that **both our algorithm and the greedy algorithm can achieve fair matching when there are sufficient items**. Moreover, the greedy algorithm outperforms our algorithm in terms of the total weight of the final matching by 19.6%. We attribute this to the nature of the greedy algorithm, which always matches agents to items with

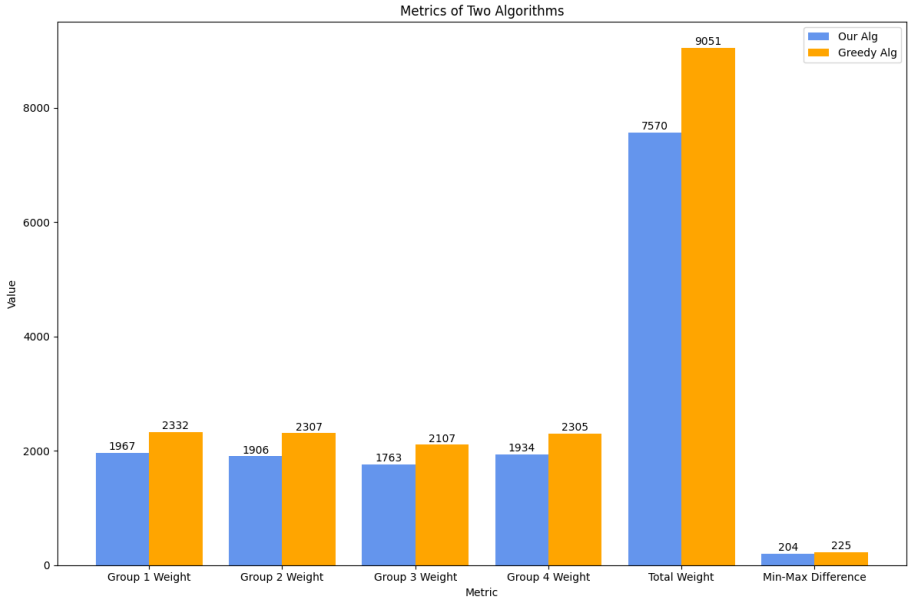

Figure 3: Experiment result with 200 items

the highest weight. When enough items are available, nearly every agent can obtain their most desired items, resulting in a higher total weight. However, in real-life scenarios, agents, such as job applicants or cities, may not have access to enough items, such as jobs or resources. Therefore, to simulate such situations, we conducted our second experiment restricting the number of items to 20.

The fig. 4 presents the experimental results when restricted to only 20 items. As expected, the total weight achieved by the greedy algorithm surpasses that of our algorithm by 9.8%. However, a notable observation is that the greedy algorithm fails to ensure fairness when the number of items is insufficient. In contrast, **our algorithm achieves better fairness** among different groups. The disparity between the group obtaining the most weight and the group obtaining the least weight is 658, roughly 9 times higher than that of our algorithm. This highlights the advantage of our algorithm, which consistently ensures fairness among different groups. Moreover, this scenario, where the number of items is restricted, is more reflective of real-life situations. For instance, in reality, job applicants often encounter limited job opportunities, making it challenging to secure ideal positions.

3 Concluding Remarks

In this project, we conducted a review of several papers spanning various perspectives regarding fairness and bipartite matching. Building upon the work of Hosseini et al. [6], we extended their non-weighted edge model to our weighted edge model and devised an algorithm to address this problem. Furthermore, we analyzed the performance of our algorithm using three metrics adapted from their work.

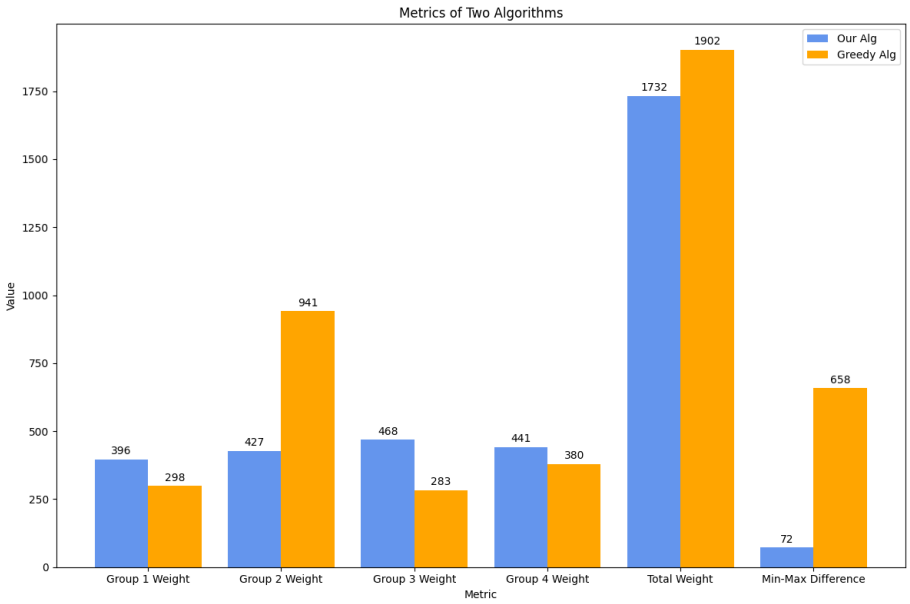

Figure 4: Experiment result with 20 items

To facilitate our analysis, we diligently implemented our algorithm in code and established the greedy algorithm as our baseline for comparison. Through this meticulous approach, we were able to demonstrate that our algorithm achieves a balance between fairness and efficiency. By simultaneously optimizing both aspects, our algorithm offers a promising solution to bipartite matching problems in real-world applications.

Indeed, there are several avenues for future exploration and enhancement of our algorithm. One potential direction is to extend our approach to address divisible matching problems, where items can be divided into fractions. Investigating strategies for efficiently handling divisible matching would be valuable for scenarios where resources can be divided among multiple agents.

Additionally, incorporating randomization techniques into our algorithm could offer opportunities for further performance improvement. Randomized algorithms have been shown to provide efficient solutions in various optimization problems, and exploring their application to bipartite matching could yield interesting insights.

Furthermore, leveraging advice and prediction mechanisms could enhance the capabilities of our algorithm. By integrating predictive models or advice from experts, we could make more informed decisions during the matching process, potentially leading to better outcomes in terms of fairness and efficiency.

Overall, the exploration of these avenues holds promise for advancing the effectiveness and applicability of our algorithm in real-world settings. Continued research in these areas could lead to significant advancements in bipartite matching algorithms and their practical utility.

References

[1] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[2] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[3] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time, 2022.

[4] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *Acm Sigact News*, 39(1):80–87, 2008.

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

[6] Hadi Hosseini, Zhiyi Huang, Ayumi Igarashi, and Nisarg Shah. Class fairness in online matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5673–5680, 2023.

[7] Jugal Garg, Peter McGlaughlin, and Setareh Taki. Approximating maximin share allocations. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.

[8] Jugal Garg and Setareh Taki. An improved approximation algorithm for maximin shares. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 379–380, 2020.

[9] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. Edge-weighted online bipartite matching, 2020.

[10] Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.

[11] Eric Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.