Computers and Operations Research 116 (2020) 104868

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Online scheduling of jobs with favorite machines

Cong Chen[a,*], Paolo Penna[b], Yinfeng Xu[c,d]

[a] School of Business Administration, South China University of Technology, Guangzhou, China
[b] Department of Computer Science, ETH Zurich, Zurich, Switzerland
[c] School of Management, Xi'an Jiaotong University, Xi'an, China
[d] The State Key Lab for Manufacturing Systems Engineering, Xi'an, China

A R T I C L E I N F O

Article history:
Received 6 July 2019
Revised 21 October 2019
Accepted 12 December 2019
Available online 17 December 2019

Keywords:
Scheduling
Online scheduling
Competitive ratio
Favorite machines
Unrelated machines

A B S T R A C T

A long distance link may cause high data latency in cloud computing systems, and thus a computing task may be processed faster by nearby servers than distant servers. To address this class of job (task) scheduling problems, we propose the *favorite machine* model. Specifically, we are interested in the online version where jobs arrive *one by one* and must be allocated irrevocably upon each arrival without knowing the future jobs. The objective is to design efficient online algorithms for allocating jobs in order to minimize the *makespan*.

Theoretical performance guarantees are presented for the Greedy algorithm and the Assign-U algorithm, where the latter is shown to be the best-possible online algorithm for this problem. Our theoretical results generalize the results for several classical problems, e.g. the *unrelated* machines and the *identical* machines. We also study a restriction of the model, called the *symmetric favorite machine* model. A 2.675-competitive algorithm is developed and proved to be the best-possible algorithm for the two machines case. Moreover, computational results show that the algorithms perform quite well for random instances, and reveal some insights for choosing algorithms for practical applications.

1. Introduction

Online scheduling on the *unrelated* machines is a classical and well-studied problem. In this problem, there are n jobs that need to be processed by one of m different machines. The time to process a job changes from machine to machine, and the goal is to allocate all jobs so as to minimize the *makespan*, that is, the maximum load over the machines. The load of a machine is the sum of the processing times of the jobs allocated to that machine. The jobs arrive *one by one* and must be assigned to one machine upon their arrival, without knowing the future jobs. Since it is generally impossible to guarantee an optimal allocation, the algorithms for online problems are evaluated through the *competitive ratio*. An online algorithm whose competitive ratio is $\rho \geq 1$ guarantees an allocation whose makespan is at most ρ times the optimal makespan.

Online algorithms have a rather “bad” performance in terms of the competitive ratio for scheduling on *unrelated* machines. For example, the simple Greedy algorithm[1] has a competitive ratio of m, the number of machines; the best-possible online algorithm achieves a competitive ratio of $\Theta(\log m)$. However, some well-known restrictions (e.g., the *identical* machines and *related* machines) admit a much better performance, that is, a constant competitive ratio.

In fact, many practical problems are neither as simple as the identical (or related) machines cases, nor so complicated as the general unrelated machines. In a sense, practical problems are somewhat “intermediate” as the following examples suggest.

Example 1 (Two types of machines (Vakhania et al., 2014))**.** Considering the production of spare parts for cars, the manufacturer may decide to use the machines for the production of spare part 1 to produce spare part 2, and vice versa. The machine for spare part 1 (spare part 2, respectively) takes time p for part 1 (part 2, respectively), but it can also manage to produce part 2 (part 1, respectively) in time $q > p$.

Example 2 (CPU-GPU cluster (Chen et al., 2014))**.** A graphics processing unit (GPU) has the ability to handle various tasks more efficiently than the central processing unit (CPU). These tasks include video processing, image analysis, and signal processing. Nevertheless, the CPU is still more suitable for a wide range of other tasks. A heterogeneous CPU-GPU system consists of a set M_1 of GPU processors and a set M_2 of CPU processors. The processing

* Corresponding author.
E-mail address: chencong@scut.edu.cn (C. Chen).

[1] This algorithm assigns each job to a machine whose load after this job assignment is minimized.

https://doi.org/10.1016/j.cor.2019.104868
0305-0548/© 2019 Elsevier Ltd. All rights reserved.

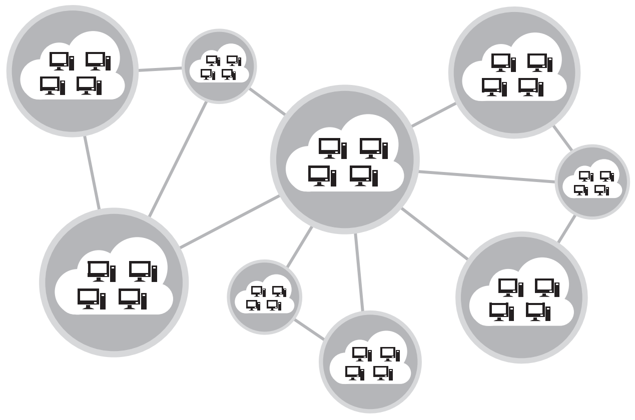


Fig. 1. Clusters of servers in cloud computing system.

time of job j is p_{j1} on a GPU processor and p_{j2} on a CPU processor. Therefore, some jobs are more suitable for GPU and others for CPU.

Example 3 (Cloud computing)**.** The servers of a cloud computing system may consist of many clusters of computers located worldwide (see Fig. 1). While the computing tasks may also arise from worldwide customers. Due to a long distance link between a sever and a task, the problem of data latency may occur. Consequently, the speed for a server to process a task is location-dependent. In other words, a task can be processed faster by nearby servers, and different tasks have different nearby servers. Since a cluster often have many computers with the same configuration in cloud computing system, each task may have several nearby servers.

Inspired by these examples, we consider the general unrelated machines case by observing that each job in the system has a certain set of *favorite* machines which correspond to the shortest processing time for this particular job. Take Example 3 for instance, the nearby severs of a task are the favorite machines of this task.

1.1. Our contributions and connections with prior work

We study the online scheduling problem on what we call the *favorite machine* model. Denote the processing time of job j on machine i by p_{ji} and the minimum processing time of job j by $p_j = \min_i p_{ji}$. Thus the set of favorite machines of job j is defined as $F_j = \{i|\ p_{ji} = p_j\}$ and the favorite machine model is as follows:

(*f-favorite machines*) This model is simply the unrelated machine setting when every job j has at least f favorite machines ($|F_j| \geq f$). The processing time of job j on any *favorite* machine $i \in F_j$ is p_j, and on any *non-favorite* machine $i \notin F_j$ is an arbitrary value $p_{ji} > p_j$.

This model is motivated by the task scheduling problem in cloud computing systems described in Example 3. Besides, the model also captures the features of other practical scheduling problems. For example, the *two types of machines* problems mentioned in Examples 1 and 2, workers with different levels of proficiency for different jobs in manufacturing and so on.

It is worth noting that this model interpolates between the *unrelated* machines where possibly only one machine has minimal processing time for the job ($f = 1$) and the *identical* machines case ($f = m$). The f-favorite machines setting can also be seen as a "relaxed" version of *restricted assignment* problem where each job j can be allocated only to a subset F_j of machines: the restricted assignment problem is essentially the case where the processing time of a job on a non-favorite machine is always ∞.

We first discuss the performance of Greedy for our problem, since Greedy is widely used in practical problems and often with a provable good performance. However, the results for Greedy show that it is not optimal. Thus we carry on exploring the optimal algorithm and find out that the Assign-U algorithm is optimal for our problem. For the Greedy algorithm, we obtain tight bound $\frac{m+f-1}{f}$ on the competitive ratio, which generalizes the well-known bounds on the competitive ratio of Greedy for unrelated machines ($f = 1$) and identical machines ($f = m$), that is, m and $2 - \frac{1}{m}$, respectively. For the Assign-U algorithm, we show that the competitive ratio is $\Theta(\log \frac{m}{f})$, which matches the lower bound $\Omega(\log \frac{m}{f})$ we proved for this problem.

Experimental evaluation Although the theoretical bounds on competitive ratio are non constant, the experimental results (Section 5) show that the algorithms behave quite well in terms of the average cases. The results also reveal some insights for practical problems: (1) The Assign-U performs better than the Greedy to process a large number of jobs which is consistent with the theoretical results. (2) A modified Assign-U algorithm (i.e., Assign-U*) outperforms both the Greedy and the Assign-U in almost any circumstance. (3) All the algorithms perform better when f gets larger, especially increasing f from 1 to 2. Therefore, the system cost can be distinctly improved by providing "several" good choices for each job. (4) If f is large enough (close to identical machines), the simple Greedy is almost as good as the Assign-U *. Thus, for scheduling systems that are close to identical machines, the simple Greedy is enough.

Special cases and the impact of "speed ratio" Note that whenever $f = \Theta(m)$, the competitive ratio is constant asymptotically. In particular, for $f = \frac{m}{2}$, Greedy has a competitive ratio of $3 - \frac{2}{m}$. To get further results and analysis the impact of "speed ratio", we consider the following restriction of the model above where a finer analysis is possible.

(*symmetric $\frac{m}{2}$-favorite machines*) All machines are partitioned into *two groups* of equal size $\frac{m}{2}$, and each job has favorite machines as exactly one of the two groups. Moreover, the processing time on non-favorite machines is s times that on favorite machines, where $s \geq 1$ is the "speed ratio" between favorite and non-favorite machines.

We show that the competitive ratio of Greedy is at most $\min\{1 + (2 - \frac{2}{m})\frac{s^2}{s+1},\ \ s + (2 - \frac{2}{m})\frac{s}{s+1},\ \ 3 - \frac{2}{m}\}\ (< 3)$. A modified greedy algorithm, called GreedyFavorite, which assigns each job greedily but only among its favorite machines, has a competitive ratio of $2 - \frac{1}{f} + \frac{1}{s}$ (< 3). As one can see, the Greedy is better than GreedyFavorite for smaller s, and GreedyFavorite is better for larger s. Thus we can combine the two algorithms to obtain a better algorithm (GGF) with a competitive ratio of at most $\min\{2 + \frac{s^2+s-2}{s+1}, 2 + \frac{1}{s}\}$ (≤ 2.675). Indeed, we characterize the *optimal* competitive ratio for the two machines case. That is, for *symmetric 1-favorite machines*, we show that the GGF algorithm is $\min\{1 + \frac{s^2}{s+1}, 1 + \frac{1}{s}\}$-competitive and there is a matching lower bound.

For this problem, our results show the *impact of the speed ratio* s on the competitive ratio. This is interesting because this problem generalizes the case of *two related* machines (Epstein et al., 2001, for which s is the speed ratio between the two machines (see Fig. 3). Fig. 2 shows the "structural" differences between the two problems, and why ours is a generalization. The two machines case has also been studied earlier from a game theoretic point of view and compared to the two related machines (Chen et al., 2017; Epstein, 2010).

Relations with prior work As mentioned above, the *f-favorite machines* is a special case of the *unrelated machines* and a general case of the *identical machines*. The *symmetric 1-favorite machines* is a generalization of the *2 related machines*. Besides, there are several other "intermediate" problems that have connections with our models and results in the literature (see Fig. 4 for an overview and Section 1.2 for details on prior results).

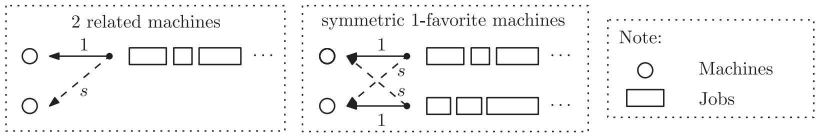


Fig. 2. Relationship between two related machines and the symmetric 1-favorite machines setting.

