Omega 38 (2010) 3 – 11

Contents lists available at ScienceDirect

Omega

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

A single-machine learning effect scheduling problem with release times ☆

Wen-Chiung Lee, Chin-Chia Wu*, Peng-Hsiang Hsu

Department of Statistics, Feng Chia University, Taichung, Taiwan, ROC

ARTICLE INFO

Article history:
Received 29 February 2008
Accepted 16 January 2009
Available online 30 January 2009

Keywords:
Single machine
Learning effect
Makespan
Release time

ABSTRACT

In this paper, we investigate a single-machine problem with the learning effect and release times where the objective is to minimize the makespan. A branch-and-bound algorithm incorporating with several dominance properties and lower bounds is developed to derive the optimal solution. A heuristic algorithm is proposed to obtain a near-optimal solution. The computational experiments show that the branch-and-bound algorithm can solve instances up to 36 jobs, and the average error percentage of the proposed heuristic is less than 0.11%.

1. Introduction

In classical scheduling problems, job processing times are assumed to be fixed and known. However, Heizer and Render [1] and Russell and Taylor [2] demonstrated through empirical studies that unit costs decline as firms produce more of a product and gain knowledge or experience in several kinds of industry. Wang and Lee [3] addressed learning by means of overall equipment effectiveness to increase the productivity of plant and equipment through company-led small group activities and autonomous maintenance by operators. Biskup [4] also pointed out that repeated processing of similar tasks improves workers' skills, e.g., workers are able to perform setups, deal with machine operations or software, or handle raw materials and components at a faster pace.

Scheduling with learning effect has received considerable attention recently. Gawiejnowicz [5], Donheti and Mohanty [6], Biskup [4] and Cheng and Wang [7] are among the pioneers that brought the learning effect into the scheduling field. Biskup [4] introduced the job-position-based learning effect model in which the actual job processing time is a decreasing function of its position in schedule. He showed that the single-machine problems to minimize total deviations of job completion times from a common due date and to minimize the sum of job completion times are polynomially solvable. Since then, many researchers have devoted attention on the relatively young but vivid area. For example, Mosheiov [8] gave several examples to demonstrate that the optimal schedules of some problems with learning effect may be different from those of the classical ones without learning consideration. Lee et al. [9] considered a bi-criteria single-machine scheduling problem to minimize the sum of the total completion time and the maximum tardiness, while Lee and Wu [10] considered a two-machine flowshop problem to minimize the total completion time. Wang and Xia [11] provided the worst-case bound for the shortest processing time (SPT) rule on the flowshop problems to minimize the makespan and the total completion time. They also showed that two special cases of the problems remain polynomially solvable. Chen et al. [12] considered a two-machine flowshop scheduling problem where the objective is to minimize a weighted sum of the total completion time and the maximum tardiness. Wang [13] considered a situation where both the phenomenon of learning and deterioration are present. He showed some single-machine problems are solvable in polynomial time. Recently, Eren and Güner [14] applied the 0–1 integer programming approach to derive the optimal solution, and used the random search, the tabu search, and the simulated annealing methods to obtain near-optimal solutions for the total tardiness problem. Wu and Lee [15] considered a multiple machine flowshop problem to minimize the total completion time.

On the other hand, Koulamas and Kyparisis [16] expressed the learning effect as a function of the normal processing times of jobs already processed. They showed that the SPT sequence is optimal for the single-machine makespan and the total completion time problems. In addition, they also proved that the two-machine flowshop makespan and the total completion time problems are optimally solved by the SPT rule when the job processing times are ordered. Besides, Wang [17] considered some single-machine problems with the effects of learning and deterioration, and proved that the makespan and sum of completion times (square) minimization problems remain polynomially solvable. They also showed that the weighted shortest processing time (WSPT) first rule and the earliest

☆This manuscript was processed by Associate Editor Jozefowska.

* Corresponding author.
E-mail address: cchwu@fcu.edu.tw (C.-C. Wu).

0305-0483/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2009.01.001

due date (EDD) first rule provide the optimal sequence for the weighted sum of completion times and the maximum lateness in some special cases. Janiak and Rudek [18] proposed an experience-based learning effect. They proved that the problem to minimize the makespan under Bachman and Janiak [19] model remains polynomially solvable when the experience-based approach is applied, but the same problem under the more general Cheng and Wang [7] model becomes strongly NP-hard in the presence of the new learning effect. Janiak and Rudek [20] introduced a new model of learning into the scheduling field that relaxes one of the rigid constraints by assuming that each job provides a different experience to the processor. They formulated the shape of the learning curve as a non-increasing k-stepwise function. Furthermore, they proved that the makespan problem is polynomially solvable if every job provides the same experience to the processor, and it becomes NP-hard if the experiences are different. Janiak et al. [21] investigated the single-machine makespan problem with a new learning model, where job processing times are described by S-shaped functions. They showed that the problem is strongly NP-hard even with the time-based learning effect. They also provided branch-and-bound and heuristic algorithms for the general problem.

Recently, Biskup [22] provided a comprehensive survey of scheduling problems with learning effects. In particular, he emphasized that as there is a significant involvement of humans in scheduling environments, the amount of learning activities is high. Hence it seems to be reasonable to consider learning in scheduling environments. He further clarified some economic fundamentals of scheduling and learning. To the best of our knowledge, most of the research assumes that jobs are available at all times. Bachman and Janiak [19] considered the learning effect scheduling problem with release times. They proved the makespan problem is NP-hard in the strong sense for two different learning models. However, no reports on developing the exact solution or the approximate solutions are found. In this paper, we provide the exact solution and near-optimal solutions for the single-machine makespan problem with release times.

The rest of the paper is organized as follows. In the next section, the problem formulation and the description of notation are given. In Section 3, some dominance properties and two lower bounds are developed to enhance the search efficiency for the optimal solution, followed by descriptions of the heuristic and the branch-and-bound algorithms. The results of a computational experiment are given in Section 4. Conclusions are given in the last section.

2. Notation and problem formulation

In this section, we first formulate the problem and then introduce some notation used throughout this paper.

There are n jobs to be scheduled. Each job i has a normal processing time p_i, and a release time r_i. The actual processing time of job i is $p_{iv} = p_i v^a$ if it is scheduled in the vth position where $a < 0$ is a learning ratio common for all jobs. The objective of this paper is to find an optimal schedule S^* such that

$$\max\{C_1(S^*), C_2(S^*), \ldots, C_n(S^*)\} \leqslant \max\{C_1(S), C_2(S), \ldots, C_n(S)\} \quad (1)$$

for any schedule S.

