European Journal of Operational Research 238 (2014) 54–64

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Fast approximation algorithms for bi-criteria scheduling with machine assignment costs

Kangbok Lee [a], Joseph Y-T. Leung [b,1], Zhao-hong Jia [c,2], Wenhua Li [d,3], Michael L. Pinedo [e,*,4], Bertrand M.T. Lin [f,5]

[a] Department of Business & Economics, York College, The City University of New York, 94-20 Guy R. Brewer Blvd, Jamaica, NY 11451, USA
[b] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA
[c] Key Lab of Intelligent Computing & Signal Processing of Ministry of Education, Anhui University, Hefei, Anhui 230039, PR China
[d] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, PR China
[e] Department of Information, Operations & Management Sciences, Stern School of Business, New York University, 44 West 4th Street, New York, NY 10012-1126, USA
[f] Department of Information & Finance Management, National Chiao Tung University, Taiwan

ARTICLE INFO

Article history:
Received 28 May 2013
Accepted 17 March 2014
Available online 29 March 2014

Keywords:
Bi-criteria scheduling
Maximum machine cost
Total machine cost
Makespan
Total completion time
Heuristics

ABSTRACT

We consider parallel machine scheduling problems where the processing of the jobs on the machines involves two types of objectives. The first type is one of two classical objective functions in scheduling theory: either the total completion time or the makespan. The second type involves an actual cost associated with the processing of a specific job on a given machine; each job-machine combination may have a different cost. Two bi-criteria scheduling problems are considered: (1) minimize the maximum machine cost subject to the total completion time being at its minimum, and (2) minimize the total machine cost subject to the makespan being at its minimum. Since both problems are strongly NP-hard, we propose fast heuristics and establish their worst-case performance bounds.

1. Introduction

In traditional scheduling theory, most problems are concerned with the minimization of certain functions of the completion times of the jobs. This type of objectives relate in certain ways to customer satisfaction since they tend to result in schedules that have either early completion times or on-time completions of the jobs. In reality, there are other important aspects in the evaluation of schedules, namely, aspects that are related to the machines themselves. When a job is assigned to a machine, the assignment results in a cost (or profit) that depends on the job as well as on the machine. The machines' objective may be the minimization of the total cost incurred by all the machines or the maximum cost incurred by any machine.

This type of multi-criteria scheduling problem occurs nowadays quite often in manufacturing as well as in services industries. Consider, for example, a manufacturing company with multiple plants in different locations. The production cost of a customer order at one plant may be completely different from the production cost at another plant. The company has to worry about the timing of the production (in order to provide good customer service) as well as the production cost (in order for the company to remain profitable).

In the services industries also, such multi-criteria scheduling problems have become more and more important in recent years. Consider, for example, an organization that provides professional services and has, say, m service providers (e.g., medical doctors, teams of consultants, lawyers, etc.) and n tasks (e.g., patients, projects, legal cases, etc.). Each task has to be handled by one of the service providers. A service provider could be regarded as a "machine" and a task may be regarded as a "job". The service providers in such an environment are typically not identical, i.e., different providers have different skill sets and different experience levels and charge therefore differently. The tasks may also be different in such a way that each may require a provider with a specific skill

* Corresponding author. Tel.: +1 212 998 0287; fax: +1 212 995 4227.
E-mail addresses: klee5@york.cuny.edu (K. Lee), leung@cis.njit.edu (J.Y-T. Leung), zhjia@mail.ustc.edu.cn (Z.-h. Jia), liwenhua@zzu.edu.cn (W. Li), mpinedo@stern.nyu.edu (M.L. Pinedo), bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

[1] Work supported in part by the NSF Grant CMMI-0969830.
[2] Work supported in part by the Science Foundation of Anhui University Grant 33050044, NSFC Grant 71171184, and China Scholarship Council Grant 201206505002.
[3] Work supported in part by the NSFC Grant 11171313.
[4] Work supported in part by the NSF Grant CMMI-0969755.
[5] Work supported in part by the National Science Council of Taiwan Grant NCS-101-2918-I-009-008.

http://dx.doi.org/10.1016/j.ejor.2014.03.026
0377-2217/

set. From the organization's point of view (e.g., a clinic, a consulting company, a legal firm, etc.), the objective is to minimize the total completion time of the tasks or to minimize the latest completion time so as to increase the clients' satisfaction levels, while at the same time minimize the total service providers' cost in order to maximize the organization's profit or minimize the maximum profit of any service provider in order to balance the profits over the service providers. Balancing the profits of the service providers is an important goal in maintaining the morale of the employees in the company.

Clearly, the framework considered in this paper applies to any organization that has to assign a resource to a specific task or project while taking into account objectives related to customer satisfaction levels as well as objectives related to the costs of utilization of resources.

In this paper we consider such bi-criteria scheduling problems with the second objective (either maximum machine cost or total machine cost) to be minimized subject to the constraint that the first objective (either makespan or total completion time) is at its minimum.

Many papers have dealt with bi- or multi-objective scheduling problems and there are several survey papers in this area. T'kindt and Billaut (2001) and T'kindt and Billaut (2006) presented a review on scheduling problems with various different machine environments including single machine, parallel machines, and flowshop environments. Hoogeveen (2005) paid more attention to due-date related objectives and scheduling with controllable processing times. Lei (2009) provided a more recent review. Nonetheless, most research initiatives focused on two or more scheduling objectives that are all related to the customers' perspectives. Thus, all the objectives are typically functions of the completion times.

Typical examples are papers that focus on minimizing two traditional scheduling objectives. Smith (1956) considered a single machine scheduling problem to minimize the total completion time subject to the constraint that all jobs should be completed at their due dates or before. Leung and Young (1989) studied a parallel machine scheduling problem to minimize the makespan subject to the constraint that the total completion time is at its minimum. Leung and Pinedo (2003) considered a parallel machine scheduling problem to minimize the total completion time subject to the constraint that the makespan is at its minimum.

Multi-objective scheduling can be found in many different settings. There are studies on agent scheduling problems that focus on multiple objectives which agents selfishly optimize; see, for example, Agnetis, Mirchandani, Pacciarelli, and Pacifici (2004), Agnetis, Pacciarelli, and Pacifici (2007), Lee, Choi, Leung, and Pinedo (2009) and Leung, Pinedo, and Wan (2010). Distributed scheduling can also lead to multi-objective scheduling problems in which each job's objective as well as the overall objective function are optimized at the same time. Several other papers focus on coordination mechanisms; see, for example, Lee, Leung, and Pinedo (2011) and Lee, Leung, and Pinedo (2012).

In the scheduling literature, there are two streams of research that deal with service providers' objectives. The first and most popular stream involves a machine activation cost. The number of machines used is a variable and the overall objective is to minimize the sum of a traditional scheduling objective (e.g., makespan) and the cost of activating machines; see, for example, Imreh and Noga (1999) and Dòsa and He (2004).

Another approach assumes a machine assignment cost. When a job is scheduled on a machine, a machine assignment cost is incurred. In most cases, the objective is either the sum of a scheduling objective and the total machine assignment cost or a prioritization of the two objectives; see, for example, Shmoys and Tardos (1993), Vignier, Sonntag, and Portmann (1999), T'kindt, Billaut, and Proust (2001), Khuller, Li, and Saha (2010), and Leung, Lee, and Pinedo (2012). Khuller et al. (2010) considered a problem with machine activation cost as well as machine assignment cost. Shmoys and Tardos (1993) formulated a general bi-objective scheduling problem with machine assignment costs as a generalized assignment problem. They considered a prioritization of the two objectives and provided an approximation approach by combining a linear programming algorithm with a rounding technique. Leung et al. (2012) described a bi-objective scheduling problem to minimize the scheduling objective and the total machine assignment cost at the same time. The makespan and the total completion time were considered as scheduling objectives and both non-preemptive and preemptive versions were dealt with. They considered weighted combinations of the two objectives as well as prioritizations of the two objectives. They restricted themselves to an analysis of the problems in terms of their time complexity under different assumptions; they proved NP-hardness for some cases and presented polynomial time algorithms for other cases.

In this paper, we will follow in our problem formulations the framework presented in Leung et al. (2012). However, we consider several aspects that are different. Unlike Leung et al. (2012), we only focus on a prioritization of objectives, which implies that we try to optimize one objective first and then try to optimize the second one with the first one being at its minimum. Also, we develop approximation algorithms with their worst-case performance analyses.

In Section 2, we formally describe the problem and introduce notations. We provide in Section 3 approximation algorithms for the problem in which the total completion time and maximum machine cost have to be minimized. We present in Section 4 approximation algorithms for minimizing the makespan and the total machine cost. We conclude the paper in the last section with a discussion on future research directions.

2. Problem description

We consider the problem of scheduling n jobs on m identical machines to minimize at the same time a customers' objective and a service providers' objective. The processing time of job j is p_j. Let C_j denote the completion time of job j. The customers' objective is to minimize either the total completion time ($\sum C_j$) or the last completion time, commonly referred to as the makespan ($C_{\max}$). In addition, a cost c_{ij} is incurred when job $j, 1 \leqslant j \leqslant n$, is processed on machine $i, 1 \leqslant i \leqslant m$. Let $x_{ij} = 1$ if job j is processed on machine i and $x_{ij} = 0$ otherwise. Thus, the total machine cost, for short *TMC*, is $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$, and the maximum machine cost, for short *MMC*, is $\max_{i=1}^{m}\{\sum_{j=1}^{n} c_{ij}x_{ij}\}$. The service providers' objective is to minimize either *TMC* or *MMC*. The goal is to find a schedule that minimizes either the total completion time or the makespan in addition to the total machine cost or the maximum machine cost. We consider in what follows only nonpreemptive scheduling models. Let $M = \{1, 2, \ldots, m\}$ denote the set of m machines and $J = \{J_1, J_2, \ldots, J_n\}$ denote the set of n jobs.