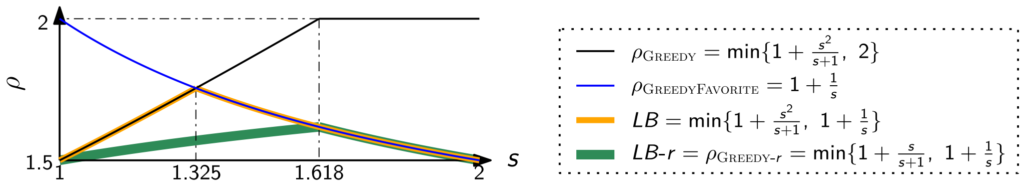
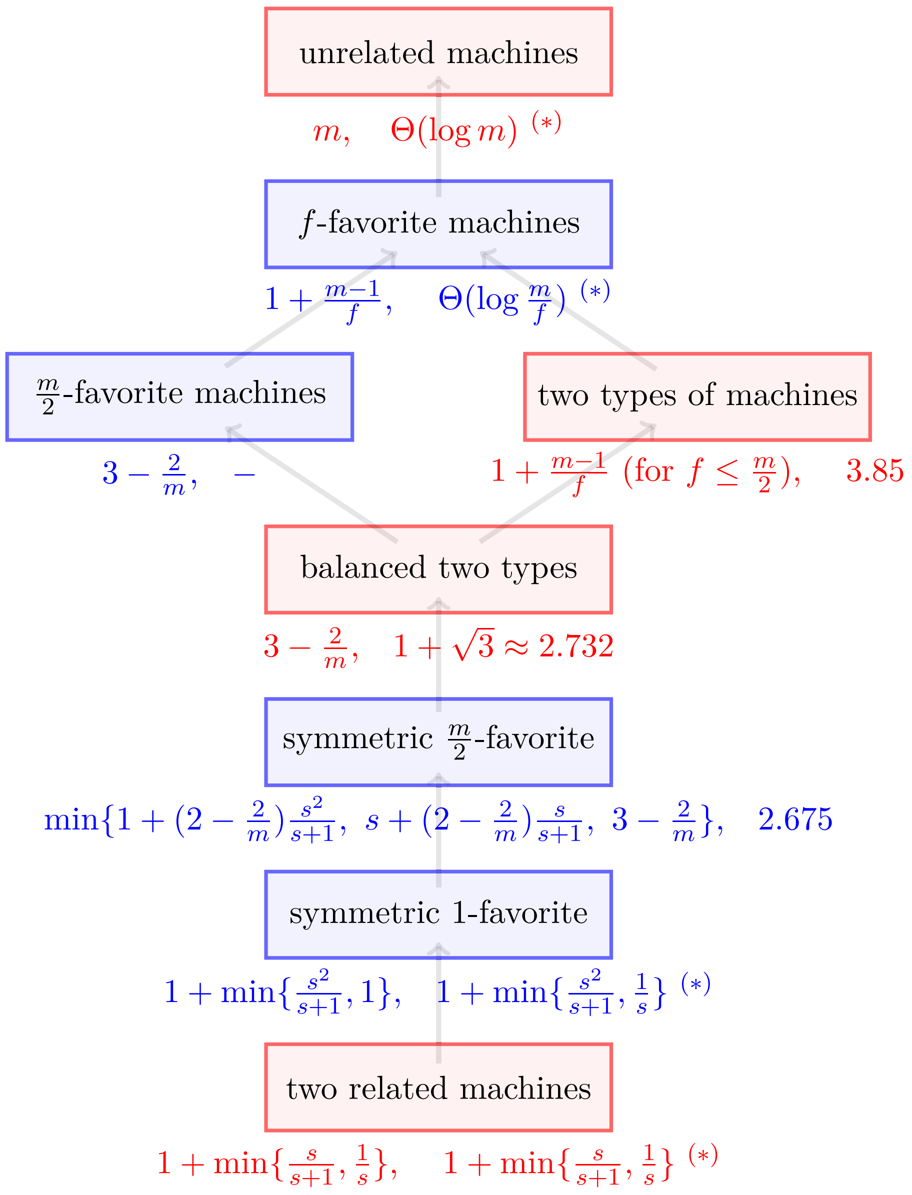


Fig. 3. The competitive ratio for the 2-machine case (related machines vs our model).



unrelated machines
$m, \quad \Theta(\log m)\ ^{(*)}$
f-favorite machines
$1 + \frac{m-1}{f}, \quad \Theta(\log \frac{m}{f})\ ^{(*)}$
$\frac{m}{2}$-favorite machines
$3 - \frac{2}{m}, \quad -$
two types of machines
$1 + \frac{m-1}{f}$ (for $f \le \frac{m}{2}$), $\quad 3.85$
balanced two types
$3 - \frac{2}{m}, \quad 1 + \sqrt{3} \approx 2.732$
symmetric $\frac{m}{2}$-favorite
$\min\{1 + (2 - \frac{2}{m})\frac{s^2}{s+1},\ s + (2 - \frac{2}{m})\frac{s}{s+1},\ 3 - \frac{2}{m}\}, \quad 2.675$
symmetric 1-favorite
$1 + \min\{\frac{s^2}{s+1}, 1\}, \quad 1 + \min\{\frac{s^2}{s+1}, \frac{1}{s}\}\ ^{(*)}$
two related machines
$1 + \min\{\frac{s}{s+1}, \frac{1}{s}\}, \quad 1 + \min\{\frac{s}{s+1}, \frac{1}{s}\}\ ^{(*)}$

Fig. 4. Comparison between prior results (in red) and our results (in blue). The two bounds below each problem are the competitive ratios for Greedy and best-known algorithm, respectively, and the "(*)" mark represent the optimality of the best-known algorithm. Arrows go from a problem to a more general one, and therefore the upper bounds for the general problem apply to the special one as well. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Specifically, in the *two types of machines* case (Chen et al., 2014; Imreh, 2003), they have two sets of machines, M_1 and M_2, and the machines in each set are identical. Each job j has processing time p_{j1} for any machine in M_1, and p_{j2} for any machine in M_2. The so-called *balanced* case is the restriction in which the numbers of machines in the two sets are equal. Note that our $\frac{m}{2}$-favorite machines model is a generalization of the balanced case, because each job has either M_1 or M_2 as favorite machines, and thus $f = |M_1| = |M_2| = \frac{m}{2}$. The symmetric $\frac{m}{2}$-favorite machines case is the restriction of the balance case in which the processing time on non-favorite machines is s times that of favorite ones, i.e., either $p_{j1} = s \cdot p_{j2}$ or $p_{j2} = s \cdot p_{j1}$.

1.2. Related work

We have discussed the relations between several classical models with our models. The detailed results for related work will be reviewed in the following.

Identical machines. For m identical machines, the Greedy algorithm has a competitive ratio exactly $2 - 1/m$ (Graham, 1966), and this is optimal for $m = 2, 3$ (Faigle et al., 1989). For arbitrary larger m, better online algorithms exist and the bound is still improving (Albers, 1999; Bartal et al., 1995; Karger et al., 1996). Till now the best-known upper bound is 1.9201 (Fleischer and Wahl, 2000) and the lower bound is 1.88 (Rudin III, 2001).

Related machines (uniform processors). The competitive ratio of Greedy is $(1+\sqrt{5})/2 \approx 1.618$ for two machines ($m = 2$), and it is at most $1 + \sqrt{2m-2}/2$ for $m \geq 3$. The bounds are tight for $m \leq 6$ (Cho and Sahni, 1980).

When the number of machines m becomes larger, the Greedy algorithm is far from optimal, as its competitive ratio is $\Theta(\log m)$ for arbitrary m (Aspnes et al., 1997). Aspnes et al. (1997) also devise the Assign-R algorithm with the first constant competitive ratio of 8 for related machines. The constant is improved to 5.828 by Berman et al. (2000).

Some research focuses on the dependence of the competitive ratio on speed s when m is rather small. For the 2 related machines case, Greedy has a competitive ratio of $1 + \min\{\frac{s}{s+1}, \frac{1}{s}\}$ and there is a matching lower bound (Epstein et al., 2001).

Unrelated machines. As to the unrelated machines, Aspnes et al. (1997) show that the competitive ratio of Greedy is rather large, namely m. In the same work, they present the algorithm Assign-U with a competitive ratio of $\mathcal{O}(\log m)$. A matching lower bound is given by Azar et al. (1992) in the problem of online restricted assignment.

Restricted assignment. The online restricted assignment problem is also known as online scheduling subject to arbitrary machine eligibility constraints. Azar et al. (1992) show that Greedy has a competitive ratio less than $\lceil \log_2 m \rceil + 1$, and there is a matching lower bound $\lceil \log_2(m+1) \rceil$. For other results of online scheduling with machine eligibility we refer the reader to the survey by Lee et al. (2013) and references therein.

Two types of machines (CPU-GPU cluster, hybrid systems). Imreh (2003) proves that Greedy algorithm is $(2 + \frac{m_1 - 1}{m_2})$-competitive for this case, where m_1 and m_2 ($\leq m_1$) are the number of two sets of machines, respectively. In our terminology, $f = m_2$ and $m_1 = m - f$, meaning that Greedy is $(1 + \frac{m-1}{f})$-competitive for $f \leq \frac{m}{2}$. The same work also improves the bound to $4 - \frac{2}{m_1}$ with a modified Greedy algorithm. Chen et al. (2014) gives a 3.85-competitive algorithm for the problem, and a simple 3-competitive algorithm and a more involved $1 + \sqrt{3} \approx 2.732$-competitive algorithm for the *balanced case*, that is, the case $m_1 = m_2$.

Further models and results. Some works consider other restrictions (Shabtay and Karhi, 2012; Vakhania et al., 2014; Zhang and Wirth, 2009). Specifically, Vakhania et al. (2014) consider the case of *two processing times*, where each processing time $p_{ji} \in \{q, p\}$ for all i and j. Shabtay and Karhi (2012) study the *two types of jobs* where there are two different job types, where each job type can be processed on a unique subset of machines. Some works also study similar models in a game-theoretic setting (Auletta et al., 2015; Heydenreich et al., 2010; Lavi and Swamy, 2009; Leung et al., 2010). Regarding online algorithms, several works consider *restricted assignment* with additional assumptions on the problem structure like *hierarchical server topologies* (Bar-Noy et al., 2001) (see also Crescenzi et al. (2007)). For other results of processing time restrictions, we refer the reader to the survey by Leung and Li (2008). Finally, the f-favorite machine model in our paper has been recently analyzed in a follow-up paper in the offline game-theoretic setting (Chen and Xu, 2019).

1.3. Outline of the paper

Section 2 introduces the preliminary definitions and notations. In Section 3, we focus on the general favorite machine model. Precise upper and lower bounds are obtained for the Greedy algorithm. We then analyze the best-possible competitive ratio any online algorithm can get and prove that the Assign-U algorithm achieves this bound. We further investigate the *symmetric* favorite machine model in Section 4. Theoretical bounds are provided for several algorithms. We also design a best-possible algorithm for the two machines case. In Section 5, we present computational results on the algorithms. Finally, we conclude our paper in Section 6.

2. Model and preliminary definitions

The favorite machines setting. There are m machines, $\mathcal{M} := \{1, 2, \ldots, m\}$, to process n jobs, $\mathcal{J} := \{1, 2, \ldots, n\}$. Denote the processing time of job j on machine i by p_{ji}, and the minimum processing time of job j by $p_j = \min_{i \in \mathcal{M}} p_{ji}$. We define the *favorite machines* of a job as the machines with the minimum processing time for the job. Let $F_j \subseteq \mathcal{M}$ be the set of favorite machines of job j. Assume that each job has at least f favorite machines, i.e., $|F_j| \geq f$ for $j \in \mathcal{J}$. Thus, the processing time of job j on its favorite machines equals to the minimum processing time p_j (i.e., $p_{ji} = p_j$ for $i \in F_j$), while the processing time on its non-favorite machines can be any value that is greater than p_j (i.e., $p_{ji} > p_j$ for $i \notin F_j$).

The symmetric favorite machines setting. This setting is the following restriction of the favorite machines. There are $m = 2f$ machines partitioned into two subsets of f machines each, that is, $\mathcal{M} = M_1 \cup M_2$, where $M_1 = \{1, 2, \ldots, f\}$ and $M_2 = \{f+1, f+2, \ldots, 2f\}$. Each job j has either M_1 or M_2 as its favorite machines, i.e., $F_j \in \{M_1, M_2\}$. The processing time of job j is p_j on its favorite machines and $s \cdot p_j$ on non-favorite machines, where $s \geq 1$.

Other notations. We say that job j is a *good job* if it is allocated to one of its *favorite* machines, and it is a *bad job* otherwise.

Let $l_i(j)$ denote the load on machine i after jobs 1 through j are allocated by online algorithm:

$$l_i(j) = \begin{cases} l_i(j-1) + p_{ji}\,, & \text{if job } j \text{ is assigned to machine } i\,, \\ l_i(j-1)\,, & \text{otherwise.} \end{cases}$$

In the analysis, we shall sometimes consider the machines in non-increasing order of their loads. For a sequence jobs, we denote by $\mu_i(j)$ the i^{th} highest load over all machines after the first j jobs are allocated, i.e.,

$$\mu_1(j) \geq \mu_2(j) \geq \cdots \geq \mu_m(j)\,, \quad \text{for any } j \in \mathcal{J}.$$

The jobs arrive *one by one* (over a list) and must be allocated irrevocably upon each arrival without knowing the future jobs. An allocation determines the *load* of each machine, and the goal is to minimize the *makespan*, the maximum load over all machines. The *competitive ratio* of an algorithm A is defined as $\rho_A := \sup_I \frac{C_A(I)}{C_{\text{OPT}}(I)}$, where I is taken over all possible sequences of jobs, $C_A(I)$ is the cost (makespan) of algorithm A on sequence I, and $C_{\text{OPT}}(I)$ is the optimal cost on the same sequence. We write C_A and C_{OPT} for simplicity whenever the job sequence is clear from the context.