The notation is summarized as follows.

n	number of jobs.
S, S'	sequences of jobs.
p_i	normal processing time of job i.
r_i	release time of job i.
$C_i(S)$	completion time of job i in S.
a	learning ratio, $a < 0$.
$p_{[i]}(S)$	normal processing time for the job scheduled in the i th position in S.
$r_{[i]}(S)$	release time of a job scheduled in the i th position in S.
$C_{[i]}(S)$	completion time of a job scheduled in the i th position in S.
$p_{(i)}$	the i th job processing time when they are in a non-decreasing order.
$r_{(i)}$	the i th job release time when they are in a non-decreasing order.

3. Algorithms

Bachman and Janiak [19] showed that the problem under consideration is NP-hard in the strong sense. Thus, we will apply the branch-and-bound technique to search for the optimal solution in this paper. In order to facilitate the searching process, we first develop several dominance properties to fathom the searching tree, followed by some adjacent pairwise interchange properties. In addition, two lower bounds are also provided in the next subsection, and the procedures of the heuristic and branch-and-bound algorithms are described at last.

3.1. Dominance properties

First, we provide a result that can reduce the size of solution space of the problem under certain condition.

Property 1. For a given schedule S, if there is a k such that $C_{[k]}(S) \leqslant \min_{k+1 \leqslant j \leqslant n} r_{[j]}(S)$, then there exists an optimal solution in which the order of the first k jobs is the same as that of S.

Proof. We prove the theorem by claiming that moving a job in the first k positions to a later place does not reduce the makespan. Let $S = (J_{[1]}, \ldots, J_{[i]}, \ldots, J_{[j]}, \ldots, J_{[n]})$. Assume that S' is derived from S by moving the job in the i th $(i < k)$ position to the j th position. That is, $S' = (J_{[1]}, \ldots, J_{[i-1]}, J_{[i+1]}, \ldots, J_{[j]}, J_{[i]}, J_{[j+1]}, \ldots, J_{[n]})$. To compare S and S', we consider two cases.

The first case is $j \leqslant k$. Assume that it yields $C_{[k]}(S') \leqslant C_{[k]}(S)$ after the movement. Note that this is a more restrictive case. Since $C_{[k]}(S) \leqslant \min_{k+1 \leqslant j \leqslant n} r_{[j]}(S)$, it implies that the $(k+1)$th job cannot be started until $r_{[k+1]}(S)$ in both sequences. Thus, $C_{[n]}(S) = C_{[n]}(S')$ in this case since jobs after the k th position are processed in the same order in both sequences.

The second case is $j > k$. Since $C_{[k]}(S) \leqslant \min_{k+1 \leqslant j \leqslant n} r_{[j]}(S)$, it implies that $C_{[l]}(S') = C_{[l]}(S)$ for $k \leqslant l \leqslant j$. Since $J_{[i]}$ is processed after $J_{[j]}$ in S', it implies that $C_{[l]}(S') \geqslant C_{[l]}(S)$ for $j+1 \leqslant l \leqslant n$. In particular, $C_{[n]}(S') \geqslant C_{[n]}(S)$. This completes the proof. □

Corollary 1. *If $C_{[n-1]}(S) \leqslant r_{[n]}(S)$, then S is an optimal schedule.*

The corollary is a straightforward result from Property 1 when $k = n-1$. Nevertheless, it is useful to determine the optimality of a given schedule.

Next, we present some properties to determine the ordering of the remaining unscheduled jobs to further speed up the searching process. Assume that $S = (\pi, \pi^c)$ is a sequence of jobs where π is the scheduled part containing k jobs and π^c is the unscheduled part. Let $S_1 = (\pi, \pi')$ be the sequence in which the unscheduled jobs are arranged in the non-decreasing orders of job processing times, i.e., $p_{(k+1)} \leqslant p_{(k+2)} \leqslant \cdots \leqslant p_{(n)}$. In addition, let $S_2 = (\pi, \pi'')$ be the sequence in which the unscheduled jobs are arranged in the non-decreasing orders of job release times, i.e., $r_{(k+1)} \leqslant r_{(k+2)} \leqslant \cdots \leqslant r_{(n)}$.

Property 2. If there exists job $j \in \pi^c$ such that $\max\{C_{[k-1]}(S), r_j\} + p_j k^a < r_{[k]}$, then $S = (\pi, \pi^c)$ is a dominated sequence.

Proof. S is a dominated sequence since job j can move to a forward position without changing the completion time of the last job in π. □

Property 3. If $C_{[k]}(S_1) > \max_{j\in\pi^c}\{r_j\}$, then $S_1 = (\pi, \pi')$ dominates sequences of the type (π, π^c) for any unscheduled sequence π^c.

Proof. Since $C_{[k]}(S_1) > \max_{j\in\pi^c}\{r_j\}$, it implies that all the unscheduled jobs are ready to be processed on time $C_{[k]}(S_1)$. Thus, the SPT rule yields the optimal subsequence [8]. The next property follows from Property 1. □

Property 4. If $C_{[n-1]}(S_2) < r_{[n]}(S_2)$, then $S_2 = (\pi, \pi'')$ dominates sequences of the type (π, π^c) for any unscheduled sequence π^c.

Next, we will develop several dominance properties based on a pairwise interchange of two adjacent jobs. Assume that $S=(\pi,J_i,J_j,\pi')$ and $S'=(\pi,J_j,J_i,\pi')$ where π and π' are partial sequences. To show that S dominates S', it suffices to show that $C_j(S) - C_i(S') < 0$. In addition, let A be the completion time of the last job in the subsequence π.

Property 5. If $r_j > \max\{A, r_i\} + p_i k^a$, then S dominates S'.

Proof. From $r_j > \max\{A, r_i\} + p_i k^a$, we have

$$C_j(S) = r_j + p_j(k+1)^a. \tag{2}$$

Since $r_j > A$, the completion time for job j in S' is

$$C_j(S') = r_j + p_j k^a. \tag{3}$$

From $r_j > r_i$ and $p_j > 0$, the completion time for job i in S' is

$$C_i(S') = r_j + p_j k^a + p_i(k+1)^a. \tag{4}$$

Thus, the difference between the completion times of job j in S and job i in S' is

$$C_i(S') - C_j(S) = p_j[k^a - (k+1)^a] + p_i(k+1)^a > 0 \tag{5}$$

since $a < 0$. Therefore, S dominates S'.