We consider three types of cost functions:

(i) $c_{ij} = c_j$,
(ii) $c_{ij} = a_i + c_j$, and
(iii) $c_{ij} = a_i \times c_j$,

where a_i is a non-negative integer parameter belonging to machine i and c_j is a non-negative integer parameter belonging to job j. The first cost function (i) is the case of identical costs; i.e., the processing cost of a job is independent of the machine the job is assigned to. The second cost function (ii) plays a role when the assignment of a job to a machine involves an additional setup cost (a_i) that is

machine dependent. The third cost function (iii) plays a role when one machine operates in a more cost efficient manner than another machine; i.e., the efficiency factor of machine i is captured by the factor a_i. For cases (ii) and (iii), we can assume, without loss of generality, that $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_m$. Obviously, (i) is a special case of (ii) when $a_i = 0$ for all $i \in M$ and (i) is a special case of (iii) when $a_i = 1$ for all $i \in M$. Throughout the paper, we also assume that, for case (ii) $a_1 = 0$ and for case (iii) $a_1 = 1$.

Basically, the bi-criteria problems we study can be referred to as LEX (γ_1, γ_2). In LEX (γ_1, γ_2), the γ_1 represents the primary objective and the γ_2 represents the secondary objective; i.e., the problem is to minimize objective γ_2 subject to the constraint that objective γ_1 is at its minimum.

Since we consider either $\sum C_j$ or $C_{\max}$ as the customers' objective, and consider either *TMC* or *MMC* as the service providers' (machines') objective, we have four possible combinations: $(\sum C_j, MMC)$, $(\sum C_j, TMC)$, $(C_{\max}, MMC)$ and $(C_{\max}, TMC)$. Leung et al. (2012) already showed that LEX $(\sum C_j, TMC)$ can be solved optimally in polynomial time. The problem LEX $(C_{\max}, MMC)$ is not completely investigated, but a special case of this problem was considered by Lee, Leung, and Pinedo (2013). In this paper we will focus on the following two problems:

1. LEX $(\sum C_j, MMC)$; i.e., minimize the maximum machine cost subject to the constraint that $\sum C_j$ is minimized.
2. LEX $(C_{\max}, TMC)$; i.e., minimize the total machine cost subject to the constraint that $C_{\max}$ is minimized.

The two scheduling problems above can be shown to be unary NP-hard via a simple reduction from the 3-Partition problem (Garey & Johnson, 1979). For LEX $(C_{\max}, TMC)$, the primary objective can be reduced from the 3-Partition problem. For LEX $(\sum C_j, MMC)$, the secondary objective can be reduced from 3-Partition.

Because of the computational complexity of the problems, we are interested in fast heuristics. We say that heuristic $\mathcal{A}$ is an (α, β)-approximation for problem LEX (γ_1, γ_2) if $\mathcal{A}$ produces a schedule where $\gamma_1(\mathcal{A}) \leqslant \alpha \times \gamma_1^*$ and $\gamma_2(\mathcal{A}) \leqslant \beta \times \gamma_2^*$. Here, γ_1^* is the optimal value for the first objective, and γ_2^* is the optimal value for the second objective provided that the schedule attains the optimal value (γ_1^*) for the first objective. For example, if $\mathcal{A}$ is a $(\frac{3}{2}, 2)$-approximation for LEX $(C_{\max}, TMC)$, then $C_{\max}(\mathcal{A}) \leqslant \frac{3}{2} \times C_{\max}^*$ and $TMC(\mathcal{A}) \leqslant 2 \times TMC^*$, where TMC^* is the optimal value for the total machine cost provided that the makespan of the schedule is $C_{\max}^*$.

Shmoys and Tardos (1993) consider a bi-objective scheduling problem that is more general than those we consider; they assume processing times p_{ij} (i.e., the processing time depends on the job as well as on the machine), and assignment costs c_{ij} (i.e., the assignment costs depend also on the job as well as on the machine). They do develop a polynomial time approximation algorithm for this framework; however, their algorithm has a higher time complexity than ours. Moreover their algorithm seems to be hard to implement, since it depends on the solution of a Linear Program, followed by the solution of a matching problem. We will compare our results with their results in the conclusion section.

The following notation will be used throughout this paper. Let

$$p_{\max} = \max(p_1, \ldots, p_n)$$

and

$$c_{\max} = \max(c_1, \ldots, c_n).$$

For a given set of jobs $J\prime$, let $P(J\prime) = \sum_{j \in J\prime} p_j$ and $V(J\prime) = \sum_{j \in J\prime} c_j$. Let $C_{\max}^*$ and $\sum C_j^*$ be the optimal objective values of the makespan and the total completion time, respectively. Let TMC^* and MMC^* be the optimal objective values of *TMC* and *MMC*, respectively, when the primary objective is at its minimum.

3. Total completion time and maximum machine cost

In this section we consider the problem LEX $(\sum C_j, MMC)$. It is well-known that the Shortest-Processing-Time first (SPT) rule yields an optimal schedule for $\sum C_j$; see Pinedo (2012). The SPT rule, in its most general form, operates as follows: Let integer k be such that $n = (k-1)m + \ell$ for some integer $\ell, 1 \leqslant \ell \leqslant m$. Sort the jobs in ascending order of their processing times, i.e., $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_n$. We refer to the last m jobs (i.e., jobs $n-m+1, n-m+2, \ldots, n,$) as the rank-k jobs, the second last m jobs (i.e., jobs $n-2m+1, \ldots, n-m$) as the rank-$(k-1)$ jobs, and so on; the first ℓ jobs (i.e., jobs $1, \ldots, \ell$) are the rank-1 jobs. The rank-1 jobs are assigned first, one job per machine, followed by the rank-2 jobs, again one job per machine. This process is repeated until jobs in the last rank (rank-k) are assigned. Clearly, there may be many different schedules that can be generated according to this SPT rule and all these schedules have a minimum $\sum C_j$. A schedule generated by the SPT rule is referred to as an SPT schedule. While limiting ourselves to SPT schedules, we try to find for the problem LEX $(\sum C_j, MMC)$ a $(1, \beta)$-approximation algorithm with the smallest possible β.

In order to guarantee a minimum $\sum C_j$, we will only consider SPT schedules. The following procedure to setup ranks will be used as a subroutine in the algorithms proposed.

Setup Ranks

(1) Sort the jobs in ascending order of their processing time; i.e., $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$. Let $n = (k-1)m + \ell$, where $1 \leqslant \ell \leqslant m$.
(2) Assign the last m jobs to set R_k, the second last m jobs to set R_{k-1}, and so on. Assign the first ℓ jobs to set R_1.
(3) Return $\langle R_1, R_2, \ldots, R_k \rangle$.

Note that the **Setup Ranks** procedure returns a possible rank composition and there may be a different rank composition for SPT schedules when there are jobs with equal processing times.

3.1. The special case $c_{ij} = c_j$

We first consider the cost function $c_{ij} = c_j$ for all $1 \leqslant i \leqslant m$. We consider a heuristic, **H1**, that schedules jobs according to SPT, but jobs in each rank are judiciously assigned to the machine so as to minimize the maximum machine cost. For each $1 \leqslant i \leqslant m$, we let B_i denote the set of jobs assigned to machine i. Below is a description of **H1**.

Heuristic H1(J, M)
(1) Call **Setup Ranks**
(2) $B_i := \emptyset$ for each $1 \leqslant i \leqslant m$.
(3) For r from 1 to k do

(a) Sort the machines in ascending order of their costs; i.e., $V(B_{l_1}) \leqslant V(B_{l_2}) \leqslant \cdots \leqslant V(B_{l_m})$.
(b) Let $\{J_{i_1}, J_{i_2}, \ldots, J_{i_y}\}$ be all the jobs in R_r (i.e., R_r has y jobs) where the jobs are sorted in descending order of their costs; i.e., $c_{i_1} \geqslant c_{i_2} \geqslant \cdots \geqslant c_{i_y}$.
(c) For each j from 1 to y do

(i) $B_{l_j} := B_{l_j} \cup \{J_{i_j}\}$.

The analysis of the performance bound for heuristic **H1** is similar to the analysis of a list scheduling algorithm for the classical $P\|C_{\max}$ problem. Before we prove the bound for heuristic **H1**, we need to prove the following lemma.

Lemma 1. *Let $V(i,r)$ denote the total cost of jobs assigned to machine i after assigning the jobs in rank r by heuristic* **H1**. *Then, we have* $0 \leqslant |V(i,r) - V(h,r)| \leqslant c_{\max}$ for $1 \leqslant i,h \leqslant m$.

