RAIRO-Oper. Res. 54 (2020) 507–527
https://doi.org/10.1051/ro/2020009

RAIRO Operations Research
www.rairo-ro.org

NEAR-OPTIMAL SOLUTIONS AND TIGHT LOWER BOUNDS FOR THE PARALLEL MACHINES SCHEDULING PROBLEM WITH LEARNING EFFECT

Lotfi Hidri[1] and Mahdi Jemmali[2,3,*]

Abstract. In this paper, the parallel machines scheduling problem with Dejong's learning effect is addressed. The considered problem has a practical interest since it models real-world situations. In addition, this problem is a challenging one because of its NP-Hardness. In this work, a set of heuristics are proposed. The developed heuristics are categorized into two types. The first category is based on the dispatching methods, with new enhancement variants. The second type is more sophisticated and requires solving NP-Hard problems. Furthermore, several lower bounds are developed in order to assess the performance of the proposed heuristics. These lower bounds are based on solving the problem of the determination of the minimum average load under taking into account some observations. Among these observations, the existence of a limit position that the jobs are not allowed to exceed in any optimal schedule. Finally, an extensive experimental study is conducted over benchmark test problems, with up to 1500 jobs and 5280 instances. The obtained results are outperforming those proposed in the literature.

Mathematics Subject Classification. 90B35, 90C27, 90C59, 90C90.

Received April 23, 2019. Accepted January 27, 2020.

1. Introduction

Parallel scheduling problems are well studied in literature [16]. This particular focus on these scheduling problems is due to the fact that they model a wide range of real-life situations, especially in the manufacturing process and parallel computing. Besides their practical interesting aspect, these problems are challenging ones from a theoretical point of view, since the most studied problems are NP-Hard. Despite all the theoretical advancements while studying these problems, there still remains a huge gap between the theoretical models and real faced problems. Therefore, new scheduling models are proposed to overcome this drawback and to narrow this gap. These new models include, for example, the multiprocessor task concept [7] and controllable job processing times [21]. In this context, the learning effect concept is considered in this paper.

Several observations in industrial facilities reveal that a worker learns more while producing more items. This is because of the accumulation of more expertise. Consequently, the processing times are reduced while treating

Keywords. Parallel machines, learning effect, lower bounds, heuristics.

[1] Industrial Engineering Department, College of Engineering, King Saud University, 11421 Riyadh, Saudi Arabia.

[2] Department of Computer Science and Information, College of Science, Majmaah University, 11952 Majmaah, Saudi Arabia.

[3] Department of Computer Science, Higher Institute of Computer Science and Mathematics of Monastir, Monastir University, 5000 Monastir, Tunisia.

*Corresponding author: m.jemmali@mu.edu.sa, mah_jem_2004@yahoo.fr

items. This observation was quoted firstly by Wright in 1936 when observing an aircraft industrial facility [5]. Following this observation, plenty of empirical studies was conducted to quantify the learning impact in the processing times. In this context, several models have been proposed [1, 15]. Taking into account the learning effect while planning the production process might reduce the total production cost [2, 3, 22].

In scheduling theory, the learning effect is introduced as a learning curve where the processing time of a given job depends on its position. In this context several learning curves are proposed [1, 15]. For example, Wright's curve, DeJong's model and exponential models.

In this work, the identical parallel machine scheduling problem with learning effect is examined. The adopted learning effect curve is the one of DeJong. The selection of this kind of curve is justified by the fact that DeJong's curve does not vanish (converges to zero) while the position increases. This is in accordance with the real life applications. The studied problem presents a lot of differences with its counterpart without learning effect (identical parallel machines). As an example, using all the parallel machines might not provide an optimal solution for the problem with learning effect. Therefore,the majority of the developed procedures, originally for the identical parallel machines do not apply for the same problem with learning effect. In addition, the studied problem is proofed to be harder than its counterpart without learning effect. Indeed, several experimental studies provide moderate results in term of the maximum reached number of jobs, the number of machines, and the hight consumed time [17]. Therefore, new procedures overcoming the previous quoted drawbacks should be proposed. In the context of this research work, new heuristics are proposed. These heuristics are categorized into two classes. The first one is exploring the well known dispatching rules (SPT, LPT, . . .) that are adjusted to fit with the studied problem. The second class of heuristics is using NP-Hard problems' solution, such as the exact solution of identical parallel machines problem. In order to asses the performance of the proposed heuristics, new efficient lower bounds are proposed and the average relative gap is considered as a performance measure.

This paper is composed of the following sections. Section 2 presents a brief literature review of the studied problem. Section 3 introduces the studied problem definition and some of its proprieties. In Section 3, new lower bounds, based on the average minimum load, are developed and presented. Furthermore, a maximum limit position based lower bound is developed. In Section 4, several enhancement methods are presented. In addition, several advanced heuristics, using the solution of NP-Hard problems, are proposed. Section 5 is devoted to an extensive numerical study in order to assess the performance of the proposed procedures. Finally, a conclusion containing some research extensions is presented.

2. Literature review

Below we present a brief literature review of the parallel machine scheduling problems with learning effect. Since the references [5, 11, 17, 18] gave a detailed review for such a scheduling problems until 2014, we just mention the other ones until nowadays.

Authors in [10] studied the parallel machine scheduling problem with learning effect and deteriorating jobs. Particular objective functions are considered for which the latter problem is shown to be polynomially solvable. The uniform parallel machine problem with learning effect and identical jobs is presented in [13]. Classical objective functions are considered. An experimental comparative study between the linear program formulation and assignment one is carried out. The assignment formulation is shown to be more efficient than the linear program formulation. The parallel machine problem with learning effect, fuzzy processing times and makespan minimization are considered in [25]. To solve the latter problem, genetic and simulated annealing algorithms are proposed. In [23] the unrelated parallel machine problem with learning effect and deteriorating jobs is studied. Special objective functions and assumptions are considered. For the latter objectives, the problem is shown to have a polynomial solution. The parallel machine scheduling problem with DeJong's learning effect is addressed in [12]. A polynomial approximation heuristic is proposed for this problem in order to minimize the makespan. The presented work in [19] focuses on non-identical parallel machine problem with learning effect, deteriorating jobs and fuzzy data. Two simultaneous objectives are considered: earliness/tardiness and maximum completion time. An exact solution based on the branch and bound method is provided to solve the proposed problem. Experimental tests show the effectiveness of the proposed procedure. In [20], authors studied the parallel machine scheduling problem with learning effect and deteriorating jobs. The considered objective function is the maximum completion time. For this NP-Hard problem, a polynomial-time heuristic is proposed. In addition, a polynomial approximation algorithm heuristic is presented. Authors in [18] considered the multi-objective uniform parallel machine scheduling problem with learning effect. An efficient particle swarm based approximation scheme is provided to solve this problem.

In [24], authors addressed the parallel machine scheduling with job splitting, learning effect and minimizing of the total completion time. An exact algorithm, utilizing the branch and bound, is proposed for small size problems. Besides, some meta-heuristics and constructive heuristics are developed. The parallel machine resource allocation scheduling problem with deteriorating jobs and learning effect is considered in [14]. This problem is shown to be polynomial for some classical objective functions if the number of machines is assumed to be constant.

3. Problem definition and properties

3.1. Problem definition

The studied problem in this paper is the identical parallel machines scheduling problem with learning effect, which is stated as follows. A set $J = \{1, 2, \ldots, n\}$ of n jobs has to be scheduled on m identical parallel machines $M_1, \ldots, M_m$. The scheduling is performed taking into account the learning effect. More precisely, a job $j \in J$ scheduled at position k $(k = 1, \ldots, n)$ has to be processed during $p_{j,k} = p_j \left(M + (1 - M) k^a\right)$ units of time. The curve $\left(M + (1 - M) k^a\right)$ is said the Dejong's learning curve [4]. The scalar a $(a \leq 0)$ is the learning index and M $(M \in [0, 1])$ is the incompressibility factor. The initial processing time p_j is the duration time treating job j without learning effect.

The scheduling of the jobs is done under the following assumptions. Each job is available for processing from time zero and the preemption is not allowed. Each machine processes at most one job at the same time. The completion time of a job $j \in J$ in a feasible schedule is denoted by C_j. The makespan which is the objective function to be minimized is $C_{\max} = \max_{1 \leq j \leq n} C_j$. The optimal makespan value is denoted hereafter by $C^*_{\max}$. According to the three-field notation [8], the treated scheduling problem will be denoted as $P_m \mid p_{j,k} = p_j \left(M + (1 - M) k^a\right) \mid C_{\max}$. Seeking simplicity, the following notation will be used for the remaining of this paper: $f^a_M(k) = M + (1 - M) k^a$.

3.2. Problem properties

In this subsection, several useful proprieties of the studied problem are presented. The complexity of the studied scheduling problem is given in the following proposition.

Proposition 3.1. *The $P_m \mid p_{j,k} = p_j f^a_M(k) \mid C_{\max}$ is NP-hard [17].*

Proof. $P_m \mid \mid C_{\max}$ is NP-Hard [9] and it is a special case of $P_m \mid p_{j,k} = p_j f^a_M(k) \mid C_{\max}$ when $M = 1$. □

The first important result of an optimal schedule is presented in the following remark.

Remark 3.2. In any optimal schedule, the jobs assigned to a machine M_l $(l = 1, \ldots, m)$ are arranged according to their non-decreasing initial processing time p_j. Indeed, the optimal solution for the one machine problem is obtained by the SPT rule [17].

The scheduling problem $P_m \mid p_{j,k} = p_j f^a_M(k) \mid C_{\max}$ is more difficult than its counterpart $P_m \mid \mid C_{\max}$, since some new unusual situations appear.

Situation 1: some of the available machines, might not be used at all in an optimal schedule for the parallel machine with learning effect. This is not the case for the classic parallel machine problem. The following example is an illustration of such a situation.

Example 3.3. Consider the following instance: $n = 11$, $m = 2$, $p_j = 1\,(j = 1, \ldots, 10)$, $p_{11} = 100$, $M = 0.5$, and $a = -0.1$.

