Computers & Operations Research 38 (2011) 1025–1034

Contents lists available at ScienceDirect

Computers & Operations Research

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

Genetic algorithm for minimizing the total weighted completion time scheduling problem with learning and release times

Chin-Chia Wu [a,*], Peng-Hsiang Hsu [a], Juei-Chao Chen [b], Nae-Sheng Wang [a]

[a] Department of Statistics, Feng Chia University, Taichung, Taiwan, ROC
[b] Department of Statistics and Information Science & Graduate Institute of Applied Statistics, Fu-Jen Catholic University, Taipei County, Taichung, Taiwan, ROC

ARTICLE INFO

Available online 9 November 2010

Keywords:
Genetic heuristic algorithm
Scheduling
Sum-of-processing time based learning effect
Release time

ABSTRACT

This paper considers a single-machine problem with the sum-of-processing time based learning effect and release times. The objective is to minimize the total weighted completion times. First, a branch-and-bound algorithm incorporating with several dominance properties and two lower bounds are developed for the optimal solution. Then a genetic heuristic-based algorithm is proposed for a near-optimal solution. Finally, a computational experiment is conducted to evaluate the performances of the proposed algorithms. The results show that the branch-and-bound algorithm can solve instances up to 15 jobs, and the average error percentage of the genetic heuristic algorithm is less than 0.105%.

1. Introduction

In classical scheduling theory, most researchers assume that the job processing times are fixed and known from the first job to be processed to the last job to be completed. However, Heizer and Render [1] and Russell and Taylor [2] pointed out that recent empirical studies in several industries have demonstrated that unit costs decline as firms produce more of a product and gain knowledge or experience. Biskup [3] claimed 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 in this setting is known as "scheduling with learning effects".

Biskup [3] and Cheng and Wang [4] were the pioneers to introduce the concept of learning into the field of scheduling. Since then, many researchers have devoted much effort to study this nascent and vivid area of scheduling. For example, Janiak and Rudek [5] proposed an experience-based learning effect. They proved that the problem to minimize the makespan under the Bachman and Janiak [6] model remains polynomially solvable when the experience-based approach is applied. Wang [7] considered the single-machine scheduling problems with the effects of learning and deterioration in which job processing times are functions of their starting times and positions in the sequence. Cheng et al. [8] introduced a new scheduling model with learning effects in which the actual processing time of a job is a function of the total normal processing times of the jobs already processed and of its position in a schedule. Wang et al. [9] considered the single-machine scheduling problem with a time-dependent learning effect. They assumed that job processing time is a function of the total normal processing time of the jobs scheduled in front of the job. Janiak and Rudek [10] brought a new model of the learning effect into the scheduling field that relaxed the rigorous constraints of the position-dependent approach by assuming that each job can provide different experience to the processor. They also described the shape of the learning curve by a k-stepwise function that is not restricted to a certain learning curve; it can thereby accurately fit various learning functions or even an irregular dependency between the experience possessed by the processor and the job processing times. Wang et al. [11] considered the single machine scheduling problem with exponential time-dependent learning effect and past-sequence-dependent (p-s-d) setup times. Wang [12] considered the single-machine scheduling problem with a sum-of processing-times-based learning effect. Wang [13] studied the single-machine scheduling problem with time-dependent learning effect and setup times. Sun [14] introduced a new scheduling model in which deteriorating jobs and learning effect are both considered simultaneously. Yin et al. [15] developed a general model with learning effects where the actual processing time of a job is not only a function of the total normal processing times of the jobs already processed, but also a function of its position in a schedule. In addition, Wu and Lee [16] proposed a learning model which considers both the machine and human learning effects simultaneously. They presented the solution procedures for some single machine and some flowshop problems. Toksari et al. [17] investigated several single machine scheduling problems under the joint effect of nonlinear job deterioration and time-dependent learning. They assumed that the processing time of a job increases when its processing is delayed, and meanwhile the machine undergoes a learning process decreases the time