The proofs of Properties 6–10 are omitted since they are similar to that of Property 5. □

Property 6. If $A > \max\{r_i, r_j\}$ and $p_i < p_j$, then S dominates S'.

Property 7. If $r_j > A > \max\{r_i, r_j - p_i k^a\}$ and $p_i < p_j$, then S dominates S'.

Property 8. If $r_i > A > \min\{r_j, r_i - p_j k^a\}$, and $r_i - A + [k^a - (k+1)^a](p_i - p_j) < 0$, then S dominates S'.

Property 9. If $r_j > r_i > A$, $r_i + p_i k^a > r_j$ and $p_i < p_j$, then S dominates S'.

Property 10. If $r_i > r_j > A$, $r_j + p_j k^a > r_i$ and $r_i - r_j + [k^a - (k+1)^a](p_i - p_j) < 0$, then S dominates S'.

3.2. Lower bounds

In this subsection, we will develop two lower bounds to curtail the branching tree. Assume that PS is a partial schedule in which the order of the first k jobs has been determined and S is a complete schedule obtained from PS. By definition, the completion time for the $(k+1)$th job is

$$\begin{aligned} C_{[k+1]}(S) &= \max\{C_{[k]}(S), r_{[k+1]}\} + p_{[k+1]}(k+1)^a \\ &\geqslant C_{[k]}(S) + p_{[k+1]}(k+1)^a. \end{aligned} \tag{6}$$

Similarly, the completion time for the $(k+l)$th job is

$$C_{[k+l]}(S) \geqslant C_{[k]}(S) + \sum_{j=1}^{l} p_{[k+j]}(k+j)^a. \tag{7}$$

Therefore, the makespan for S is

$$C_{[n]}(S) \geqslant C_{[k]}(S) + \sum_{j=1}^{n-k} p_{[k+j]}(k+j)^a. \tag{8}$$

It is observed that the first term on the right-hand side of Eq. (8) is known, and a lower bound of the makespan for the partial sequence PS can be obtained by minimizing the second term. Since the value of $(k+j)^a$ is a decreasing function of j, the makespan is minimized by sequencing the unscheduled jobs according to the SPT rule. Consequently, the first lower bound is

$$LB_1 = C_{[k]}(S) + \sum_{j=1}^{n-k} p_{(k+j)}(k+j)^a, \tag{9}$$

where $p_{(k+1)} \leqslant p_{(k+2)} \leqslant \cdots \leqslant p_{(n)}$. On the other hand, this lower bound may not be tight if the release times are large. To overcome this situation, a second lower bound is established by taking account of the release time. The completion time for the $(k+1)$th job is

$$\begin{aligned} C_{[k+1]}(S) &= \max\{C_{[k]}(S), r_{[k+1]}\} + p_{[k+1]}(k+1)^a \\ &\geqslant r_{[k+1]}(S) + p_{[k+1]}(k+1)^a. \end{aligned} \tag{10}$$

Thus, the completion time for the last job in S is

$$C_{[n]}(S) \geqslant r_{[k+1]}(S) + \sum_{j=1}^{n-k} p_{[k+j]}(k+j)^a. \tag{11}$$

By induction, it is derived that

$$C_{[n]}(S) \geqslant \max_{1\leqslant l\leqslant j}\{r_{[k+l]}(S)\} + \sum_{j=l}^{n-k} p_{[k+j]}(k+j)^a, \tag{12}$$

where $1 \leqslant j \leqslant n-k$. Note that since the first term on the right-hand side of Eq. (12) is the maximal release time of the first j unscheduled jobs, it is greater than or equal to $r^*_{(k+j)}$, where $r^*_{(k+1)} \leqslant r^*_{(k+2)} \leqslant \cdots \leqslant r^*_{(n)}$ denote the release times of the unscheduled jobs arranged in a non-decreasing order. The second term is minimized by the SPT rule since $(k+j)^a$ is a decreasing function of j. Thus, it is derived from equation that the second lower bound is

$$LB_2 = \max_{1\leqslant j\leqslant n-k}\left\{ r^*_{(r+j)} + \sum_{l=1}^{n-k+1-j} p^*_{(k+l)}(k+j+l-1)^a \right\}, \tag{13}$$

where $p^*_{(k+1)} \leqslant p^*_{(k+2)} \leqslant \cdots \leqslant p^*_{(n)}$ denote the processing times of the unscheduled jobs arranged in a non-decreasing order. Note that $p^*_{(k+j)}$ and $r^*_{(k+j)}$ do not necessarily come from the same job. In order to make the lower bound tighter, we choose the maximum value from Eqs. (9) and (13) as the lower bound of PS. That is,

$$LB^* = \max\{LB_1, LB_2\}. \tag{14}$$

3.3. The heuristic algorithm

A typical approach to an NP-hard problem is to provide a heuristic algorithm. French [23] pointed out that apart from the choice of a search strategy and lower bounds, there are other ways in which we may attempt to increase the speed of finding an optimal solution. Firstly, we may prime the procedure with a near-optimal schedule

as the first trial. The better the first trial the more nodes we may expect to be eliminated in the early stages. Therefore, assuming that we can find a near optimal schedule, we may save ourselves much computation. Such solution can be provided by heuristic methods. In this section, a three-stage heuristic algorithm is proposed. The main idea of the first stage is to arrange jobs in earlier positions if the jobs have smaller values of the release times and processing times with the learning consideration. However, the schedule might have many idle periods, which in turn yields a large value of the makespan. To overcome this problem, NEH algorithm [24] is utilized to improve the quality of the solution in Stage II. That is, it selects the k th job from the solution obtained in the first stage, and inserts it in k possible positions. A local search (pairwise interchange) is inserted in the third stage to further improve the quality of the solution. The steps of the proposed algorithm are described as follows.