Let π_1 the schedule in which the jobs $\{1, 2, 3, \ldots, 10\}$ are scheduled on M_1 and job 11 is scheduled on M_2. The makespan of π_1 is $C_{\max}(\pi_1) = 100$. Consider now the schedule π_2 where all the jobs (including job 11) are assigned to M_1 only, then $C_{\max}(\pi_2) = 98.64$.

Situation 2: omitting jobs from J may increase the optimal solution value, which is not the case for the $P_m \mid \mid C_{\max}$, as it will be shown in the next example.

Example 3.4. Let the instance with data: $n = 5, m = 1$, $M = 0$ and $a = -1$. The initial processing times are given as follows, $p_j = \{3, 5, 7, 9, 50\}$. For this example, the makespan $C_{\max}\,(J) = 20.08$ and $C_{\max}\,(J \setminus 4) = 20.33$

4. Lower bounds

In this section, existing lower bounds from the literature will be presented and new ones will be developed. The new lower bounds generalize and improve the existing ones. These new lower bounds will be introduced gradually.

4.1. Lower bounds from literature

Prior to the presentation of the existing lower bounds the following notation is presented. Let $p_{(j)}$ be the jth $(1 \leq j \leq n)$ smallest initial processing time p_j $(j \in J)$, sorted in the non-decreasing order.

Lemma 4.1. *For an optimal schedule, the job j_k having the kth initial processing time $p_{(k)}$ is scheduled in a machine at most at position k $(1 \leq k \leq n)$.*

Proof. By contradiction, assume that j_k is scheduled at a position $s_k > k$, then $s_k - 1$ jobs with initial processing times less or equal to $p_{(k)}$, are scheduled on the same machine before j_k. This is because in an optimal schedule the assigned jobs for the same machine are arranged according to the SPT rule. Thus, $s_k - 1$ jobs have an initial processing time less than $p_{(k)}$, with $s_k - 1 \geq k$, which is in contradiction with the fact that the job j_k has the kth initial processing time $p_{(k)}$. □

According to [17] the Proposition 4.2 holds.

Proposition 4.2.

$$\begin{cases} \mathrm{LB}_0 = \max\limits_{1\,\leq\,j\,\leq\,n} \{p_j\}\,(M + (1 - M)\,n^a) & (4.1) \\ \mathrm{LB}_1 = \dfrac{1}{m}\sum\limits_{j=1}^{n} p_{(j)}\,(M + (1 - M)\,j^a) & (4.2) \\ \mathrm{LB}_2 = \max\,(\mathrm{LB}_0, \mathrm{LB}_1) & (4.3) \end{cases}$$

are valid lower bounds for the $P_m \mid p_{j,k} = p_j f_M^a(k) \mid C_{\max}$.

Proof. The proof is presented for the three lower bounds separately.

- Proof of equation (4.1): in an optimal schedule, there exists a position k $(1 \leq k \leq n)$ where the job with the largest initial processing time $p_{\max} = \max_{1\,\leq\,j\,\leq\,n}\{p_j\}$ is scheduled, thus, $C^*_{\max} \geq p_{\max} f_M^a(k)$. Since $f_M^a(k) \geq f_M^a(n)$ for any $1 \leq k \leq n$ due to the decreased curve $f_M^a(.)$. Therefore, $C^*_{\max} \geq p_{\max} f_M^a(n) = \mathrm{LB}_0$.
- Proof of equation (4.2): based on Lemma 4.1, the contribution of j_k (the job j_k having the kth initial processing time $p_{(k)}$) in the total load is $tl_k = p_{(k)} f_M^a(s_k)$ with $s_k \leq k$. Since, $s_k \leq k$ then $f_M^a(s_k) \geq f_M^a(k)$ ($f_M^a(.)$ is a decreasing curve) and $tl_k \geq p_{(k)} f_M^a(k)$. Denoting the total load on a machine M_i $(1 \leq i \leq n)$ by TL_i, then $\mathrm{TL}_i \leq C^*_{\max}$ and consequently $\frac{1}{m}\sum_{i=1}^{m} \mathrm{TL}_i \leq C^*_{\max}$. Furthermore, $\sum_{i=1}^{m} \mathrm{TL}_i = \sum_{k=1}^{n} tl_k$, and by taking into account all the previous remarks, we conclude that $\mathrm{LB}_1 = \frac{1}{m}\sum_{k=1}^{n} p_{(k)} f_M^a(k) \leq \frac{1}{m}\sum_{k=1}^{n} tl_k = \frac{1}{m}\sum_{i=1}^{m} \mathrm{TL}_i \leq C^*_{\max}$.

- Proof of equation (4.3): obviously, $\text{LB}_2 = \max(\text{LB}_0, \text{LB}_1)$ is a valid lower bound since LB_0 and LB_1 are valid lower bounds.

□

Remark 4.3. According to the expression of LB_1, a lower bound of the total load is obtained by scheduling all the jobs on only one machine according to the SPT rule.

4.2. New developed lower bounds

In this subsection, a set of new lower bounds will be presented.

4.2.1. Max processing time based lower bound

A first improvement of LB_0 is LB_0^1 is given throughout the Lemma 4.4.

Lemma 4.4. *A valid lower bound for the* $P_m \mid p_{j,k} = p_j f_M^a(k) \mid C_{\max}$ *is:*

$$\text{LB}_0^1 = \max_{1 \le j \le n} \{p\,(_j)\, f_M^a(j)\}$$

Proof. In an optimal schedule, the job j_k (with the kth initial processing time $p_{(k)}$) is scheduled at a position s_k. Thus $C^*_{\max} \ge p_{(k)} f_M^a(s_k)$. According to the Lemma 4.1, $s_k \le k$, therefore $C^*_{\max} \ge p_{(k)} f_M^a(k)$ for $1 \le k \le n$. Consequently, $C^*_{\max} \ge \max_{1 \le k \le n} \left(p_{(k)} f_M^a(k)\right) = \text{LB}_0^1$. □

4.2.2. Max position based lower bounds

The lower bounds LB_0^1, LB_1, and LB_2 might be enhanced, by observing that the last position in any machine and for an optimal schedule might be strictly less than n. In this context, the Proposition 4.5 is presented.

Proposition 4.5. *Let* UB *be an upper bound and h the maximum number of scheduled jobs from J according to SPT, in a single machine, without exceeding* UB *(taking into account the learning effect). Then, in an optimal schedule and in any machine, the number of scheduled jobs does not exceed h.*

Proof. By contradiction, assume that there exists an optimal schedule σ^* with makespan value $C^*_{\max}$ and a machine M_i containing more than h jobs. Let $j_1, j_2, \ldots, j_l$ $(l > h)$ be the assigned jobs to M_i regarding σ^*. Since the jobs are sorted in the non-decreasing order of their processing times then $p_{j_k} \ge p_{(k)}$ $(1 \le k \le l)$ and consequently $C^*_{\max} \ge \sum_{k=1}^{l} p_{j_k} f_M^a(k) \ge \sum_{k=1}^{l} p_{(k)} f_M^a(k)$. Clearly, $\sum_{k=1}^{l} p_{(k)} f_M^a(k) = \sum_{k=1}^{h} p_{(k)} f_M^a(k) + \sum_{k=h+1}^{l} p_{(k)} f_M^a(k)$. According to the definition of h, one could observe that $\sum_{k=1}^{h} p_{(k)} f_M^a(k) + \sum_{k=h+1}^{l} p_{(k)} f_M^a(k) > \text{UB}$. Thus, $C^*_{\max} > \text{UB}$ which contradicts the fact that $C^*_{\max}$ is the optimal makespan value. □

An illustration of Proposition 4.5 is presented over the following example.

Example 4.6. Let $n = 5$, $m = 2$, $a = -0.1, M = 0.5$. The processing times are given as follows, $\{71, 29, 9, 24, 22\}$. Applying the LPT rule, a feasible schedule with $C_{\max} = 79.69 = \text{UB}$ is obtained. Sorting the jobs according to their non-decreasing initial processing times results in the following list: $3, 5, 4, 2, 1$. Applying the SPT rule for the sequence $3, 5, 4$ the obtained value is 53.01. The same latter procedure applied to $3, 5, 4, 2$ gives a completion time of 80.13, therefore $h = 3$.

According to Example 4.6, $h = 3$, which involves that in any optimal schedule a machine can only contain at most three jobs. Taking into account the Proposition 4.5, a new lower bound is deduced and presented over the Proposition 4.7.

Proposition 4.7. *Let h be the position found by Proposition 4.5, then*

$$\mathrm{LB}_0^2 = \max\left\{\max_{1\ \leq\ k\ \leq h-1}\left(p_{(k)} f_M^a(k)\right),\ p_{(n)} f_M^a(h)\right\}$$

is a valid lower bound for $P_m \mid p_{j,k} = p_j f_M^a(k) \mid C_{\max}$.

Proof. In an optimal schedule the job j_k (with the kth initial processing time $p_{(k)}$) is scheduled at a position s_k with $s_k \leq k$. For $1 \leq k \leq h-1$, we have $C^*_{\max} \geq p_{(k)} f_M^a(s_k) \geq p_{(k)} f_M^a(k)$. In addition, for $h \leq k \leq n$, the optimal value satsfies $C^*_{\max} \geq p_{(k)} f_M^a(s_k) \geq p_{(k)} f_M^a(h)$ since h is the last position. Therefore, $C^*_{\max} \geq$

$\max\left\{\max_{1\ \leq\ k\ \leq\ h-1}\left(p_{(k)} f_M^a(k)\right),\ p_{(n)} f_M^a(h)\right\} = \mathrm{LB}_0^2$. □

In Example 4.6, $h = 3$. Consequently, $\mathrm{LB}_0^2 = \max\{p_{(1)},\ p_{(2)} f_M^a(2),\ p_{(5)} f_M^a(3)\} = \max(9,\ 21.25,\ 67.30) = 67.30 > \mathrm{LB}_0^1$.

In the same context, LB_1 might be improved throughout Proposition 4.8.