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

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.11.001

required to process a given job. Zhang and Yan [18] proposed a general learning model which considers the position-based learning and the sum-of-processing-time-based learning effects at the same time. Huang et al. [19] considered the single machine scheduling problems with time-dependent deterioration and exponential learning effect. Wang [20] dealt with the single machine scheduling problems with a time-dependent learning effect and deteriorating jobs. Toksari [21] introduced simultaneous effects of learning and linear deterioration into assembly line balancing problem. Wang and Liu [22] considered a two-machine flowshop scheduling problem with effects of deterioration and learning. Recently, Biskup [23] presented a comprehensive review of research on scheduling with learning effects. In addition, Wang and Guo [24] studied the concepts of due-date assignment problems and the effects of learning and deterioration simultaneously. Janiak et al. [25] investigated the single-processor problem to minimize the makespan with an S-shaped learning model. They proved that the problem is strongly NP-hard even if the experience provided by each job is equal to its normal processing time. They constructed a branch-and-bound algorithm and some fast heuristic methods to find the optimal and near-optimal solutions, respectively. Wang [26] considered the single-machine scheduling problems with a sum-of-actual-processing-time-based learning effect. Wang et al. [27] considered the single machine scheduling problems with exponential sum-of-logarithm-processing-times based learning effect. Janiak and Rudek [28] proposed a new experience-based learning model, where the job processing times are described by S-shaped functions dependent on the experience of the processor. They proved that the problem to minimize the makespan on a single processor is NP-hard or strongly NP-hard with most of the commonly considered learning models. Janiak and Rudek [29] brought into scheduling a new approach called *multi-ability* learning that generalizes the existing ones and models more precisely real-life settings.

However, research with learning and release times is relatively unexplored. To the best of our knowledge, Bachman and Janiak [6] proved the makespan single-machine job position-based learning problem with release times is NP hard in the strong sense. Lee et al. [30] further provided exact and heuristic algorithms for the problem. Eren [31] considered a single machine scheduling problem with unequal release times and a position-based learning where the objective is to minimize the total weighted completion time. Lee et al. [32] studied a single-machine position-based learning scheduling problem with release times where the objective is to minimize the sum of makespan and total completion time. Motivated by this observation, in this paper we consider the single-machine total weighted completion time problem with sum of processing times-based learning and release times.

The rest of the paper is organized as follows. In the next section, the description of notation and the problem formulation 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 genetic heuristic-based algorithm 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. Problem formulation

The formulation of the proposed problem is as follows. There are n jobs to be scheduled. Preemption is not allowed and the machine is only able to process one job at a time. Each job j has a weight w_j, a normal processing time p_j, and a release time r_j. The actual processing time of job j is $p_{jr}=p_j(1-\sum_{l=1}^{r-1}p_{[l]}/\sum_{l=1}^{n}p_l)^a$ if it is scheduled in the rth position where $a\geq 1$ is a learning ratio common for all jobs. The main objective of this paper is to find an optimal schedule S^* to minimize the total weighted completion times, i.e. $\sum_{j=1}^{n}w_jC_j(S^*)\leq\sum_{j=1}^{n}w_jC_j(S)$ for any schedule S.

3. Dominance properties

Lenstra et al. [33] pointed out that the same problem without learning consideration is NP-Hard. 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 some adjacent pairwise interchange properties to fathom the searching tree. 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.

Before presenting the adjacent pairwise interchange properties, we provide several lemmas, which will be used in the proofs of the properties in the sequel.

Lemma 1. Let $h(x)=c[1-(1-x)^a]-\frac{1}{c}[1-(1-cx)^a]$. Then $h(x)\geq 0$ for $c\geq 1$, $0<x<1$, and $a\geq 1$.