Stage I:

Set $J=\{J_1,J_2,\ldots,J_n\}$;
For $l \leftarrow 1$ **to** n **do**
  Choose job J_i with $\min\{r_i+p_i l^a\}$ from J and place it on the l th position.
  Delete job J_i from J.
end
Output $S=(J_{[1]},J_{[2]},\ldots,J_{[n]})$;

Stage II:

Set $S^0=(J_{[1]},-,\ldots,-)$ where the first job is from S;
For $l \leftarrow 2$ **to** n **do**
  Select the l th job from S and insert it in l possible positions in the current partial sequence S^0.
  Among the l partial sequences, select the one with a minimum $C_{[l]}$ as the current partial sequence S^0.
end
Output S^0;

Stage III:

For $k \leftarrow 1$ **to** $n-1$ **do**
  For $i \leftarrow k+1$ **to** n **do**
    Create a new sequence S^1 by interchanging jobs in the i th and the k th positions from S^0.
    Replace S^0 by S^1 if the value of the makespan of S^1 is less than that of S^0;
  end
end
Output S^0;

The complexities of the three stages in the proposed heuristic algorithm are $O(n^2)$, $O(n^3)$ and $O(n^2)$, respectively. Therefore, the complexity for the proposed heuristic algorithm is $O(n^3)$.

3.4. The branch-and-bound algorithm

The depth-first search is adopted in the branching procedure. This search strategy has the advantage that it only needs to store the lower bounds for at most $n-1$ active nodes through the branch procedure. This algorithm assigns jobs in a forward manner starting from the first position. In the searching tree, we choose a branch and systematically work down the tree until we either eliminate it by virtue of the dominance properties, the lower bounds or reach its final node, in which case this sequence either replace the initial solution or is eliminated. The steps of the branch-and-bound algorithm are described as follows.

Step 1. {Initialization} Perform the proposed heuristic algorithm to obtain a sequence as the initial solution.

Step 2. {Reduction} First apply Corollary 1 to determine whether the initial solution is optimal. If not, utilize Property 1 to reduce the number of jobs.

Step 3. {Branching} Use the depth-first search for the reduced problem.

For each node **do**
Apply Property 2 and Properties 5–10 to eliminate the dominated partial sequences.
For the non-dominated nodes, utilize Properties 3 and 4 to attempt to determine the order of the unscheduled jobs.
Calculate the lower bound of the makespan for the unscheduled partial sequences or the makespan for the completed sequences.
If the lower bound for the unscheduled partial sequence is greater than the initial solution, eliminate that node and all nodes beyond it in the branch.
If the value of the completed sequence is less than the initial solution, replace it as the new solution. Otherwise, eliminate it.
end
Return the solution

4. Computational experiment

A computational experiment was conducted to evaluate the efficiency of the branch-and-bound algorithm, and the accuracy of the heuristic algorithm. The algorithms were coded in Fortran and run on Compaq Visual Fortran version 6.6 on a 3.4 GHz Pentium 4 CPU with 1 GB RAM on Windows XP. The experimental design followed Chu's [25] framework. The job processing times were generated from a uniform distribution over the integers between 1 and 100. The release times were generated from a uniform distribution over the integers on $(0, 50.5n\lambda)$ where n is the number of job and λ is a control variable.

For the branch-and-bound algorithm, the average and the maximum numbers of nodes as well as the average and the maximum execution times (in seconds) were recorded. The heuristic with only the first stage was denoted as Phase I, the heuristic with the first two stages was denoted as Phase II, and the heuristic with all the stages was denoted as Phase III. For the heuristic algorithms, the mean and the maximum error percentages were recorded, where the error percentage was calculated as

$$(C_{\max}-C^*_{\max})/C^*_{\max}\times 100\%,$$

where $C_{\max}$ is the makespan obtained from the heuristic algorithm and $C^*_{\max}$ is the makespan of the optimal schedule.

The computational experiment was divided into four parts. In the first part of the experiment, the number of jobs was fixed at 10. The branch-and-bound algorithm with neither lower bounds nor dominance properties was used as the base, and the dominance properties were included one each time. The results were given in Table 1. It can be seen that all the properties are useful in cutting the search tree, especially Properties 1–6. To further analyze the efficiency of Properties 7–10, we conducted another computational experiment with one property (Properties 7–10) excluded each time. It was observed from Table 2 that the branch-and-bound algorithm with all properties included is the most efficient in term of the execution time. Moreover, the impact of particular phases of the proposed heuristic on the quality of the final solution was presented in Table 3. It can be seen that without neither lower bounds nor dominance properties, the impact of the initial solutions is limited.

In the second part of the experiment, the number of job was fixed at 20 and the learning effect a took the values of 70%, 80%, and 90% in order to study the impact of the control variable λ. The control variable λ took the values from 0.2 to 3.0, with a jump of

Table 1
The performance of the branch and bound algorithm with $n = 10$.

Property included	a	λ	Branch-and-bound algorithm				Rate
			CPU time		Node		
			Mean	Max	Mean	Max	
None	70	0.20	4.8341	10.7188	439 854	960 624	1.000000
Only Property 1			4.8925	12.1719	426 731	960 624	0.970165
Only Property 2			3.7916	12.4844	279 517	857 404	0.635477
Only Property 3			0.0044	0.0781	170	4441	0.000386
Only Property 4			7.5672	16.0312	439 854	960 624	1.000000
Only Property 5			4.1269	10.1094	364 884	903 032	0.829557
Only Property 6			0.1256	0.2969	12 477	29 621	0.028366
Only Property 7			4.7027	10.6406	415 885	909 148	0.945507
Only Property 8			4.7358	10.5156	430 276	960 624	0.978225
Only Property 9			4.7373	10.0156	407 842	871 375	0.927221
OnlyProperty 10			4.5664	9.4219	416 616	871 791	0.947169
None		1.00	1.4356	6.5625	155 321	984 152	1.000000
Only Property 1			0.0363	1.6094	3175	131 009	0.020442
Only Property 2			0.0356	1.3281	2774	97 103	0.017860
Only Property 3			1.0294	7.3594	96 335	983 149	0.620232
Only Property 4			1.0300	6.1875	58 176	342 975	0.374553
Only Property 5			0.2933	1.2188	30 244	138 768	0.194719
Only Property 6			0.2737	0.7500	27 787	81 217	0.178900
Only Property 7			1.3712	6.2500	147 122	954 015	0.947213
Only Property 8			1.4433	6.3438	153 659	970 758	0.989300
Only Property 9			1.3778	6.3125	147 147	970 452	0.947374
Only Property 10			1.4595	6.3750	152 787	984 152	0.983685
None	90	0.20	20.1514	27.5625	1 841 588	2 539 133	1.000000
Only Property 1			19.4723	29.8594	1 699 918	2 539 133	0.923072
Only Property 2			9.7431	28.9688	786 352	2 294 059	0.426997
Only Property 3			0.0016	0.0312	45	1075	0.000024
Only Property 4			30.5973	45.8281	1 841 588	2 539 133	1.000000
Only Property 5			17.3422	29.4219	1 517 547	2 402 410	0.824043
Only Property 6			0.1381	0.5000	13 751	47 475	0.007467
Only Property 7			19.8456	28.0469	1 769 083	2 539 133	0.960629
Only Property 8			20.6541	29.1250	1 830 974	2539133	0.994236
only Property 9			17.9703	23.9531	1629285	2 182 066	0.884717
Only Property 10			19.9439	27.8125	1 811 076	2 539 133	0.983432
None			1.8753	11.5781	172 380	1 049 682	1.000000
Only Property 1			0.6562	11.9219	56 160	1 049 682	0.325792
Only Property 2			0.2125	4.8125	18 024	395 526	0.104560
Only Property 3			0.3687	3.6250	26 777	372 757	0.155337
Only Property 4			2.7559	20.2031	149 845	1 049 682	0.869271
Only Property 5			0.8531	8.5312	73 750	734 800	0.427834
Only Property 6			0.3361	1.1250	30 543	103 199	0.177184
Only Property 7			1.7344	10.3750	156 884	933 736	0.910106
Only Property 8			1.9178	12.1406	171 337	1 045 935	0.993949
Only Property 9			1.7963	10.9844	157 260	955 806	0.912287
Only Property 10			1.9197	12.5000	171 185	1 049 682	0.993068