3. The favorite machine model

In this section, we first analyze the performance of the Greedy algorithm and show its competitive ratio is precisely $\frac{m+f-1}{f}$. We then show that no online algorithm can be better than $\Omega(\log \frac{m}{f})$ and prove algorithm Assign-U (Aspnes et al., 1997) has an optimal competitive ratio of $\mathcal{O}(\log \frac{m}{f})$ for our problem.

3.1. Greedy algorithm

Algorithm Greedy : Each job is assigned to a machine that minimizes the completion time of this job, i.e., each job j is assigned to machine i^*, where $i^* = \arg\min_i \left(l_i(j-1) + p_{ji}\right)$.

We first provide a key lemma showing that Greedy maintains the following invariant: the sum of the f largest machines' loads never exceeds the sum of all jobs' minimal processing times.

Lemma 1. *For f-favorite machines, and for every sequence of n jobs, the allocation of Greedy satisfies the following condition:*

$$\mu_1(n) + \mu_2(n) + \cdots + \mu_f(n) \le p_1 + p_2 + \cdots + p_n\,. \tag{1}$$

Proof. The proof is by induction on the number n of jobs released so far. The base case is $n = 1$. Since Greedy allocates the first job to one of its favorite machines, and the other machines are empty, we have $\sum_{i=1}^{f} \mu_i(1) = \mu_i(1) = p_1$, and thus (1) holds for $n = 1$.

As for the inductive step, we assume that (1) holds for $n-1$, i.e.,

$$\mu_1(n-1) + \mu_2(n-1) + \cdots + \mu_f(n-1) \le p_1 + p_2 + \cdots + p_{n-1}\,, \tag{2}$$

and show that the same condition holds for n, i.e., after job n is allocated.

If job n is allocated as a *good* job, then the left-hand side of (2) will increase by at most p_n, while the right-hand side will increase by exactly p_n. Thus, equation (1) follows from the inductive hypothesis (2).

If job n is allocated as a *bad* job on some machine b, then the following observation allows to prove the statement: before job n is allocated, the load of each favorite machine for job n must be higher than $l_b(n-1)$ (otherwise Greedy would allocate n as a *good* job). Since there are at least f favorite machines for job n, there must be a favorite machine a with

$$l_b(n-1) < l_a(n-1) \le \mu_f(n-1). \tag{3}$$

Thus, $l_b(n-1)$ is not one of the f largest loads before job n is allocated. After allocating job n, the load of machine b increases to $l_b(n) = l_b(n-1) + p_{nb}$. We then have two cases depending on whether $l_b(n)$ is one of the f largest loads after job n is allocated:

Case 1 ($l_b(n) \le \mu_f(n-1)$). In this case, the f largest loads remain the same after job n is allocated, meaning that the left-hand side of (2) does not change, while the right-hand side increases (when adding job n). In other words, (1) follows from the inductive hypothesis (2).

Case 2 ($l_b(n) > \mu_f(n-1)$). In this case, $\mu_f(n-1)$ will be no longer included in the first f largest loads, after job n is allocated, while $l_b(n)$ will enter the set of f largest loads:

$$\sum_{i=1}^{f} \mu_i(n) = \sum_{i=1}^{f} \mu_i(n-1) - \mu_f(n-1) + l_b(n)\,. \tag{4}$$

Since Greedy allocates job n to machine b, it must be the case

$$l_b(n) = l_b(n-1) + p_{nb} \le l_a(n-1) + p_n \le \mu_f(n-1) + p_n\,, \tag{5}$$

where a is the favorite machine satisfying (3). Substituting (5) into (4) and by inductive hypothesis (2), we obtain $\sum_{i=1}^{f} \mu_i(n) \le \sum_{i=1}^{f} \mu_i(n-1) + p_n \le \sum_{i=1}^{n-1} p_i + p_n$, and thus (1) holds. This completes the proof of the lemma. □

Theorem 2. *The competitive ratio of Greedy is at most* $\frac{m+f-1}{f}$.

Proof. Without loss of generality, we assume that the makespan of the allocation of Greedy is determined by the last job n (otherwise, we can consider the last job n' which determines the makespan, and ignore all jobs after n' since they neither increase C_{Greedy} nor decrease C_{opt}).

After allocating the last job n, the cost of Greedy becomes

$$\begin{aligned} C_{Greedy} &\le \min\left\{\min_{i\in F_n} l_i(n-1) + p_n,\ \min_{i\in\mathcal{M}\setminus F_n} (l_i(n-1) + p_{ni})\right\} \\ &\le \min_{i\in F_n} l_i(n-1) + p_n \\ &\le \mu_f(n-1) + p_n\,, \end{aligned} \tag{6}$$

where the last inequality holds because job n has at least f favorite machines, and thus the least loaded among them must have load at most $\mu_f(n-1)$.

Since the optimum must allocate all jobs on m machines, and job n itself requires p_n on a single machine, we have

$$C_{opt} \ge \max\left\{\frac{\sum_{j=1}^{n} p_j}{m},\ p_n\right\} \ge \max\left\{\frac{f\cdot\mu_f(n-1) + p_n}{m},\ p_n\right\}. \tag{7}$$

where the second inequality is due to Lemma 1 applied to the first $n-1$ jobs only (specifically, we have $\sum_{j=1}^{n-1} p_j \ge \sum_{i=1}^{f} \mu_i(n-1) \ge f\cdot\mu_f(n-1)$). By combining (6) and (7), we obtain

$$\begin{aligned} \frac{C_{Greedy}}{C_{opt}} &\le \frac{\mu_f(n-1) + p_n}{\max\left\{\frac{f\cdot\mu_f(n-1)+p_n}{m},\ p_n\right\}} \\ &= \min\left\{\frac{m\left(\frac{\mu_f(n-1)}{p_n} + 1\right)}{f\frac{\mu_f(n-1)}{p_n} + 1},\ \frac{\mu_f(n-1)}{p_n} + 1\right\} \\ &\le \frac{m+f-1}{f}\,, \end{aligned}$$

where the last inequality holds because the first term decreases in $\mu_f(n-1)/p_n$ and the second term increases in $\mu_f(n-1)/p_n$. □

Theorem 3. *The competitive ratio of Greedy is at least* $\frac{m+f-1}{f}$.

Proof. We will provide a sequence of jobs whose schedule by Greedy has a makespan $\frac{m+f-1}{f}$, and the optimal makespan is 1. For simplicity, we assume that in case of a tie, the Greedy algorithm will allocate the job as a *bad job* to a machine with the *smallest index*. Without loss of generality, suppose m is divisible by f. We partition the m machines into $m' := \frac{m}{f}$ groups, $M_1, M_2, \ldots, M_{m'}$, each of them containing f machines, i.e., $M_i = \{(i-1)f + 1,\ (i-1)f + 2, \ldots,\ (i-1)f + f\}$ for $i = 1, \ldots, m'$.

The jobs are released in two phases (described in detail below): In the first phase, we force Greedy to allocate each machine in M_i a load of *bad* jobs equals to $i-1$, and a load of good jobs equals to $1 - \frac{i}{s}$ for a suitable $s > 1$ (except for $M_{m'}$); In the second phase, several jobs with favorite machines in $M_{m'}$ are released and contribute an additional $2 - \frac{1}{f}$ to the load of one machine in $M_{m'}$, and thus the makespan is $\frac{m+f-1}{f}$.

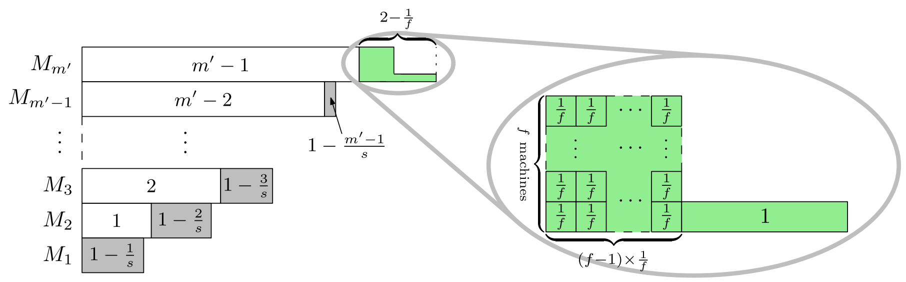


Fig. 5. Proof of Theorem 3 (bad jobs in white).

We use the notation $r \times (p, F)$ to represent a sequence of r identical jobs whose favorite machines are F, the processing time on favorite machines is p, and on non-favorite machines is $s \cdot p$ with $s \geq 1$.

Phase 1. For each i from 1 to $m'-1$, release a sequence of jobs $f \times (1-\frac{i}{s}, M_i)$ followed by a sequence $f \times (\frac{i}{s}, M_i)$. In this phase, we take $s > m'-1+\sqrt{(m'-1)(m'-2)}$, so that, for each i, the jobs $f \times (1-\frac{i}{s}, M_i)$ will be assigned as good jobs to the f machines in M_i, one per machine, and the jobs $f \times (\frac{i}{s}, M_i)$ will be assigned as bad jobs to the f machines in M_{i+1}, one per machine (as shown in Fig. 5).

At the end of Phase 1, each machine in M_i ($i \in [1, m'-1]$) will have a load of $\frac{i-1}{s} \cdot s + 1 - \frac{i}{s} = i(1-\frac{1}{s})$, and each machine in $M_{m'}$ will have a load of $m'-1$. The optimal schedule assigns every job to some favorite machine evenly so that all machines have a load of 1, except for the machines in the last group $M_{m'}$ which are left empty.

Phase 2. In this phase, all jobs released have $M_{m'}$ as their favorite machines, specifically $f(f-1) \times (\frac{1}{f}, M_{m'})$ followed by a single job $(1, M_{m'})$. By taking $s > m$, no jobs will be allocated as bad jobs by Greedy. Thus, this phase can be seen as scheduling on f identical machines. Consequently, all jobs will be allocated as in Fig. 5 increasing the maximum load of machines in $M_{m'}$ by $2-\frac{1}{f}$, while the optimal schedule can have a makespan 1.

At the end of Phase 2, the maximum load of machines in $M_{m'}$ is $m'-1+2-\frac{1}{f} = \frac{m+f-1}{f}$, while the optimal makespan is 1. □

3.2. A general lower bound

In this section, we provide a general lower bound for the f-favorite machines problem. The bound borrows the basic idea of the lower bound for the restricted assignment (Azar et al., 1992). The trick of our proof is that we always partition the selected machines into arbitrary groups of f machines each, and apply the idea to each of the groups respectively.

Theorem 4. *Every deterministic online algorithm must have a competitive ratio at least* $\frac{1}{2}\lfloor \log_2 \frac{m}{f} \rfloor + 1$.

Proof. Suppose first that $\frac{m}{f}$ is a power of 2, i.e., $\frac{m}{f} = 2^{u-1}$. We provide a sequence of m jobs having optimal makespan equal to 1, while any online algorithm will have a makespan at least $\frac{u+1}{2} = \frac{1}{2}\log_2 \frac{m}{f} + 1$.

We denote by $r \times (1, F)$ a sequence of r identical jobs whose favorite machines are F, the processing time on favorite machines is 1, and the processing time on non-favorite machines is greater than $\frac{u+1}{2} = \frac{1}{2}\log_2 \frac{m}{f} + 1$, so that all jobs must be allocated to favorite machines.

We consider subsets of machines $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_u$, where the first subset is $\mathcal{M}_1 = \mathcal{M}$ and the others satisfy $\mathcal{M}_i \subset \mathcal{M}_{i-1}$ and $|\mathcal{M}_i| = \frac{1}{2}|\mathcal{M}_{i-1}| = \frac{m}{2^{i-1}}$ for $i = 2, \ldots, u$. We then release u sets of jobs iteratively. At each iteration i ($i = 1, 2, \ldots, u$), a set of jobs $\mathcal{J}_i$ with favorite machines $\mathcal{M}_i$ are released for allocation, and after the allocation, half of the "highly loaded" machines in $\mathcal{M}_i$ are chosen for the next set $\mathcal{M}_{i+1}$. More in detail, for each iteration i ($i = 1, 2, \ldots, u$) we proceed as follows:

- Partition the current set $\mathcal{M}_i$ of machines into arbitrary groups of f machines each, that is, into groups $M_{i,1}, M_{i,2}, \ldots, M_{i,m'_i}$ where $m'_i = \frac{|\mathcal{M}_i|}{f} = \frac{m}{f \cdot 2^{i-1}} = 2^{u-i}$. Then, release a set of jobs $\mathcal{J}_i$:

$$\mathcal{J}_i := \begin{cases} J_{i,1} \cup J_{i,2} \cup \ldots \cup J_{i,m'_i}, & \text{if } 1 \leq i \leq u-1\,; \\ \{f \times (1, \mathcal{M}_u)\}, & \text{if } i = u\,; \end{cases}$$

  where $J_{i,l} := \left\{ \frac{f}{2} \times (1, M_{i,l}) \right\}$.
- The next set $\mathcal{M}_{i+1}$ of machines consists of the $f/2$ most loaded machines in each subgroup after jobs $\mathcal{J}_i$ have been allocated. Specifically, we let $\mathcal{M}_{i+1} := M'_{i,1} \cup M'_{i,2} \cup \cdots \cup M'_{i,m'_i}$, for $M'_{i,l} \subset M_{i,l}$ being the subset of $f/2$ most loaded machines in group $M_{i,l}$ after jobs $\mathcal{J}_i$ have been allocated.