Lemma 2. $f(x)=1+c[1-(1-x)^a-ax(1-cx)^{a-1}]$. Then $f(x)\geq 0$ for $c\geq 1$, $0<x<1$, and $a\geq 1$.

Lemma 3. Let $g(\theta)=(\theta-1)+c\theta[1-(1-x)^a]-\frac{1}{c}[1-(1-c\theta x)^a]$. Then $g(\theta)\geq 0$ for $\theta\geq 1$, $c\geq 1$, $a\geq 1$, and $0<x<1$.

Lemma 4. Let $f(x)=1-ax(1-x)^{a-1}-(1-x)^a$, then $f(x)\geq 0$ for $a\geq 1$ and $0<x<1$.

Lemma 5. Let $g(\theta)=(\theta-1)+(1-\theta x)^a-\theta(1-x)^a$, then $g(\theta)\geq 0$ for $\theta\geq 1$, $a\geq 1$, and $0<x<1$.

Next, we will develop some dominance properties based on a pairwise interchange of two adjacent jobs. Let $S=(\pi,J_i,J_j,\pi')$ and $S'=(\pi,J_j,J_i,\pi')$ be two sequences in which π and π' denote partial sequences. To show that S dominates S', it suffices to show that $[w_iC_i(S)+w_jC_j(S)]\leq[w_jC_j(S')+w_iC_i(S')]$ and $C_j(S)<C_i(S')$. In addition, let t be the completion time of the last job in the subsequence π with $(r-1)$ jobs.

Property 1. If $(p_j/p_i)>(w_j/w_i)>1$, and $\max\{r_i,r_j\}\leq t$, then S dominates S'.

Proof. Since $\max\{r_i,r_j\}\leq t$, we have

$$C_i(S)=t+p_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a \tag{1}$$

$$C_j(S)=t+p_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a+p_j\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_i}{\sum_{l=1}^{n}p_l}\right)^a \tag{2}$$

$$C_j(S')=t+p_j\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a \tag{3}$$

and

$$C_i(S')=t+p_j\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a+p_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_j}{\sum_{l=1}^{n}p_l}\right)^a \tag{4}$$

After taking the following difference between S and S', we have

$$[w_jC_j(S')+w_iC_i(S')]-[w_iC_i(S)+w_jC_j(S)]=(w_ip_j-w_jp_i)\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a$$

$$+w_jp_j\left[\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a-\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_i}{\sum_{l=1}^{n}p_l}\right)^a\right]$$
$$-w_ip_i\left[\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a-\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_j}{\sum_{l=1}^{n}p_l}\right)^a\right] \quad (5)$$

Taking $\theta=(p_j/w_j)/(p_i/w_i), c=w_j/w_i, x=p_i/(\sum_{l=1}^{n}p_l-\sum_{l=1}^{r-1}p_{[l]})$, Eq. (5) can be simplified to

$$[w_jC_j(S')+w_iC_i(S')]-[w_iC_i(S)+w_jC_j(S)]$$
$$=w_jp_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a\left\{(\theta-1)+c\theta[1-(1-x)^a]-\frac{1}{c}[1-(1-c\theta x)^a]\right\} \quad (6)$$

where $c\geq 1$, $\theta\geq 1$, $a\geq 1$, and $0<x<1$. From Lemma 3, we have Eq. (6) is non-negative. Hence, we have

$$[w_jC_j(S')+w_iC_i(S')]\geq[w_iC_i(S)+w_jC_j(S)].$$

After taking the difference of Eqs. (2) and (4), we have

$$C_i(S')-C_j(S)=(p_j-p_i)\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a+p_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_j}{\sum_{l=1}^{n}p_l}\right)^a$$
$$-p_j\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}+p_i}{\sum_{l=1}^{n}p_l}\right)^a \quad (7)$$

On substituting $\theta=p_j/p_i$, $u=(1-(\sum_{l=1}^{r-1}p_{[l]}/\sum_{l=1}^{n}p_l))$, and $x=(p_i/\sum_{l=1}^{n}p_l)$ into (7), and simplifying, we obtain

$$C_i(S')-C_j(S)=p_iu^a[(\theta-1)+(1-\theta x)^a-\theta(1-x)^a] \quad (8)$$

From Lemma 5, and $\theta\geq 1$, $a\geq 1$ and $0<x<1$, we have $C_i(S')-C_j(S)>0$. □

Therefore, S dominates S'.

