TR-IIS-08-011

Bi-objective Optimization : An Online Algorithm for Job Assignment

Chien-Min Wang, Xiao-Wei Huang, and Chun-Chen Hsu

November 25, 2008 || Technical Report No. TR-IIS-08-011
http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2008/tr08.html

Bi-objective Optimization : An Online Algorithm for Job Assignment

Chien-Min Wang[1], Xiao-Wei Huang[1], and Chun-Chen Hsu[1,2]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan
{cmwang,xwhuang,tk}@iis.sinica.edu.tw
[2] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
d95006@csie.ntu.edu.tw

Abstract. We study an online problem that occurs when the capacities of machines are heterogeneous and all jobs are identical. Each job is associated with a subset, called feasible set, of the machines that can be used to process it. The problem involves assigning each job to a single machine in its feasible set, i.e., to find a feasible assignment. The objective is to maximize the throughput, which is the sum of the bandwidths of the jobs; and minimize the total load, which is the sum of the loads of the machines. In the online setting, the jobs arrive one-by-one and an algorithm must make decisions based on the current state without knowledge of future states. By contrast, in the offline setting, all the jobs with their feasible sets are known in advance to an algorithm. Let m denote the total number of machines, α denote the competitive ratio with respect to the throughput and β denote the competitive ratio with respect to the total load. In this paper, our contribution is that we propose an online algorithm that finds a feasible assignment with a throughput-competitive upper bound $\alpha = O(\sqrt{m})$, and a total-load-competitive upper bound $\beta = O(\sqrt{m})$. We also show a lower bound $\alpha\beta = \Omega(\sqrt{m})$, of the problem in the offline setting, which implies a lower bound $\alpha\beta = \Omega(\sqrt{m})$, of the problem in the online setting.

Keywords: Online algorithms, job assignment, bi-objective optimization, throughput, load.

1 Introduction

In the scenario where a number of machines with different positive capacities are ready to provide services for a set of jobs, each job is associated with one non-negative unit weight and a subset, called the *feasible set*, of the machines that can be used to process it. The problem involves assigning each job to a single machine in its feasible set, i.e., to find a feasible assignment. The objective is to maximize the throughput, which is the sum of the bandwidths of the jobs; and minimize the total load, which is the sum of the loads of the machines. We consider the online problem in the following model. There are m machines with different capacities, and n jobs. Each job i has the same weight and a feasible set F_i. In the online setting, the job arrives with its weight and its feasible set. An online algorithm must assign the job to a single machine in its feasible set without knowledge of future states and the decision cannot be revoked at a later stage. By contrast, in the offline setting, all the jobs and their feasible sets are known in advance to an algorithm.

Given a feasible assignment, which is a mapping from the jobs to the machines, the load on a machine in the assignment is the sum of the number of the jobs assigned to it divided by its capacity [1]. The amount of bandwidth allocated to a job in an assignment, which represents the quality of service, depends on the total weight of jobs that share the resource with it [6, 11]. Specifically, in our model, if l jobs are assigned to the same machine with capacity c, then each one will be allocated a bandwidth of c/l, since all jobs have the same weight. We measure the assignment by the throughput as the utility, defined as the sum of the bandwidths of all jobs, and the total load as the congestion, defined as the sum of the loads of all machines. As mentioned earlier, our goal is to find a feasible assignment in order to simultaneously maximize the throughput and minimize the total load.

Our contribution in this paper is twofold. First we present an online algorithm for the online job assignment problem when considering the throughput and the total load. The algorithm has the throughput-competitive ratio $\alpha = O(\sqrt{m})$ and the total-load-competitive ratio $\beta = O(\sqrt{m})$, which will be equal to the ratio with respect to the average of the loads. Our second contribution is that we show a lower bound of this problem in the offline setting with $\alpha\beta = \Omega(\sqrt{m})$, where α is the competitive ratio with respect to the throughput and β is the competitive ratio with respect to the total load. Note that the problem in the offline setting must be easier than or equivalent to the problem in the online setting.

The remainder of this paper is organized as follows. Section 2 contains a literature review. In Section 3, we define the problem formally. Section 4 presents the proposed online algorithm and its properties, and Section 5 shows a lower bound of the offline problem.

2 Related Work

Many approaches for measuring the quality of a job assignment have been proposed. For example, a popular measure that minimizes the maximum load [1, 3], measures for minimizing the l_2-norm or any other l_p-norm of the machine load vector [2, 8], measures that consider both fairness and balancing issues [6, 11], and some other quality measures are discussed in [16]. However, it is not always clear how to properly measure the quality of an assignment in general. Hence, it is desirable to find a solution that can approximate several measures simultaneously [2, 4, 6, 7, 9–11, 13, 17]. In this work, we focus on a measure that maximizes the throughput and minimizes the total load simultaneously.

In [1, 3], the authors studied the load balancing problem with the objective of minimizing the maximum load. The restricted assignment model was studied with respect to this measure in [3]. In the restricted assignment model, there are m identical machines and n jobs. Each job is associated with a non-negative weight and a feasible set of machines to which it can be assigned. It was showed that the maximum load generated by the greedy online strategy is within $O(\log m)$ factor of the optimal load.

Kleinberg et al. [14] studied fairness issues in several routing and load balancing models in an offline setting. They defined the notion of prefix competitiveness and a stronger notion of coordinate-wise competitiveness and considered several offline problems in terms of these measures.

In [11], the authors studied the $1-\infty$ model in an online setting from the fairness and load balancing perspective. Under the $1-\infty$ model, there are m identical machines and n jobs. Each job is associated with the same weight and with a feasible set of machines to which it can be assigned. The $1-\infty$ model is a special case of the restricted model; the only difference is that in the $1-\infty$ model, all jobs have the same (one unit) weight. Goel et al. [11] proved that the greedy strategy, which always assigns a job to the machine with the smallest load in its feasible set based on the current state is globally $O(\log n)$-fair and globally $O(\log m)$-balanced, where m is the number of machines and n is the number of jobs. They also showed that any online algorithm must be globally $\Omega(\log m)$-fair as well as globally $\Omega(\log m)$-balanced in [11]. Buchbinder and Naor [6] solved the open problem in [11] and proved that the greedy strategy is globally $O(\log m)$-fair and globally $O(\log m)$-balanced in the $1-\infty$ model.