Table 2
The performance of the branch-and-bound algorithm with $n = 20$.

Property excluded	a	λ	Branch-and-bound algorithm			
			CPU time		Node	
			Mean	Max	Mean	Max
None	70	0.20	0.0505	0.2344	186	1178
Property 7			0.0591	0.2969	195	1194
Property 8			0.0538	0.2969	196	1461
Property 9			0.0522	0.2500	187	1178
Property 10			0.0531	0.2500	190	1268

0.05 each time. For each condition, 1000 instances were randomly generated. The results were presented in Figs. 1 and 2. When the learning effect is fixed, it can be seen in Fig. 1 that the average number of nodes in the branch-and-bound algorithm increases first and then decrease as λ increases. When the value of λ is small, it implies that all the job release times are small and Property 3 is useful in that case since it is relatively easy for the job completion time to surpass the maximal release time of the unscheduled jobs after some jobs have been scheduled. This explains the increment of the number of nodes for small values of λ. However, as the value of λ

Table 3
The performance of the branch and bound algorithm with $n = 10$.

Initial solution	a	λ	Branch-and-bound			
			CPU time		Node	
			Mean	Max	Mean	Max
Phase I	70	0.20	5.0422	11.0625	456 487	1 012 725
		1.00	1.4525	6.3906	155 686	984 152
	90	0.20	20.5477	27.9688	1 853 813	2 539 139
		1.00	1.9431	12.8281	173 243	1 069 982
Phase II	70	0.20	4.8531	10.4688	440 449	960 624
		1.00	1.4781	6.5469	155 321	984 152
	90	0.20	20.5716	27.9531	1 843 016	2 539 135
		1.00	1.9041	11.7812	172 386	1 049 682
Phase III	70	0.20	4.8341	10.7188	439 854	960 624
		1.00	1.4356	6.5625	155321	984 152
	90	0.20	20.1514	27.5625	1 841 588	2 539 133
		1.00	1.8753	11.5781	172 380	1 049 682

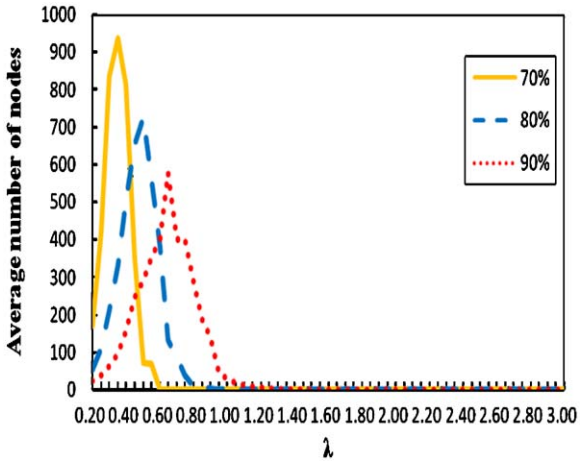

Fig. 1. The performance of the branch-and-bound algorithm with respect to λ.

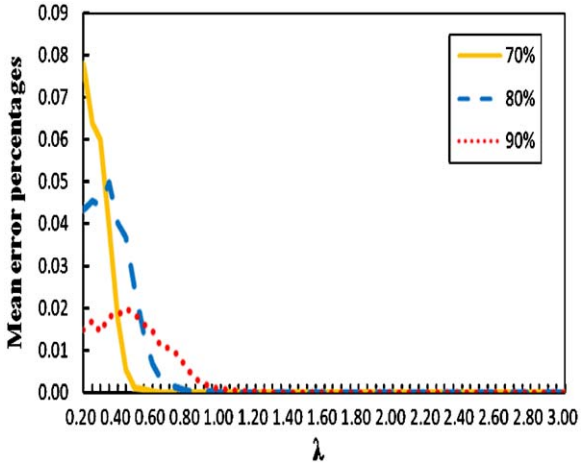

Fig. 2. The performance of Phase III algorithm with respect to λ.

becomes large, it implies that the range of the job release times is large, thus Property 1 and Corollary 1 are useful in that case. This explains the decreasing trend of the number of nodes as the value of λ becomes large. The same phenomenon was also observed in Fig. 2. The mean error percentages decreased to zero as the value of λ became large. This is due to the fact that only the ordering of the last few jobs will affect the value of the makespan if some of the job release times are relatively large. Moreover, a further experiment was conducted to evaluate the performance of the heuristic algorithms. The job size was fixed at 20 and the learning effect a took the value of 90%. The performance of Phase I was very poor, as shown in Table 4. Thus, Fig. 3 only includes the performance of Phase II and Phase III. It can be seen that the contribution of the pairwise interchange movement is significant for small values of λ by comparing the performance of Phase II and Phase III. Thus, we only considered the cases in which the values of λ fell within the interval $(0.2, 1.0)$ in later computational analysis.

In the third part of the experiment, the performance of the branch-and-bound algorithm and the heuristic algorithms was evaluated as the value of the learning effect varied. The job size was again fixed at 20 while the learning effect was from 70% to 90% with a jump of 1% each time, and λ took the values of 0.2, 0.4, 0.6, 0.8 and 1.0. A total of 105 cases were tested. For each case, 1000 instances were randomly generated, and the results were presented in Figs. 4 and 5.