Property 2. If $r_i\leq t\leq r_j\leq t+p_i(1-(\sum_{l=1}^{r-1}p_{[l]}/\sum_{l=1}^{n}p_l))^a$, and $1<(p_i/w_i)<(p_j/w_j)$, then S dominates S'.

Property 3. If $t\geq r_i$ and $t+p_i\left(1-\frac{\sum_{l=1}^{r-1}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a<r_j$ then S dominates S'.

Property 4. If $t\leq r_i\leq r_j$, $r_i+p_i(1-(\sum_{l=1}^{r-1}p_{[l]}/\sum_{l=1}^{n}p_l))^a\geq r_j$, and $1<(p_i/w_i)<(p_j/w_j)$, then S dominates S'.

Property 5. If $t\leq r_i$, and $r_i+p_i(1-(\sum_{l=1}^{r-1}p_{[l]}/\sum_{l=1}^{n}p_l))^a<r_j$, then S dominates S'.

3.1. Lower bound

In this subsection, we develop lower bounds by using a lemma from Hardy et al. [34].

Lemma 6. . [34] Suppose that there are two sequences of numbers a_i and b_i. The sum $\sum_{i=1}^{n}a_ib_i$ of products of the corresponding elements is the least if the sequences are monotonic in the opposite sense.

First, let S=(*PS*, *US*) be a sequence of jobs in which *PS* is a scheduled subsequence with k jobs and *US* is an unscheduled subsequence with n–k jobs. The symbol [ ] is used to signify the order of jobs in a sequence. Furthermore, let $w_{(1)}\leq w_{(2)}\leq\ldots\leq w_{(n-k)}$, $p_{(1)}\leq p_{(2)}\leq\ldots\leq p_{(n-k)}$, and $r_{(1)}\leq r_{(2)}\leq\ldots\leq r_{(n-k)}$ are the non-decreasing orders of the weights, normal processing times, and ready times in *US*. (Notice that $w_{(j)}$, $p_{(j)}$, and $r_{(j)}$ are not necessarily from the same job.) By definition, the completion time for the $(k+1)$th job is

$$C_{[k+1]}(S)=\max\{C_{[k]}(S),r_{[k+1]}\}+p_{[k+1]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq C_{[k]}(S)+p_{[k+1]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq C_{[k]}(S)+p_{(1)}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a \quad (9)$$

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

$$C_{[k+j]}(S)\geq C_{[k]}(S)+\sum_{i=1}^{j}p_{[k+i]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=1}^{i-1}p_{[k+l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq C_{[k]}(S)+\sum_{i=1}^{j}p_{(i)}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=n-k-i+2}^{n-k}p_{(l)}}{\sum_{l=1}^{n}p_l}\right)^a \quad \text{for } 1\leq j\leq n-k. \quad (10)$$

where $\sum_{l=1}^{0}p_{[k+l]}=0$, $\sum_{l=n-k+1}^{n-k}p_{(l)}=0$, and we let

$$\hat{C}_{(j)}=C_{[k]}(S)+\sum_{i=1}^{j}p_{(i)}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=n-k-i+2}^{n-k}p_{(l)}}{\sum_{l=1}^{n}p_l}\right)^a \quad \text{for } 1\leq j\leq n-k. \quad (11)$$

Therefore, based on Lemma 6 and Eqs. (9)–(11), the first lower bound is

$$LB_1=\sum_{i=1}^{k}w_{[i]}C_{[i]}(S)+\sum_{j=1}^{n-k}w_{(n-k-j+1)}\hat{C}_{(j)} \quad (12)$$