Proof. We will prove the lemma by induction on r. The base case, $r = 1$, is obvious. This is because each of B_i and B_h has at most one job and thus $0 \leqslant V(i,1) \leqslant c_{\max}$ and $0 \leqslant V(h,1) \leqslant c_{\max}$. Thus, we have

$$0 \leqslant |V(i,1) - V(h,1)| \leqslant c_{\max}.$$

Assume the lemma is true for all ranks up to $r-1$. We want to show that the lemma is true for rank-r. If $V(i,r-1) = V(h,r-1)$, then it immediately holds that $|V(i,r) - V(h,r)| \leqslant c_{\max}$. Otherwise, by symmetry, we may assume that $V(i,r-1) > V(h,r-1)$. Then, from the induction hypothesis, we have

$$0 < V(i,r-1) - V(h,r-1) \leqslant c_{\max}.$$

By heuristic **H1**, the job assigned to machine h has a cost that is at least as large as the cost of the job assigned to machine i. Since the cost of the job assigned to machine h is simply $V(h,r) - V(h,r-1)$ and the cost of the job assigned to machine i is simply $V(i,r) - V(i,r-1)$, we have

$$V(h,r) - V(h,r-1) \geqslant V(i,r) - V(i,r-1),$$

and hence

$$V(i,r) - V(h,r) \leqslant V(i,r-1) - V(h,r-1) \leqslant c_{\max}.$$

Moreover,

$$V(i,r) - V(h,r) \geqslant V(i,r-1) - V(h,r) \geqslant V(i,r-1) - (V(h,r-1) + c_{\max}) \geqslant (V(i,r-1) - V(h,r-1)) - c_{\max} \geqslant -c_{\max},$$

where the last inequality is due to the fact that $V(i,r-1) > V(h,r-1)$. Therefore, we have

$$|V(i,r) - V(h,r)| \leqslant c_{\max}.$$

By induction, the lemma holds for all r. □

Theorem 1. *Heuristic* **H1** *is a $(1, 2-1/m)$-approximation for the problem LEX ($\sum C_j, MMC$) when $c_{ij} = c_j$ for all $1 \leqslant i \leqslant m$. Moreover, the bound is tight.*

Proof. Since heuristic **H1** schedules jobs according to the SPT rule, it generates an optimal solution for the $\sum C_j$ objective. Consider now the *MMC* objective. By Lemma 1, we have

$$\max_{i\in M}\{V(i,k)\} - \min_{i\in M}\{V(i,k)\} \leqslant c_{\max} \leqslant MMC^*.$$

There are two cases to consider.

Case 1: $\min_{i\in M}\{V(i,k)\} \leqslant \frac{m-1}{m} MMC^*$.
In this case, we have

$$\max_{i\in M}\{V(i,k)\} \leqslant \min_{i\in M}\{V(i,k)\} + c_{\max} \leqslant \frac{m-1}{m} MMC^* + MMC^* = \left(2 - \frac{1}{m}\right) MMC^*.$$

Case 2: $\min_{i\in M}\{V(i,k)\} > \frac{m-1}{m} MMC^*$.

A lower bound for the total cost on all the machines is $\max_{i\in M}\{V(i,k)\} + (m-1) \times \min_{i\in M}\{V(i,k)\}$; i.e., one machine has the maximum cost and $m-1$ machines have the minimum cost. Thus, we have

$$\max_{i\in M}\{V(i,k)\} + (m-1) \times \min_{i\in M}\{V(i,k)\} \leqslant \sum_{i=1}^{m} V(i,k) \leqslant m \times MMC^*,$$

and hence,

$$\max_{i\in M}\{V(i,k)\} \leqslant m \times MMC^* - (m-1) \times \min_{i\in M}\{V(i,k)\} < \left\{m - \frac{(m-1)^2}{m}\right\} MMC^* = \left(2 - \frac{1}{m}\right) MMC^*.$$

To show that the bound is tight, consider $3m$ jobs, where the rank-1 jobs all have a processing time of 1 unit, the rank-2 jobs all have a processing time of 2 units, and the rank-3 jobs all have a processing time of 3 units. The costs of the rank-1 and rank-2 jobs are $0, 1, 2, \ldots, m-1$, while the cost of the rank-3 jobs are zero except one job that has a cost of m. Heuristic **H1** will produce a schedule with $MMC(\mathbf{H1}) = 2m-1$, while the optimal schedule has $MMC^* = m$. Therefore, the ratio is $2 - 1/m$. □

3.2. The special case $c_{ij} = a_i + c_j$

We now consider the cost function $c_{ij} = a_i + c_j$. We propose a different heuristic, **H2**, which is slightly more complicated.

First, the **Setup Ranks** routine is called to obtain a possible rank composition for the minimum total completion time objective. In a SPT schedule with k ranks such that $n = k(m-1) + \ell, \ell$ machines have k jobs each and $(m-\ell)$ machines have $(k-1)$ jobs each. Thus, the cost at machine i is $|B_i|a_i + V(B_i)$ where $|B_i| = k$ or $(k-1)$. The basic idea of heuristic **H2** is to construct two schedules – the first one for jobs in rank 1 and the second one for jobs in ranks $2, \ldots, k$ – and combine these to create a final schedule.

For the first schedule, the jobs in R_1 are sorted in descending order of their costs; i.e., $c_{j_1} \geqslant c_{j_2} \geqslant \cdots \geqslant c_{j_\ell}$. Then, for each $1 \leqslant i \leqslant \ell$, job J_{j_i} is assigned to machine i and A_i is set to be $k \times a_i + c_{j_i}$. For each $\ell + 1 \leqslant i \leqslant m, A_i$ is set to be $(k-1) \times a_i$. Then, the machines are sorted in descending order of A_i; i.e., the machines $l_1, l_2, \ldots, l_m$ are such that

$$A_{l_1} \geqslant A_{l_2} \geqslant \cdots \geqslant A_{l_m}.$$

Now, heuristic **H1** is then called to generate the second schedule with the set of jobs $R_2 \cup R_3 \cup \cdots \cup R_k$. When heuristic **H1** returns, let B_i be the set of jobs assigned to machine i. Then, the B_i's are sorted in an ascending order of the total cost of the jobs in B_i; i.e.,

$$V(B_{i_1}) \leqslant V(B_{i_2}) \leqslant \cdots \leqslant V(B_{i_m}).$$

Finally, for the minimum maximum cost, for each $1 \leqslant h \leqslant m$, the jobs in B_{i_h} are assigned to machine l_h. The cost of machine l_h is $A_{l_h} + V(B_{i_h})$. Shown below is a description of heuristic **H2**.

Heuristic H2 (J, M)

(1) Call **Setup Ranks**
(2) Let $\{J_{j_1}, J_{j_2}, \ldots, J_{j_\ell}\}$ be the jobs in R_1, sorted in descending order of their costs; i.e., $c_{j_1} \geqslant c_{j_2} \geqslant \cdots \geqslant c_{j_\ell}$.
(3) Sort the machines in ascending order of a_i; i.e., $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$.
(4) For i from 1 to ℓ, assign job J_{j_i} to machine i and let $A_i = k \times a_i + c_{j_i}$.
(5) For i from $\ell + 1$ to m, let $A_i = (k-1) \times a_i$.
(6) Sort A_i values such that $A_{l_1} \geqslant A_{l_2} \geqslant \cdots \geqslant A_{l_m}$.
(7) Call heuristic **H1**$(R_2 \cup R_3 \cup \cdots \cup R_k, M)$ to schedule jobs in $R_2 \cup R_3 \cup \cdots \cup R_k$ on m identical machines. When heuristic **H1** returns, let B_i be the set of jobs assigned to machine i in the schedule produced by heuristic **H1**.
(8) Sort the B_i in ascending order of $V(B_i)$; such that $V(B_{i_1}) \leqslant V(B_{i_2}) \leqslant \cdots \leqslant V(B_{i_m})$.
(9) For each $1 \leqslant h \leqslant m$, assign the jobs in B_{i_h} to machine l_h and let the cost of machine l_h be $A_{l_h} + V(B_{i_h})$.

Theorem 2. *Heuristic* **H2** *is a* $(1, 2-1/m)$*-approximation for the problem LEX* $(\sum C_j, MMC)$ *when* $c_{ij} = a_i + c_j$ *for all* $1 \leqslant i \leqslant m$*. Moreover, the bound is tight.*

Proof. For each $1 \leqslant i \leqslant m$, let V_i be $V(B_i)$ for simplicity. Recall that B_i is the set of jobs scheduled on machine i from $R_2 \cup R_3 \cup \cdots \cup R_k$. Thus, the cost of machine i is $A_i + V_i$.

We claim that $\max_{i\in M}\{A_i\} \leqslant MMC^*$ by the following reason. We consider another set of jobs J' where all costs of jobs in rank 1 is the same as the original problem instance and all costs of jobs in ranks $2, \ldots, k$ become zero. Then, $\max_{i\in M}\{A_i\}$ is the optimal *MMC* value for J'. Obviously, the optimal *MMC* value for J is greater than or equal to the optimal *MMC* value for J'. Therefore, $\max_{i\in M}\{A_i\} \leqslant MMC^*$.

Thus, $|A_i - A_h| \leqslant MMC^*$ for $1 \leqslant i, h \leqslant m$. By Lemma 1, we have $|V_i - V_h| \leqslant MMC^*$ for $1 \leqslant i, h \leqslant m$.

Without loss of generality, we may assume that $A_i \geqslant A_h$. Thus, $0 \leqslant A_i - A_h \leqslant MMC^*$. Since $A_i \geqslant A_h$, we have $V_i \leqslant V_h$, by the nature of heuristic **H2**. Thus, we have

$$(A_i + V_i) - (A_h + V_h) \leqslant A_i - A_h \leqslant MMC^*,$$

and

$$(A_i + V_i) - (A_h + V_h) \geqslant V_i - V_h \geqslant -MMC^*.$$

Therefore, we have $|(A_i + V_i) - (A_h + V_h)| \leqslant MMC^*$. We now consider two cases. Let $A_i + V_i = \max_{l\in M}\{A_l + V_l\}$ and $A_h + V_h = \min_{l\in M}\{A_l + V_l\}$.