It is important to note that our model is different from the $1-\infty$ model in [6, 11], since the machines in the $1-\infty$ model must be identical and the machines in our model have different positive capacities. Clearly, the greedy strategy proposed in [6, 11], which always assigns a job to the least loaded machine in the feasible set, can not work well in our model. To illustrate this point, consider a simple setting with only two machines with capacities c_1 and c_2, where $c_1 \gg c_2$, and two jobs that can be processed by the both machines. After assigning the first job to one machine, the greedy strategy assigns the second job to the other machine whose loaded is zero and results in an assignment such that the total load is $\frac{1}{c_1} + \frac{1}{c_2}$. Since there exists another assignment such that the total load is $\frac{2}{c_1}$, we can see that the ratio is $\frac{1}{2} + \frac{c_1}{2c_2}$ and may be greater than any given constant if $c_1 \gg c_2$.

3 The Problem Definition

In this section, we formally define our job assignment problem.

Definition 1 (A feasible assignment). *Given the machines' index set $\mathcal{M} = \{1, \cdots, m\}$, the jobs' index set $\mathcal{J} = \{1, \cdots, n\}$ and the non-empty feasible set $F_i \subset \mathcal{M}$ of job i for all $i \in \mathcal{J}$, a feasible assignment $\phi : \mathcal{J} \to \mathcal{M}$ is a mapping from $\mathcal{J} = \{1, \cdots, n\}$ to $\mathcal{M} = \{1, \cdots, m\}$ subject to $\phi(i) \in F_i$ for all $i \in \mathcal{J}$.*

Given a feasible assignment, the bandwidth allocated to a job in the assignment is the quality of service it gets depends on the total weight of jobs that share the resource together with it [6, 11]. In our model, there are m machines with different capacities and n jobs with the same (one unit) weight. Hence, the amount of bandwidth allocated to a job depends on the number of jobs that share the resource with it. We define the bandwidth vector of the feasible assignment as follows:

Definition 2 (The bandwidth vector of a feasible assignment). *Given m machines with capacities $c_1, c_2, \cdots, c_m$, n jobs with their feasible sets, and a feasible assignment $\phi : \mathcal{J} \to \mathcal{M}$, where $\mathcal{J}$ is the jobs' index set and $\mathcal{M}$ is the machines' index set, the bandwidth of job i in the assignment is $b_i = \frac{c_{\phi(i)}}{|A^{\phi}(\phi(i))|}$, where $A^{\phi}(k)$ is the set of jobs assigned to machine k in the assignment ϕ. The bandwidth vector of the assignment $\boldsymbol{B}_{\phi} = (b_1, b_2, \cdots, b_n)$.*

The load on a machine in a feasible assignment is the sum of the number of jobs assigned to it divided by its capacity [1]. We define the load vector of the feasible assignment as follows:

Definition 3 (The load vector of a feasible assignment). *Given m machines with capacities $c_1, c_2, \cdots, c_m$, n jobs with their feasible sets, and a feasible assignment $\phi : \mathcal{J} \to \mathcal{M}$, where $\mathcal{J}$ is the jobs' index set and $\mathcal{M}$ is the machines' index set, the load of machine j in the assignment is $l_j = \frac{|A^{\phi}(j)|}{c_j}$, where $A^{\phi}(k)$ is the set of jobs assigned to machine k in the assignment ϕ. The load vector of the assignment $\boldsymbol{L}_{\phi} = (l_1, l_2, \cdots, l_m)$.*

We measure a feasible assignment ϕ of the job assignment problem by taking the throughput as the utility function $U(\boldsymbol{B}_{\phi}) = \sum_{i=1}^{n} b_i$ and the total load as the congestion function $C(\boldsymbol{L}_{\phi}) = \sum_{i=1}^{m} l_i$ in the following model. Our goal is to find a feasible assignment in order to simultaneously maximize the throughput and minimize the total load. In multi-objective optimization problems, it is unlikely that the different objectives could be optimized simultaneously by the same alternative parameter choices, especially for some conflicting objectives. Hence, to ensure that a design is satisfactory, there must be a trade-off between the criteria.

Definition 4. *[The job assignment problem envloves maximize the throughput and minimize the total load in an offline setting] Given m machines with capacities $c_1, c_2, \cdots, c_m$ and n jobs with their feasible sets, the problem is to find a feasible assignment $\phi : \mathcal{J} \to \mathcal{M}$ such that*

$$\alpha U(\boldsymbol{B}_{\phi}) \geq U(\boldsymbol{B}_{\phi'}) \text{ for all other feasible assignmets } \phi',$$

and

$$C(\boldsymbol{L}_{\phi}) \leq \beta C(\boldsymbol{L}_{\phi''}) \text{ for all other feasible assignmets } \phi'',$$

where the competitive ratio α, β are as small as possible simultaneously.

In the online setting, jobs arrive with their feasible sets one-by-one and the algorithm must immediately assign job i to machine $\phi(i)$ when job i arrives. This contrasts with the offline setting, where all the jobs and feasible sets are given initially.

4 The Proposed Online Algorithm and Its Properties

In this section, we introduce the proposed online algorithm and its properties. We begin by introducing notations. Let $A(i,j)$ denote the set of the jobs assigned to machine j when job i arrives, and $A(i,j) \subseteq \{1,2,\dots,i-1\}$. Let S_i denote the index set of machines which no job is assigned to when job i arrives. Let $max(i)$ denote the index of the machine with the most capacity in the feasible set F_i of job i, and $un(i)$ denote the index of the machine with the most capacity in $F_i \cap S_i$. In addition, let $\gamma_{k,i}$ denote the number of times that the machine k is regarded as the most powerful machine in the feasible sets of the first i jobs, i.e., $\gamma_{k,i} = |\{j|max(j) = k, 1 \le j \le i\}|$. The proposed online algorithm is detailed in Algorithm 1.