It can be seen in Fig. 4 that the mean number of nodes increases as the learning effect becomes strong when $\lambda = 0.2$ and 0.4. It can be concluded from Fig. 1 that Property 3 is more useful as the value of λ is smaller. However, as the learning effect is stronger, Property 3 is less efficient since it is more difficult for the job completion time to surpass the maximal job release times. On the other hand, the mean number of nodes decreases as the learning effect becomes stronger when $\lambda = 0.6$ and 0.8. It can also be concluded in Fig. 1 that Property 1 and Corollary 1 are useful when the value of λ is large. In addition, Property 1 and Corollary 1 are more efficient in those cases since it is easier to finish the available jobs before the arrival of other jobs. This phenomenon is even stronger when $\lambda = 1.0$. In that case, the mean number of nodes tends to be zero as the learning effect becomes stronger. It can be observed in Fig. 5 that the performance of Phase III is very good with average error percentages of less than 0.08%. However, there is no trend between the performance of the heuristic algorithm and the learning effect. As seen in Fig. 2, the error percentage is smaller as the value of λ is larger when the learning effect is fixed.

In the four part of the experiment, the branch-and-bound algorithm and the proposed heuristic algorithm were tested with five different numbers of jobs ($n = 20, 24, 28, 32$, and 36). The control variable λ took the values of 0.2, 0.4, 0.6, 0.8, and 1.0, and the learning effect a took the values of 70%, 80% and 90%. For each situation, 100 instances were randomly generated, and the results are

Table 4
The performance of the heuristics with $n = 20$ and 90% learning effect.

λ	leftPhase I		leftPhase II		leftPhase III	
	Error percentage					
	Mean	Max	Mean	Max	Mean	Max
0.20	3.7673	11.9658	0.0853	1.6104	0.0149	0.7117
0.40	3.4478	12.8487	0.0739	1.5174	0.0185	0.3117
0.60	2.3827	12.6857	0.0493	0.6856	0.0147	0.3221
0.80	1.2549	8.6577	0.0158	0.4283	0.0059	0.2444
1.00	0.5729	6.7380	0.0032	0.2972	0.0012	0.1084
1.25	0.3041	4.8578	0.0006	0.0760	0.0002	0.0297
1.50	0.2034	3.6634	0.0003	0.0517	0.0000	0.0279
1.75	0.1196	3.5272	0.0001	0.0225	0.0000	0.0189
2.00	0.1101	2.7306	0.0001	0.0375	0.0000	0.0168
3.00	0.0468	1.5731	0.0000	0.0043	0.0000	0.0000

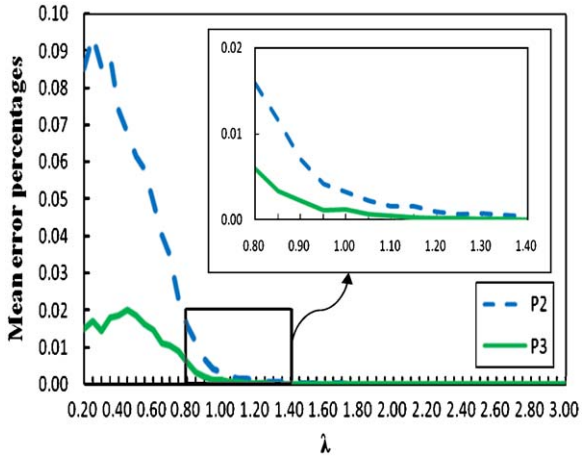

Fig. 3. The performance of Phase II and Phase III algorithms with respect to λ.

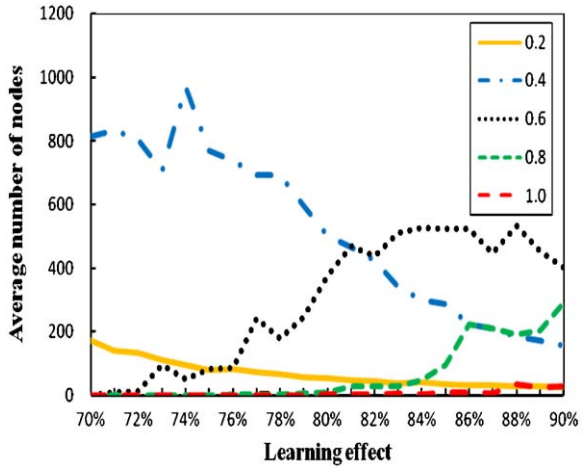

Fig. 4. The performance of the branch-and-bound algorithm with respect to the learning effect.

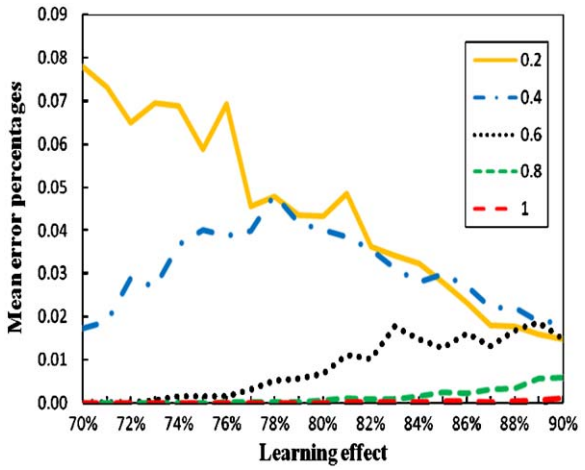

Fig. 5. The performance of Phase III algorithm with respect to the learning effect.

presented in Table 5. The branch-and-bound algorithm can solve most of the problems in a small amount of time when the job size is less than or equal to 36. Properties 1, 4, 5 and Corollary 1 are more useful when the range of release times is wider or the learning effect is stronger. Property 2 is useful when the range of release times is wide while Properties 3 and 6 are useful when the range of release times is narrow. Moreover, Properties 7–10 are less effective when the range of release times is wider or the learning effect is weaker. However, the execution time and the number of nodes increase dramatically as the job size increases, especially when the learning effect is strong and the value of λ is small. The worst case took about 11.57 h when $n = 36$, $\lambda = 0.4$ and the learning effect was 80%. On the other hand, the mean error percentages for Phase II and Phase III algorithm seemed to remain stable as the number of job increased. Moreover, the computational experiments showed that Phase II and Phase III algorithms were quite accurate with average error percentages of less than 0.1862% and 0.1088%, respectively. Moreover, the mean error percentages of both heuristic algorithms tended to be zero when $\lambda = 1.0$. Thus, Phase III algorithm is recommended for small values of λ due to its accuracy.