Proposition 4.8. *Let h be the position determined by Proposition 4.5, then:*

$$\mathrm{LB}_1^1 = \frac{1}{m}\left\{\sum_{j=1}^{h-1} p_{(j)} f_M^a(j) + \left(\sum_{j=h}^{n} p_{(j)}\right) f_M^a(h)\right\}$$

is a valid lower bound.

Proof. In an optimal schedule, the job j_k (with the kth initial processing time $p_{(k)}$) is scheduled at a position s_k with $s_k \leq k \leq h$. For $1 \leq k \leq h-1$, the contribution of j_k in the total load is $tl_k = p_{(k)} f_M^a(s_k) \geq p_{(k)} f_M^a(k)$ and for $h \leq k \leq n$, the corresponding contribution of j_k in the total load is $tl_k = p_{(k)} f_M^a(s_k) \geq p_{(k)} f_M^a(h)$. Denoting the total load on a machine M_i $(1 \leq i \leq n)$ by TL_i, then $\mathrm{TL}_i \leq C^*_{\max}$ and consequently $\frac{1}{m}\sum_{i=1}^{m} \mathrm{TL}_i \leq C^*_{\max}$. Furthermore, $\sum_{i=1}^{m} \mathrm{TL}_i = \sum_{k=1}^{n} tl_k$, and $\mathrm{LB}_1^1 = \frac{1}{m}\left(\sum_{k=1}^{h-1} p_{(k)} f_M^a(k) + f_M^a(h)\sum_{k=h}^{n} p_{(k)}\right) \leq \frac{1}{m}\sum_{k=1}^{n} tl_k = \frac{1}{m}\sum_{i=1}^{m} \mathrm{TL}_i \leq C^*_{\max}$. □

Based on Example 4.6 and Proposition 4.8 the enhancement will be: $\mathrm{LB}_1^1 = \frac{1}{2}\left(\sum_{j=1}^{2} p_{(j)} f_M^a(j) + f_M^a(3)\left\{p_{(3)} + p_{(4)} + p_{(5)}\right\}\right) = 73.90$. $\mathrm{LB}_1^1 = 73.90$, instead of $\mathrm{LB}_1 = 72.93$, obtained by the classical way.

4.2.3. Max position-capacity based lower bound

The latter lower bound LB_1^1 could be improved by observing that in any optimal schedule a position is assigned at maximum for m machines, this is because of the capacity constraint. Therefore, for the latter example (Example 4.6) position 3 is at most assigned to two jobs, which is not satisfied by LB_1^1. Instead of LB_1^1 an improvement could be made by considering $\mathrm{LB}_1^2 = \frac{1}{2}\{p_{(1)} f_M^a(1) + p_{(2)} f_M^a(2) + p_{(3)} f_M^a(2) + p_{(4)} f_M^a(3) + p_{(5)} f_M^a(3)\} = 74.12$. In this example, position 3 is assigned to only two jobs. The candidates are jobs with largest processing times $p_{(5)}$ and $p_{(4)}$. Position 2, will be occupied by two jobs with processing times $p_{(2)}$ and $p_{(3)}$, which are the jobs with largest processing times among the unassigned jobs $1, 2, 3$. In other words the positions that will be assigned are:

- h positions in the first machine M_1 from position 1 to position h.
- $n-h$ positions in the $m-1$ remaining machines, starting from the last position h.

The $n-h$ positions are distributed on the $m-1$ machines $M_2, M_3, \ldots, M_m$ as follows. First, let $n-h = \left\lfloor \frac{n-h}{m-1} \right\rfloor (m-1) + r_h$ with $0 \leq r_h < m-1$. Then

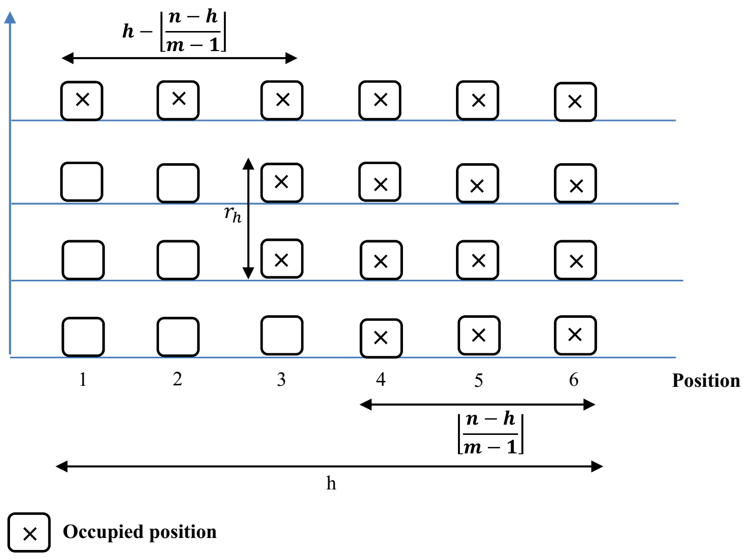
$h - \left\lfloor \frac{n-h}{m-1} \right\rfloor$
r_h
1
2
3
4
5
6
Position
$\left\lfloor \frac{n-h}{m-1} \right\rfloor$
h
Occupied position

FIGURE 1. Positions distribution for LB_1^2.

– If $r_h = 0$, the number of positions to be considered in each machine $M_2, M_3, \ldots, M_m$ is $\left\lfloor \frac{n-h}{m-1} \right\rfloor$, starting from position $h - \left\lfloor \frac{n-h}{m-1} \right\rfloor + 1$ to position h.
– If $r_h \neq 0$, the number of positions to be considered in each machine $M_2, M_3, \ldots, M_{r_h+1}$ is $\left\lfloor \frac{n-h}{m-1} \right\rfloor + 1$, starting from position $h - \left\lfloor \frac{n-h}{m-1} \right\rfloor$ to position h, and $\left\lfloor \frac{n-h}{m-1} \right\rfloor$ positions from position $h - \left\lfloor \frac{n-h}{m-1} \right\rfloor + 1$ to position h, in machines $M_{r_h+2}, M_{r_h+3}, \ldots, M_m$. This distribution of positions is illustrated in Figure 1, where $n = 17$ and $h = 6$.

Once all the required positions are selected, the jobs are assigned according to the SPT rule to these positions. The obtained average load is a lower bound for the considered problem as expressed in Proposition 4.9.

Proposition 4.9. *Let h be the position determined by Proposition 4.5, and $n - h = \left\lfloor \frac{n-h}{m-1} \right\rfloor (m-1) + r_h$ where $0 \leq r_h < m-1$. Considering the following cases:*

– *If $r_h = 0$ then $\mathrm{LB}_1^2 = \frac{1}{m} \left\{ \sum_{k=1}^{\alpha_h} p_{(k)} f_M^a(k) + \sum_{k=\alpha_h+1}^{h} f_M^a(k) \left(\sum_{i=\sigma_k}^{\delta_k} p_{(i)} \right) \right\}$ with $\alpha_h = h - \left\lfloor \frac{n-h}{m-1} \right\rfloor$, $\sigma_k = m(k - \alpha_h - 1) + \alpha_h + 1$, and $\delta_k = m(k - \alpha_h) + \alpha_h$.*
– *If $r_h \neq 0$ then*
$\mathrm{LB}_1^2 = \frac{1}{m} \left\{ \sum_{k=1}^{\alpha_h - 1} p_{(k)} f_M^a(k) + f_M^a(\alpha_h) \sum_{k=\alpha_h}^{\alpha_h + r_h} p_{(k)} + \sum_{k=\alpha_h+1}^{h} f_M^a(k) \left(\sum_{i=\sigma_k}^{\delta_k} p_{(i)} \right) \right\}$ with $\alpha_h = h - \left\lfloor \frac{n-h}{m-1} \right\rfloor$, $\sigma_k = m(k - \alpha_h) + 1 + r_h$, and $\delta_k = m(k + 1 - \alpha_h) + r_h$.

LB_1^2 is a valid lower bound.

Proof. The proof is based on the three following observations.

– All positions from 1 to h have to be filled at least in one machine.
– A position k has to be filled with m jobs before any assignment to any other position l with $l < k$. Indeed, let j be a job with a processing time p_j placed at position l where at least a position k is empty, then the contribution of j in the total load is $p_j f_M^a(l)$. However placing j in the empty position k will reduce the total load by $p_j \left(f_M^a(l) - f_M^a(k) \right)$.

- Given two jobs i and j with respective processing times p_i and p_j, If i and j are placed in respective positions l and k such that $l \leq k$ then $p_i \leq p_j$. Indeed, assuming that $p_i > p_j$ and comparing the contributions of i and j in the total load, placed respectively in l, k and k, l. The comparison is: $A = p_i f_M^a(l) + p_j f_M^a(k) - p_i f_M^a(k) - p_j f_M^a(l) = (p_i - p_j) f_M^a(l) - (p_i - p_j) f_M^a(k) = (p_i - p_j)(f_M^a(l) - f_M^a(k))$. Since, $f_M^a(l) \geq f_M^a(k)$ then $A > 0$. Therefore, placing i and j in the respective positions l, k with $p_i \leq p_j$ reduces the total load, which matches with the minimization of the total load.