Algorithm 1 The proposed online algorithm: When a job i arrives, the algorithm assigns the job to a machine in its feasible set F_i.

```
$S_i := \{j||A(i,j)| = 0, 1 \le j \le m\}$
if $max(i) \in S_i$ then
  assign job $i$ to machine $max(i)$
else if $max(i) \notin S_i$ and $S_i \cap F_i = \emptyset$ then
  assign job $i$ to machine $max(i)$
else if $max(i) \notin S_i$ and $S_i \cap F_i \neq \emptyset$ then
  if $\sqrt{\gamma_{max(i),i}} c_{un(i)} \le c_{max(i)}$ then
    assign job $i$ to machine $max(i)$
  else if $\sqrt{\gamma_{max(i),i}} c_{un(i)} > c_{max(i)}$ then
    assign job $i$ to machine $un(i)$
  end if
end if
```

4.1 The $O(\sqrt{m})$-Competitive Ratio For the Total Load

Lemma 1. *Given a feasible assignment ϕ, the total load is*

$$C(\boldsymbol{L}_\phi) = \sum_{k=1}^{m} l_k = \sum_{k=1}^{m} \sum_{i \in A^\phi(k)} \frac{1}{c_{\phi(i)}} = \sum_{i=1}^{n} \frac{1}{c_{\phi(i)}}.$$

Proof. Since the load of machine k is $l_k = \frac{|A^\phi(k)|}{c_k} = \sum_{i \in A^\phi(k)} \frac{1}{c_k}$, we have

$$C(\boldsymbol{L}_\phi) = \sum_{k=1}^{m} l_k = \sum_{k=1}^{m} \sum_{i \in A^\phi(k)} \frac{1}{c_k}.$$

Note that $i \in A^\phi(k)$ means job i is assigned to machine k in the assignment ϕ, i.e., $\phi(i) = k$. Hence,

$$C(\boldsymbol{L}_\phi) = \sum_{k=1}^{m} \sum_{i \in A^\phi(k)} \frac{1}{c_k} = \sum_{k=1}^{m} \sum_{i \in A^\phi(k)} \frac{1}{c_{\phi(i)}} = \sum_{i \in \bigcup_{k=1}^{m} A^\phi(k)} \frac{1}{c_{\phi(i)}}.$$

Moreover, all jobs in $\mathcal{J} = \{1, 2, \cdots, n\}$ must be assigned to some machine $k \in \mathcal{M}$ in the assignment ϕ; thus, we can see that $\bigcup_{k=1}^{m} A^{\phi}(k) = \mathcal{J}$ and

$$C(\boldsymbol{L}_{\phi}) = \sum_{i \in \bigcup_{k=1}^{m} A^{\phi}(k)} \frac{1}{c_{\phi(i)}} = \sum_{i \in \mathcal{J}} \frac{1}{c_{\phi(i)}} = \sum_{i=1}^{n} \frac{1}{c_{\phi(i)}}.$$

□

According to Lemma 1, we can find an optimal assignment ϕ^L for the total load by assigning each job i to machine $max(i)$. It is easy to see that ϕ^L has $\beta =$ 1-competitive ratio for the total load. We refer to ϕ^L as the optimal assignment for the total load. In the following, Lemma 2 shows the proposed online algorithm has $\beta = O(\sqrt{m})$-competitive ratio for the total load.

Lemma 2. *The proposed online algorithm results in a feasible assignment ϕ with $\beta = O(\sqrt{m})$ such that $C(\boldsymbol{L}_{\phi}) \leq \beta C(\boldsymbol{L}_{\phi^L})$, where $\boldsymbol{L}_{\phi}$ is the load vector of the assignment ϕ derived by the proposed algorithm and $\boldsymbol{L}_{\phi^L}$ is the load vector of the optimal assignment ϕ^L for the total load.*

Proof. First of all, we evaluate the total load of the assignment ϕ^L. By Lemma 1, we calculate the total load of the assignment ϕ^L as follows:

$$C(\boldsymbol{L}_{\phi^L}) = \sum_{k=1}^{m} \sum_{i \in A^{\phi^L}(k)} \frac{1}{c_{\phi^L(i)}} = \sum_{k=1}^{m} \frac{|A^{\phi^L}(k)|}{c_k}.$$

Then, we evaluate the total load of the assignment ϕ derived by the proposed online algorithm. Observe the load, which is $\frac{|A^{\phi}(k)|}{c_k}$, on the machine k in the assignment ϕ, we have that $|A^{\phi}(k)| - |A^{\phi^L}(k)| \leq 1$ for all k according to the algorithm. Consider the upper bound of the ratio of the load on the machine k for all k in the cases of $|A^{\phi}(k)| - |A^{\phi^L}(k)| \leq 0$ and $|A^{\phi}(k)| - |A^{\phi^L}(k)| = 1$:

1. If $|A^{\phi}(k)| - |A^{\phi^L}(k)| \leq 0$, the load on the machine k in the assignment ϕ is equal to or less than the load on the machine k in the assignment ϕ^L, i.e. $\frac{|A^{\phi}(k)|}{c_k} \leq \frac{|A^{\phi^L}(k)|}{c_k}$.
2. If $|A^{\phi}(k)| - |A^{\phi^L}(k)| = 1$ and $|A^{\phi^L}(k)| \geq 1$, we have the ratio
$$\frac{\frac{|A^{\phi}(k)|}{c_k}}{\frac{|A^{\phi^L}(k)|}{c_k}} = 1 + \frac{1}{|A^{\phi^L}(k)|} \leq 2.$$
3. If $|A^{\phi}(k)| - |A^{\phi^L}(k)| = 1$ and $|A^{\phi^L}(k)| = 0$, we can see that the unique job i, assigned to the machine k, in the set $A^{\phi}(k)$ must be assigned to some machine k' such that $\sqrt{\gamma_{k',i}} \frac{1}{c_{k'}} > \frac{1}{c_k}$ in the assignment ϕ^L. Hence, the ratio will be bounded by the load of the machine k', i.e.
$$\frac{\frac{|A^{\phi}(k)|}{c_k}}{\frac{|A^{\phi^L}(k')|}{c_{k'}}} = \frac{\frac{1}{c_k}}{\frac{|A^{\phi^L}(k')|}{c_{k'}}} < \frac{\sqrt{\gamma_{k',i}} \frac{1}{c_{k'}}}{\frac{|A^{\phi^L}(k')|}{c_{k'}}} = \frac{\sqrt{\gamma_{k',i}}}{|A^{\phi^L}(k')|}.$$

Note that there are at most $m-1$ jobs in this case since there are at most m machines. Therefore, the ratio of the load of these machines in this case will be bounded by $\frac{\frac{1}{c_k}\sum_{i=|A^{\phi^L}(k)|-m+1}^{|A^{\phi^L}(k)|}\sqrt{i}}{\frac{|A^{\phi^L}(k)|}{c_k}}$, for some machine k.