We then show that the following invariant holds at every iteration.

Lemma 5. *The average load of machines in* $\mathcal{M}_i$ *is at least* $\frac{i-1}{2}$ *before the jobs in* $\mathcal{J}_i$ *are assigned for* $1 \leq i \leq u$.

Proof. The proof is done by induction on i. For the base case $i = 1$, the lemma is trivial. As for the inductive step, suppose the lemma holds for i.

Denote the average load of $M_{i,l}$ *before* the allocation of jobs $\mathcal{J}_i$ by $Avg(i, l)$, and the average load of $M'_{i,j}$ *after* the allocation of jobs $\mathcal{J}_i$ by $Avg'(i, l)$. We claim that $Avg'(i,l) \geq Avg(i,l) + 1/2$ after the allocation of jobs $\mathcal{J}_i$. Note that $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$ are the union of all groups $M_{i,l}$ and $M'_{i,l}$, respectively. Thus, we have that the average load of $\mathcal{M}_{i+1}$ is at least $\frac{i-1}{2} + \frac{1}{2} = \frac{i}{2}$, since the average load of $\mathcal{M}_i$ is $\frac{i-1}{2}$ (the inductive hypothesis). This concludes the proof of the inductive step, and thus the lemma follows. Next, we prove that the claim $Avg'(i,l) \geq Avg(i,l) + 1/2$ holds:

Case 1. There is a machine in $M_{i,l} \backslash M'_{i,l}$ that has a load at least $Avg(i,l) + 1/2$. Since $M'_{i,l}$ consists of the $f/2$ most loaded machines in $M_{i,l}$, each machine in $M'_{i,l}$ will also have a load at least $Avg(i,l) + 1/2$. Thus, we obtain that $Avg'(i,l) \geq Avg(i,l) + 1/2$.

Case 2. Each machine in $M_{i,l} \backslash M'_{i,l}$ has load no greater than $Avg(i,l) + 1/2$. Observe that the total load of $M_{i,l}$ is $f \cdot Ave(i,l) + f/2$ after the allocation of $\mathcal{J}_i$. Since $M_{i,l} \backslash M'_{i,l}$ has a load at most $f/2 \cdot (Avg(i,l) + 1/2)$, the average load of the $f/2$ machines in $M'_{i,l}$ must be at least

$$Avg'(i,l) \geq \frac{f \cdot Avg(i,l) + \frac{f}{2} - \frac{f}{2}(Avg(i,l) + \frac{1}{2})}{f/2} = Avg(i,l) + \frac{1}{2}\,.$$

□

By applying Lemma 5 with $i = u$, we have that the average load of $\mathcal{M}_u$ is at least $\frac{u-1}{2}$ before the allocation of jobs $\mathcal{J}_u$. Thus, after jobs in $\mathcal{J}_u$ are allocated, the average load of $\mathcal{M}_u$ is at least $\frac{u-1}{2}+1=\frac{u+1}{2}$, i.e., the online cost is at least $\frac{u+1}{2}$.

An optimal cost of 1 can be achieved by using the machines in $M_{i,l}\backslash M'_{i,l}$ for $\mathcal{J}_i$ (for $1\le i\le u-1$ and $1\le l\le m'_i$) and using machines in $\mathcal{M}_u$ for $\mathcal{J}_u$. Therefore, the competitive ratio is at least $\frac{u+1}{2}=\frac{1}{2}\log_2\frac{m}{f}+1$. Finally, if $\frac{m}{f}$ is not a power of 2, we simply apply the previous construction to a subset of m^* machines, for $\frac{m^*}{f}=2^{\lfloor u\rfloor-1}$ and $u=1+\log_2\frac{m}{f}$. This gives the desired lower bound $\frac{\lfloor u\rfloor+1}{2}=\frac{1}{2}\lfloor\log_2\frac{m}{f}\rfloor+1$. □

3.3. General upper bound (algorithm Assign-U is optimal)

In this section, we prove a matching upper bound for f-favorite machines. Specifically, we show that the optimal online algorithm for unrelated machines (algorithm Assign-U by Aspnes et al. (1997) described below) yields an optimal upper bound:

Theorem 6. *Algorithm Assign-U can be used to achieve $\mathcal{O}(\log\frac{m}{f})$ competitive ratio.*

We show that the algorithm has an optimal competitive ratio ρ (Lemma 7) for the case the optimal cost $C_{opt}(\mathcal{J})$ is *known*, and then apply the *doubling* approach to get an algorithm with competitive ratio 4ρ for the case the optimum is not known in advance (Theorem 6). In the following, we use a "tilde" notation to denote a normalization by the optimal cost $C_{opt}(\mathcal{J})$, i.e., $\tilde{x}=x/C_{opt}(\mathcal{J})$.

Algorithm Assign-U : Each job j is assigned to machine i^*, where i^* is the index minimizing $\Delta_i=a^{\tilde{l}_i(j-1)+\tilde{p}_{ji}}-a^{\tilde{l}_i(j-1)}$, where $a>1$ is a suitable constant.

Lemma 7. *If the optimal cost is known, Assign-U has a competitive ratio of $\mathcal{O}(\log\frac{m}{f})$.*

Proof. Without loss of generality, we assume that the makespan of the allocation of Assign-U is determined by the last job n (this is the same as the proof of Theorem 2 above). Let $l_i^*(j)$ be the load of machine i in the optimal schedule after the first j jobs are assigned. Define the potential function:

$$\Phi(j)=\sum_{i=1}^{m}a^{\tilde{l}_i(j)}\left(\gamma-\tilde{l}_i^*(j)\right), \tag{8}$$

where a, $\gamma>1$ are constants. We have that the potential function (8) is non-increasing for $a=1+1/\gamma$ (see A.1). Since $\Phi(0)\ge\Phi(n-1)$ and $\tilde{l}_i^*(n-1)\le 1$, we have

$$\begin{aligned}\gamma\cdot m&\ge\sum_{i=1}^{m}a^{\tilde{l}_i(n-1)}\left(\gamma-\tilde{l}_i^*(n-1)\right)\\&\ge\sum_{i=1}^{m}a^{\tilde{l}_i(n-1)}(\gamma-1)\\&\ge f\cdot a^{\tilde{\mu}_f(n-1)}(\gamma-1),\end{aligned}$$

where the last inequality holds because $\mu_f(n-1)$ is the f-th largest load after the first $n-1$ jobs have been allocated. Thus, it follows that

$$\mu_f(n-1)\le\log_a\left(\frac{\gamma}{\gamma-1}\cdot\frac{m}{f}\right)\cdot C_{opt}. \tag{9}$$

Claim 8. $C_{Assign-U}\le\mu_f(n-1)+p_n$.

Proof. Suppose Assign-U assigns the last job n to machine i', it holds that

$$C_{Assign-U}=l_{i'}(n-1)+p_{ni'},$$

since we suppose job n is the last completed job.

Case 1. Job n is allocated as a good job, i.e., $p_{ni'}=p_n$. According to the rule of Assign-U, machine i' must be one of the machines in F_n that has the minimum load. Since there are at least f favorite machines, it holds that $l_{i'}(n-1)\le\mu_f(n-1)$. Thus, $C_{Assign-U}\le\mu_f(n-1)+p_n$.

Case 2. Job n is allocated as a bad job, i.e., $p_{ni'}>p_n$. Let i'' be the machine who has the minimum load in F_n. Note that $l_{i''}(n-1)\le\mu_f(n-1)$. According to the rule of Assign-U, it holds that $l_{i'}(n-1)\le l_{i''}(n-1)$; otherwise, we have $\Delta_{i'}\ge\Delta_{i''}$, which is contradictory to that Assign-U assigns job n to machine i'.

Since $\Delta_{i'}\le\Delta_{i''}$ and $l_{i'}(n-1)\le l_{i''}(n-1)\le\mu_f(n-1)$, we have

$$a^{\frac{l_{i'}(n-1)+p_{ni'}}{C_{opt}}}-a^{\frac{l_{i'}(n-1)}{C_{opt}}}\le a^{\frac{l_{i''}(n-1)+p_{ni''}}{C_{opt}}}-a^{\frac{l_{i''}(n-1)}{C_{opt}}},$$

so that $\frac{l_{i'}(n-1)+p_{ni'}}{C_{opt}}\le\frac{l_{i''}(n-1)+p_{ni''}}{C_{opt}}\le\frac{\mu_f(n-1)+p_{ni''}}{C_{opt}}$, implying

$$C_{Assign-U}=l_{i'}(n-1)+p_{ni'}\le\mu_f(n-1)+p_{ni''}=\mu_f(n-1)+p_n.$$

□

According to Claim 8 and (9), we have

$$\begin{aligned}\frac{C_{Assign-U}}{C_{opt}}&\le\frac{\mu_f(n-1)+p_n}{C_{opt}}\\&\le\log_a\left(\frac{\gamma}{\gamma-1}\cdot\frac{m}{f}\right)+\frac{p_n}{C_{opt}}\\&\le\log_a\left(\frac{\gamma}{\gamma-1}\cdot\frac{m}{f}\right)+1.\end{aligned}$$

Since the latter quantity is $\mathcal{O}(\log\frac{m}{f})$, this proves Lemma 7. □

By using the standard *doubling* approach, Lemma 7 implies Theorem 6 (see A.2).

4. The symmetric favorite machine model

This section focuses on the *symmetric* favorite machines problem, where a finer analysis is possible such that we can get further results and analyze the impact of "speed ratio". In this model, $\mathcal{M}=\{M_1,M_2\}$ ($|M_1|=|M_2|=f$), $F_j\in\{M_1,M_2\}$ for each job j, and the processing time of job j on a non-favorite machine is s (≥ 1) times that on a favorite machine. We analyze the competitive ratio of Greedy as a function of the parameter s. Though Greedy has a constant competitive ratio for this problem, another natural algorithm called GreedyFavorite performs better for larger s. At last, a combination of the two algorithms, GreedyOrGreedyFavorite, obtains a better competitive ratio, and the algorithm is optimal for the two machines case.

4.1. Greedy algorithm

Theorem 9. *For the symmetric $\frac{m}{2}$-favorite machines case, the Greedy algorithm has a competitive ratio at most*

$$\hat{\rho}_{\text{Greedy}}:=\min\left\{1+\left(2-\frac{1}{f}\right)\frac{s^2}{s+1},\ s+\left(2-\frac{1}{f}\right)\frac{s}{s+1},\ 3-\frac{1}{f}\right\},$$

where $f=\frac{m}{2}$ and m is the number of machines.

Proof. Because this is a special case of the general model, we have $\hat{\rho}_{Greedy}\le 3-\frac{1}{f}$ from Theorem 2 (by recalling that $m=2f$). Thus, we just need to prove

$$\frac{C_{Greedy}}{C_{opt}}\le\min\left\{1+\left(2-\frac{1}{f}\right)\frac{s^2}{s+1},\ s+\left(2-\frac{1}{f}\right)\frac{s}{s+1}\right\}.$$

Without loss of generality, we assume that the makespan of the allocation of Greedy is determined by the last job n.

Suppose that the first $n-1$ jobs have been already allocated by Greedy, and denote $l_\alpha = \min_{i\in M_1} l_i^{(n-1)}$, $L_\alpha = \sum_{i\in M_1} l_i^{(n-1)}$, $l_\beta = \min_{i\in M_2} l_i^{(n-1)}$, $L_\beta = \sum_{i\in M_2} l_i^{(n-1)}$. Also let L_α^{good} and L_β^{good} be the total load of good jobs of L_α and L_β, respectively. Observe that $l_\alpha \le \frac{L_\alpha}{f}$ and $l_\beta \le \frac{L_\beta}{f}$. Without loss of generality, we assume that machines in M_1 are the favorite machines of the last job n.