Case 1: $A_h + V_h \leqslant \frac{m-1}{m} MMC^*$.
In this case, we have

$$A_i + V_i \leqslant A_h + V_h + MMC^* \leqslant \frac{m-1}{m} MMC^* + MMC^* = \left(2 - \frac{1}{m}\right) MMC^*.$$

Case 2: $A_h + V_h > \frac{m-1}{m} MMC^*$.
In this case, we have

$$A_i + V_i + (m-1)(A_h + V_h) \leqslant \sum_{l\in M} (A_l + V_l) \leqslant m \times MMC^*.$$

Therefore, we have

$$A_i + V_i \leqslant m \times MMC^* - (m-1)(A_h + V_h) < \left\{ m - \frac{(m-1)^2}{m} \right\} MMC^*$$
$$= \left(2 - \frac{1}{m}\right) MMC^*.$$

The set of jobs achieving a tight bound in Theorem 1 can be used to show the tight bound for this theorem as well. Simply let $a_i = 0$ for all i. □

3.3. The special case $c_{ij} = a_i \times c_j$

We now consider the cost function $c_{ij} = a_i \times c_j$. Recall that we assume that $a_1 = 1$ and $a_i \leqslant a_{i+1}$ for $i = 1, \ldots, m-1$. We can assume that $n = km$; otherwise without changing the problem structure we can add $\lceil n/m \rceil m - n$ dummy jobs with zero processing times and zero costs.

We consider the third heuristic, **H3**, which is similar to heuristic **H1**. Again, jobs are ordered according to the SPT rule, rank by rank. Jobs within the same rank are assigned to machines as follows: Jobs are considered in descending order of their cost c_j. When job j is being assigned, we consider all the eligible machines and assign job j to that machine i that results in the smallest total cost on machine i. After job j is assigned to machine i, machine i becomes ineligible until we deal with the jobs in the next rank. The heuristic is described as follows.

Heuristic H3(J, M)

(1) Call **Setup Ranks**
(2) $B_i := \emptyset$ for each $1 \leqslant i \leqslant m$.
(3) For r from 1 to k do
  (a) Let $M' = \{1, 2, \ldots, m\}$ be the eligible machines.
  (b) Let $R_i = \{J_{i_1}, J_{i_2}, \ldots, J_{i_m}\}$, sorted in descending order of their cost; i.e., $c_{i_1} \geqslant c_{i_2} \geqslant \cdots \geqslant c_{i_m}$.
  (c) For each j from 1 to m do
    (i) $l := \arg\min_{f\in M'}\{V(B_f) + a_f \times c_{i_j}\}$, with ties broken by choosing the machine with a larger a_f.
    (ii) $B_l := B_l \cup \{J_{i_j}\}$.
    (iii) $M' := M' \setminus \{l\}$.

Even though heuristic **H3** looks reasonable, its theoretical bound may be poor. We can derive a lower bound of the ratio from the literature. Cho and Sahni (1980) considered a list scheduling algorithm for a uniform parallel machine scheduling problem to minimize the makespan. They showed that the worst case performance ratio of the list scheduling algorithm is at least $O(\log m)$ by presenting a problem instance. We can construct a problem instance from the prohlem instance in Cho and Sahni (1980) and show that the approximation ratio of heuristic **H3**, for problem LEX $(\sum C_j, MMC)$ when $c_{ij} = a_i \times c_j$ for all $1 \leqslant i \leqslant m$, is at least $(1, O(\log m))$. However, we will prove an unbounded approximation ratio of heuristic **H3** in the following theorem.

Theorem 3. *The approximation ratio of heuristic* **H3***, for problem LEX* $(\sum C_j, MMC)$ *when* $c_{ij} = a_i \times c_j$ *for all* $1 \leqslant i \leqslant m$*, is at least* $(1, O(a_m))$*.*

Proof. Since heuristic **H3** schedules jobs according to SPT, it generates an optimal solution for the $\sum C_j$ objective. Thus, we will prove the lower bound of $MMC(\mathbf{H3})/MMC^*$ by providing an example.

Consider two machines and four jobs. The four jobs have identical processing times, say 1 unit. Their costs are: $c_1 = c_2 = 1, c_3 = c_4 = a$, where $a \gg 1$. For machine 1, $a_1 = 1$ and for machine 2, $a_2 = a$.

Suppose the algorithm puts jobs 1 and 2 in the first rank, and jobs 3 and 4 in the second rank, then $V(B_1) = V(B_2) = a + 1$. In this case, we have $MMC(\mathbf{H3}) = a(a+1)$.

On the other hand, the optimal solution puts jobs 1 and 3 in the first rank, and jobs 2 and 4 in the second rank. Then we assign the job with the smallest cost in each rank to machine 2. In this case, we have jobs 1 and 2 in machine 2, and jobs 3 and 4 in machine 1. We have $MMC^* = 2a$.

The approximation ratio is $a(a+1)/(2a) = (a+1)/2$, which approaches infinity when a gets large. This completes the proof. □

Since heuristic **H3** does not have a constant approximation ratio, we consider a special case of LEX $(\sum C_j, MMC)$ where $c_{ij} = a_i \times c_j, a_1 = \cdots = a_{m-1} = 1$ and $a_m \geqslant 1$. This setting implies that there is only one inefficient machine. For this special case, we present an approximation algorithm with a constant ratio.

The basic idea is that we consider two cases for the a_m value separately. When a_m is not so large, we just use heuristic **H1** and modify the schedule. When a_m is large, we need to assign jobs with small costs to machine m within the structure of SPT schedules. However, if we fix ranks by **Setup Ranks** like heuristics **H1, H2** and **H3**, it might be impossible to assign jobs with small costs to machine m. Therefore, we first consider a procedure to assign a job from each rank to machine m to minimize the total cost of jobs assigned to machine m among all possible rank compositions. The algorithm can be described as follows:

Minimum Cost for Machine m

(1) Call **Setup Ranks**
(2) Construct a weighted bipartite graph $G = (U_1 \cup U_2, E; w)$ where
  (a) $U_1 = \{j | J_j \in J\}, U_2 = \{1, \ldots, k+1\}$
  (b) $E_1 = \{(\mu, v) | \min_{j \in R_v}\{p_j\} \leqslant p_\mu \leqslant \max_{j \in R_v}\{p_j\} \text{for} \mu \in U_1, v \in U_2 \setminus \{k+1\}\}$,
  $E_2 = \{(\mu, k+1) | \mu \in U_1\}$ and
  $E = E_1 \cup E_2$.
  (c) $w_{\mu,v} = c_\mu$ for $\mu \in U_1, v \in U_2 \setminus \{k+1\}$.
(3) Solve the following transportation problem on G and obtain an optimal solution $y^*_{\mu,v}$.

$$\begin{aligned} \text{minimize} \quad & \sum_{(\mu,v)\in E_1} w_{\mu,v} y_{\mu,v} \\ \text{subject to} \quad & \sum_{v:(\mu,v)\in E} y_{\mu,v} = 1 \quad \text{for } \mu \in U_1 \\ & \sum_{\mu:(\mu,v)\in E_1} y_{\mu,v} = 1 \quad \text{for } v \in U_2 \setminus \{k+1\} \\ & \sum_{\mu:(\mu,k+1)\in E_2} y_{\mu,k+1} = n-k \\ & y_{\mu,v} \in \{0,1\} \quad \text{for } (\mu,v) \in E \end{aligned}$$

(4) Job J_μ is scheduled on rank v of machine m if and only if $y^*_{\mu,v} = 1$ for $(\mu, v) \in E_1$.

Instead of solving the mathematical program described above through a transportation algorithm, we will solve it by an alternative procedure that will significantly reduce the running time. This alternative procedure is described in the Appendix. Its running time is $O(n \log n)$.

Note that $V(B_i)$ is the sum of c_j values of the jobs assigned to machine i. The cost of machine i is $V(B_i)$ for $i = 1, \ldots, m-1$ and is $a_m V(B_m)$ for $i = m$ and thus

$$MMC = \max\{\max\{V(B_i) | i \in \{1, \ldots, m-1\}\}, a_m V(B_m)\}.$$

Now, we are ready to present an algorithm with a constant ratio for LEX $(\sum C_j, MMC)$ where $c_{ij} = a_i \times c_j, a_1 = \cdots = a_{m-1} = 1$ and $a_m \geqslant 1$.

Heuristic H4

(1) If $a_m \geqslant 2$,
  (a) Call **Minimum Cost for Machine** m. Let J' be the set of jobs scheduled on machine m.
  (b) Apply **H1**$(J \setminus J', M \setminus \{m\})$ for the schedule of the remaining jobs on machines $1, \ldots, m-1$.
(2) If $1 \leqslant a_m < 2$,
  (a) Apply **H1**(J, M) and obtain B_i for $i \in M$.
  (b) Let $h := \arg\min\{V(B_i) | i \in M\}$. Swap B_m and B_h.

Theorem 4. *Heuristic* **H4** *is a* $(1, 2 - \frac{1}{2(m-1)})$*-approximation for problem LEX* $(\sum C_j, MMC)$ *when* $c_{ij} = a_i \times c_j, a_1 = \cdots = a_{m-1} = 1$ *and* $a_m \geqslant 1$.

Proof. *Case 1.* Suppose that $a_m \geqslant 2$. We consider two sub-cases: (i) *MMC* is determined at machine m, and (ii) *MMC* is determined at machine i, for $i \neq m$.

(i) If *MMC* is determined by machine m, by the property of the schedule by **Minimum Cost for Machine** m, the current schedule is optimal.
(ii) Suppose that *MMC* is determined by a machine that is not machine m. By Lemma 1, $|V(B_i) - V(B_h)| \leqslant c_{\max}$ for $i, h \in M \setminus \{m\}$.