Let k^* denote the machine's index with

$$\max_{1\le k\le m}\{\frac{\frac{|A^{\phi^L}(k)|}{c_k}+\frac{|A^{\phi^L}(k)|+1}{c_k}+\frac{1}{c_k}\sum_{i=|A^{\phi^L}(k)|-m+1}^{|A^{\phi^L}(k)|}\sqrt{i}}{\frac{|A^{\phi^L}(k)|}{c_k}}\}.$$

We obtain that

$$\beta \le \frac{C(\boldsymbol{L}_\phi)}{C(\boldsymbol{L}_{\phi^L})} = \frac{\sum_{k=1}^m \frac{|A^\phi(k)|}{c_k}}{\sum_{k=1}^m \frac{|A^{\phi^L}(k)|}{c_k}} \le \frac{\frac{|A^{\phi^L}(k^*)|}{c_{k^*}}+\frac{|A^{\phi^L}(k^*)|+1}{c_{k^*}}+\frac{1}{c_{k^*}}\sum_{i=|A^{\phi^L}(k^*)|-m+1}^{|A^{\phi^L}(k^*)|}\sqrt{i}}{\frac{|A^{\phi^L}(k^*)|}{c_{k^*}}}$$

$$\le 3+\frac{1}{|A^{\phi^L}(k^*)|}\sum_{i=|A^{\phi^L}(k^*)|-m+1}^{|A^{\phi^L}(k^*)|}\sqrt{i}.$$

Consider the cases of that $|A^{\phi^L}(k^*)|\le m$ and $|A^{\phi^L}(k^*)|>m$:

1. If $|A^{\phi^L}(k^*)|\le m$, it follows that

$$\beta \le 3+\frac{1}{|A^{\phi^L}(k^*)|}\sum_{i=|A^{\phi^L}(k^*)|-m+1}^{|A^{\phi^L}(k^*)|}\sqrt{i} \le 3+c\frac{|A^{\phi^L}(k^*)|^{\frac{3}{2}}}{|A^{\phi^L}(k^*)|} \le 3+c\sqrt{m}=O(\sqrt{m}),$$

   where c is a constant.
2. If $|A^{\phi^L}(k^*)|>m$, it follows that

$$\beta \le 3+\frac{1}{|A^{\phi^L}(k^*)|}\sum_{i=|A^{\phi^L}(k^*)|-m+1}^{|A^{\phi^L}(k^*)|}\sqrt{i} \le 3+\frac{m\sqrt{|A^{\phi^L}(k^*)|}}{|A^{\phi^L}(k^*)|} \le 3+\sqrt{m}=O(\sqrt{m}).$$

□

4.2 The $O(\sqrt{m})$-Competitive Ratio For the Throughput

In this subsection, we show our proposed algorithm has a $O(\sqrt{m})$-competitive ratio for the throughput by exploring the relation between our proposed algorithm and an online greedy assignment algorithm.

The online greedy assignment algorithm works as follows. It assigns job i to machine $un(i)$ when $F_i\cap S_i\ne\emptyset$ and assigns job i to machine $max(i)$ when $F_i\cap S_i=\emptyset$. We will show that the online greedy assignment algorithm has a 2-competitive ratio for the throughput in Lemma 4. Before introducing Lemma 4, we first introduce Lemma 3 used in Lemma 4.

Lemma 3 states that the throughput of an assignment ϕ can be calculated as the sum of the capacities of those machines which there is at least one job assigned to in the assignment ϕ.

Lemma 3. *Given a feasible assignment ϕ, we can calculate the throughput of ϕ as follows.*

$$U(\boldsymbol{B}_\phi) = \sum_{i=1}^{n} b_i = \sum_{k \in T} c_k,$$

where T denotes the set of machines with at least one job in the assignment ϕ, i.e., $T = \{k || A^\phi(k)| > 0, 1 \leq k \leq m\}$,

Proof. By Definition 2,

$$U(\boldsymbol{B}_\phi) = \sum_{i=1}^{n} b_i = \sum_{i=1}^{n} \frac{c_\phi(i)}{|A^\phi(\phi(i))|} = \sum_{i \in \mathcal{J}} \frac{c_\phi(i)}{|A^\phi(\phi(i))|}.$$

Since $J = \bigcup_{k=1}^{m} A^\phi(k)$,

$$\begin{aligned} U(\boldsymbol{B}_\phi) &= \sum_{i \in \bigcup_{k=1}^{m} A^\phi(k)} \frac{c_\phi(i)}{|A^\phi(\phi(i))|} \\ &= \sum_{1 \leq k \leq m, |A^\phi(k)| \neq 0} \sum_{i \in A^\phi(k)} \frac{c_{\phi(i)}}{|A^\phi(\phi(i))|} = \sum_{k \in T} \sum_{i \in A^\phi(k)} \frac{c_{\phi(i)}}{|A^\phi(\phi(i))|}. \end{aligned}$$

Note that $i \in A^\phi(k)$ means that $\phi(i) = k$,

$$U(\boldsymbol{B}_\phi) = \sum_{k \in T} \sum_{i \in A^\phi(k)} \frac{c_k}{|A^\phi(k)|} = \sum_{k \in T} |A^\phi(k)| \frac{c_k}{|A^\phi(k)|} = \sum_{k \in T} c_k.$$

□

Lemma 4 shows that the online greedy assignment algorithm results in a feasible assignment with a 2-competitive ratio for the throughput.

Lemma 4. *The online greedy assignment algorithm results in an assignment with $\alpha_1 = 2$ such that $\alpha_1 U(\boldsymbol{B}_g) \geq U(\boldsymbol{B}_{\phi^*})$, where $\boldsymbol{B}_g$ is the bandwidth vector of the assignment g derived by the online greedy assignment algorithm and $\boldsymbol{B}_{\phi^*}$ is the bandwidth vector of the optimal assignment.*

The proof of Lemma 4 is deferred to Appendix A.

In Lemma 5, we show our proposed algorithm has a $O(\sqrt{m})$-competitive ratio with respect to the throughput by exploring the relation between the proposed online algorithm and the online greedy assignment algorithm since the online greedy assignment algorithm results in a feasible assignment with a constant competitive ratio $\alpha_1 = 2$ with respect to the throughput.