We first give a lower bound of the optimal cost C_{opt}:

Claim 10. $C_{opt} \ge \max\{\frac{f\cdot s\cdot l_\alpha + f\cdot l_\beta + s\cdot p_n}{f\cdot s^2 + f\cdot s}, \frac{f\cdot l_\alpha + f\cdot s\cdot l_\beta + s^2\cdot p_n}{f\cdot s^2+f\cdot s}, p_n\}$

Proof. Denote by P_α and P_β the total *minimum processing time* of the jobs which have M_1 and M_2 as their favorite machines, respectively, i.e.,

$$P_\alpha = \sum_{j:\ F_j = M_1} p_j = L_\alpha^{good} + \frac{1}{s}\left(L_\beta - L_\beta^{good}\right) + p_n\,, \tag{10}$$

$$P_\beta = \sum_{j:\ F_j = M_2} p_j = L_\beta^{good} + \frac{1}{s}\left(L_\alpha - L_\alpha^{good}\right). \tag{11}$$

A lower bound on the optimal cost can be obtained by considering the following *fractional* assignment. First, allocate all jobs as *good jobs*, i.e., assign P_α to M_1 and P_β to M_2. Then, reassign a fraction of them to make all machines to have the same load. We next distinguish two cases:

Case 1 ($P_\alpha \ge P_\beta$). In this case, the reassignment is to move $\frac{1}{s+1}(P_\alpha - P_\beta)$ of P_α to machines in M_2 so that

$$P_\alpha - \frac{1}{s+1}(P_\alpha - P_\beta) = P_\beta + \frac{s}{s+1}(P_\alpha - P_\beta).$$

Therefore, along with (10) and (11), the optimal cost is at least:

$$C_{opt} \ge \frac{s\cdot P_\alpha + P_\beta}{f(s+1)} = \frac{L_\alpha + s\cdot L_\beta + s^2\cdot p_n + \left(s^2-1\right)L_\alpha^{good}}{f\cdot s^2 + f\cdot s}. \tag{12}$$

Furthermore, substituting (10) and (11) into $P_\alpha \ge P_\beta$ we have

$$L_\alpha^{good} - L_\beta^{good} \ge \frac{1}{s+1}\left(L_\alpha - L_\beta\right) - \frac{s}{s+1}p_n\,.$$

Therefore,

$$L_\alpha^{good} \ge \max\left\{\frac{1}{s+1}\left(L_\alpha - L_\beta\right) - \frac{s}{s+1}p_n,\ 0\right\}. \tag{13}$$

Substituting (13) into (12), we have

$$C_{opt} \ge \max\left\{\frac{s\cdot L_\alpha + L_\beta + s\cdot p_n}{f\cdot s^2 + f\cdot s}, \frac{L_\alpha + s\cdot L_\beta + s^2\cdot p_n}{f\cdot s^2 + f\cdot s}\right\}.$$

Along with $C_{OPT} \ge p_n$, $L_\alpha \ge f\cdot l_\alpha$ and $L_\beta \ge f\cdot l_\beta$, we obtain the inequation of this claim.

Case 2 ($P_\alpha < P_\beta$). Similarly to the previous case, we have

$$C_{opt} \ge \frac{P_\alpha + s\cdot P_\beta}{f(s+1)} = \frac{s\cdot L_\alpha + L_\beta + s\cdot p_n + \left(s^2-1\right)L_\beta^{good}}{f\cdot s^2 + f\cdot s} \tag{14}$$

In this case, (10) and (11) imply

$$L_\beta^{good} \ge \max\left\{-\frac{1}{s+1}\left(L_\alpha - L_\beta\right) + \frac{s}{s+1}p_n,\ 0\right\}, \tag{15}$$

Substituting (15) into (14), we have

$$C_{opt} \ge \max\left\{\frac{L_\alpha + s\cdot L_\beta + s^2\cdot p_n}{f\cdot s^2 + f\cdot s}, \frac{s\cdot L_\alpha + L_\beta + s\cdot p_n}{f\cdot s^2 + f\cdot s}\right\}.$$

Along with $C_{OPT} \ge p_n$, $L_\alpha \ge f\cdot l_\alpha$ and $L_\beta \ge f\cdot l_\beta$, we obtain the inequation of this claim and complete the proof. □

We next consider the cost of the Greedy algorithm. Recall that job n has M_1 as favorite machines. After job n is allocated, we have

$$C_{Greedy} \le \min\left\{l_\alpha + p_n,\ l_\beta + s\cdot p_n\right\}. \tag{16}$$

Two cases arise depending on the largest between the two quantities in (16):

Case 1 ($l_\alpha + p_n \le l_\beta + s\cdot p_n$). This case implies

$$l_\beta \ge l_\alpha - (s-1)\cdot p_n\,, \tag{17}$$

$$C_{Greedy} \le l_\alpha + p_n\,. \tag{18}$$

Therefore, we have

$$\begin{aligned}\frac{C_{Greedy}}{C_{opt}} &\le \frac{l_\alpha + p_n}{\max\left\{\frac{f\cdot s\cdot l_\alpha + f\cdot l_\beta + s\cdot p_n}{f\cdot s^2+f\cdot s}, \frac{f\cdot l_\alpha + f\cdot s\cdot l_\beta + s^2\cdot p_n}{f\cdot s^2 + f\cdot s}, p_n\right\}}\\ &\le \frac{l_\alpha + p_n}{\max\left\{\frac{f(s+1)l_\alpha + (f+s-fs)p_n}{fs^2+fs}, \frac{f(s+1)l_\alpha + \left(fs+s^2-fs^2\right)p_n}{fs^2+fs}, p_n\right\}}\\ &= \min\left\{\frac{\left(fs^2+fs\right)(x+1)}{f(s+1)x+f+s-fs}, \frac{\left(fs^2+fs\right)(x+1)}{f(s+1)x+fs+s^2-fs^2}, x+1\right\}\\ &\le \min\left\{1+\left(2-\frac{1}{f}\right)\frac{s^2}{s+1},\ s+\left(2-\frac{1}{f}\right)\frac{s}{s+1}\right\},\end{aligned}$$

where the first inequality is by (18) and Claim 10; the second inequality is by (17); the third equation is obtained by defining $x := l_\alpha/p_n$; the first term of the last inequality is obtained by the second and third terms of the third equation (one decreases in x and one increases in x); similarly, the second term of the last inequality is obtained by the first and third terms of the third equation.

Case 2 ($l_\alpha + p_n \ge l_\beta + s\cdot p_n$). This case implies

$$l_\alpha \ge l_\beta + (s-1)\cdot p_n \quad\text{and}\quad C_{Greedy} \le l_\beta + s\cdot p_n\,.$$

Similarly to the previous case, we can obtain

$$\frac{C_{Greedy}}{C_{opt}} \le \min\left\{1+\left(2-\frac{1}{f}\right)\frac{s^2}{s+1},\ s+\left(2-\frac{1}{f}\right)\frac{s}{s+1}\right\}.$$

The above two cases conclude the proof of the theorem. □

Theorem 11. *The upper bound in Theorem 9 is tight ($\rho_{Greedy} = \hat{\rho}_{Greedy}$) in each of the following cases:*

1. *For $f=1$, $\rho_{\text{Greedy}} = \min\{1+\frac{s^2}{s+1}, 2\}$;*
2. *For $f=2$ and $1\le s\le 1.605$, $\rho_{\text{Greedy}} = 1+\frac{3s^2}{2(s+1)}$;*
3. *For $3\le f\le \frac{s}{s-1}$ (implying $1\le s\le 1.5$), $\rho_{\text{Greedy}} = 1+(2-\frac{1}{f})\frac{s^2}{s+1}$;*
4. *For $f > \frac{s}{s-1}$ and $1\le s\le \frac{1+\sqrt{5}}{2}$, $\rho_{\text{Greedy}} = s+(2-\frac{1}{f})\frac{s}{s+1}$;*
5. *For $2\le f < s$, $\rho_{\text{Greedy}} = 3-\frac{1}{f}$.*

4.2. GreedyFavorite algorithm

We next consider another algorithm called GreedyFavorite which simply assigns each job j to one of its favorite machines in F_j.

Algorithm GreedyFavorite : Assign each job to one of its *favorite* machines, chosen in a *greedy* fashion (minimum load).

It turns out that this natural variant of Greedy performs better for large s.

Theorem 12. *For symmetric $\frac{m}{2}$-favorite machines, the GreedyFavorite algorithm has a competitive ratio of $2-\frac{1}{f}+\frac{1}{s}$, where $f=\frac{m}{2}$ and m is the number of machines.*

Proof. Note that in GreedyFavorite all jobs are assigned as good jobs. Suppose the overall maximum load occurred on machine 1 in M_1. Moreover, if there is any job executed on machines in M_2, we can remove all of it, which will not decrease $\frac{C_{\text{GreedyFavorite}}}{C_{\text{Opt}}}$. Therefore, all jobs have the same favorite machines M_1, and GreedyFavorite assigns all of them to M_1.

We also use l_α to represent the minimum load over M_1 before job n is allocated. Denote by P_α the total minimum processing time of the jobs who have M_1 as their favorite machines, which is also the total minimum processing time of all the jobs here. Obviously,

$$P_\alpha \geq f \cdot l_\alpha + p_n \tag{19}$$

$$C_{\text{GreedyFavorite}} \leq l_\alpha + p_n \,. \tag{20}$$

The optimal schedule can allocate some of the jobs to M_2 to balance the load over all machines. Thus, the optimal cost will be at least

$$C_{\text{OPT}} \geq \max\left\{(P_\alpha - x)\cdot\frac{1}{f},\ x\cdot\frac{s}{f}\right\} \geq \frac{s}{s+1}P_\alpha\cdot\frac{1}{f} \geq \frac{s}{s+1}\left(l_\alpha + \frac{p_n}{f}\right), \tag{21}$$

where x is the load of jobs that are assigned to M_2 to balance the load over all machines.

According to (20), (21) and $C_{\text{OPT}} \geq p_n$, we have

$$\frac{C_{\text{GreedyFavorite}}}{C_{\text{OPT}}} \leq \min\left\{\frac{l_\alpha + p_n}{\frac{s}{s+1}(l_\alpha + \frac{p_n}{f})},\ \frac{l_\alpha + p_n}{p_n}\right\} \leq 2 - \frac{1}{f} + \frac{1}{s}\,. \tag{22}$$

Thus the upper bound on the competitive ratio is proved.

To see that this bound is tight for any f and s, consider the following sequence of jobs:

$$f(f-1)\times(\tfrac{1}{f}, M_1), \quad f\times(\tfrac{1}{s}, M_1), \quad (1, M_1)\,.$$

According to algorithm GreedyFavorite, all these jobs are assigned to M_1 in a greedy fashion, and thus $C_{\text{GreedyFavorite}} = \frac{1}{f}(f-1) + \frac{1}{s} + 1 = 2 - \frac{1}{f} + \frac{1}{s}$. The optimal solution will instead assign the f jobs $(\frac{1}{s}, M_1)$ to M_2, thus implying $C_{\text{OPT}} = 1$. □

4.3. A better algorithm

As one can see, the Greedy is better than GreedyFavorite for smaller s, and GreedyFavorite is better for larger s. Thus we can combine the two algorithms to obtain a better algorithm.

Algorithm GreedyOrGreedyFavorite (GGF): If $s \leq s^*$, run Greedy ; otherwise run GreedyFavorite.

Corollary 13. *For symmetric $\frac{m}{2}$-favorite machines, if $s^* \simeq 1.481$ then $\rho_{\text{GGF}} \leq \min\{2 + \frac{s^2+s-2}{s+1}, 2 + \frac{1}{s}\} \leq 2.675$.*

Proof. By Theorem 9, we have

$$\rho_{\text{Greedy}} \leq s + \left(2 - \frac{1}{f}\right)\frac{s}{s+1} \leq s + \frac{2s}{s+1} = 2 + \frac{s^2+s-2}{s+1}\,.$$

By Theorem 12, we have

$$\rho_{GreedyFavorite} \leq 2 - \frac{1}{f} + \frac{1}{s} \leq 2 + \frac{1}{s}\,.$$

Note that if $s \leq 1.481$, $2 + \frac{s^2+s-2}{s+1} \leq 2 + \frac{1}{s}$, otherwise $2 + \frac{s^2+s-2}{s+1} > 2 + \frac{1}{s}$, thus

$$\rho_{\text{GGF}} \leq \min\left\{2 + \frac{s^2+s-2}{s+1}, 2 + \frac{1}{s}\right\} \leq 2.675\,.$$

□

4.4. Tight bounds for two machines (symmetric 1-favorite machines)

In this section, we show that the GGF algorithm is optimal for the symmetric case with *two* machines, i.e., the *symmetric 1-favorite machines.*

Theorem 14. *For symmetric 1-favorite machines, any deterministic online algorithm has competitive ratio $\rho \geq \min\left\{1 + \frac{s^2}{s+1},\ 1 + \frac{1}{s}\right\}$.*

Proof. Consider a generic algorithm Alg. Note that we have two machines, M_1 contains machine 1 and M_2 contains machine 2 only. Without loss of generality, assume the first job is assigned to machine 1 and this machine then has a load of 1, that is, job 1 is either $(1, M_1)$ or $(1/s, M_2)$.