5. Conclusions

In this paper, we addressed a single-machine makespan problem where the job processing times are decreasing functions of their position and each job has its release time. The problem is NP-hard in the strong sense. Thus, we developed a branch-and-bound algorithm incorporating with several dominances and two lower bounds to derive the optimal solution. In addition, we also proposed a heuristic algorithm to obtain a near-optimal solution.

The computational results showed that with the help of the proposed heuristic initial solution, the branch-and-bound algorithm performs well in terms of the number of nodes and the execution time when the number of jobs is less than or equal to 36. Moreover, the computational experiments also showed that the performance of the proposed Phase III algorithm is very accurate. The extension to

Table 5
The performance of the branch-and-bound and the heuristic algorithms.

n	λ	α	Branch-and-bound algorithm				Phase II		Phase III	
			CPU time		Number of nodes		Error percentage			
			Mean	Max	Mean	Max	Mean	Max	Mean	Max
20	0.20	70	0.0452	0.2031	186	1158	0.1862	2.0845	0.1088	2.0845
		80	0.0197	0.3438	60	1410	0.1065	1.3610	0.0315	0.6771
		90	0.0128	0.0469	26	124	0.0963	1.6104	0.0159	0.3346
	0.40	70	0.1250	3.4219	952	25 529	0.0313	0.7032	0.0188	0.3197
		80	0.0770	1.0000	496	8697	0.0948	0.8741	0.0485	0.3998
		90	0.0355	0.4219	150	2007	0.0808	0.6238	0.0109	0.1495
	0.60	70	0.0000	0.0000	0	6	0.0000	0.0000	0.0000	0.0000
		80	0.0683	2.7812	593	29 246	0.0238	0.3585	0.0068	0.1470
		90	0.0631	0.4219	439	2831	0.0547	0.3331	0.0109	0.1178
	0.80	70	0.0000	0.0000	0	5	0.0000	0.0000	0.0000	0.0000
		80	0.0034	0.2812	20	1737	0.0012	0.0575	0.0002	0.0170
		90	0.0195	0.5000	198	3824	0.0141	0.1527	0.0064	0.1194
	1.00	70	0.0000	0.0000	0	11	0.0000	0.0000	0.0000	0.0000
		80	0.0002	0.0156	0	12	0.0000	0.0000	0.0000	0.0000
		90	0.0028	0.1406	38	2173	0.0026	0.0644	0.0006	0.0181
24	0.20	70	0.4775	6.1875	1418	17 233	0.1786	2.1627	0.0643	0.8916
		80	0.1055	1.4219	192	3563	0.1465	1.8826	0.0512	1.2911
		90	0.0508	0.3906	69	583	0.0812	0.4795	0.0116	0.2000
	0.40	70	0.3497	14.6094	1975	91 769	0.0211	0.8308	0.0072	0.5246
		80	0.6603	5.8281	2819	28 116	0.1100	1.1302	0.0344	1.1302
		90	0.2053	1.5000	659	5639	0.0793	0.5819	0.0185	0.1632
	0.60	70	0.0000	0.0000	0	6	0.0002	0.0146	0.0001	0.0103
		80	0.2528	10.3125	1399	54 504	0.0077	0.1578	0.0043	0.1578
		90	0.4361	9.1250	2039	29 388	0.0444	0.5076	0.0123	0.3032
	0.80	70	0.0000	0.0000	0	35	0.0000	0.0000	0.0000	0.0000
		80	0.0002	0.0156	0	5	0.0001	0.0061	0.0000	0.0000
		90	0.0887	5.0469	517	22 521	0.0091	0.1778	0.0027	0.1113
	1.00	70	0.0002	0.0156	0	6	0.0001	0.0075	0.0000	0.0000
		80	0.0005	0.0156	0	8	0.0001	0.0125	0.0000	0.0000
		90	0.0042	0.1875	56	3185	0.0009	0.0255	0.0005	0.0142
28	0.20	70	4.3563	47.6250	9618	120 420	0.1760	2.4619	0.0574	1.2740
		80	0.5961	3.4531	694	5678	0.1103	1.6357	0.0343	0.6211
		90	0.2649	2.4688	263	2902	0.0592	0.4913	0.0135	0.2757
	0.40	70	0.0188	1.7969	272	26 602	0.0018	0.0539	0.0003	0.0193
		80	16.3044	284.5000	54 474	761 881	0.0940	1.0432	0.0424	0.3798
		90	2.9578	139.2656	6687	309 145	0.0755	0.6555	0.0193	0.2761
	0.60	70	0.0000	0.0000	0	4	0.0003	0.0138	0.0000	0.0000
		80	3.2812	279.9219	21 169	1 910 931	0.0055	0.1619	0.0018	0.0606
		90	3.3005	30.0156	14 883	95 887	0.0350	0.2600	0.0113	0.1897
	0.80	70	0.0000	0.0000	0	2	0.0000	0.0000	0.0000	0.0000
		80	0.0003	0.0156	5	406	0.0002	0.0211	0.0000	0.0023
		90	1.2034	44.1719	6513	232 332	0.0050	0.0988	0.0012	0.0384
	1.00	70	0.0000	0.0000	0	1	0.0000	0.0040	0.0000	0.0000
		80	0.0002	0.0156	0	6	0.0000	0.0000	0.0000	0.0000
		90	0.0112	0.7344	97	5547	0.0030	0.0803	0.0008	0.0333
32	0.20	70	115.4145	2122.9062	235 343	6 548 117	0.1526	1.6864	0.0919	1.6864
		80	6.6736	177.1406	7112	217 764	0.1304	1.0920	0.0341	0.4391
		90	0.7325	16.7031	516	11 851	0.0750	0.5763	0.0094	0.1454
	0.40	70	2300.0701	230 006.9688	7 020 932	702 092 992	0.0006	0.0318	0.0000	0.0000
		80	110.5523	2678.6094	385 262	11 609 855	0.0920	0.7987	0.0196	0.1749
		90	22.8186	890.1094	45 904	2 141 226	0.0653	0.5807	0.0186	0.2783
	0.60	70	0.0002	0.0156	0	10	0.0001	0.0150	0.0000	0.0000
		80	2.8531	283.6719	8311	825 377	0.0020	0.1135	0.0010	0.0353
		90	134.4742	4040.2812	445 553	20 446 428	0.0431	0.3119	0.0137	0.2360
	0.80	70	0.0000	0.0000	0	4	0.0000	0.0000	0.0000	0.0000
		80	0.0006	0.0156	3	186	0.0003	0.0087	0.0001	0.0084
		90	9.8114	659.4688	32 525	1 800 263	0.0053	0.0803	0.0023	0.0709
	1.00	70	0.0000	0.0000	0	1	0.0000	0.0000	0.0000	0.0000
		80	0.0002	0.0156	0	8	0.0001	0.0076	0.0001	0.0076
		90	0.0012	0.0313	7	209	0.0004	0.0196	0.0002	0.0166
36	0.20	70	1881.4626	41 645.7969	2 943 904	77 158 511	0.1877	1.7338	0.0873	1.3438
		80	43.6572	1445.9531	39 434	1 615 989	0.1187	1.1016	0.0527	1.0258
		90	3.7502	57.2188	2148	37 576	0.0911	0.5691	0.0170	0.5054
	0.40	70	0.0000	0.0000	1	38	0.0001	0.0061	0.0000	0.0000
		80	5653.2881	186 009.4375	15 573 308	553 483 772	0.0786	0.9170	0.0328	0.2085
		90	212.9334	5443.1406	250 190	4 478 008	0.0789	0.7406	0.0155	0.1822
	0.60	70	0.0000	0.0000	0	5	0.0000	0.0035	0.0000	0.0000
		80	8.2377	819.8281	28 699	2 849 042	0.0024	0.1459	0.0005	0.0284
		90	776.8614	19 880.5625	1 803 784	49 277 848	0.0210	0.1847	0.0098	0.1158