Lemma 5. *The proposed online algorithm results in a feasible assignment ϕ with $\alpha_2 = O(\sqrt{m})$ such that $\alpha_2 U(\boldsymbol{B}_\phi) \geq U(\boldsymbol{B}_g)$, where $\boldsymbol{B}_\phi$ is the bandwidth vector of the assignment ϕ derived by the proposed algorithm and $\boldsymbol{B}_g$ is the bandwidth vector derived by the online greedy assignment algorithm.*

Proof. We begin by introducing notations. Let ϕ be the assignment derived by the proposed algorithm, and g be the assignment derived by the online greedy assignment algorithm. Let T denote the set of machines with at least one job in the assignment ϕ, and U denote the set of machines with at least one job in the assignment g. Let $V = \{max(i) | 1 \leq i \leq n\}$. By Lemma 3, we have

$$\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} = \frac{\sum_{i=1}^{n} \frac{c_{g(i)}}{|A^g(g(i))|}}{\sum_{i=1}^{n} \frac{c_{\phi(i)}}{|A^\phi(\phi(i))|}} = \frac{\sum_{p\in U} c_p}{\sum_{k\in T} c_k}.$$

Note that $V \subset T$ and $U \subset T \cup (U\backslash T)$,

$$\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} = \frac{\sum_{p\in U} c_p}{\sum_{k\in T} c_k} \leq \frac{\sum_{p\in T} c_p + \sum_{p\in U\backslash T} c_p}{\sum_{k\in T} c_k} = 1 + \frac{\sum_{p\in U\backslash T} c_p}{\sum_{k\in T} c_k}.$$

For each job $p \in U\backslash T$, the proposed algorithm assigns no job to p while the greedy assigns at least one job to p. Let p be a machine in the set $U\backslash T$, and i_p be a job in the set $A^g(p)$. Job i_p is assigned to $p = un(i_p) \neq max(i_p)$ in the online greedy assignment algorithm due to $F_{i_p} \cap S_{i_p} \neq \emptyset$ while i_p is assigned to $max(i_p)$ in the proposed algorithm due to $\sqrt{\gamma_{max(i_p),i_p}}c_p \leq c_{max(i_p)}$. Hence, $c_p \leq \frac{1}{\sqrt{\gamma_{k,i_p}}}c_k$ where $k = max(i_p)$ for some job $i_p \in A^g(p)$. Note that $\gamma_{max(i_p),i_p} \geq 2$ since $p = un(i_p) \neq max(i_p)$, which implies that machine $max(i_p)$ has been regarded as the most powerful machine in the feasible sets of at least two jobs when job i_p arrives. We have

$$\begin{aligned}\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} &\leq 1 + \frac{\sum_{p\in U\backslash T} c_p}{\sum_{k\in T} c_k} \leq 1 + \frac{\sum_{p\in U\backslash T} c_p}{\sum_{k\in V} c_k} \leq 1 + \frac{\sum_{p\in U\backslash T} \frac{1}{\sqrt{\gamma_{max(i_p),i_p}}} C_{max(i_p)}}{\sum_{k\in V} c_k} \\ &\leq 1 + \frac{\sum_{k\in V}\sum_{i=2}^{\gamma_{k,n}} \frac{1}{\sqrt{i}}c_k}{\sum_{k\in V} c_k} = \frac{\sum_{k\in V}\sum_{i=1}^{\gamma_{k,n}} \frac{1}{\sqrt{i}}c_k}{\sum_{k\in V} c_k}.\end{aligned}$$

Since there are only m machines,

$$\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} \leq= \frac{\sum_{k\in V}\sum_{i=1}^{\gamma_{k,n}} \frac{1}{\sqrt{i}}c_k}{\sum_{k\in V} c_k} \leq \frac{\sum_{k\in V}\sum_{i=1}^{m} \frac{1}{\sqrt{i}}c_k}{\sum_{k\in V} c_k}.$$

Furthermore,

$$\frac{\sum_{i=1}^{m} \frac{1}{\sqrt{i}}c_k}{c_k} = \sum_{i=1}^{m} \frac{1}{\sqrt{i}} = O(\sqrt{m}), \text{ for all } k \in V.$$

That is, for all $k \in V$, there exist constant δ, n_0 such that $\sum_{i=1}^{\gamma_{k,n}} \frac{1}{\sqrt{i}}c_k \leq \delta\sqrt{m}c_k$ for $m \geq m_0$. It follows that there exist constant $\delta' = \delta$, $m_0' = m_0$ such that

$$\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} \leq \frac{\sum_{k\in V}\sum_{i=1}^{\gamma_{k,n}} \frac{1}{\sqrt{i}}c_k}{\sum_{k\in V} c_k} \leq \frac{\sum_{k\in V} \delta\sqrt{m}c_k}{\sum_{k\in V} c_k} = \delta'\sqrt{m} \text{ for } m \geq m_0'.$$

Therefore, we conclude that $\frac{U(\boldsymbol{B}_g)}{U(\boldsymbol{B}_\phi)} = O(\sqrt{m})$. □

Finally, we can obtain Theorem 1.

Theorem 1. *The proposed online algorithm for our problem results in a feasible assignment ϕ with the bandwidth vector $\boldsymbol{B}_\phi$ and the load vector $\boldsymbol{L}_\phi$ such that $\alpha U(\boldsymbol{B}_\phi) \geq U(\boldsymbol{B}_{\phi'})$ for the bandwidth vector $\boldsymbol{B}_{\phi'}$ of all other feasible assignments ϕ' and $C(\boldsymbol{L}_\phi) \leq \beta C(\boldsymbol{L}_{\phi''})$ for the load vector $\boldsymbol{L}_{\phi''}$ of all other feasible assignments ϕ'', where $\alpha = O(\sqrt{m})$ and $\beta = O(\sqrt{m})$.*

Proof. By Lemma 2, Lemma 4 and Lemma 5, we obtain Theorem 1. □

5 Lower Bounds

We now show a lower bound of our problem in the offline setting , where an algorithm knows all the jobs with their feasible sets in advance. The problem in the offline setting must be easier than or equivalent to the problem in the online setting.

Theorem 2. *If an algorithm for this problem in the offline setting results in a feasible assignment ϕ with the bandwidth vector $\boldsymbol{B}_\phi$ and the load vector $\boldsymbol{L}_\phi$, such that $\alpha U(\boldsymbol{B}_\phi) \geq U(\boldsymbol{B}_{\phi'})$ for the bandwidth vector $\boldsymbol{B}_{\phi'}$ of all other feasible assignments ϕ', and $C(\boldsymbol{L}_\phi) \leq \beta C(\boldsymbol{L}_{\phi''})$ for the load vector $\boldsymbol{L}_{\phi''}$ of all other feasible assignments ϕ'', then $\alpha\beta = \Omega(\sqrt{m})$.*

Proof. We first construct a problem instance P as follows:

1. There are $n = m$ jobs with feasible sets $F_1 = \{1\}$ and $F_i = \{1, i\}$ for $2 \leq i \leq n$.
2. There are m machines with capacity $c_1 = \sqrt{m}$ and $c_i = 1$ for $2 \leq i \leq m$.