□

According to the three previous observations, the machines should be totally filled from the last position h with the jobs having the largest processing times among the unassigned jobs and the following two cases have to be considered:

- If $r_h = 0$, only the first machine M_1 has positions 1 to position $\alpha_h = h - \frac{n-h}{m-1}$ to be assigned. The jobs that will be placed are those with processing times $p_{(k)}$ with $1 \leq k \leq \alpha_h$. Therefore the resulted load is $\sum_{k=1}^{\alpha_h} p_{(k)} f_M^a(k)$. in addition, from position $\alpha_h + 1$ to position h all the machines are filled with jobs and the resulted load is: $\sum_{k=\alpha_h+1}^{h} f_M^a(k) \left(\sum_{i=\sigma_k}^{\delta_k} p_{(i)}\right)$ where $\sigma_k = m(k - \alpha_h - 1) + \alpha_h + 1$ and $\delta_k = m(k - \alpha_h) + \alpha_h$.
- If $r_h \neq 0$ then there is three cases. The positions from 1 to $\alpha_h - 1$ in only M_1 are assigned with jobs and the participation in the load is $\sum_{k=1}^{\alpha_h - 1} p_{(k)} f_M^a(k)$. In the position α_h there are $1 + \alpha_h$ providing their position to be used and the contribution in the total load is $f_M^a(\alpha_h) \sum_{k=\alpha_h}^{\alpha_h + r_h} p_{(k)}$. The positions from $\alpha_h + 1$ to h are available for all the machines and their contribution in the total load time is $\sum_{k=\alpha_h+1}^{h} f_M^a(k) \left(\sum_{i=\sigma_k}^{\delta_k} p_{(i)}\right)$ with $\sigma_k = m(k - \alpha_h) + 1 + r_h$ and $\delta_k = m(k + 1 - \alpha_h) + r_h$.

4.2.4. Max position-capacity and connectivity based lower bound

In addition, if a position i $(1 < i \leq h)$ of a machine is assigned in an optimal schedule then the position $i - 1$ should be assigned as well, this is the connectivity constraints while assigning jobs to positions. In Example 4.6, in the second machine, the position 2 is assigned without having any job in position 1. Therefore, taking into account the last observation, the LB_1^2 could be improved as follows.

$$\text{LB}_2^2 = \frac{1}{2}\left\{p_{(1)} f_M^a(1) + p_{(2)} f_M^a(1) + p_{(3)} f_M^a(2) + p_{(4)} f_M^a(2) + p_{(5)} f_M^a(3)\right\} = 74.76.$$

Similarly to the elaboration of LB_1^2, the positions that will be occupied based on the connectivity constraint, is expressed in the following remark.

Remark 4.10. First, let $n = \left\lfloor \frac{n}{h} \right\rfloor h + r$ with $0 \leq r < h$. Then

- If $r = 0$, the number of the positions to be considered in the $\left\lfloor \frac{n}{h} \right\rfloor$ machine $M_1, M_3, \ldots, M_{\left\lfloor \frac{n}{h} \right\rfloor}$ is h, starting from position 1 to position h.
- If $r \neq 0$, the number of the positions to be considered in each machine $M_1, M_3, \ldots, M_{\left\lfloor \frac{n}{h} \right\rfloor}$ is $\left\lfloor \frac{n}{h} \right\rfloor$, starting from position 1 to position h, and r positions from position 1 to position r, in the machine $M_{\left\lfloor \frac{n}{h} \right\rfloor + 1}$. This distribution of the positions is illustrated in Figure 2, where $n = 17$ and $h = 6$.

The latter lower bound will be expressed over the following proposition.

Proposition 4.11. *Let $n = \left\lfloor \frac{n}{h} \right\rfloor h + r$ where $0 \leq r < h$ then*

- *If $r = 0$ then $\text{LB}_1^3 = \frac{1}{m}\left\{\sum_{k=1}^{h} f_M^a(k)\left(\sum_{i=\alpha_k}^{\beta_k} p_i\right)\right\}$ with $\alpha_k = (k-1)\left\lfloor \frac{n}{h} \right\rfloor + 1$ and $\beta_k = (k)\left\lfloor \frac{n}{h} \right\rfloor$ $(k = 1, \ldots, h)$.*
- *If $r \neq 0$ then $\text{LB}_1^3 = \frac{1}{m}\left\{\sum_{k=1}^{r} f_M^a(k)\left(\sum_{i=\alpha_k}^{\beta_k} p_i\right) + \sum_{j=r+1}^{h} f_M^a(j)\left(\sum_{i=\gamma_j}^{\delta_j} p_i\right)\right\}$ with $\alpha_k = (k-1)\left\lfloor \frac{n}{h} \right\rfloor + k$ and $\beta_k = (k)\left\lfloor \frac{n}{h} \right\rfloor + k$, $(k = 1, \ldots, r).\gamma_j = (j-1)\left\lfloor \frac{n}{h} \right\rfloor + r + 1$ and $\delta_j = j\left\lfloor \frac{n}{h} \right\rfloor + r$ $(k = r+1, \ldots, h)$,*

LB_1^3 is a valid lower bound.

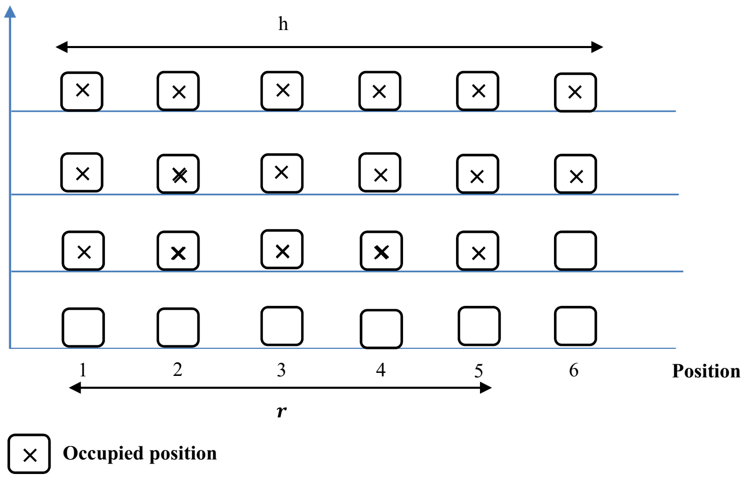

FIGURE 2. Positions distribution for LB_1^3.

Proof. In general, the elementary transformation which consists in omitting a position k $(k = 1,\dots,h-1)$ and replacing it by another position $l(l = 1,\dots,h-1)$ with $l < k$, while respecting the connectivity constraint, conducts to an increase in the total load. In particular this holds for $k = h$. In addition, obtaining a new distribution of the occupied positions, that satisfies the connectivity constraint, is done by a successive elementary transformations. Therefore any modification in the occupied positions as described in 4.10, will conduct to an increase in the total load. Consequently, LB_1^3 is a valid lower bound. $\square$

In the sequel, the Example 4.12 is presented in order to illustrate the lower bound LB_1^3.

Example 4.12. Let $n = 10$, $m = 5$, $a = -0.1$, $M = 0$ and the processing times are presented as follows $p_j = \{12, 99, 90, 13, 25, 16, 11, 27, 34, 10\}$. $\mathrm{LB}_1^3 = 58.70$, $h = 6$ then $\mathrm{LB}_0^2 = \max\{10, 10.26, 10.75, 11.32, 13.62, 82.75\} = 82.75$, then $\mathrm{LB}_0^2 > \mathrm{LB}_1^3$.

Corollary 4.13. $\mathrm{LB} = \max\left(\mathrm{LB}_1^3, \mathrm{LB}_0^2\right)$ *is a valid lower bound for the consider problem.*

Remark 4.14. LB_1^3 dominates LB_1^2.

Proof. A job j is assigned to a position $a_j\,(1 \le a_j \le h)$ for LB_1^2 and to a position $b_j\,(1 \le b_j \le h)$ for LB_1^3. Trivially, $a_j \ge b_j$ then the respective contributions of job j in LB_1^2 and LB_1^3 are $p_j f_M^a(a_j)$ and $p_j f_M^a(b_j)$. Since $f_M^a(a_j) \le f_M^a(b_j)$ ($f_M^a(.)$ is a decreasing curve), consequently, $\mathrm{LB}_1^3 \ge \mathrm{LB}_1^2$. $\square$

The h value might be improved (decreased) and the following proposition is an example.

Proposition 4.15. *Let* $J_h = \{1, 2, \dots, h\}$ *. Applying the SPT on* $J_r = J \setminus J_h$ *yields* h_1 *the maximum number of jobs that can be placed on each machine* $M_2, \dots, M_m$*. If* $(m-1)\times h_1 < |J_r|$ *and SPT on* $\{1, 2, \dots, h-1, h+1\} >$ UB *then* $h := h - 1$*.*

Proof. If $(m-1) \times h_1 < |J_r|$ then the set of jobs J_r can't be scheduled on the $(m-1)$. Therefore, in any optimal schedule the set J_h can't be placed on only one machine. Let S_{M_1} the number of scheduled jobs on M_1, then $S_{M_1} = h$ or $S_{M_1} < h$. If $S_{M_1} = h$ then the set J_h should be modified. The first modification consists on replacing the job h by job $h+1$. If SPT on $\{1, 2, \dots, h-1, h+1\}$ gives a completion time greater than UB, this involves that the set $\{1, 2, \ldots, h-1, h+1\}$ is not a candidate to be scheduled on that machine. Consequently, any other subset without job h with cardinality h is not also a candidate to be placed on M_1. If there is such a subset $a_1, a_2, \ldots, a_h$ then $p_{a_i} \geq p_i$ for $1 \leq i \leq h-1$ and $p_{a_h} \geq p_{h+1}$ and then the corresponding completion time $\sum_{k=1}^{h} p_{a_k} f_M^a(k) + p_{h+1} \geq \sum_{k=1}^{h-1} p_k f_M^a(k) + p_{h+1} f_M^a(h) > \mathrm{UB}$. Thus $S_{M_1} < h$ so h will be enhanced and the new value is $h-1$. □

Example 4.16. Considering the following instance $n = 10$, $m = 2$, $a = -0.1$ and $M = 0.5$.
The processing times are given as follows, $p_j = \{12, 5, 3, 14, 13, 3, 2, 17, 19, 16\}$. A feasible schedule is obtained by application of a picked heuristic. This schedule has a maximum completion time UB $= 48.92$. The SPT in a single machine gives the maximum sequence $7, 6, 3, 2, 1, 5, 4$ for which the completion time does not exceed UB. Therefore $h = 7$ and $J_r = \{10, 8, 9\}$. Applying the SPT on J_r in a single machine gives a completion time $C_{10,8,9} = 50.44$ which exceeds UB, thus $h_1 = 2$. On the other hand we have $(m-1) \times h_1 = 2 \times (2-1) < 3 = |J_r|$. In addition, applying SPT on the set $\{1, 2, \ldots, h-1, h+1\} = \{7, 6, 3, 2, 1, 5, 10\}$ gives a completion time 50.04 which exceeds UB, hence $h = 6$ instead of 7 according to the previous proposition.