If

$$\min_{i\in M\setminus\{m\}}\{V(B_i)\} \leqslant \left(1 - \frac{1}{2(m-1)}\right) MMC^*,$$

then

$$\max_{i\in M\setminus\{m\}}\{V(B_i)\} \leqslant \min_{i\in M\setminus\{m\}}\{V(B_i)\} + c_{\max} \leqslant \left(2 - \frac{1}{2(m-1)}\right) MMC^*.$$

If

$$\min_{i\in M\setminus\{m\}}\{V(B_i)\} > \left(1 - \frac{1}{2(m-1)}\right) MMC^*,$$

then by the lower bound for MMC^*, we have $MMC^* \geqslant V(J)/(m - 1 + 1/a_m)$. We consider the following relationship:

$$\begin{aligned} &\max_{i\in M\setminus\{m\}}\{V(B_i)\} + (m-2)\min_{i\in M\setminus\{m\}}\{V(B_i)\} \leqslant V(J) \\ &\leqslant (m - 1 + 1/a_m) MMC^*. \end{aligned}$$

Thus, we have

$$\begin{aligned} &\max_{i\in M\setminus\{m\}}\{V(B_i)\} < (m - 1 + 1/a_m) MMC^* \\ &- (m-2)\left(1 - \frac{1}{2(m-1)}\right) MMC^* = \left(\frac{3}{2} + \frac{1}{a_m} - \frac{1}{2(m-1)}\right) MMC^* \\ &\leqslant \left(2 - \frac{1}{2(m-1)}\right) MMC^*. \end{aligned}$$

Case 2. Suppose that $1 \leqslant a_m < 2$. We again consider two subcases: (i) *MMC* is determined by machine m, and (ii) *MMC* is determined by machine i, for $i \neq m$.

(i) Suppose that *MMC* is determined at machine m. If $V(B_m) > \frac{1}{m}(m - 1 + \frac{1}{a_m}) MMC^*$, then $\sum_{i\in M} V(B_i) > (m - 1 + \frac{1}{a_m})\ MMC^*$, which is a contradiction. If $V(B_m) \leqslant \frac{1}{m}\ (m - 1 + \frac{1}{a_m}) MMC^*$, then we have

$$\begin{aligned} a_m V(B_m) &\leqslant \frac{a_m}{m}\left(m - 1 + \frac{1}{a_m}\right) MMC^* = \left(a_m\left(1 - \frac{1}{m}\right) + \frac{1}{m}\right) MMC^* \\ &\leqslant \left(2 - \frac{1}{m}\right) MMC^*. \end{aligned}$$

(ii) Suppose that *MMC* is determined at a machine that is not machine m. By Lemma 1, $|V(B_i) - V(B_h)| \leqslant c_{\max}$ for $i, h \in M$. If $V(B_m) \leqslant (1 - \frac{1}{m}) MMC^*$, then

$$\max_{i\in M\setminus\{m\}}\{V(B_i)\} \leqslant V(B_m) + c_{\max} \leqslant \left(2 - \frac{1}{m}\right) MMC^*.$$

If $V(B_m) > (1 - \frac{1}{m}) MMC^*$, then consider the relationship

$$\max_{i\in M}\{V(B_i)\} + (m-1)V(B_m) \leqslant V(J) \leqslant (m - 1 + 1/a_m) MMC^*.$$

Thus, we have

$$\begin{aligned} \max_{i\in M}\{V(B_i)\} &\leqslant (m - 1 + 1/a_m) MMC^* - (m-1)\left(1 - \frac{1}{m}\right) MMC^* \\ &= \left(\frac{1}{a_m} + \frac{m-1}{m}\right) MMC^* \leqslant \left(2 - \frac{1}{m}\right) MMC^*. \end{aligned}$$

Note that $2 - \frac{1}{m} \leqslant 2 - \frac{1}{2(m-1)}$ for $m \geqslant 2$. The proof is complete. □

4. Makespan and total machine cost

In this section we consider the problem LEX $(C_{\max}, TMC)$. For the cost function $c_{ij} = c_j$, this problem is equivalent to the problem $P||C_{\max}$, for which many heuristics have been proposed and analyzed. Therefore, we will not consider this special case here.

On both cost functions $c_{ij} = a_i + c_j$ and $c_{ij} = a_i \times c_j$, we will obtain an upper bound of the optimal makespan by the following steps. We first order the n jobs on the m machines according to the Largest-Processing-Time (LPT) rule, ignoring the cost of the jobs. The LPT rule schedules the jobs in descending order of their processing times. The next job will be assigned to the machine that finishes the earliest. The makespan of the LPT schedule is denoted by L and is used as an upper bound for the optimal makespan. It is well known that $L \leqslant (\frac{4}{3} - \frac{1}{3m})C^*_{\max}$, where $C^*_{\max}$ is the optimal makespan; see Graham (1969).

Since the primary objective, the makespan, is hard to optimize, we may consider using LPT to optimize the primary objective. Unfortunately, such an algorithm can perform very poorly with regard to the secondary objective for both cost functions $c_{ij} = a_i + c_j$ and $c_{ij} = a_i \times c_j$.

For the cost function $c_{ij} = a_i + c_j$, we consider an example with two machines having $a_1 = 0$ and $a_2 = a > 0$ and n jobs, where n is even and $n \geqslant 6$. The job information is as follows: $p_1 = p_2 = 1/2$ and $p_3 = \cdots = p_n = 1/(n-2)$ and $c_1 = \cdots = c_n = 0$. Obviously, $C^*_{\max} = 1$. In the LPT schedule, jobs J_1 and J_2 are scheduled on different machines and the remaining jobs are evenly scheduled on both machines. Thus, $TMC(LPT) = n/2 \times a$. However, in the optimal schedule, jobs J_1 and J_2 are scheduled on machine 2 and the other jobs are scheduled on machine 1. Thus, $TMC^* = 2 \times a$. Therefore, $TMC(LPT)/TMC^* = O(n)$.

For the cost function $c_{ij} = a_i \times c_j$, we consider an example with two machines having $a_1 = 1$ and $a_2 = a > 1$ and n jobs, where n is even and $n \geqslant 6$. The job information is as follows: $p_1 = p_2 = 1/2$ and $p_3 = \cdots = p_n = 1/(n-2)$ and $c_1 = c_2 = c/2$ and $c_3 = \cdots = c_n = 0$. Obviously, $C^*_{\max} = 1$. In the LPT schedule, jobs J_1 and J_2 are scheduled on different machines and the remaining jobs are evenly scheduled on both machines. Thus, $TMC(LPT) = \frac{c}{2}(1+a)$. However, in the optimal schedule, jobs J_1 and J_2 are scheduled on machine 1 and the other jobs are scheduled on machine 2. Thus, $TMC^* = c$. Since

$$\frac{TMC(LPT)}{TMC^*} = \frac{1+a}{2}$$

and a can be arbitrarily large, $TMC(LPT)/TMC^*$ is unbounded. Therefore, we need to design an algorithm that considers both objectives simultaneously.

4.1. The special case $c_{ij} = a_i + c_j$

For the cost function $c_{ij} = a_i + c_j$, we propose a fast heuristic, heuristic **H5**, which works as follows. Let the makespan of the LPT schedule be L. Then the jobs are sorted in ascending order of their processing times; i.e., $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$. The jobs are scheduled on the machines, starting with the first machine. Specifically, the first λ jobs are scheduled on machine 1, where λ is the smallest index such that the total processing time of the first λ jobs is larger than L. We then delete the λ jobs and machine 1 from consideration, and repeat the process on machine 2. This process is repeated until all jobs have been scheduled. Shown below is a description of heuristic **H5**.

Heuristic H5(J, M)

(1) Disregarding the cost of the jobs, schedule the n jobs on the m machines according to LPT. Let L denote the makespan of the LPT schedule.
(2) Sort the jobs in ascending order of their processing times. Let $J = (J_1, J_2, \ldots, J_n)$ be a list of the n jobs such that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$.
(3) Set $i = 1$.
(4) If the total processing time of the jobs in J is less than or equal to L, then schedule all jobs in J on machine i and stop.
(5) Let λ be the smallest integer such that the total processing time of the first λ jobs is larger than or equal to L.
(6) Assign the first λ jobs on machine i.
(7) Delete the first λ jobs from J.
(8) $i := i + 1$.
(9) Goto Step 4.

Let B_i and B^*_i be the sets of jobs assigned to machine i in the schedule generated by heuristic **H5** and in the optimal schedule, respectively. Let $B = \langle B_1, B_2, \ldots, B_m\rangle$ and $B^* = \langle B^*_1, B^*_2, \ldots, B^*_m\rangle$. With this notation, we will prove a lemma that is instrumental in proving Theorem 5.

Lemma 2. *Let B_i be the set of jobs assigned to machine i in the schedule generated by heuristic* **H5**. *Then, we have $\sum_{h=1}^{i} P(B_h) \geqslant \min\{i \times L, \sum_{j=1}^{n} p_j\}$ for all $i \in M$.*

Proof. If $P(B_h) \geqslant L$ for all $h, h = 1, \ldots, i$, then we have $\sum_{h=1}^{i} P(B_h) \geqslant i \times L$. Otherwise, there is a machine f such that $P(B_f) < L$ for $1 \leqslant f \leqslant i$. This means that all n jobs have been scheduled on machines $1, 2, \ldots, f$. Thus, we have $\sum_{h=1}^{i} P(B_h) = \sum_{j=1}^{n} p_j$. □

We are now ready to prove Theorem 5.