Job 2 is (s, M_1). If Alg assigns job 2 to machine 1, then $C_{Alg} = 1 + s$ while $C_{\text{OPT}} = s$, thus implying $\rho_{\text{ALG}} \geq 1 + \frac{1}{s}$. Otherwise, if job 2 is assigned to machine 2, then a third job $(s+1, M_2)$ arrives. No matter where Alg assigns job 3, the cost for Alg will be $C_{Alg} = s^2 + s + 1$. As the optimum is $C_{\text{OPT}} = s + 1$, we have $\rho_{Alg} \geq 1 + \frac{s^2}{s+1}$ in the latter case. □

By combining Theorem 9, Theorem 12, and Theorem 14, we obtain the following result:

Corollary 15. *For symmetric 1-favorite machines, if $s^* \simeq 1.481$ then $\rho_{\text{GGF}} = \min\{1 + \frac{s^2}{s+1},\ 1 + \frac{1}{s}\} \leq 1.7549$. Therefore, the GGF algorithm is optimal.*

5. Experimental results

In this section, we present experimental results for the Greedy algorithm and the Assign-U algorithm. As the Assign-U algorithm is applied to the case the optimal cost is known, we use the standard *doubling* approach to get an algorithm for the case where the optimum is not known in advance. However, the standard *doubling* approach is designed for the sake of theoretical analysis, and may lose some efficiency in practice. Thus, we also use a *modified doubling* approach to achieve a better performance for practical problems. We denote by Assign-U* the algorithm that uses the *modified doubling* approach, and by Assign-U the algorithm that use the *standard doubling* approach in this section.

In the *standard doubling* approach (A.2) each phase is independent, i.e., in each phase jobs are assigned assuming that all machines are empty at the beginning of this phase, regardless of the jobs assigned in the previous phases. However, in the *modified doubling* approach, we remove the assumption, i.e., in each phase jobs are assigned according to the accumulated load of each machine.

We consider the general model where the job has the same processing time on its favorite machines and some arbitrary longer processing times on its non-favorite machines. In these experiments, to clear the influence of the factor f, we let each job have exactly f favorite machines. The processing time of each job on each machine is uniformly distributed in the interval [0,10], and there are arbitrary f favorite machines with the smallest processing time. Specifically, for each job we first generate a sequence of $m - f + 1$ real values chosen uniformly at random in the interval [0,10], and insert the minimum value among them into the sequence randomly for $f - 1$ times one by one. Hence, we have a sequence of m real values and the minimum occurs on f different positions of the sequence, which gives the processing time of a job for m machines.

For simplicity, instead of the optimal cost, we calculate the competitive ratio by a lower bound of the optimal cost. Thus the competitive ratios in the tables are slightly larger (worse) than the real ratios. The lower bound of the optimal cost is the total minimum processing time divided by the number of machines, i.e., $\text{OPT}^* = \sum_j p_j/m$. We take the parameter a of Assign-U as $a = 1.9$. We test the case $m = 10$ for $n = 50, 100, 500, 1000, 2000$ jobs, and the case $m = 50$ for $n = 500, 1000, 2000, 3000, 5000$ jobs. For each case, 10 different values of f are considered including $f = 1$ and $f = m$ (corresponding to unrelated and identical machine models respectively). Each ratio in the table is the mean competitive ratio over 30 runs.

Table 1
Computational results ($m = 10$).

	n \ f	1	2	3	4	5	6	7	8	9	10
Assign-U	50	1.94	1.62	1.48	1.48	1.45	1.49	1.41	1.39	1.38	1.32
Greedy		1.68	1.45	1.36	1.32	1.31	1.30	1.26	1.23	1.23	1.17
Assign-U*		1.62	1.40	1.33	1.30	1.30	1.30	1.26	1.23	1.23	1.17
Assign-U	100	1.54	1.30	1.23	1.24	1.22	1.23	1.21	1.21	1.19	1.15
Greedy		1.51	1.28	1.20	1.16	1.15	1.17	1.13	1.13	1.10	1.09
Assign-U*		1.36	1.20	1.17	1.14	1.14	1.16	1.13	1.13	1.10	1.09
Assign-U	500	1.21	1.08	1.06	1.05	1.05	1.05	1.04	1.05	1.04	1.04
Greedy		1.36	1.14	1.07	1.05	1.04	1.03	1.03	1.03	1.02	1.02
Assign-U*		1.15	1.05	1.04	1.04	1.04	1.03	1.03	1.03	1.02	1.02
Assign-U	1000	1.15	1.04	1.03	1.03	1.03	1.03	1.02	1.02	1.02	1.02
Greedy		1.36	1.12	1.06	1.04	1.02	1.02	1.02	1.01	1.01	1.01
Assign-U*		1.10	1.02	1.02	1.02	1.01	1.01	1.01	1.01	1.01	1.01
Assign-U	2000	1.10	1.02	1.02	1.01	1.02	1.01	1.01	1.01	1.01	1.01
Greedy		1.34	1.11	1.05	1.03	1.02	1.01	1.01	1.01	1.01	1.00
Assign-U*		1.06	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.00

From Tables 1 and 2, we can observe the following properties:

1. The performances of Assign-U, Greedy and Assign-U* get better when f gets larger. Especially when f is increased from 1 to 2, the performance will have distinct improvement (e.g. Figs. 6 and 7).
2. The larger the number of jobs n is, the better the performances of Assign-U, Greedy and Assign-U* will be (e.g. Fig. 8).
3. For smaller n, the Greedy usually has much better performance than Assign-U (e.g. Fig. 6).

Table 2
Computational results ($m = 50$).

	n \ f	1	2	10	15	20	25	30	35	40	50
Assign-U	500	1.93	1.59	1.45	1.51	1.43	1.53	1.44	1.47	1.45	1.22
Greedy		1.71	1.54	1.30	1.36	1.28	1.30	1.27	1.33	1.30	1.11
Assign-U*		1.60	1.45	1.30	1.36	1.28	1.30	1.27	1.33	1.30	1.11
Assign-U	1000	1.60	1.29	1.24	1.25	1.22	1.26	1.26	1.23	1.20	1.12
Greedy		1.57	1.30	1.17	1.15	1.16	1.17	1.15	1.15	1.14	1.05
Assign-U*		1.40	1.21	1.16	1.15	1.16	1.17	1.15	1.15	1.14	1.05
Assign-U	2000	1.39	1.16	1.12	1.13	1.13	1.12	1.13	1.11	1.12	1.07
Greedy		1.47	1.22	1.08	1.08	1.08	1.08	1.08	1.08	1.07	1.03
Assign-U*		1.23	1.10	1.08	1.08	1.08	1.08	1.08	1.08	1.07	1.03
Assign-U	3000	1.31	1.12	1.09	1.08	1.08	1.08	1.09	1.08	1.08	1.04
Greedy		1.45	1.19	1.06	1.06	1.05	1.05	1.05	1.05	1.05	1.02
Assign-U*		1.20	1.07	1.05	1.06	1.05	1.05	1.05	1.05	1.05	1.02
Assign-U	5000	1.24	1.08	1.06	1.05	1.05	1.05	1.05	1.05	1.05	1.03
Greedy		1.42	1.17	1.04	1.03	1.04	1.03	1.03	1.03	1.03	1.01
Assign-U*		1.14	1.05	1.03	1.03	1.04	1.03	1.03	1.03	1.03	1.01

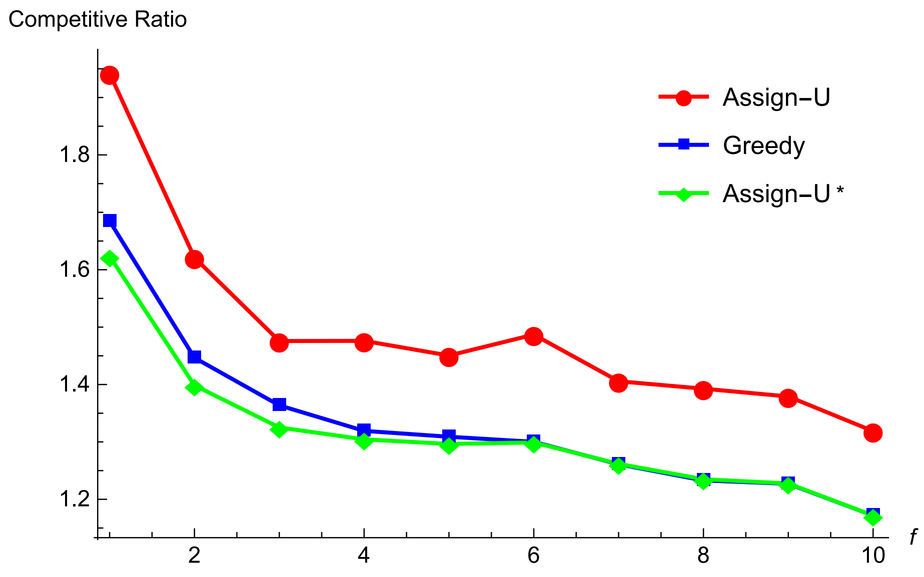


Fig. 6. Computation results for the case $m = 10$ and $n = 50$.

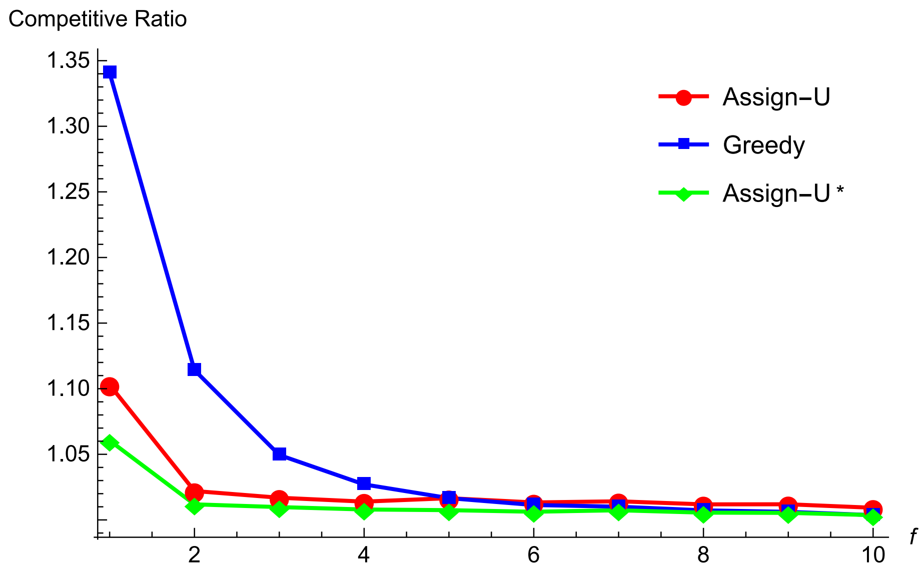


Fig. 7. Computation results for the case $m = 10$ and $n = 2000$.

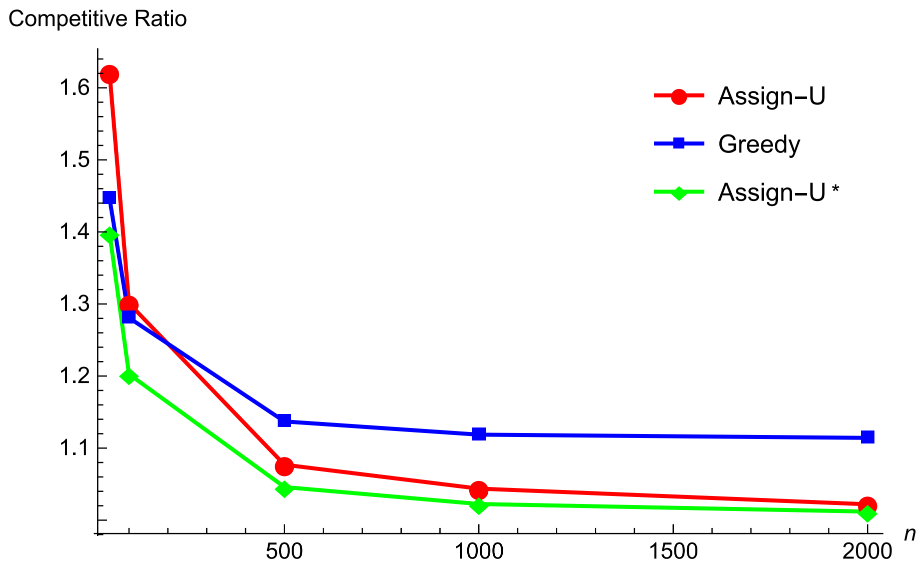


Fig. 8. Computation results for the case $m = 10$ and $f = 2$.

4. For larger n, the Assign-U is better than the Greedy especially for smaller f (e.g. Fig. 7).
5. The Assign-U* outperforms both the Greedy and the Assign-U in almost any circumstance.