5. Heuristics

In this section, several heuristics will be developed. These heuristics are of two types. The first type is based on greedy algorithms with some variants. The second type is based on solving more advanced problems as a knapsack problem and the classic parallel machines problem.

Remark 5.1. After applying any heuristic, the jobs in any machine should be reordered in the non-decreasing order of their initial processing time p_j.

5.1. Dispatching rules

5.1.1. Shortest processing time (SPT)

The jobs are sorted in the non-decreasing order of their processing times and scheduled on the parallel machines according to this order on the most available machine. Other heuristics might provide a better solution as observed in the following heuristic.

5.1.2. Longest processing time (LPT)

The jobs are scheduled on the parallel machines according to the non-increasing order of their processing time on the most available machine. The result given by the LPT rule encourages the enhancement of the SPT rule as for the following proposed heuristic.

5.1.3. Modified SPT (MSPT)

The jobs are sorted in the non-decreasing order of their processing times. At each iteration k $(k = 1, \ldots, n)$ of this heuristic the set of already scheduled jobs is denoted J_S^k. The first job among the unscheduled jobs $J_{\mathrm{US}}^k = J \backslash J_S^k$, is k. This job is selected to be placed on a machine M_l. This machine M_l is picked such that the minimum completion time of the job k is reached (M_l might not be the machine with the smallest available time). In other words, M_l is a machine such that:

$$u_l + p_{k,n_l+1} = \min_{1 \leq i \leq m} (u_i + p_{k,n_i+1})$$

where n_i is the number of the scheduled jobs on M_i and u_i its availability. This heuristic (MSPT) is illustrated over the Example 5.2.

Example 5.2. Let $n = 10$, $m = 2$, $a = -0.1$ and $M = 0.5$. The processing times are given as follows, $p_j = \{209, 131, 89, 159, 185, 143, 110, 92, 1, 158\}$. Sorting the jobs according to their non-decreasing initial processing times (p_j) yields the following list: $9, 3, 8, 7, 2, 6, 10, 4, 5, 1$. The SPT rule provides the schedule: on M_1: $9, 8, 2, 10, 5$

and on M_2: $3, 7, 6, 4, 1$. The obtained maximum completion time is $C_{\max} = 673.05$. Using now the modified SPT heuristic (MSPT) provides the following feasible schedule. On M_1: $9, 3, 7, 6, 4, 1$ and on M_2: $8, 2, 10, 5$. Then the makespan obtained by MSPT is $U_{\text{MSPT}} = C_{\max} = 664.08$.

Remark 5.3. The experimental results show that there is no dominance between SPT and MSPT.

5.1.4. Modified LPT (MLPT)

For this heuristic (Modified LPT), the jobs are sorted in the non-increasing order of their initial processing times p_j. Keeping the same procedure like MSPT, a feasible schedule will be obtained. After finishing the schedule, the jobs in each machine are rescheduled according to the SPT rule. The whole procedure for the MLPT heuristic is illustrated over Example 5.4.

Example 5.4. Let $n = 6$, $m = 2$, $a = -0.1$ and $M = 0.5$ and the processing times is given as follows, $p_j = \{48, 27, 99, 26, 31, 53\}$.

The LPT rule gives the following schedule: on $M_1 : 3, 5$ and on $M_2 : 6, 1, 2, 4$. After rearranging the jobs on each machine in the non-decreasing order of their processing time, the following schedule is obtained: on $M_1 : 5, 3$ and on $M_2 : 4, 2, 1, 6$, with $C_{\max} = 147.16$. However, applying the MLPT, the obtained schedule is: on $M_1 : 3, 2, 4$ and on $M_2 : 6, 1, 5$.

The rearrangement of the jobs on each machine, in the non-decreasing order of their processing time, yields the following schedule: on M_1:$4, 2, 3$ and on M_2:$5, 1, 6$, with $C_{\max} = 145.94$.

The experimental results show that there is no dominance between LPT and MLPT.

Remark 5.5. The complexity of the previous heuristics is in $O\left(n \ln(n)\right)$ time since only the sorting of the jobs in the non-decreasing (respectively, non-increasing) order is required.

5.1.5. Randomised SPT (RSPT)

This heuristic is based on the following idea. At the kth $(k = 1, \ldots, n)$ iteration, instead of scheduling the first unscheduled job, we select randomly one job among the pair of the two first unscheduled jobs. This is done by assigning randomly a probability α to the first unscheduled job and a probability $1 - \alpha$ to the second one, and the job with the largest probability is picked. Example 5.6 presents the details RSPT.

Example 5.6. For $n = 5$, $m = 2$, $a = -0.1$, $M = 0.5$ and processing times as is given as follows, $p_j = \{100, 86, 160, 104, 83\}$.

The obtained schedule after performing the SPT rule is: on $M_1 : 5, 1, 3$ and on $M_2 : 2, 4$, with $C_{\max} = 331.32$. Moreover, the RSPT provides the following schedule: on $M_1 : 2, 4, 5$ and on $M_2 : 1, 3$ where $C_{\max} = 265.19$.

The corresponding algorithm of the RSPT heuristic are detailed and given in Algorithm 1.

The latter procedure could be repeated several times in order to obtain a better solution. In our case, an experimental study has been conducted to enable us an efficient choice of the repetitions number. The balancing between the consumed time and the quality of the solution, suggested that 100 times is sufficient.

5.1.6. Randomised LPT (RLPT)

First, the jobs are sorted according to the non-increasing order of their processing times. The remaining of the Randomised LPT (RLPT) is similar to the RSPT procedure (described above), except that for the obtained schedule a reassignment of the jobs in each machine according to the SPT is required.

5.2. Advanced heuristics

In this subsection, more advanced heuristics than the previous ones are developed and presented. These heuristics are based on solving exactly NP-Hard problems using efficient procedures, such as the parallel machine scheduling problem $P_m \mid \mid C_{\max}$.

Algorithm 1. RSPT heuristic algorithm.

```
1: Sorting jobs on non-decreasing order of $p_j$.
2: Initialize $n_{\text{rest}} = n$ and $\tilde{J} = J$.
3: while $(n_{\text{rest}} > 1)$ do
4:     $J1$ is the first largest job and $J2$ is the second largest job.
5:     $r = Random[1, 100]$.
6:     if $r \in [1, 20]$ then
7:         Schedule $J1$ on the most available machine.
8:         $\tilde{J} = J \setminus J1$.
9:     else
10:        Schedule $J2$ on the most available machine.
11:        $\tilde{J} = J \setminus J2$.
12:    end if
13:    $n_{\text{rest}} = n_{\text{rest}} - 1$.
14: end while
15: Schedule the last remaining job on the most available machine.
16: Calculate $C_{\max}$.
17: Stop. Return $C_{\max}$.
```

5.2.1. Optimal Parallel machine based heuristic (Opt)

For the current heuristic, a $P_m \,||\, C_{\max}$ problem is associated with the $P_m \,|p_{j,k} = p_j f_M^a(k)|\, C_{\max}$ problem. The $P_m \,||\, C_{\max}$ problem is obtained by setting the processing times as p_j $(j \in J)$. This parallel machine scheduling problem is exactly solved using the efficient branch and bound algorithm provided in [9]. The optimal obtained schedule is denoted by σ. Naturally, a feasible schedule for $P_m \mid p_{j,k} = p_j f_M^a(k) \mid C_{\max}$, is deduced from σ as follows. The same assignment of jobs to machines is kept as in the schedule σ. In addition, in each machine the jobs are rescheduled according to SPT rule. At this stage the learning effect is reconsidered, the final makespan is computed and the obtained value is denoted U_{Opt}. This heuristic is denoted *Opt*.

A time limit is fixed, to run the presented branch and bound in [9], since the $P_m \,||\, C_{\max}$ is an NP-Hard problem. If the branch and bound does not reach an optimal solution within this time limit, then the best reached feasible schedule while running the branch and bound is considered instead of σ.

5.2.2. Knapsack based heuristic (KN)

Let LB be a lower bound for the studied problem. For each iteration i $(1 \le i \le m-1)$, the machine M_i (empty) is selected and a knapsack problem with the following settings is solved.

– Set of items is the set of unscheduled jobs.
– The size $size_j$ of item j is p_j $(j \in J_{\text{US}})$.
– Capacity C is $\lfloor \text{LB} \rfloor$.
– Objective function: maximizing the number of selected jobs.

The latter knapsack problem is polynomially solvable [6]. This procedure is repeated until finishing the iteration $m-1$. The remaining jobs are scheduled on the last machine M_m. Finally, in each machine, the SPT rule is applied. Now, the learning effect is considered and the makespan noted U_{KN} is calculated. The described heuristic is denoted KN. Denoted by KN(S, C) the function that apply the knapsack problem applied to the set S and capacity C.

In the sequel a brief description of the corresponding KN algorithm (Algorithm 2).

5.2.3. A Bandwidth LPT based heuristic (BLPT)

Let LB be a lower bound for the studied problem $P_m \mid p_{j,k} = p_j f_M^a(k) \mid C_{\max}$. For each iteration i $(1 \le i \le m)$, the (empty) machine M_i is selected, and the LPT rule is applied to the unscheduled jobs for this machine until reaching LB, with learning effect consideration. These iterations are repeated until m machines are loaded.