Given an algorithm D with α-competitive ratio for throughput and β-competitive ratio for total load, let ϕ_D denote the assignment generated by algorithm D for the constructed problem instance P, and x denote the number of machines that are assigned at least one job in the assignment ϕ_D, i.e., $x = |\{k||A^{\phi_D}(k)| > 0, 1 \leq k \leq m\}|$.

We obtain the throughput $U(\boldsymbol{B}_{\phi_D})$ and total load $C(\boldsymbol{L}_{\phi_D})$ of ϕ_D as follows:

$$U(\boldsymbol{B}_{\phi_D}) = \sqrt{m} + (x-1) \text{ and } C(\boldsymbol{L}_{\phi_D}) \geq \frac{m-x+1}{\sqrt{m}} + (x-1).$$

We also construct two assignments, ϕ' and ϕ'' as follows:

1. In ϕ', each job i is assigned to machine i.
2. In ϕ'', all jobs is assigned to machine 1.

The throughput $U(\boldsymbol{B}_{\phi'})$ of ϕ' and the total load $C(\boldsymbol{L}_{\phi''})$ of ϕ'' are as follows:

$$U(\boldsymbol{B}_{\phi'}) = \sum_{i=1}^{n} c_i = \sqrt{m} + (m-1) \text{ and } C(\boldsymbol{L}_{\phi''}) = \frac{m}{\sqrt{m}} = \sqrt{m}.$$

It follows that

$$\alpha \geq \frac{U(\boldsymbol{B}_{\phi'})}{U(\boldsymbol{B}_{\phi_D})} = \frac{\sqrt{m} + (m-1)}{\sqrt{m} + (x-1)}$$

and

$$\begin{aligned}\beta &\geq \frac{C(\boldsymbol{L}_{\phi_D})}{C(\boldsymbol{L}_{\phi''})} \geq \frac{\frac{m-x+1}{\sqrt{m}} + (x-1)}{\sqrt{m}} \\ &= \frac{m-x+1}{m} + \frac{(x-1)}{\sqrt{m}} = 1 - \frac{x}{m} + \frac{1}{m} + \frac{\sqrt{m}(x-1)}{m} \\ &= 1 + \frac{\sqrt{m}x + 1 - x - \sqrt{m}}{m}.\end{aligned}$$

Since $1 \leq x \leq m = n$, we consider two cases where $1 \leq x < \sqrt{m}$ and $\sqrt{m} \leq x \leq m$.

1. if $1 \leq x < \sqrt{m}$, we have

$$\alpha \geq \frac{\sqrt{m} + (m-1)}{\sqrt{m} + (x-1)} \geq \frac{\sqrt{m} + (m-1)}{2\sqrt{m} - 1} \geq \frac{m}{2\sqrt{m}} = \frac{\sqrt{m}}{2}$$

and

$$\beta \geq 1 + \frac{\sqrt{m}x + 1 - x - \sqrt{m}}{m} \geq 1,$$

which implies that $\alpha\beta = \Omega(\sqrt{m})$.

2. if $\sqrt{m} \leq x \leq m$, we have

$$\alpha \geq \frac{\sqrt{m} + (m-1)}{\sqrt{m} + (x-1)} \geq \frac{\sqrt{m} + (m-1)}{2x - 1} \geq \frac{m}{2x}$$

and

$$\beta \geq 1 + \frac{\sqrt{m}x + 1 - x - \sqrt{m}}{m} \geq 1 + \frac{\sqrt{m}x + 1 - m - \sqrt{m}}{m} \geq \frac{\sqrt{m}x - \sqrt{m}}{m}.$$

Then

$$\alpha\beta \geq \frac{m}{2x}\frac{\sqrt{m}x - \sqrt{m}}{m} = \frac{\sqrt{m}}{2} - \frac{\sqrt{m}}{2x} \geq \frac{\sqrt{m}}{2} - \frac{1}{2},$$

which implies $\alpha\beta = \Omega(\sqrt{m})$ also.

□

Note that Theorem 2 also implies a lower bound $\alpha\beta = \Omega(\sqrt{m})$ of the problem in the online setting.

References

1. J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, "On-line routing of virtual circuits with applications to load balancing and machine scheduling," in Journal of the ACM, Vol. 44, No. 3, pp. 486-504, May 1997.
2. Y. Azar, L. Epstein, Y. Richter and G. J. Woeginger, "All-norm approximation algorithms," in Journal of Algorithms, Vol. 52, No. 2, pp. 120-133, August 2004.
3. Y. Azar, J. Naor, and R. Rom, "The competitiveness of on-line assignments," in Journal of Algorithms, Vol. 18, No. 2, pp. 221-237, March 1995.

4. J.A. Aslam, A. Rasala, C. Stein, and N. Young, "Improved bicriteria existence theorems for scheduling," in Proceedings of the 10th annual ACM-SIAM symposium on Discrete algorithms, pp. 846-847, January 1999.
5. J.A. Bondy and U.S.R. Murty, "Graph Theory with Applications," Elsevier North-Holland, 1976.
6. N. Buchbinder and J. Naor, "Fair online load balancing," in Proceedings of the 18th annual ACM symposium on Parallelism in algorithms and architectures, 2006.
7. N. Buchbinder and J. Naor, "Improved Bounds for Online Routing and Packing Via a Primal-Dual Approach," in 47th Annual IEEE Symposium on Foundations of Computer Science, 2006.
8. I. Caragiannis, "Better bounds for online load balancing on unrelated machines," in Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms, 2008.
9. S. Cho and A. Goel, "Pricing for fairness: distributed resource allocation for multiple objectives," in Proceedings of the 38th ACM Symposium on Theory of Computing, pp. 197-204, May 2006.
10. A. Goel and A. Meyerson, "Simultaneous optimization via approximate majorization for concave profits or convex costs," in Algorithmica, Vol. 44, No. 4, pp.301-323, May 2006.
11. A. Goel, A. Meyerson, and S. Plotkin, "Approximate majorization and fair online load balancing," in ACM Transactions on Algorithms, Vol. 1, No. 2, pp. 338-349, 2005.
12. A. Goel, A. Meyerson, and S. Plotkin, "Combining fairness with throughput: online routing with multiple objectives," in Journal of Computer and System Sciences, Vol. 63, No. 1, pp. 62-79, August 2001.
13. A. Goel and H. Nazerzadeh, "Price based protocols for fair resource allocation: convergence time analysis and extension to Leontief utilities," in Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms, 2008.
14. J. Kleinberg, E. Tardos, and Y. Rabani, "Fairness in routing and load balancing," in Proceedings of the 40th Annual Symposium on Foundations of Computer Science, October 1999.
15. A. Kumar , J. Kleinberg, "Fairness measures for resource allocation," in Proceedings of the 41st Annual Symposium on Foundations of Computer Science, November 2000.
16. R. K. Lain, D.-M. Chiu, and W. Howe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems," in DEC Res. Rep. TR-301, 1984.
17. C. Stein and J. Wein, "On the existence of schedules that are near-optimal for both makespan and total weighted completion time," Technical Report TR96-295, 1996.