Therefore, we can get some insights of choosing algorithms for practical problems. Although the Assign-U has analyzable theoretical bound, Assign-U* performs better for random instances. If the scheduling system is close to the unrelated machine model (smaller f), algorithm Assign-U* is the best choice. Otherwise, the simple Greedy is almost as good as the Assign-U* for larger f, and hence the Greedy is also a suitable choice. However, if considering the long-term plan (to process a large number of jobs), the Assign-U and Assign-U* are more efficient than the Greedy (e.g. Fig. 8).

6. Conclusion and open questions

This work studies online scheduling for the *favorite* machine model. Our results are supplements to several classical problems and reveal the relations among them (as indicated in Fig. 4). For the general f-favorite machines case, we provide tight bounds on both Greedy and Assign-U algorithms and show that the latter is the best-possible online algorithm. To some extent, the key factor f in our model captures some of the main features that make the model perform well or badly: low or high competitive ratio. In particular, when $f = 1$, the model is exactly the *unrelated* machines; when $f = m$, the model is exactly the *identical* machines. The experimental results show that the algorithms perform better on random instances with f-favorite machines, and reveal some insights of choosing algorithms for practical use. Finally, the analysis of symmetric favorite machines allows a direct comparison with the two related machines.

Regarding future work, it might be interesting to further study "intermediate" classes of instances between unrelated and related machines, where greedy performs better than in the general case, possibly incorporating some "scaling factor". One interesting direction is to "relax" the definition of favorite machine, by considering those machines where processing times are "close" to the minimum as favorite machines. For instance, one can say that the processing time of each job j on its favorite machines is between p_j and $\alpha \cdot p_j$, for some constant $\alpha \geq 1$.

Acknowledgments

This work was partially done while the first author was visiting ETH Zurich and was a PhD student at School of Management, Xi'an Jiaotong University, China. This work was supported by the National Natural Science Foundation of China [71601152, 71720107002]; and the Natural Science Fund of Guangdong Province [2017A030312001].

Appendix A. Postponed Proofs

A1. The potential function is non-increasing (for proof of Lemma 7)

The proof is based on Aspnes et al. (1997). Recall that the potential function is defined as $\Phi(j) = \sum_{i=1}^{m} a^{\tilde{l}_i(j)}(\gamma - \tilde{l}_i^*(j))$. Assume that job j is assigned to machine i' by algorithm Assign-U and to machine i by the optimal schedule, i.e., $\tilde{l}_{i'}(j) = \tilde{l}_{i'}(j-1) + \tilde{p}_{ji'}$ and $\tilde{l}_i^*(j) = \tilde{l}_i^*(j-1) + \tilde{p}_{ji}$. Then we have

$$\begin{aligned}\Phi(j) - \Phi(j-1) &= (\gamma - \tilde{l}_{i'}^*(j-1))(a^{\tilde{l}_{i'}(j)} - a^{\tilde{l}_{i'}(j-1)}) - a^{\tilde{l}_i(j-1)}\tilde{p}_{ji} \\ &\leq \gamma(a^{\tilde{l}_{i'}(j-1)+\tilde{p}_{ji'}} - a^{\tilde{l}_{i'}(j-1)}) - a^{\tilde{l}_i(j-1)}\tilde{p}_{ji} \\ &\leq \gamma(a^{\tilde{l}_i(j-1)+\tilde{p}_{ji}} - a^{\tilde{l}_i(j-1)}) - a^{\tilde{l}_i(j-1)}\tilde{p}_{ji} \qquad (\text{by } \Delta_{i'} \leq \Delta_i) \\ &= a^{\tilde{l}_i(j-1)}(\gamma(a^{\tilde{p}_{ji}} - 1) - \tilde{p}_{ji}).\end{aligned}$$

By taking $a = 1 + 1/\gamma$, we get $\gamma(a^{\tilde{p}_{ji}} - 1) - \tilde{p}_{ji} \leq 0$ since $0 \leq \tilde{p}_{ji} \leq 1$, so that the potential function is non-increasing.

A2. Doubling approach (proof of Theorem 6)

Let ρ be the competitive ratio of Assign-U when the optimal cost C_{opt} is known. By using *doubling* approach one can easily get a 4ρ-competitive algorithm for the case optimal cost is not known. This approach has been used in Aspnes et al. (1997). We report the details below for completeness.

We run Assign-U in phases, and let Λ_i be the estimation of C_{opt} at the beginning of phase i. Initially (beginning of phase 1) when the first job arrives, let Λ_1 be the minimum processing time of the first job. Whenever the makespan exceeds ρ times the current estimation, $\rho\Lambda_i$, the current phase i ends and the next phase $i+1$ begins with doubled estimation $\Lambda_{i+1} = 2\Lambda_i$ as the new estimation of the C_{opt} to run Assign-U. During a single phase, jobs are assigned independently of the jobs assigned in the previous phases. It is easy to see that this approach increases the competitive ratio ρ by at most a multiplicative factor 4 (a factor of 2 due to the load in all but the last phase, and another factor of 2 due to imprecise estimation of C_{opt}).

More in detail, each phase i can increase the load of every machine by at most $\rho\Lambda_i$. If u denotes the number of phases, then the final makespan will be no more than $\rho\sum_{i=1}^{u}\Lambda_i$. Note that $\sum_{i=1}^{u}\Lambda_i = (1 + \frac{1}{2} + \cdots + \frac{1}{2^{u-1}})\Lambda_u = (2 - \frac{1}{2^{u-1}})\Lambda_u$, since $\Lambda_{i+1} = 2\Lambda_i$. We also have $\Lambda_u = 2\Lambda_{u-1} < 2C_{\text{OPT}}$, because $\Lambda_{u-1} < C_{\text{OPT}}$ (otherwise in phase $u-1$ the makespan will not exceed $\rho\Lambda_{u-1}$ according to Lemma 7). Thus we have $\rho\sum_{i=1}^{u}\Lambda_i = (2 - \frac{1}{2^{u-1}})\rho\Lambda_u < 4\rho C_{\text{OPT}}$.

A3. Proof of Theorem 11

In some of the proofs we shall make use of the following initial set of "small" jobs:

$$\underbrace{f \times (\epsilon, M_1),\ f \times (\frac{2\epsilon}{s}, M_1),\ f \times (\frac{2\epsilon}{s}, M_2),\ f \times (\frac{2\epsilon}{s}, M_1),\ f \times (\frac{2\epsilon}{s}, M_2) \ldots}_{t/\epsilon \text{ blocks}} \tag{A.1}$$

where the total number of jobs is $f \cdot t/\epsilon$, and ϵ is chosen so that t/ϵ is integer.

According to the algorithm Greedy, only the first first f jobs of length ϵ are assigned as good jobs, while all other jobs are assigned as bad jobs. Moreover, all machines in each class will have the same load of t and $t - \epsilon$. These jobs can be redistributed to the machines in order to build arbitrary load (up to some arbitrarily small additive ϵ). Taking $\epsilon \to 0$, we can obtain the following result.

Observation 16. At the beginning of a schedule by algorithm Greedy, if $s < 2$, each machine can have a load of t so that all jobs executed during $[0, t]$ are bad jobs, and each bad job is extremely "small" so that they can be redistributed to create an arbitrary load on any machine.

Proof of Theorem 11. We give five instances each of them resulting in a lower bound for the corresponding case.

Case 1 ($f = 1$)**.** If $1 \leq s \leq \frac{1+\sqrt{5}}{2}$, the jobs sequence is $(\frac{1}{s+1}, M_2)$, $(\frac{s}{s+1}, M_2)$ and $(1, M_1)$. The Greedy algorithm assigns the first job to machine 2, and the last two jobs to machine 1, which leads to $C_{\text{Greedy}} = 1 + \frac{s^2}{s+1}$. In optimal schedule all jobs are allocated as good jobs, i.e., $C_{\text{OPT}} = 1$.

If $s > \frac{1+\sqrt{5}}{2}$, the jobs sequence is $(\frac{s-1}{s}, M_2)$, $(\frac{1}{s}, M_2)$ and $(1, M_1)$. The Greedy assigns the first job to machine 2, and the last two jobs to machine in 1, which leads to $C_{\text{GREEDY}} = 2$. Again, in optimal schedule all jobs are allocated as good jobs, i.e., $C_{\text{OPT}} = 1$.

Therefore, $\rho_{\text{GREEDY}} = \min\{1 + \frac{s^2}{s+1},\ 2\}$ is tight for any $s \geq 1$.

Case 2 (**$f = 2$ and $1 \leq s \leq 1.605$**). Let $l_\alpha = \frac{3s^2}{2(s+1)}$, $S_6 = \sum_{i=1}^{6}(s-1)^i = \frac{s-1}{2-s}(1-(s-1)^6)$ and $l_\beta^{good} = \frac{2-s}{2(s+1)}$. The sequence of jobs corresponds to the following three steps:

Step 1: According to Observation 16, we let each machine have an initial load $l_\alpha - S_6 - l_\beta^{good} + (s-1)^6$ of bad jobs, where $l_\alpha - S_6 - l_\beta^{good} + (s-1)^6 \geq 0$ due to $1 \leq s \leq 1.605$.

Step 2: Four jobs arrive: $2 \times (l_\beta^{good} - (s-1)^6, M_2)$ and $2 \times (\frac{l_\beta^{good} - (s-1)^6}{s}, M_2)$. According to GREEDY, the first two jobs will be assigned to machine 3 and 4 respectively as good jobs. But the last two jobs will be assigned to machine 1 and 2 as bad jobs. At this point, all the four machines have the same load $l_\alpha - S_6$.

Step 3: This sequence of jobs arrive: $2 \times ((s-1)^6, M_2)$, $2 \times ((s-1)^5, M_2)$, $2 \times ((s-1)^4, M_1)$, $2 \times ((s-1)^3, M_2)$, $2 \times ((s-1)^2, M_1)$, $2 \times (s-1, M_2)$. According to GREEDY, the first two jobs are allocated as good jobs, while the others are allocated as bad jobs. At this point, the loads of machine 1 and 2 are l_α, while machine 3 and 4 have the same load of $l_\alpha - (s-1)$.

Step 4: Job $(1, M_1)$ arrives, which will be assigned to machine 3 as a bad job. Therefore, $C_{\text{GREEDY}} = l_\alpha + 1 = 1 + \frac{3s^2}{2(s+1)}$. The optimal schedule is to assign all jobs as good jobs. By calculation we have $C_{\text{OPT}} = 1$. Thus, $\rho_{\text{GREEDY}} = 1 + \frac{3s^2}{2(s+1)}$ is tight for $1 \leq s \leq 1.605$.

Case 3 (**$3 \leq f \leq \frac{s}{s-1}$**). Let $l_\alpha = (2 - 1/f)\frac{s^2}{s+1}$, $a_i = (s-1)^i$, $S_u = \sum_{i=1}^{u} a_i = \frac{s-1}{2-s}(1-a_u)$ and $l_\beta^{good} = \frac{f+s-f\cdot s}{f(s+1)}$, where u is even number. Suppose $3 \leq f \leq \frac{s}{s-1+(s+1)a_u}$ and $1 \leq s \leq 1.5$. Note that when $u \to \infty$, we have $a_u \to 0$, i.e. $3 \leq f \leq \frac{s}{s-1}$.

Step 1: According to Observation 16, we let each machine have an initial load $l_\alpha - S_u - l_\beta^{good} + a_u$ of bad jobs, where $l_\alpha - S_u - l_\beta^{good} + a_u \geq 0$ due to $3 \leq f \leq \frac{s}{s-1+(s+1)a_u}$ and $1 \leq s \leq 1.5$.

Step 2: These $2f$ jobs arrive: $f \times (l_\beta^{good} - a_u, M_2)$ and $f \times \left(\frac{l_\beta^{good} - a_u}{s}, M_2\right)$. According to GREEDY, the first f jobs will be assigned to M_2 as good jobs with one machine each, while the last f jobs will be assigned to M_1 as bad jobs with one machine each. At this point, all the $2f$ machines have the same load of $l_\alpha - S_u$.

Step 3: This sequence of jobs arrives: $f \times (a_u, M_2)$, $f \times (a_{u-1}, M_2)$, $f \times (a_{u-2}, M_1)$, $f \times (a_{u-3}, M_2)$, $f \times (a_{u-4}, M_1)$,..., $f \times (a_2, M_1)$, $f \times (a_1, M_2)$. According to GREEDY, the first f jobs will be allocated as good jobs while the others as bad jobs. At this point, each machine in M_1 has a load of l_α, and each machine in M_2 has a load of $l_\alpha - (s-1)$.

Step 4: Job $(1, M_1)$ arrives, which is assigned to one machine in M_2 as a bad job. Therefore, $C_{\text{GREEDY}} = l_\alpha + 1 = 1 + (2 - 1/f)\frac{s^2}{s+1}$.