Algorithm 2. KN heuristic algorithm.

```
1: Calculate LB for the $P_m \mid p_{j,k} = p_j f_M^a(k) | C_{\max}$ problem.
2: Set $C = \lfloor \text{LB} \rfloor$ , $J_{\text{US}} = J$, and $size_j = p_j$.
3: for $i = 1$ to $m - 1$ do
4:     $J_{\text{KN}} = \text{KN}(J_{\text{US}}, C)$.
5:     Assign $J_{\text{KN}}$ to $M_i$ and $J_{\text{US}} := J_{\text{US}} \setminus J_{\text{KN}}$
6: end for
7: Assign $J_{\text{US}}$ on $M_m$.
8: Apply SPT in each machine.
9: Compute makespan $U_{\text{KN}}$ with learning effect.
10: Stop. Return $U_{\text{KN}}$.
```

For the remaining jobs, MLPT and MSPT are applied separately to all machines even thought to exceed LB. The heuristics' value is selected. The choice of the LPT, MLPT and MSPT procedures to be applied in this heuristic is justified experimentally as it will be shown in the experimental study section. Seeking clarity, the above-described heuristic (BLPT) will be presented over the Algorithm 3.

Algorithm 3. BLPT heuristic algorithm.

```
1: Calculate LB for the $P_m \mid p_{j,k} = p_j f_M^a(k) | C_{\max}$ problem.
2: Set $J_{\text{US}} = J$.
3: for $i = 1$ to $m$ do
4:     Apply LPT rule on machine $M_i$ and $J_{\text{US}}$, until reaching LB, with learning effect.
5:     Assign the set of selected jobs $J_{LPT}$ on $M_i$ and $J_{\text{US}} = J_{\text{US}} \setminus J_{LPT}$.
6: end for
7: Apply MLPT on $J_{\text{US}}$, and all machines with makespan $C^1_{\max}$.
8: Apply MSPT on $J_{\text{US}}$, and all machines with makespan $C^2_{\max}$.
9: Select from STEP 3 and STEP 4 the rule giving the $C = \min\left(C^1_{\max}, C^2_{\max}\right)$.
10: Apply SPT in each machine $M_i$.
11: Compute makespan $U_{\text{BLPT}}$.
12: Stop. Return $U_{\text{BLPT}}$.
```

5.2.4. Min–Max restarting heuristic (MMR)

The current heuristic is composed of two phases. The first one is a constructive phase, allowing the development of an initial feasible solution. The second phase is an improved one. During the first phase, the LPT rule is used to develop a feasible schedule. After, that the SPT rule is applied to each machine. For the improvement phase, the most loaded and the least loaded machines are selected. The MLPT is applied to these two machines and the subset of jobs that are already scheduled on them. This might improve the makespan. The improvement phase is repeated until there is no detected enhancement.

5.3. Further enhancements

In this subsection, some enhancements procedures will be presented and performed on the previous heuristics.

5.3.1. Decreasing the number of the machines

In an optimal schedule, we may not use all the m machines. This can be seen in Example 5.7. Thus, decreasing the number of machines may results in an improvement of a given feasible solution. This will be applied to all the presented heuristics. Clearly, the complexity of each one of the improved heuristics will be multiplied only by m.

Example 5.7. Let $n = 5$, $m = 2$, $a = -0.322$, $M = 0.5$. The processing times are given as follows, $p_1 = 126, p_2 = 1, p_3 = 3, p_4 = 1$, and $p_5 = 2$. By application of the SPT rule for the above data, we obtain the following schedule: on $M_1 : 2, 5, 1$ and on $M_2 : 4, 3$ with $C_{\max} = 110.02$. Now, applying the same rule SPT with the same data but only on one machine (decreasing the number of machines). This provides the following schedule: $2, 4, 5, 3, 1$ and the corresponding $C_{\max} = 106.58 < 110.02$.

Remark 5.8. The decreasing of the number of machines may result in the non-decreasing of the makespan value as it will be shown in Example 5.9.

Example 5.9. Let $n = 7$, $m = 4$, $a = -0.322$, $M = 0.5$ and the processing times are as follows, $p_j = \{101, 3, 8, 1, 9, 8, 2\}$. Applying the MSPT rule on $m = 4$, we obtain the following schedule: on $M_1 : 4, 6$, on $M_2 : 7, 5$, on $M_3 : 2, 1$ and on $M_4 : 3$ with $C_{\max} = 93.89$. Despite, decreasing the number of machines to $m = 3$, we obtain the following schedule: on $M_1 : 4, 3, 1$, on $M_2 : 7, 6$ and on $M_3 : 2, 5$ with $C_{\max} = 94.15$. In addition, for $m = 2$, we obtain the following schedule: on $M_1 : 4, 2, 6, 1$ and on $M_2 : 7, 3, 5$ with $C_{\max} = 93.32$.

Remark 5.10. Clearly, the value of the makespan does not respect any monotony in decreasing of the number of machines. Consequently, all the number of machines should be tested.

Motivated by the last remark (Rem. 5.10), Example 5.9 has been investigated in depth and all the feasible solutions are explored $\left(1024 = 4^5 \text{ possibilities}\right)$. The optimal schedule was reached for only 2 machines instead of 4 provided machines. The corresponding optimal schedule is: on $M_1 : 4, 7, 2, 1$ and on $M_2 : 6, 3, 5$ with $C^*_{\max} = 88.16$.

5.3.2. SPT filling enhancement

Let UB be the makespan value of a given heuristic, then this value might be enhanced. A feasible schedule could be proposed as follows. In the first machine M_1, the jobs are scheduled according to the SPT rule without exceeding UB. Assume that the completion time of the scheduled jobs in the first machine is $\text{UB}_1 < \text{UB}$. Iteratively, the unscheduled jobs are assigned to each one of the remaining machines $M_i \,(2 \le i \le m)$, using the SPT rule and without exceeding UB_1. At the last machine M_m, if the completion time of the remaining scheduled jobs is less than or equal to UB_1 then a feasible schedule with less value UB_1 is obtained. Otherwise, the old schedule is maintained. It is worth noting that the complexity of SPT filling enhancement is in $O\left(mn \ln(n)\right)$ time. Example 5.11 illustrates the different steps of the presented enhancement heuristic.

Example 5.11. Let $n = 10$, $m = 2$, $a = -0.1$, $M = 0.5$ and the processing times are as follows, $p_j = \{38, 37, 38, 79, 94, 37, 15, 65, 92, 50\}$.

The applied heuristic is the LPT rule. The obtained schedule is: on $M_1 : 7, 1, 10, 8, 5$ and on $M_2 : 2, 6, 3, 4, 9$ with $\text{UB} = 267.83$. Now, applying the enhancement phase as described above, conducts to the following schedule: on $M_1 : 7, 2, 6, 1, 3, 10, 8$ and on $M_2 : 4, 9, 5$ with $\text{UB}_1 = 261.70 < \text{UB} = 267.83$.

6. Experimental results

In this section, the performance of the proposed heuristics and lower bounds, for the studied problem, will be discussed. All the procedures were coded in C++ and simulations were carried out on PC core i7 − 3337U with 1.8 GHz and 8 GB RAM. The experiments are carried out over benchmark test problems. These test problems are detailed in the following subsection.

6.1. Test problems

The learning index a and the incompressibility factor M, which are the parameters of the DeJong's learning curve, are selected as in [17]. This means that $a \in \{-0.1, -0.322\}$ and $M \in \{0, 0.5\}$. The number of jobs $n \in \{10, 20, 50, 100, 150, 200, 300, 500, 1000, 1500\}$. The number of machines m depends on n as follows.

– Type 1: $n \in \{10, 20, 50\}$ and $m \in \{2, 3\}$.

TABLE 1. The performance of LB with respect to a and M.

	$a=-0.1$		$a=-0.322$	
	$M=0$	$M=0.5$	$M=0$	$M=0.5$
Perc	100%	100%	100%	100%
GAP_l	9.9%	3.8%	34.2%	6.7%

- Type 2: $n \in \{100\}$ and $m \in \{2,3,5\}$.
- Type 3: $n \in \{150,200,300,500,1000,1500\}$ and $m \in \{2,3,5,10\}$.

The processing times are generated according to the following probability distribution:

- Class 1: p_j is generated from the discrete uniform distribution on $[1,20]$.
- Class 2: p_j is generated from the discrete uniform distribution on $[1,100]$.
- Class 3: p_j is generated from the discrete uniform distribution on $[50,100]$.
- Class 4: p_j is generated from the normal distribution with mean 100 and standard deviation 20.

For each type, each class and each pair (a,M), 10 instances are generated. Therefore, the total generated instances are 5280. This data is derived from those proposed in [17] for class 1 and class 2. Classes 3 and 4 are added to widen the variability of the data and to have a more efficient analysis.

6.2. Procedures performances

In this section, the performance of the proposed procedures (lower bounds and heuristics) is assessed.

6.2.1. Lower bound performance

To the best of our knowledge, the best proposed lower bound in literature is LB_2 (see Sect. 4.1). In this subsection, a comparison study between the proposed lower bound LB and LB_2, will be conducted. In this study, the considered performance measures are:

- Perc: is the percentage of instances where $\mathrm{LB} > \mathrm{LB}_2$.
- GAP_l: is the average of $rg = \frac{\mathrm{LB}-\mathrm{LB}_2}{\mathrm{LB}} \times 100$ for the considered class of instances, where rg measures the relative distance between LB and LB_2.
- Time: is the average consumed time while computing LB (in seconds). If Time < 0.001 s in an entire column or line then this column or line is omitted.

First, an overall comparison is necessarily to show the performance of the proposed lower bounds comparing with those from literature review. Remarkably, the experimental results gives that for 100% of the instances $\mathrm{LB} > \mathrm{LB}_2$. In addition, the average consumed time running LB is less than $0.001s$. The dominance of LB over LB_2 is clearly exhibited with $\mathrm{GAP}_l = 13.7\%$. The latter presented values provide strong evidence of the performance of the proposed lower bound LB compared to the existing ones. In the sequel, a more detailed study about the behavior of LB with respect to some parameters is carried out. The first detailed study is presenting the performance of LB with regard to the learning effect parameters a and M. The results are displayed in Table 1.

In Table 1, we remark that the average gap (GAP_l) is decreasing as M increases. In contrast, GAP_l increases as $|a|$ increases. Moreover, the maximum average relative gap 34.2% is reached for $a=-0.322$ and $M=0$, where the learning effect is more important. For this learning effect values ($a=-0.322$ and $M=0$), LB is performing very well.