A Proof of Lemma 4

Before giving the proof of Lemma 4, we introduce some definitions about the bipartite maximum-weight perfect matching problem (or matching problem for simplicity) described in [5].

Definition 5. *Given $G = (V, E)$, a subset M of E is a matching in G if its elements are links and no two are adjacent in G; the two ends of an edge in M are said to be matched under M.*

Definition 6. *A matching M saturates a vertex v, which is said to be M-saturated if some edge of M is incident with it; otherwise, v is M-unsaturated.*

Definition 7. *If every vertex of G is M-saturated, the matching M is perfect.*

Definition 8 (the bipartite maximum-weight perfect matching problem). *Consider a weighted complete bipartite graph with a bipartition (X, Y), where $X = \{1, 2, \cdots, n\}, Y = \{1, 2, \cdots, n\}$ and each edge (i, j) has weight $w_{ij} = w((i, j))$. Find a maximum-weight perfect matching M in this weighted graph, i.e., find an perfect matching M such that*

$$U_{MP}(M) \geq U_{MP}(M'), \text{ where } U_{MP}(M) = \sum_{(i,j)\in M} w_{ij}.$$

for all other matchings M'. We refer to such a matching as an optimal matching.

Proof of Lemma 4.
We first show that any instance of the offline version of our problem can be converted into an instance of the matching problem.

Given an instance of the offline version of our problem, i.e., the jobs' index set $\mathcal{J} = \{1, 2, \cdots, n\}$ with a feasible set F_i for each job $i \in \mathcal{J}$ and the machines' index set $\mathcal{M} = \{1, 2, \cdots, m\}$ with $c_1, c_2, \cdots, c_m$, we construct an instance of the matching problem as follows: Construct a weighted complete bipartite graph with a bipartition (X, Y), where $X = \{1, 2, \cdots, n+m\}$ and $Y = \{1, 2, \cdots, n+m\}$ and each edge (i, j) with weight

$$w_{ij} = \begin{cases} c_j, & \text{if } i \leq n \text{ and } j \in F_i, \\ 0, & \text{otherwise.} \end{cases}$$

Next, we show how to construct a solution ϕ_M for our problem instance from the solution M of the constructed matching problem instance. Given a perfect matching M of the constructed bipartite graph, we define an assignment $\phi_M : \{1, 2, \cdots, n\} \to \{1, 2, \cdots, m\}$ of the original job assignment problem instance as follows:

$$\phi_M(i) = \begin{cases} j, & \text{if } (i, j) \in M, j \leq m \text{ and } w_{ij} > 0, \\ max(i), & \text{if } (i, j) \in M \text{ and } j > m, \\ max(i), & \text{if } (i, j) \in M \text{ and } w_{ij} = 0, \end{cases}$$

where $max(i)$ is the index of the machine with the most capacity in the feasible set F_i of job i. In the first case, $\phi_M(i) = j$ is in the feasible set F_i since $w_{ij} > 0$, and in the other cases, $\phi_M(i) = max(i)$ is also in the feasible set F_i. Therefore, the constructed ϕ_M is feasible.

Note that $w_{ij} = c_j$ if $i \leq n$ and $j \in F_i$ and $w_{ij} = 0$ else, in the constructed matching problem instance. We can see that $U(\boldsymbol{B}_{\phi_M}) \geq U_{MP}(M)$ since $U(\boldsymbol{B}_{\phi_M}) \geq \sum_{(i,j)\in M, i\leq n, j\leq m} c_j$ and $U_{MP}(M) = \sum_{(i,j)\in M} w_{ij} = \sum_{(i,j)\in M, i\leq n, j\in F_i} c_j$.

Given an assignment ϕ, we now show how to construct a matching M_ϕ of the constructed bipartite graph as follows:

1. Let $R_\phi = \{(i, j) | \phi(i) = j\}$ initially.

2. If there is more than one edge incident with the same vertex $j \in Y$ in R_ϕ, we keep only one of them selected arbitrarily and remove the others from R_ϕ.
3. Add edge (i, j) into M_ϕ if $(i, j) \in R_\phi$.
4. Add edge $(n + m + 1 - j, n + m + 1 - i)$ into M_ϕ if $(i, j) \in R_\phi$.
5. Add edge $(i, n + m + 1 - i)$ into M_ϕ for vertex $i \in X$ is M_ϕ-unsaturated.

We show that the constructed M_ϕ is a perfect matching. It is not hard to see that after steps 3-5, all $i \in X$ are M_ϕ-saturated. We only need to show that vertices $j \in Y$ selected in steps 3-5 are not the same. It is clear that the vertex $j \in Y$ selected in step 3 will not be selected as vertex $n + m + 1 - i$ in step 4 since j is ranged from 1 to m in step 3 and $n + m + 1 - i$ is ranged from $m + 1$ to $n + m$ in step 4.

Also, the selected $n+m+1-i \in Y$ in step 4 cannot be selected as $n+m+1-i \in Y$ in step 5. If $n + m + 1 - i \in Y$ is selected in step 4 due to $(i, j) \in R_\phi$, then (i, j) must have been selected in step 3, which implies that vertex i is not M-unsaturated at step 5. Therefore, $n + m + 1 - i$ will not be selected in step 5.

Finally, the vertex $j \in Y$ selected in step 3 will not be selected as vertex $n + m + 1 - i$ in step 5. If a vertex j selected in step 3 is also selected in step 5, i.e., $j = n + m + 1 - i$, i is $n + m + 1 - j$ in step 5. However, $n + m + 1 - j$ must have been selected in step 4 since j is selected in step 3, which contradicts to that i is M_ϕ-unsaturated in step 5. Therefore, the constructed matching M_ϕ is a perfect matching.