Table 5 (*Continued*).

n	λ	a	Branch-and-bound algorithm				Phase II		Phase III	
			CPU time		Number of nodes		Error percentage			
			Mean	Max	Mean	Max	Mean	Max	Mean	Max
	0.80	70	0.0000	0.0000	0	3	0.0000	0.0034	0.0000	0.0000
		80	0.0003	0.0156	0	10	0.0004	0.0296	0.0000	0.0000
		90	162.4353	5346.8125	660 432	24 807 410	0.0052	0.0872	0.0029	0.0581
	1.00	70	0.0002	0.0156	0	4	0.0000	0.0016	0.0000	0.0000
		80	0.0003	0.0156	0	19	0.0003	0.0107	0.0000	0.0025
		90	0.0042	0.2656	38	1896	0.0004	0.0111	0.0000	0.0000

the bi-criterion or multiple machine system seems to be an interesting topic for future research.

Acknowledgments

The authors are grateful to the editor and the referees, whose constructive comments have led to a substantial improvement in the presentation of the paper.

References

[1] Heiser J, Render B. Operations management. 5th ed., Englewood Cliffs, NJ: Prentice-Hall; 1999.
[2] Russell R, Taylor BW. Operations management: multimedia version. 3rd ed., Upper Saddle River, NJ: Prentice-Hall; 2000.
[3] Wang FK, Lee W. Learning curve analysis in total productive maintenance. OMEGA—The International Journal of Management Science 2001;29:491–9.
[4] Biskup D. Single-machine scheduling with learning considerations. European Journal of Operational Research 1999;115:173–8.
[5] Gawiejnowicz S. A note on scheduling on a single processor with speed dependent on a number of executed jobs. Information Processing Letters 1996;56:297–300.
[6] Dondeti VR, Mohanty BB. Impact of learning and fatigue factors on single machine scheduling with penalties for tardy jobs. European Journal of Operational Research 1998;105:509–24.
[7] Cheng TCE, Wang G. Single machine scheduling with learning effect considerations. Annals of Operations Research 2000;98:273–90.
[8] Mosheiov G. Scheduling problems with a learning effect. European Journal of Operational Research 2001;132:687–93.
[9] Lee WC, Wu CC, Sung HJ. A bi-criterion single-machine scheduling problem with learning considerations. Acta Informatica 2004;40: 303–15.
[10] Lee WC, Wu CC. Minimizing total completion time in a two-machine flowshop with a learning effect. International Journal of Production Economics 2004;88: 85–93.
[11] Wang JB, Xia ZQ. Flow-shop scheduling with a learning effect. Journal of the Operational Research Society 2005;56:1325–30.
[12] Chen P, Wu CC, Lee WC. A bi-criteria two-machine flowshop scheduling problem with a learning effect. Journal of the Operational Research Society 2006;57: 1113–25.
[13] Wang JB. A note on scheduling problems with learning effects and deteriorating jobs. International Journal of Systems Science 2006;37:827–33.
[14] Eren T, Güner E. Minimizing total tardiness in a scheduling problem with a learning effect. Applied Mathematical Modelling 2007;31: 1351–61.
[15] Wu CC, Lee WC. A note on the total completion time problem in a permutation flowshop with a learning effect. European Journal of Operational Research 2009;192:343–7.
[16] Koulamas C, Kyparisis GJ. Single-machine and two-machine flowshop scheduling with general learning functions. European Journal of Operational Research 2007;178:402–7.
[17] Wang JB. Single-machine scheduling problems with the effects of learning and deterioration. OMEGA—The International Journal of Management Science 2007;35:397–402.
[18] Janiak A, Rudek R. The learning effect: getting to the core of the problem. Information Processing Letters 2007;103:183–7.
[19] Bachman A, Janiak A. Scheduling jobs with position-dependent processing times. Journal of the Operational Research Society 2004;55:257–64.
[20] Janiak A, Rudek R. A new approach to the learning effect: beyond the learning curve restrictions. Computers & Operations Research 2008;35:3727–36.
[21] Janiak A, Janiak W, Rudek R, Wielgus A. Solution algorithms for the makespan minimization problem with the general learning model, Computers & Industrial Engineering 2008, doi:10.1016/j.cie.2008.07.019.
[22] Biskup D. A state-of-the-art review on scheduling with learning effect. European Journal of Operational Research 2008;188:315–29.
[23] French S. Sequencing and scheduling: an introduction to the mathematics of the job shop. Ellis Horwood Limited; 1982.
[24] Nawaz M, Enscore EE, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA: The International Journal of Management Science 1983;11:91–5.
[25] Chu CB. A branch-and-bound algorithm to minimize total flow time with unequal release dates. Naval Research Logistics 1992;39: 859–75.