The variation of GAP_l according to the different classes is presented in Table 2.

Based on Table 2, one could observe that the GAP_l is more important for classes 3 and 4, where $\mathrm{GAP}_l = 17\%$. These two classes are characterized by large values of initial processing times. However, for the other classes (1

TABLE 2. The performance of LB with respect to the classes.

Class	Perc	GAP$_l$
1	100%	10.4%
2	100%	10.1%
3	100%	17.2%
4	100%	17.0%

TABLE 3. The performance of LB according to m.

m	GAP$_l$	Perc
2	8.3%	100%
3	12.1%	100%
5	16.8%	100%
10	23.5%	100%

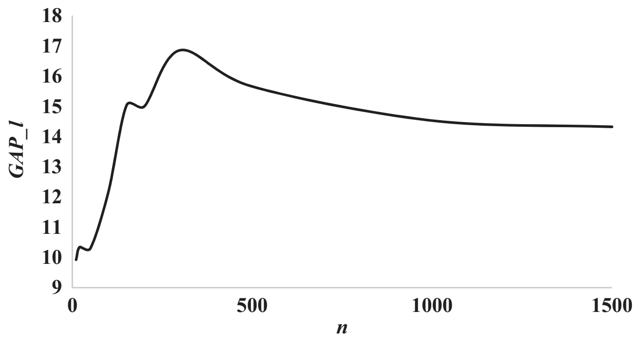

FIGURE 3. GAP$_l$ *versus* n for all instances.

and 2), where there are mixing between large and small initial processing times, the GAP$_l$ = 10%. A detailed assessment of the performance of LB is done according to the number of machines m. The results are displayed in the Table 3.

According to Table 3, we observe that the performance of the proposed LB (GAP$_l$) increases as the number of machines m increases. In addition, the minimum value GAP$_l$ = 8.3% is reached for $m = 2$, and the maximum value GAP$_l$ = 23.5% when $m = 10$. In Figure 3, the behavior of the GAP$_l$ *versus* the number of jobs n is exhibited.

According to Figure 3, we observe that the maximum value GAP$_l$ = 16.9% is reached for $n = 300$. Moreover, the GAP$_l$ value decreases toward 14% as the number of jobs n goes to infinity. More refined details about the GAP$_l$ behavior simultaneously according to n and m is presented in Table 4.

Based on Table 4, we observe that the maximum GAP$_l$ value (24%) is reached for $n = 200$ and $m = 10$. In addition, the minimum GAP$_l$ value (7.9%) for $n = 1500$ and $m = 2$.

6.2.2. *Heuristics performance analysis*

In this paper, ten heuristics have been proposed. These heuristics are assessed over the following performance measures:

Table 4. The detailed performance of LB according to n and m.

n	m	Perc	GAP$_l$
10	2	100%	8.4%
	3	100%	11.5%
20	2	100%	8.7%
	3	100%	12.0%
50	2	100%	8.6%
	3	100%	12.0%
100	2	100%	8.5%
	3	100%	12.0%
	5	100%	16.0%
150	2	100%	8.4%
	3	100%	11.9%
	5	100%	15.9%
	10	100%	23.9%
200	2	100%	8.3%
	3	100%	11.9%
	5	100%	15.9%
	10	100%	24.0%
300	2	100%	8.3%
	3	100%	11.8%
	5	100%	23.7%
	10	100%	23.7%
500	2	100%	8.1%
	3	100%	15.5%
	5	100%	15.5%
	10	100%	23.6%
1000	2	100%	8.0%
	3	100%	11.4%
	5	100%	15.5%
	10	100%	23.2%
1500	2	100%	7.9%
	3	100%	11.3%
	5	100%	15.1%
	10	100%	22.9%

Table 5. The performance of UB according a and M.

	$a = -0.1$		$a = -0.322$		All
	$M = 0$	$M = 0.5$	$M = 0$	$M = 0.5$	
GAP$_u$	3.6%	1.5%	13.7%	2.7%	5.4%
Time	0.186	0.243	0.188	0.187	0.201

– GAP$_u$: is the average of the relative gap *rgap* for a specified set of test problems. The expression of *rgap* for each test problem is given by $rgap = \frac{U-\text{LB}}{\text{LB}} \times 100$, where U is the upper bound to be evaluated and LB are the lower bound developed in the section "Lower bounds". The relative gap *rgap* is expressing the maximum relative error when the optimal solution is unknown.
– Time: the consumed running time U (in seconds). Denoted by $(-)$ when time is strictly less than 0.001 s.

Let UB denoting the minimum upper bound among the 10 proposed ones. A general overview describing the performance of UB is given in Table 5.

In Table 5, the average gap and the total time are categorized with respect to the learning effect parameters a and M. According to this table, the minimum average gap of 1.5% is reached for $a = -0.1$ and $M = 0.5$, whereas the mean running time is the highest one with a value of 0.243 s. On another hand, the maximum gap 13.7% is recorded while $a = -0.322$ and $M = 0$. Consequently, the overall average gap is 5.4% with a mean time of 0.201 s. In addition, we remark that the average gap increases when $|a|$ increases. Further, the mean time almost decreases with an increase of $|a|$. Furthermore, analysis according to the classes (initial processing times) reveals that the classes 1 and 2 present almost the maximum average gap 8.6%, whereas the two remaining classes (3 and 4) have an average gap of 2.4%. In contrast, the minimum mean time is reached for the classes 1 and 2 with a value of approximately 0.096 s. These results are retrieved from Table 6.

An analysis of the obtained results with respect to the number of the machines m, discloses that the average gap GAP$_u$ and the mean time Time increase as m increases. More precisely, for $m = 2$ the minimum are reached with the following values: GAP$_u$ = 1.1% and Time = 0.021 s. In addition, the maximums GAP$_u$ = 11.8% and Time = 0.74 s are reached for $m = 10$. These results are displayed in Table 7.

Table 6. The performance of UB with respect to the classes.

Class	GAP_u	Time
1	8.3%	0.067
2	8.6%	0.096
3	2.2%	0.180
4	2.4%	0.429

Table 7. The performance of UB with respect to the number of machines m.

m	GAP_u	Time
2	1.1%	0.021
3	4.1%	0.060
5	7.8%	0.193
10	11.8%	0.749

Table 8. The performance of UB with respect to the number of the jobs n.

n	GAP_u	Time
10	3.9%	0.006
20	2.5%	0.009
50	2.6%	0.011
100	4.4%	0.034
150	6.2%	0.163
200	6.2%	0.193
300	6.1%	0.158
500	6.1%	0.221
1000	6.0%	0.379
1500	6.0%	0.507

The effect of the number of jobs n on the performance of the proposed heuristics is presented over Table 8.

Seeking clarity, the presented results in Table 8 will be shown separately in Figures 4 and 5. According to Figure 4, the Time is almost increasing as n increases.

Moreover, Figure 5 reveals the existing of a maximum average gap 6.2% for $n = \{150, 200\}$ and a minimum average gap of 2.5% for $n = 20$. A more detailed analysis involving all the parameters (a, M, n, and m) is presented over Table 9. In Table 9, the impact of learning parameters a, M, the number of machines m and the number of jobs n are considered simultaneously. According to Table 9, the maximum gap is reached for $a = -0.322$, $M = 0$, $m \in \{5, 10\}$ and $n \geq 150$. In the sequel, a comparison of the 10 heuristics will be conducted. This comparison is done over the percentage of times the studied heuristic is equal to UB. These obtained results are exhibited in Table 10. Based on Table 10, we observe first that there is no dominance between the 10 heuristics. In addition, the RLPT reaches UB in 82% of instances. Surprisingly, the SPT based heuristic contributes only by 0.1% in UB. Further, the percentage of participation in UB is almost independent of the learning effect parameters a and M. The average running time of the heuristics is presented in Table 11. As expected, the *Opt* heuristic presents the highest consumed time with an average of 1.750 s. The average time for the RLPT is 0.127 s, which is an acceptable time. A more refined analysis is given in Table 12. The presented results are about the percent the studied heuristics participate *lonely* in the UB. We observe that The RLPT

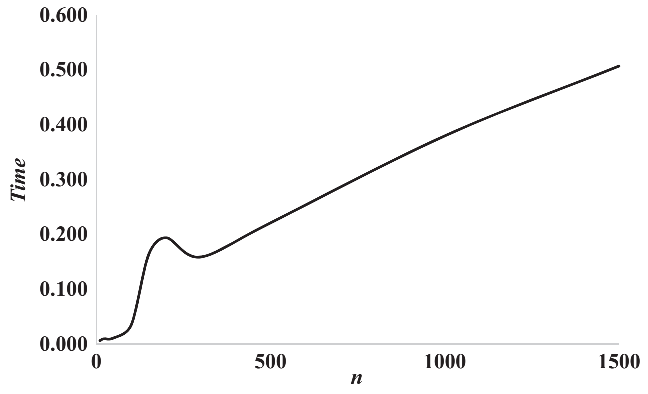

Figure 4. Behavior of Time *versus* n.

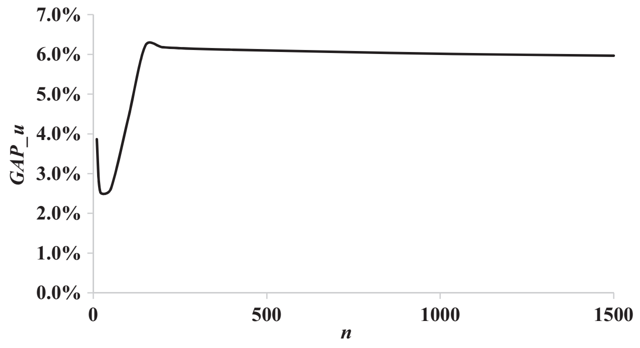

Figure 5. Behavior of GAP_u *versus* n.

still have the highest percentage with a value of 80.32%. Remarkably, some heuristics do not participate at the UB value lonely as The SPT.