Theorem 5. *Heuristic* **H5** *is a $((\frac{7}{3} - \frac{1}{3m}), 1)$-approximation for problem LEX ($C_{\max}$, TMC) when $c_{ij} = a_i + c_j$.*

Proof. First, we want to show that heuristic **H5** would be able to schedule all the jobs on m machines so that each machine finishes by time $L + p_{\max}$. Suppose not. Then there is a job that cannot be scheduled on any machine such that it completes by time $L + p_{\max}$. Since $p_j \leqslant p_{\max}$ for all j, no machine becomes available before time L when this job is being considered for scheduling. This means that the total processing time of all the jobs that have been scheduled so far is more than mL. This is impossible since $L \geqslant C^*_{\max}$. By the result of Graham (1969), we have

$$C^*_{\max} \leqslant L \leqslant \left(\frac{4}{3} - \frac{1}{3m}\right) C^*_{\max}.$$

Furthermore, we have $p_{\max} \leqslant C^*_{\max}$. Therefore, we have

$$C_{\max}(\mathbf{H5}) \leqslant \left(\frac{7}{3} - \frac{1}{3m}\right) C^*_{\max}.$$

We now consider the *TMC* objective. We will show that $TMC(\mathbf{H5}) \leqslant TMC^*$, where TMC^* is the optimal total machine cost among all schedules with makespan equal to $C^*_{\max}$.

Let $n_i = |B_i|$ and $n^*_i = |B^*_i|$. Note that $\sum_{i \in M} n_i = \sum_{i \in M} n^*_i = n$. Let S and S^* be the schedule produced by heuristic **H5** and the optimal schedule, respectively. In schedule S, the smallest $\sum_{h=1}^{i} n_h$ jobs, with respect to processing times, are scheduled on machines $1, \ldots, i$. Suppose that, in schedule S^*, $\sum_{h=1}^{i} n^*_h > \sum_{h=1}^{i} n_h$. Then, obviously, we have $\sum_{h=1}^{i} P(B^*_h) > \sum_{h=1}^{i} P(B_h)$. By Lemma 2, we have $\sum_{h=1}^{i} P(B_h) \geqslant i \times L$ or $\sum_{h=1}^{i} P(B_h) = \sum_{j=1}^{n} p_j$ for all $i \in M$.

If $\sum_{h=1}^{i} P(B_h) \geqslant i \times L$, then we have

$$\sum_{h=1}^{i} P(B^*_h) > \sum_{h=1}^{i} P(B_h) \geqslant i \times L \geqslant i \times C^*_{\max},$$

which contradicts the fact that $C^*_{\max}$ is the optimal makespan.

If $\sum_{h=1}^{i} P(B_h) = \sum_{j=1}^{n} p_j$, then

$$\sum_{h=1}^{i} P(B_h^*) > \sum_{h=1}^{i} P(B_h) = \sum_{j=1}^{n} p_j,$$

which again leads to a contradiction. Thus, we have

$$\sum_{h=1}^{i} n_h \geqslant \sum_{h=1}^{i} n_h^*$$

for all $i \in M$. Now, $TMC(\mathbf{H5}) = \sum_{i\in M} n_i \times a_i + \sum_{j=1}^{n} c_j$. Since $\sum_{j=1}^{n} c_j$ is a constant independent of the assignment, we will show that $\sum_{i\in M} n_i \times a_i \leqslant \sum_{i\in M} n_i^* \times a_i$, and hence $TMC(\mathbf{H5}) \leqslant TMC^*$.

Let $\bar{a}_i = a_i - a_{i-1}$ for all $i \in M$, where $a_0 = 0$. Since $a_i \geqslant a_{i-1}$, we have $\bar{a}_i \geqslant 0$ for all $i \in M$.

$$\begin{aligned}\sum_{i\in M} n_i a_i &= n_1\bar{a}_1 + n_2(\bar{a}_1+\bar{a}_2) + \cdots + n_m\left(\sum_{h=1}^{m}\bar{a}_h\right)\\ &= (n_1+n_2+\cdots+n_m)\bar{a}_1\\ &\quad + (n_2+\cdots+n_m)\bar{a}_2 + \cdots + n_m\bar{a}_m\\ &= n\bar{a}_1 + (n-n_1)\bar{a}_2\\ &\quad + (n-n_1-n_2)\bar{a}_3 + (n-n_1-n_2-\cdots-n_{m-1})\bar{a}_m\\ &= n(\bar{a}_1+\bar{a}_2+\cdots+\bar{a}_m) - \sum_{i=1}^{m-1}\left\{\left(\sum_{h=1}^{i} n_h\right)\bar{a}_{i+1}\right\}.\end{aligned}$$

Since $n(\bar{a}_1+\bar{a}_2+\cdots+\bar{a}_m)$ is a constant, $\sum_{h=1}^{i} n_h \geqslant \sum_{h=1}^{i} n_h^*$ for all $i \in M$ and $\bar{a}_i \geqslant 0$ for all $i \in M$, we have $TMC(\mathbf{H5}) \leqslant TMC^*$. □

4.2. The special case $c_{ij} = a_i \times c_j$

We now consider the cost function $c_{ij} = a_i \times c_j$. We propose a heuristic, heuristic **H6**, which works as follows. Like in heuristic **H5**, let the makespan of the LPT schedule be L. We then sort the jobs in descending order of the ratio $\frac{c_j}{p_j}$; i.e., $\frac{c_1}{p_1} \geqslant \frac{c_2}{p_2} \geqslant \cdots \geqslant \frac{c_n}{p_n}$. Jobs are now assigned to the machines, starting with the first machine. The first λ jobs will be assigned to machine 1, where λ is the smallest index such that the total processing time of the first λ jobs is larger than L. These λ jobs and machine 1 will be deleted from consideration, and we proceed to schedule jobs on machine 2. This process is iterated until all jobs are assigned. Below is a description of heuristic **H6**.

Heuristic H6(J, M)

(1) Disregarding the cost of the jobs, schedule the n jobs on the m machines by the LPT rule. Let L denote the makespan of this schedule.
(2) Sort the jobs in descending order of the ratios $\frac{c_j}{p_j}$. Let $J = (J_1, J_2, \ldots, J_n)$ be a list of the n jobs such that $\frac{c_1}{p_1} \geqslant \frac{c_2}{p_2} \geqslant \cdots \geqslant \frac{c_n}{p_n}$.
(3) Set $i = 1$.
(4) If the total processing time of the jobs in J is less than or equal to L then assign all jobs in J to machine i and stop.
(5) Let λ be the smallest integer such that the total processing time of the first λ jobs is greater than or equal to L.
(6) Assign the first λ jobs to machine i.
(7) Delete the first λ jobs from J.
(8) $i := i + 1$.
(9) Goto Step 4.

In the proof of Theorem 6, we will assume that the jobs have been sorted in descending order of $\frac{c_j}{p_j}$ (i.e., $\frac{c_1}{p_1} \geqslant \frac{c_2}{p_2} \geqslant \cdots \geqslant \frac{c_n}{p_n}$). Let B_i and B_i^* be the sets of jobs assigned to machine i in the schedule generated by heuristic **H6** and in the optimal schedule, respectively. Thus, $TMC(\mathbf{H6}) = \sum_{i=1}^{m} a_i \times V(B_i)$ and $TMC^* = \sum_{i=1}^{m} a_i \times V(B_i^*)$.

Theorem 6. *Heuristic* **H6** *is a* $((\frac{7}{3} - \frac{1}{3m}), 1)$*-approximation for the problem LEX* $(C_{\max}, TMC)$ *when* $c_{ij} = a_i \times c_j$ *for all* $1 \leqslant i \leqslant m$.

Proof. Similar to the proof of Theorem 5, we can show that heuristic **H6** must be able to schedule the n jobs on the m machines so that each machine finishes by time $L + p_{\max}$. Since $L \leqslant (\frac{4}{3} - \frac{1}{3m})C_{\max}^*$ and $p_{\max} \leqslant C_{\max}^*$, we have $C_{\max}(\mathbf{H6}) \leqslant (\frac{7}{3} - \frac{1}{3m})C_{\max}^*$. We now show that $TMC(\mathbf{H6}) \leqslant TMC^*$, where TMC^* is the optimal total machine cost among all schedules with a makespan of $C_{\max}^*$.