Next, we show that $U_{MP}(M_\phi) = U(\boldsymbol{B}_\phi)$. After step 3, $M_\phi = R_\phi$ and we have that $U_{MP}(M_\phi) = \sum_{(i,j)\in M_\phi} w_{ij} = \sum_{(i,j)\in R_\phi} w_{ij}$. By Lemma 3 and $w_{ij} = c_j$ if $i \leq n$ and $j \in F_i$,

$$U_{MP}(M_\phi) = \sum_{(i,j)\in R_\phi} w_{ij} = \sum_{(i,j)\in R_\phi} c_j = \sum_{j\in T} c_j = U(\boldsymbol{B}_\phi),$$

where $T = \{j||A^\phi(j)| \neq 0, 1 \leq j \leq m\}$.

Also the weight of the edges added in steps 4-5 are all 0. In step 4, the weight of each added edge is 0, i.e., $w_{n+m+1-j,n+m+1-i} = 0$ since, for all $(i, j) \in R_\phi$, $1 \leq j \leq m$, and $n + 1 \leq n + m + 1 - j \leq n + m$. In step 5, the weight of each added edge is 0, i.e., $w_{i,n+m+1-i} = 0$. This is because, if $1 \leq i \leq n$, then $n+m+1-i > m$ and $w_{i,n+m+1-i} = 0$. If $n+1 \leq i \leq n+m$, then $w_{i,n+m+1-i} = 0$. Therefore, $U_{MP}(M_\phi) = U(\boldsymbol{B}_\phi)$.

Given an optimal matching M^*, we show that $U(\boldsymbol{B}_{\phi^*}) \geq U(\boldsymbol{B}_\phi)$ for all other assignments ϕ, where the assignment ϕ^* is constructed from M^*. We prove it by contradiction. Assume there is an assignment ϕ resulting in a bandwidth vector $\boldsymbol{B}_\phi$ with $U(\boldsymbol{B}_\phi) > U(\boldsymbol{B}_{\phi^*})$. We have $U_{MP}(M_\phi) = U(\boldsymbol{B}_\phi) > U(\boldsymbol{B}_{\phi^*}) \geq U_{MP}(M^*)$, which contradicts the fact that M^* is an optimal matching. Hence, we conclude that $U(\boldsymbol{B}_{\phi^*})$ is the maximum.

The throughput of the constructed assignment ϕ^* is equal to the weight of the optimal matching M^*. This is because if $U(\boldsymbol{B}_{\phi^*}) > U_{MP}(M^*)$, we can construct a matching M_{ϕ^*} from the assignment ϕ^* with $U_{MP}(M_{\phi^*}) = U(\boldsymbol{B}_{\phi^*}) > U_{MP}(M^*)$. It leads to a contradiction. Hence $U(\boldsymbol{B}_{\phi^*}) = U_{MP}(M^*)$.

We consider an online greedy matching algorithm which always chooses the edge with the most weight that has not been chosen before. Let M' denote the matching derived by the online greedy matching algorithm, and g denote the assignment derived by the online greedy assignment algorithm. We show that $U_{MP}(M') \leq U(\boldsymbol{B}_g)$. We first construct a feasible assignment $\phi_{M'}$ from the perfect matching M' with $U_{MP}(M') \leq U(\boldsymbol{B}_{\phi_{M'}})$. Recall that

$$\phi_{M'}(i) = \begin{cases} j, & \text{if } (i,j) \in M', j \leq m \text{ and } w_{ij} > 0, \\ max(i), & \text{if } (i,j) \in M' \text{ and } j > m, \\ max(i), & \text{if } (i,j) \in M' \text{ and } w_{ij} = 0. \end{cases}$$

In the first case, if $(i,j) \in M', j \leq m, w_{ij} > 0$, and the online greedy matching algorithm chooses the edge with the most weight $w_{ij} = w_{iun(i)} = c_{un(i)}$, which implies $F_i \cap S_i \neq \emptyset$ and $\phi_{M'}(i) = j = un(i) = g(i)$. In the other cases, the online greedy matching algorithm chooses the edge with the most weight $w_{ij} = 0$, which implies that $F_i \cap S_i = \emptyset$ and $\phi_{M'}(i) = j = max(i) = g(i)$. Hence, the assignment g equals to $\phi_{M'}$, i.e., $g(i) = \phi_{M'}(i)$ for all $1 \leq i \leq n$. Therefore, we have $U_{MP}(M') \leq U(\boldsymbol{B}_{\phi_{M'}}) = U(\boldsymbol{B}_g)$.

In the following, we show that $U_{MP}(M^*) \leq 2U_{MP}(M')$. If we consider M^* as a permutation $f^* : \{1, 2, \cdots, n+m\} \to \{1, 2, \cdots, n+m\}$, defined as $f^*(i) = j,$ if $(i,j) \in M^*$, and M' as a permutation $f' : \{1, 2, \cdots, n+m\} \to \{1, 2, \cdots, n+m\}$, defined as $f'(i) = j,$ if $(i,j) \in M'$, then we have

$$\begin{aligned} U_{MP}(M^*) - U_{MP}(M') = & \sum_{w_{if^*(i)} > w_{if'(i)}} (w_{if^*(i)} - w_{if'(i)}) \\ & + \sum_{w_{if^*(i)} = w_{if'(i)}} (w_{if^*(i)} - w_{if'(i)}) \\ & + \sum_{w_{if^*(i)} < w_{if'(i)}} (w_{if^*(i)} - w_{if'(i)}) \\ \leq & \sum_{w_{if^*(i)} > w_{if'(i)}} (w_{if^*(i)} - w_{if'(i)}) \\ \leq & \sum_{w_{if^*(i)} > w_{if'(i)}} (w_{if^*(i)}) \end{aligned}$$

Recall that the greedy matching algorithm chooses the edge with most weight that has not been chosen before. For each edge $(i, f'(i))$ where $w_{if^*(i)} > w_{if'(i)}$, there must be an edge $(i', f^*(i)) \in M'$ with the same weight as $w_{if^*(i)}$; otherwise, the online greedy matching algorithm chooses the edge $(i, f^*(i))$ with $w_{if^*(i)} > w_{if'(i)}$ instead of $(i, f'(i))$. Therefore,

$$\begin{aligned} U_{MP}(M^*) - U_{MP}(M') \leq & \sum_{w_{if^*(i)} > w_{if'(i)}} (w_{if^*(i)}) \\ \leq & \; U_{MP}(M'). \end{aligned}$$

We then obtain

$$U_{MP}(M^*) \leq 2U_{MP}(M')$$

and

$$U(\boldsymbol{B}_{\phi^*}) = U_{MP}(M^*) \leq 2U_{MP}(M') \leq 2U(\boldsymbol{B}_g)$$

as required. $\square$