7. Conclusion

In this paper, the parallel machines scheduling problem with DeJong's learning effect has been addressed. A set of heuristics have been developed. These heuristics belong to two main families. The first family is composed of a dispatching rules based heuristics with new enhancement methods. However, the second family is based on the exact resolution of NP-Hard problems. Intensive experimental study shows the dominance of the so-called Randomized LPT heuristic, despite its simplicity and low complexity. In addition, a set of lower bounds have been proposed. These lower bounds improve the existing ones and outperform them. The key enhancement is the estimation of the maximum position to be occupied by a job in an optimal schedule. These lower bounds are mathematically proved to be dominant compared with the existing ones. The experimental study support this claim and provides strong evidence of their efficiency. The developed and proposed lower bounds and heuristics could be utilized in future works that include the development of exact branch and bound method for the addressed problem. Furthermore, the presented procedures could be generalized to other scheduling problems as the uniform parallel machine scheduling problem with learning effect.

Table 9. The performance of UB with respect to a, M, n and m.

n	m	$a=-0.1$		$a=-0.322$		n	m	$a=-0.1$		$a=-0.322$	
		$M=0$	$M=0.5$	$M=0$	$M=0.5$			$M=0$	$M=0.5$	$M=0$	$M=0.5$
10	2	0.7%	0.4%	2.0%	0.7%	300	2	0.8%	0.3%	3.3%	0.4%
	3	5.8%	5.1%	9.7%	6.7%		3	2.6%	1.0%	9.5%	1.6%
20	2	0.7%	0.3%	2.1%	0.7%		5	4.9%	1.9%	20.9%	3.6%
	3	3.0%	1.7%	8.3%	3.5%		10	7.5%	3.0%	30.5%	6.3%
50	2	0.8%	0.3%	2.8%	0.6%	500	2	0.8%	0.3%	3.3%	0.4%
	3	2.9%	1.5%	9.2%	2.7%		3	2.7%	1.0%	9.6%	1.5%
100	2	0.8%	0.3%	3.0%	0.5%		5	5.0%	1.9%	21.1%	3.1%
	3	2.9%	1.4%	9.4%	2.5%		10	7.7%	2.9%	30.7%	5.5%
	5	4.9%	2.0%	20.0%	4.6%	1000	2	0.9%	0.3%	3.3%	0.3%
150	2	0.8%	0.3%	3.2%	0.5%		3	2.7%	1.0%	9.7%	1.2%
	3	2.6%	1.0%	9.4%	1.9%		5	5.0%	1.8%	21.3%	2.6%
	5	5.0%	2.0%	20.3%	4.2%		10	7.8%	2.8%	31.0%	4.6%
	10	7.7%	3.2%	30.2%	7.2%	1500	2	0.9%	0.3%	3.3%	0.3%
200	2	0.9%	0.3%	3.2%	0.5%		3	2.6%	0.9%	9.6%	1.1%
	3	2.7%	1.1%	9.6%	1.9%		5	5.1%	1.7%	21.3%	2.3%
	5	5.0%	2.0%	20.4%	3.9%		10	7.9%	2.8%	31.2%	4.2%
	10	7.6%	3.1%	29.7%	6.8%						

Table 10. Heuristics comparison based on percentage of participation in UB.

		LPT	SPT	RLPT	RSPT	OPT	MSPT	MLPT	MMR	KN	BLPT
$a=-0.1$	$M=0$	9.0%	0.2%	82.3%	7.5%	3.9%	0.2%	8.8%	4.2%	0.2%	0.3%
	$M=0.5$	8.3%	0.2%	84.0%	6.0%	4.6%	0.2%	8.3%	3.5%	0.2%	0.5%
$a=-0.322$	$M=0$	6.9%	0.0%	75.5%	15.8%	3.6%	0.1%	6.8%	2.9%	0.0%	0.2%
	$M=0.5$	7.0%	0.0%	86.1%	5.8%	3.2%	0.0%	7.0%	3.0%	0.0%	0.4%
Average		7.8%	0.1%	82.0%	8.8%	3.8%	0.1%	7.7%	3.4%	0.1%	0.3%

Table 11. Time heuristics comparison.

		LPT	SPT	RLPT	RSPT	OPT	MSPT	MLPT	MMR	KN	BLPT
$a=-0.1$	$M=0$	0.000	0.000	0.127	0.125	1.601	0.001	0.000	0.001	0.005	0.001
	$M=0.5$	0.000	0.000	0.127	0.125	2.168	0.001	0.000	0.001	0.004	0.001
$a=-0.322$	$M=0$	0.000	0.000	0.126	0.125	1.621	0.001	0.000	0.001	0.005	0.001
	$M=0.5$	0.000	0.000	0.126	0.125	1.612	0.001	0.000	0.001	0.006	0.001
Average		0.000	0.000	0.127	0.125	1.750	0.001	0.000	0.001	0.005	0.001

Table 12. Lonely percentage participation of the heuristics in UB.

		LPT	SPT	RLPT	RSPT	OPT	MSPT	MLPT	MMR	KN	BLPT
$a=-0.1$	$M=0$	0.2%	0.0%	80.5%	6.2%	3.0%	0.0%	0.0%	0.0%	0.0%	0.2%
	$M=0.5$	0.2%	0.0%	82.0%	4.5%	3.0%	0.0%	0.1%	0.0%	0.0%	0.3%
$a=-0.322$	$M=0$	0.2%	0.0%	73.9%	14.4%	3.3%	0.0%	0.0%	0.0%	0.0%	0.2%
	$M=0.5$	0.1%	0.0%	84.8%	4.2%	2.3%	0.0%	0.1%	0.0%	0.0%	0.4%
Average		0.13%	0.00%	80.32%	7.35%	2.88%	0.00%	0.04%	0.00%	0.00%	0.25%

Acknowledgements. The authors gratefully acknowledge the technical and financial support of Research Center of College of Engineering, Deanship of Scientific Research, King Saud University.

References

[1] M.J. Anzanello and F.S. Fogliatto, Learning curve models and applications: literature review and research directions. *Int. J. Ind. Ergon.* **41** (2011) 573–583.

[2] L. Argote, Organizational Learning: Creating, Retaining and Transferring Knowledge. Springer Science & Business Media (2012).

[3] R.G. Askin and J.B. Goldberg, Design and Analysis of Lean Production Systems. John Wiley & Sons (2007).

[4] A.B. Badiru, Computational survey of univariate and multivariate learning curve models. *IEEE Trans. Eng. Manage.* **39** (1992) 176–188.

[5] D. Biskup, A state-of-the-art review on scheduling with learning effects. *Eur. J. Oper. Res.* **188** (2008) 315–329.

[6] J. Carlier, F. Clautiaux and A. Moukrim, New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Comput. Oper. Res.* **34** (2007) 2223–2250.

[7] M. Drozdowski, Back to scheduling models. In: Scheduling for Parallel Processing. Springer (2009) 367–377.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Vol. 5 of *Annals of Discrete Mathematics.* Elsevier (1979) 287–326.

[9] M. Haouari and M. Jemmali, Tight bounds for the identical parallel machine-scheduling problem: part ii. *Int. Trans. Oper. Res.* **15** (2008) 19–34.

[10] X. Huang, M.Z. Wang and P. Ji, Parallel machines scheduling with deteriorating and learning effects. *Optim. Lett.* **8** (2014) 493–500.

[11] M. Ji, D. Yao, Q. Yang and T. Cheng, Machine scheduling with DeJong's learning effect. *Comput. Ind. Eng.* **80** (2015) 195–200.

[12] M. Ji, X. Tang, X. Zhang and T.E. Cheng, Machine scheduling with deteriorating jobs and DeJong's learning effect. *Comput. Ind. Eng.* **91** (2016) 42–47.

[13] Y.K. Lin, Scheduling identical jobs on uniform parallel machines under position-based learning effects. *Arab. J. Sci. Eng.* **39** (2014) 6567–6574.

[14] Y.Y. Lu, J. Jin, P. Ji and J.B. Wang, Resource-dependent scheduling with deteriorating jobs and learning effects on unrelated parallel machine. *Neural Comput. Appl.* **27** (2016) 1993–2000.

[15] K. Luo, A scheduling model with a more general function of learning effects. *Comput. Ind. Eng.* **82** (2015) 159–166.

[16] L.P. Michael, Scheduling: Theory, Algorithms, and Systems. Springer (2018)

[17] D. Okołowski and S. Gawiejnowicz, Exact and heuristic algorithms for parallel-machine scheduling with DeJong's learning effect. *Comput. Ind. Eng.* **59** (2010) 272–279.

[18] S. Pakzad-Moghaddam, A Lévy flight embedded particle swarm optimization for multi-objective parallel-machine scheduling with learning and adapting considerations. *Comput. Ind. Eng.* **91** (2016) 109–128.

[19] M. Rostami, A.E. Pilerood and M.M. Mazdeh, Multi-objective parallel machine scheduling problem with job deterioration and learning effect under fuzzy environment. *Comput. Ind. Eng.* **85** (2015) 206–215.

[20] R. Rudek, Scheduling on parallel processors with varying processing times. *Comput. Oper. Res.* **81** (2017) 90–101.

[21] D. Shabtay and G. Steiner, A survey of scheduling with controllable processing times. *Disc. Appl. Math.* **155** (2007) 1643–1666.

[22] C. Teplitz, The Learning Curve Deskbook: A Reference Guide to Theory, Calculations and Applications. Quorum Books, Westport, CT (1991).

[23] X.Y. Wang and J.J. Wang, Scheduling deteriorating jobs with a learning effect on unrelated parallel machines. *Appl. Math. Model.* **38** (2014) 5231–5238.

[24] C. Wang, C. Liu, Z.H. Zhang and L. Zheng, Minimizing the total completion time for parallel machine scheduling with job splitting and learning. *Comput. Ind. Eng.* **97** (2016) 170–182.

[25] W.C. Yeh, P.J. Lai, W.C. Lee and M.C. Chuang, Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Inf. Sci.* **269** (2014) 142–158.