The ratio c_j/p_j is the cost per unit processing time. Let $\lambda_j = c_j/p_j$ for each $1 \leqslant j \leqslant n$; hence $c_j = \lambda_j \times p_j$. We have $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$. Since heuristic **H6** schedules jobs on each machine with total processing time larger than L and since $L \geqslant C_{\max}^*$, we have for each $1 \leqslant i \leqslant m$,

$$\sum_{h=1}^{i} P(B_h) \geqslant \sum_{h=1}^{i} P(B_h^*).$$

Now,

$$\sum_{h=1}^{i} V(B_h) = \sum_{j\in(B_1\cup\cdots\cup B_h)} c_j = \sum_{j\in(B_1\cup\cdots\cup B_h)} \lambda_j \times p_j.$$

Since heuristic **H6** schedules the jobs in ascending order of the indexes (and hence larger λ_j values) and since $\sum_{h=1}^{i} P(B_h) \geqslant \sum_{h=1}^{i} P(B_h^*)$, we have

$$\sum_{h=1}^{i} V(B_h) \geqslant \sum_{h=1}^{i} V(B_h^*).$$

Let $\bar{a}_i = a_i - a_{i-1}$ for all $i \in M$, where $a_0 = 0$ and $a_1 = 1$. Since $a_i \geqslant a_{i-1}$, we have $\bar{a}_i \geqslant 0$ for all $i \in M$.

$$\begin{aligned}\sum_{i\in M} a_i \times V(B_i) &= V(B_1)\bar{a}_1 + V(B_2)(\bar{a}_1+\bar{a}_2) + \cdots + V(B_m)\left(\sum_{h=1}^{m}\bar{a}_h\right)\\ &= (V(J))\bar{a}_1 + (V(J) - V(B_1))\bar{a}_2 + \cdots + V(B_m)\bar{a}_m\\ &= V(J)(\bar{a}_1+\bar{a}_2+\cdots+\bar{a}_m) - \sum_{i=1}^{m-1}\left\{\left(\sum_{h=1}^{i} V(B_h)\right)\bar{a}_{i+1}\right\}.\end{aligned}$$

Since $V(J)(\bar{a}_1+\bar{a}_2+\cdots+\bar{a}_m)$ is a constant, $\sum_{h=1}^{i} V(B_h) \geqslant \sum_{h=1}^{i} V(B_h^*)$ for all $i \in M$, and $\bar{a}_i \geqslant 0$ for all $i \in M$, we have $TMC(\mathbf{H6}) \leqslant TMC^*$. □

In heuristics **H5** and **H6**, we calculate an upper bound for the optimal makespan by using LPT and thus have a competitive ratio of $(1 + \frac{4}{3} - \frac{1}{3m}, 1)$ for our problems. However, if we use a better approximation algorithm for the makespan with a worst-case performance ratio of ρ, we can have a $(1+\rho, 1)$-approximation algorithm. For example, if we compute an upper bound by MULTIFIT, we have a $(1 + 13/11, 1)$-approximation algorithm because the worst-case performance ratio of MULTIFIT is 13/11 by Yue (1990).

The MULTIFIT algorithm is an approximation algorithm for $P||C_{\max}$, which operates as follows. First, it computes a lower bound (LB) and an upper bound (UB) for $C_{\max}^*$. An obvious lower bound is $LB = \max\left\{p_{\max}, \frac{\sum_{j=1}^{n} p_j}{m}\right\}$, and an obvious upper bound is $UB = 2 \times LB$. It then conducts a binary search in the interval $[LB, UB]$. For each value C obtained in the binary search (i.e., $C = \frac{LB+UB}{2}$), it tries to schedulle the jobs by the First-Fit-Decreasing (FFD) rule so that no job completes after time C. If it is successful in scheduling all n jobs, then it sets UB to be C; otherwise, it sets LB to be C. This process is iterated until $UB = LB + 1$. Now, if all jobs can be scheduled by the FFD rule so that no job completes after time LB, then it returns LB; otherwise, it returns UB.

Shmoys and Tardos (1993) considered the following scheduling problem: job j on machine i requires a processing time p_{ij} and

Table 1
Comparisons of heuristics and approximation algorithms.

Problem	c_{ij}	Our paper		Shmoys and Tardos (1993)
		Alg.	Ratio	Ratio
LEX($\sum C_j, MMC$)	c_j	H1	$(1, 2-\frac{1}{m})$	$(1,2)$
	$a_i + c_j$	H2	$(1, 2-\frac{1}{m})$	
	$a_i \times c_j$	H3	(1, unbounded)	
	$a_i \times c_j, \bar{a}^{\dagger}$	H4	$(1, 2-\frac{1}{2(m-1)})$	
LEX($C_{\max}, TMC$)	$a_i + c_j$	H5	$(1+\rho^{\ddagger}, 1)$	$(1+\rho^{\ddagger}, 1)$
	$a_i \times c_j$	H6	$(1+\rho^{\ddagger}, 1)$	

- $\bar{a}^{\dagger}: a_1 = \cdots = a_{m-1} = 1, a_m > 1$.
- $\rho^{\ddagger}$: an approximation ratio for the parallel machine scheduling problem to minimize the makespan.

incurs a cost c_{ij}. Assume T is given as an upper bound of the makespan, what is the machine-job assignment that minimizes the total assignment cost? They modeled this as the following linear program. Given T and C, for any $t \geqslant T$, integer solutions to the linear program, LP $(T, C : t)$, have a one-to-one correspondence with schedules that have a cost of at most C and a makespan of at most T. The LP $(T, C : t)$ problem can be formulated as follows:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \leqslant C,$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \text{for } j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} p_{ij}x_{ij} \leqslant T \quad \text{for } i = 1, \ldots, m,$$

$$x_{ij} \geqslant 0 \text{ for } i = 1, \ldots, m \ , j = 1, \ldots, n,$$

$$x_{ij} = 0 \text{ if } p_{ij} > t, \ i = 1, \ldots, m, \ j = 1, \ldots, n,$$

They proved that if LP $(T, C : t)$ has a feasible solution, then there exists a schedule that has a makespan of at most $T + t$ and a cost of at most C. They also provided an algorithm that converts a feasible solution to LP $(T, C : t)$ into the required schedule.

We can apply their result to our problem LEX $(C_{\max}, TMC)$. Note that in our problem $p_{ij} = p_j$ and $c_{ij} = a_i + c_j$ or $c_{ij} = a_i \times c_j$, which implies that the model in Shmoys and Tardos (1993) is more general than ours. According to the above result, we can get an $(1+\rho, 1)$ approximation algorithm by the following procedure where ρ is an approximation ratio of any approximation algorithm for minimizing the makespan in an environment with identical machines in parallel.

Step 1. Apply an approximation algorithm for minimizing the makespan with identical machines in parallel and obtain the makespan T'. (For example, when LPT is applied, we have T' such that $C^*_{\max} \leqslant T' \leqslant (\frac{4}{3} - \frac{1}{3m})C^*_{\max}$).

Step 2. Regard C as a variable, solve LP $(T', C : p_{\max})$ with the objective C to be minimized and get an optimal value C'.

Step 3. Convert an optimal solution to LP $(T', C' : p_{\max})$ into a feasible solution.

Since $T' \geqslant C^*_{\max}$ and we allow split jobs to multiple machines in a linear program solution, we have $C' \leqslant TMC^*$. Furthermore, $p_{\max} \leqslant C^*_{\max}$ and $T' \leqslant \rho C^*_{\max}$. The optimal solution from Step 2 is feasible to LP $(T', C' : p_{\max})$, and thus, we can get a schedule with a makespan of at most $T' + p_{\max}$ and cost at most C', which implies we have a $(1+\rho, 1)$ -approximation algorithm for LEX $(C_{\max}, TMC)$.

However, to find such an approximation, it is necessary to apply two procedures (which determine the overall time complexity), namely, a linear programming algorithm in order to obtain a fractional matching solution (in Step 2) and a weighted bipartite matching algorithm for the rounding (in Step 3). This LP is considered a fractional packing problem and thus can be solved in $O(mn^2 \log n)$; the weighted bipartite graph has $O(n)$ nodes and $O(n)$ edges and thus the weighted bipartitie matching can be solved in $O(n^2 \log n)$. Therefore, the overall time complexity is $O(mn^2 \log n)$.

The proposed algorithms for our problem LEX $(C_{\max}, TMC)$ are in both cases (i.e., $c_{ij} = a_i + c_j$ and $c_{ij} = a_i \times c_j$) much simpler and easier to implement; their time complexity is $O(n \log n)$.

5. Conclusions

We consider bi-criteria scheduling problems where customers' objectives as well as service providers' objectives have to be optimized at the same time. As a customers' objective, we consider either the total completion time or the makespan and as a service providers' objective, we consider either the total assignment cost or the maximum assignment cost. The primary objective is the customers' objective and the secondary objective is the service providers' objective. For three machine assignment cost functions ($c_{ij} = c_j, c_{ij} = a_i + c_j$, and $c_{ij} = a_i \times c_j$), we provide (α, β)-approximation algorithms.

Our models are all special cases of the more general model considered by Shmoys and Tardos (1993), who did provide a polynomial time approximation algorithm that is applicable to all models within their framework. However, their approximation scheme is clearly computationally slower than those we propose and it does not provide any intuitive insights into the respective problems. Our schemes are based on prioritization and are therefore more intuitive and easier to implement (see Table 1). All the heuristics proposed in this paper run in $O(n \log n)$, whereas the algorithm developed by Shmoys and Tardos runs in $O(mn^2 \log n)$. Our worst case approximation ratios are in three cases better than the ratio in Shmoys and Tardos; in two other cases they are the same and in one case we were not able to determine a bounded worst case approximation ratio.

We have considered only traditional scheduling objectives that are functions of completion times as the primary objective. However, considering *MMC* or *TMC* as the primary objective may lead to different approximation ratios. We have only considered cases where either $\alpha = 1$ or $\beta = 1$. Finding an (α, β)-approximation algorithm for $\alpha > 1$ and $\beta > 1$ may be a very promising research topic.