The optimal schedule will allocate all jobs as good jobs. We next give such an optimal schedule to show that $C_{\text{OPT}} = 1$ is achievable:

(*Step 1 jobs.*) All the jobs in Step 1 will be allocated as good jobs in optimal schedule, meaning $f \times \frac{l_\alpha - S_u - l_\beta^{good} + a_u}{s}$ for each of M_1 and M_2;

(*Step 2 jobs.*) All the jobs in Step 2 will be assigned to M_2;

(*Step 3 jobs.*) All the jobs in Step 3 will be allocated as good jobs, meaning $f \times (a_2 + a_4 + \cdots + a_{u-2})$ will be assigned to M_1, while the rest of them to M_2;

(*Step 4 jobs.*) The last job in Step 4 will be assigned to M_1.

For the jobs allocated to M_1, notice that every 2 jobs of $f \times (a_2 + a_4 + \cdots + a_{u-2})$ should be assigned to one machine, i.e., the loads of some $\lfloor \frac{f}{2} \rfloor$ machines are all $2 \times (a_2 + a_4 + \cdots + a_{u-2})$, where $2 \times (a_2 + a_4 + \cdots + a_{u-2}) = \frac{2a_u - 2(s-1)^2}{s(s-2)} < 1$ since $3 \leq f \leq \frac{s}{s-1+(s+1)a_u}$ and $1 \leq s \leq 1.5$. Then the other jobs can be easily arranged within time 1, since the jobs in Step 1 are all "small" jobs.

For the jobs allocated to M_2, the jobs can be equally divided into f parts with each part has $\frac{l_\alpha - S_u - l_\beta^{good} + a_u}{s} + (l_\beta^{good} - a_u) + \frac{l_\beta^{good} - a_u}{s} + (a_1 + a_3 + \cdots + a_{u-1} + a_u) = 1$. Thus all machines in M_2 also have the same load of 1. Therefore, $C_{\text{OPT}} = 1$.

Thus, $\rho_{\text{GREEDY}} = 1 + (2 - 1/f)\frac{s^2}{s+1}$ is tight for $3 \leq f \leq \frac{s}{s-1+(s+1)a_u}$ and $1 \leq s \leq 1.5$. Taking $u \to \infty$, we have $a_u \to 0$, i.e. $3 \leq f \leq \frac{s}{s-1}$.

Case 4 (**$f > \frac{s}{s-1}$ and $1 \leq s \leq \frac{1+\sqrt{5}}{2}$**). Let $l_\alpha = s + \frac{f\cdot s - f - s}{f(s+1)}$, $a_i = (s-1)^i$, $S_u = \sum_{i=1}^{u} a_i = \frac{s-1}{2-s}(1-a_u)$ and $l_\alpha^{good} = \frac{f\cdot s - f - s}{f(s+1)}$, where u is odd number. Suppose $f > \frac{s}{s-1-(s+1)a_u}$ and $1 \leq s \leq \frac{1+\sqrt{5}}{2}$. When $u \to \infty$, we have $a_u \to 0$, i.e. $f > \frac{s}{s-1-(s+1)a_u}$.

Step 1: According to Observation 16, we let each machine have an initial load $l_\alpha - S_u$ of bad jobs, where $l_\alpha - S_u > 0$ due to $f > \frac{s}{s-1-(s+1)a_u}$ and $1 \leq s \leq \frac{1+\sqrt{5}}{2}$.

Step 2: This sequence of jobs arrives: $f \times (a_u, M_1)$, $f \times (a_{u-1}, M_1)$, $f \times (a_{u-2}, M_2)$, $f \times (a_{u-3}, M_1)$, $f \times (a_{u-4}, M_2)$,..., $f \times (a_2, M_1)$, $f \times (a_1, M_2)$. According to GREEDY, the first f jobs will be allocated as good jobs while the others as bad jobs. Up to now each machine in M_1 has a load of l_α, and each machine in M_2 has a load of $l_\alpha - (s-1)$.

Step 3: Job $(1, M_1)$ arrives, which will be assigned to one machine in M_2 as a bad job. Therefore, $C_{\text{GREEDY}} = l_\alpha + 1 = s + (2 - 1/f)\frac{s}{s+1}$.

For the optimal cost, we will show a schedule so that $C_{\text{OPT}} = 1$. Part of the jobs in Step 1 will be allocated as bad jobs in optimal schedule, specifically some jobs with total minimum processing time $f \times \frac{l_\alpha^{good} - a_u}{s}$ having M_2 as their favorite machine set will be allocated to M_1 as bad jobs, i.e., M_1 will have jobs with total load of $f \times \frac{l_\alpha - S_u}{s} + f \times (l_\alpha^{good} - a_u)$ while M_2 will have jobs with total load of $f \times (\frac{l_\alpha - S_u}{s} - \frac{l_\alpha^{good} - a_u}{s})$; all the jobs in Step 2 and 3 will be allocate as good jobs, meaning $f \times (a_u + a_{u-1} + a_{u-3} + \cdots + a_2) + 1$ will be assigned to M_1, while the rest of them to M_2. To sum up, the machines of M_1 have jobs with total load of $f \times \frac{l_\alpha - S_u}{s} + f \times (l_\alpha^{good} - a_u) + f \times (a_u + a_{u-1} + a_{u-3} + \cdots + a_2) + 1 = f$, while the machines of M_2 have jobs with total load of $f \times (\frac{l_\alpha - S_u}{s} - \frac{l_\alpha^{good} - a_u}{s}) + f \times (a_{u-2} + a_{u-4} + \cdots + a_1) = f$.

Then we give a schedule so that each machine has the same load of 1. For the jobs allocated to M_1, we first arrange the $f \times (a_u + a_{u-1} + a_{u-3} + \cdots + a_2)$ and 1. The job with length 1 will be assigned to machine 1, and jobs $(f-1) \times (a_u + a_{u-1} + a_{u-3} + \cdots + a_2)$ will be assigned to the remaining $f-1$ machines with $(a_u + a_{u-1} + a_{u-3} + \cdots + a_2)$ each. The remaining $(a_u + a_{u-1} + a_{u-3} + \cdots + a_2)$ will be divided into 2 parts, a_2 assigned to machine 2 and $(a_u + a_{u-1} + a_{u-3} + \cdots + a_4)$ to machine 3. Note that $(a_u + a_{u-1} + a_{u-3} + \cdots + a_2) + a_2 < 1$ and $2(a_u + a_{u-1} + a_{u-3} + \cdots + a_2) - a_2 < 1$, due to $f > \frac{s}{s-1-(s+1)a_u}$, $1 \leq s \leq \frac{1+\sqrt{5}}{2}$ and $a_u \leq s-1$. Till now, no machine has a load more than 1, and the remaining jobs are all "small" jobs which can be arbitrary divided and assigned to make every machine with load 1. For the jobs allocate to M_2, they can be equally divided into f parts with each size 1. Therefore, all machines have the same load of 1.

Thus, $\rho_{\text{GREEDY}} = s + (2 - 1/f)\frac{s}{s+1}$ is tight for $f > \frac{s}{s-1-(s+1)a_u}$ and $1 \leq s \leq \frac{1+\sqrt{5}}{2}$. Taking $u \to \infty$, we have $a_u \to 0$, i.e. $f > \frac{s}{s-1}$.

Case 5 (**$2 \leq f < s$**). Consider this jobs sequence: $f \times (1 - 1/s, M_2)$, $f \times (1/s, M_2)$, $f(f-1) \times (1/f, M_1)$ and $(1, M_1)$.

According to algorithm GREEDY, the first f jobs will be assigned to f machines in M_2 respectively, so that each machine in M_2 has a load of $1 - 1/s$. Then the next f jobs will be assigned to f machines in M_1 respectively, so that each machine in M_1 has a load of 1. In terms of the $f(f-1)$ jobs with length $1/f$, all of them will be assigned to machines in M_1 with $f-1$ job each machine. Note that none of the $f(f-1)$ jobs will go to M_2, since $1 - \frac{1}{s} + \frac{s}{f} > 1 + \frac{f-1}{f}$. Now all machines in M_1 have the same load of $2 - \frac{1}{f}$. At last, the final job with length 1 will be assigned to one machine in M_1, since $2 - \frac{1}{f} + 1 < 1 - \frac{1}{s} + s$. Therefore, $C_{\text{GREEDY}} = 3 - \frac{1}{f}$.

For the optimal cost, it is easy to have $C_{\text{OPT}} = 1$ by assigning each job to its favorite class of machines.

Therefore, $\rho_{\text{GREEDY}} = 3 - 1/f$ is tight for $2 \le f < s$. $\square$

References

Albers, S., 1999. Better bounds for online scheduling. SIAM J. Comput. 29 (2), 459–473.

Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O., 1997. On-line routing of virtual circuits with applications to load balancing and machine scheduling. J. ACM 44 (3), 486–504.

Auletta, V., Christodoulou, G., Penna, P., 2015. Mechanisms for scheduling with single-bit private values. Theory Comput. Syst. 57 (3), 523–548.

Azar, Y., Naor, J.S., Rom, R., 1992. The competitiveness of on-line assignments. In: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 203–210.

Bar-Noy, A., Freund, A., Naor, J., 2001. On-line load balancing in a hierarchical server topology. SIAM J. Comput. 31 (2), 527–549.

Bartal, Y., Fiat, A., Karloff, H., Vohra, R., 1995. New algorithms for an ancient scheduling problem. J. Comput. Syst. Sci. 51 (3), 359–366.

Berman, P., Charikar, M., Karpinski, M., 2000. On-line load balancing for related machines. J. Algo. 35 (1), 108–121.

Chen, C., Penna, P., Xu, Y., 2017. Selfish jobs with favorite machines: Price of anarchy vs. strong price of anarchy. In: Proc. of the 11th Annual International Conference on Combinatorial Optimization and Applications (COCOA), pp. 226–240.

Chen, C., Xu, Y., 2019. Selfish load balancing for jobs with favorite machines. Operat. Res. Lett. 47 (1), 7–11.

Chen, L., Ye, D., Zhang, G., 2014. Online scheduling of mixed cpu-gpu jobs. Int. J. Foundat. Comput. Sci. 25 (06), 745–761.

Cho, Y., Sahni, S., 1980. Bounds for list schedules on uniform processors. SIAM J. Comput. 9 (1), 91–103.

Crescenzi, P., Gambosi, G., Nicosia, G., Penna, P., Unger, W., 2007. On-line load balancing made simple: greedy strikes back. J. Discrete Algo. 5 (1), 162–175.

Epstein, L., 2010. Equilibria for two parallel links: the strong price of anarchy versus the price of anarchy. Acta Inf. 47 (7), 375–389.

Epstein, L., Noga, J., Seiden, S., Sgall, J., Woeginger, G., 2001. Randomized on-line scheduling on two uniform machines. J. Schedul. 4 (2), 71–92.

Faigle, U., Kern, W., Turán, G., 1989. On the performance of on-line algorithms for partition problems. Acta Cybernet. 9 (2), 107–119.

Fleischer, R., Wahl, M., 2000. On-line scheduling revisited. J. Schedul. 3 (6), 343–353.

Graham, R.L., 1966. Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. 45 (9), 1563–1581.

Heydenreich, B., Müller, R., Uetz, M., 2010. Mechanism design for decentralized online machine scheduling. Oper. Res. 58 (2), 445–457.

Imreh, C., 2003. Scheduling problems on two sets of identical machines. Computing 70 (4), 277–294.

Karger, D.R., Phillips, S.J., Torng, E., 1996. A better algorithm for an ancient scheduling problem. J. Algo. 20 (2), 400–430.

Lavi, R., Swamy, C., 2009. Truthful mechanism design for multidimensional scheduling via cycle monotonicity. Games Econ. Behav. 67 (1), 99–124.

Lee, K., Leung, J.Y.-T., Pinedo, M.L., 2013. Makespan minimization in online scheduling with machine eligibility. Ann. Oper. Res. 204 (1), 189–222.

Leung, J.Y.-T., Li, C.-L., 2008. Scheduling with processing set restrictions: a survey. Int. J. Prod. Econ. 116 (2), 251–262.

Leung, J.Y.-T., Pinedo, M., Wan, G., 2010. Competitive two-agent scheduling and its applications. Oper. Res. 58 (2), 458–469.

Rudin III, J.F., 2001. Improved bounds for the online scheduling problem. The University of Texas at Dallas.

Shabtay, D., Karhi, S., 2012. Online scheduling of two job types on a set of multipurpose machines with unit processing times. Comput. Oper. Res. 39 (2), 405–412.

Vakhania, N., Hernandez, J.A., Werner, F., 2014. Scheduling unrelated machines with two types of jobs. Int. J. Prod. Res. 52 (13), 3793–3801.

Zhang, L., Wirth, A., 2009. On-line scheduling of two parallel machines with a single server. Comput. Oper. Res. 36 (5), 1529–1553.