Appendix A. Procedure for solving minimum cost for machine *m* problem

For SPT schedules, we sort the jobs in non-decreasing order of their processing times and in case of a tie we sort the jobs in non-decreasing order of their costs. When there are jobs with identical processing times, those jobs may be eligible to multiple consecutive ranks. Thus, we define $F_{r',r''}$ to be the set of jobs that can be

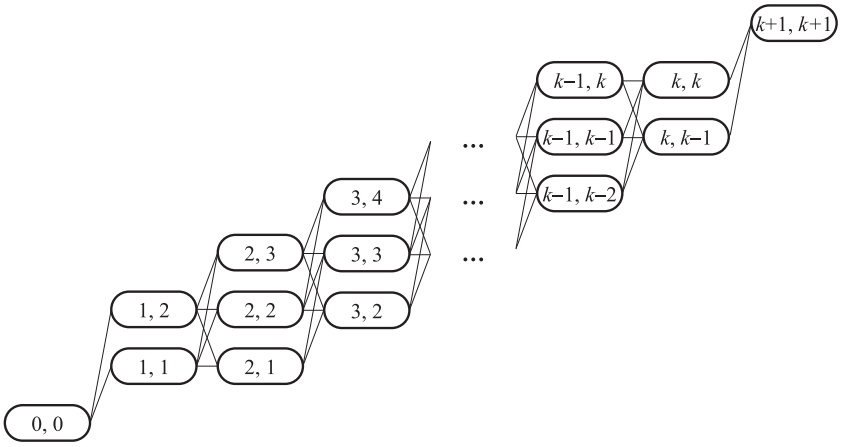

Fig. 1. A shortest path problem to solve **Minimum Cost for Machine** m.

scheduled in rank r for $r' \leqslant r \leqslant r''$. Since the number of jobs in $F_{r',r''}$ that can be scheduled on machine m in ranks $[r', r'']$ is at most $r'' - r' + 1$, we can define $\overline{F}_{r',r''}$ to be the set of $(r'' - r' + 1)$ jobs with minimum costs from $F_{r',r''}$. Now, the problem is to select and assign jobs from $\overline{F}_{r',r''}$ to rank r for $r' \leqslant r \leqslant r''$ such that each rank has exactly one job and the total cost of assigned jobs is minimum.

If $r'' \geqslant r' + 2$, then ranks $r' + 1, r' + 2, \ldots, r'' - 1$ must be filled by jobs in $\overline{F}_{r',r''}$. So, we can assign $(r'' - r' - 1)$ jobs to those ranks first and redefine the problem by taking off those ranks along with the scheduled jobs. Thus, without loss of generality, we can say that $r'' = r'$ or $r'' = r' + 1$. Moreover, $\overline{F}_{r,r}$ has at most one job and $\overline{F}_{r,r+1}$ has at most two jobs. The number of possible (r', r'') pairs for $\overline{F}_{r',r''}$ is bounded by $k + (k-1) = 2k - 1$ and the number of jobs in $\overline{F}_{r',r''}$ is bounded by $k + 2(2k - 1) = 5k - 2$.

In order to describe the subsequent steps, we use following notation.

- Let $c^1(\overline{F}_{r,r})$ be the cost of the job in $\overline{F}_{r,r}$ if $|\overline{F}_{r,r}| = 1$ and ∞ otherwise.
- Let $c^1(\overline{F}_{r,r+1})$ be the cost of the job with the lower cost in $\overline{F}_{r,r+1}$ if $|\overline{F}_{r,r+1}| \geqslant 1$ and ∞ otherwise.
- Let $c^2(\overline{F}_{r,r+1})$ be the cost of the job with the higher cost in $\overline{F}_{r,r+1}$ if $|\overline{F}_{r,r+1}| = 2$ and ∞ otherwise.

We define the following shortest path problem from a source node $(0,0)$ to the destination node $(k+1, k+1)$ in a layered graph (See Fig. 1).

Nodes are defined as follows:

- Node (0, 0) is a dummy source node.
- Node $(k+1, k+1)$ is a dummy destination node.
- Node $(r, r+1)$ denotes that rank r is assigned from a job in $\overline{F}_{r,r+1}$ for $r = 1, \ldots, k-1$.
- Node (r, r) denotes that rank r is assigned from a job in $\overline{F}_{r,r}$ for $r = 1, \ldots, k$.
- Node $(r, r-1)$ denotes that rank r is assigned from a job in $\overline{F}_{r-1,r}$ for $r = 2, \ldots, k$.

The arcs and their costs are defined as follows:

- (k,k) and $(k+1, k+1)$ are connected and the arc cost is 0.
- $(k, k-1)$ and $(k+1, k+1)$ are connected and the arc cost is 0.
- $(r-1, r)$ and $(r, r+1)$ are connected and the arc cost is $c^1(\overline{F}_{r,r+1})$.
- $(r-1, r-1)$ and $(r, r+1)$ are connected and the arc cost is $c^1(\overline{F}_{r,r+1})$.
- $(r-1, r-2)$ and $(r, r+1)$ are connected and the arc cost is $c^1(\overline{F}_{r,r+1})$.
- $(r-1, r)$ and (r, r) are connected and the arc cost is $c^1(\overline{F}_{r,r})$.
- $(r-1, r-1)$ and (r, r) are connected and the arc cost is $c^1(\overline{F}_{r,r})$.
- $(r-1, r-2)$ and (r, r) are connected and the arc cost is $c^1(\overline{F}_{r,r})$.
- $(r-1, r)$ and $(r, r-1)$ are connected and the arc cost is $c^2(\overline{F}_{r-1,r})$.
- $(r-1, r-1)$ and $(r, r-1)$ are connected and the arc cost is $c^1(\overline{F}_{r-1,r})$.
- $(r-1, r-2)$ and $(r, r-1)$ are connected and the arc cost is $c^1(\overline{F}_{r-1,r})$.

From the arcs of a shortest path, we can construct an optimal solution.

Now, consider the running time of the proposed algorithm. It takes $O(n \log n)$ time to sort the jobs. It takes $O(n)$ time to construct $F_{r',r''}$ and $\overline{F}_{r',r''}$. As for the shortest path problem, the number of arcs is less than $9k = O(k) \leqslant O(n)$ and a shortest path can be obtained in $O(n)$ time. Therefore, the proposed algorithm can be implemented in $O(n \log n)$ time.

References

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research, 52*(2), 229–242.

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Combinatorial models for multi-agent scheduling problems. In Eugene Levner (Ed.), *Multiprocessor scheduling: Theory and applications* (pp. 21–46). Vienna, Austria: I-TECH Educational and Publishing.

Cho, Y., & Sahni, S. (1980). Bounds on list schedules for uniform processors. *SIAM Journal on Computing, 9*(1), 91–103.

Dòsa, G., & He, Y. (2004). Better online algorithms for scheduling with machine cost. *SIAM Journal on Computing, 33*, 1035–1051.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Graham, R. L. (1969). Bounds on multiprocessing time anomalies. *SIAM Journal on Applied Mathematics, 17*(2), 416–429.

Hoogeveen, H. (2005). Multicriteria scheduling. *European Journal of Operational Research, 167*(3), 592–623.

Imreh, C., & Noga, J. (1999). Scheduling with machine cost. In *Proceedings of 2nd international workshop on approximation algorithms. Lecture notes in computer science* (Vol. 1671, pp. 168–176).

Khuller, S., Li, J., & Saha, B. (2010). Energy efficient scheduling via partial shutdown. In *Proceedings of the twenty-first annual ACM-SIAM symposium on discrete algorithms (SODA '10)* (pp. 1360–1372).

Lee, K., Choi, B.-C., Leung, J. Y.-T., & Pinedo, M. L. (2009). Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Information Processing Letters, 109*(16), 913–917.

Lee, K., Leung, J. Y.-T., & Pinedo, M. L. (2011). Coordination mechanisms with hybrid local policy. *Discrete Optimization, 8*, 513–524.

Lee, K., Leung, J. Y.-T., & Pinedo, M. L. (2012). Coordination mechanisms for parallel machine scheduling. *European Journal of Operational Research, 220*(2), 305–313.

Lee, K., Leung, J. Y.-T., & Pinedo, M. L. (2013). Two dimensional load balancing. Working paper.

Lei, D. (2009). Multi-objective production scheduling: A survey. *The International Journal of Advanced Manufacturing Technology, 43*(9–10), 926–938.
Leung, J. Y.-T., Lee, K., & Pinedo, M. L. (2012). Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics, 139*(1), 321–329.
Leung, J. Y-T., & Pinedo, M. L. (2003). Minimizing total completion time on parallel machines with deadline constraints. *SIAM Journal on Computing, 32*, 1370–1388.
Leung, J. Y-T., Pinedo, M. L., & Wan, G. (2010). Competitive two-agent scheduling and its applications. *Operations Research, 58*, 458–469.
Leung, J. Y-T., & Young, G. H. (1989). Minimizing schedule length subject to minimum flow time. *SIAM Journal on Computing, 18*, 314–326.
Pinedo, M. L. (2012). Scheduling: Theory, algorithms, and systems. *4th ed.*. Springer.
Shmoys, D. B., & Tardos, E. (1993). An approximation algorithm for the generalized assignment problem. *Mathematical Programming, 62*(1–3), 461–474.
Smith, W. E. (1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly, 3*, 59–66.
T'kindt, V., & Billaut, J.-C. (2001). Multicriteria scheduling problems: A survey. *RAIRO – Operations Research, 35*(2), 143–163.
T'kindt, V., & Billaut, J.-C. (2006). Multi-criteria scheduling – Theory, models and algorithms. *2nd ed.*. Springer.
T'kindt, V., Billaut, J.-C., & Proust, C. (2001). Solving a bicriteria scheduling problem on unrelated parallel machines occurring in the glass bottle industry. *European Journal of Operational Research, 135*, 42–49.
Vignier, A., Sonntag, B., & Portmann, M.-C. (1999). A hybrid method for a parallel-machine scheduling problem. In *7th IEEE international conference on emerging technologies and factory automation (ETFA '99)* (pp. 671–678).
Yue, M. (1990). On the exact upper bound for the MultiFit processor scheduling algorithm. *Annals of Operations Research, 24*, 233–259.