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

$$C_{[k+1]}(S)=\max\{C_{[k]}(S),r_{[k+1]}\}+p_{[k+1]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq r_{[k+1]}(S)+p_{[k+1]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq r_{(1)}(S)+p_{(1)}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}}{\sum_{l=1}^{n}p_l}\right)^a \quad (13)$$

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

$$C_{[k+j]}(S)\geq r_{[k+j]}(S)+\sum_{i=1}^{j}p_{[k+i]}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=1}^{i-1}p_{[k+l]}}{\sum_{l=1}^{n}p_l}\right)^a$$
$$\geq r_{(1)}(S)+\sum_{i=1}^{j}p_{(i)}(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=n-k-i+2}^{n-k}p_{(l)}}{\sum_{l=1}^{n}p_l})^a, \quad \text{for } 1\leq j\leq n-k. \quad (14)$$

Thus, we have

$$\tilde{C}_{(j)}=r_{(1)}(S)+\sum_{i=1}^{j}p_{(i)}\left(1-\frac{\sum_{l=1}^{k}p_{[l]}+\sum_{l=n-k-i+2}^{n-k}p_{(l)}}{\sum_{l=1}^{n}p_l}\right)^a \quad \text{for } 1\leq j\leq n-k. \quad (15)$$

According to Lemma 6 and Eqs. (13)–(15), the second lower bound is

$$LB_2=\sum_{i=1}^{k}w_{[i]}C_{[i]}(S)+\sum_{j=1}^{n-k}w_{(n-k-j+1)}\tilde{C}_{(j)} \quad (16)$$

In order to make the lower bound tighter, we choose the maximum value from Eqs. (12) and (16) as the lower bound

of *PS*. That is

$$LB = \max\{LB_1, LB_2\} \quad (17)$$

3.2. The details of genetic algorithms

Genetic algorithms (GAs) are intelligent random search strategies which have been successfully applied to find near-optimal solutions of many complex problems [35–39]. It originates from Darwin's [40] "survival of the fittest" concept, which means natural populations evolve according to the principles of natural selection. The basic principles of GA were first laid down rigorously by Holland [41]. A genetic algorithm starts with a set of feasible solutions (population) and iteratively replaces the current population by a new population. It requires a suitable encoding for the problem and a fitness function that represents a measure of the quality of each encoded solution (chromosome or individual). The reproduction mechanism selects the parents and recombines them using a crossover operator to generate offsprings that are submitted to a mutation operator in order to alter them locally [42]. The components of the GA applied to solve the proposed problem can be described as follows.

3.2.1. Representation of structure

In this study we adopt the method proposed by Etiler et al. [43] that a structure can be described as a sequence of the jobs in the problem.

3.2.2. Initial population

Most GA's in other contexts assume that the initial population is chosen completely at random [44]. But it is a good idea to construct the initial population drawing on other heuristics in the literature [45,46,43]. This enables us to arrive at the final solution more quickly. In this study, five initial sequences were proposed. In GA_1, jobs are arranged in earliest ready times (ERT) first rule while in GA_5, jobs are arranged according to the weighted smallest processing time (WSPT) rule which provides an optimal schedule for the classical total weighted completion time problem. The second, third, and fourth genetic algorithms, denoted as GA_2, GA_3, and GA_4, are arranged in a non-decreasing order on those weight-factors for jobs. They are $0.25p_i/w_i+0.75r_i$, $0.5p_i/w_i+0.5r_{,i}$ and $0.75p_i/w_i+0.25r_i$, respectively. Before performing GA, pairwise interchange is utilized to improve the quality of the solutions obtained from the previous rules to reduce many idle periods. The corresponding initial heuristic algorithms are recorded as GA_{01}, GA_{02}, …, GA_{05}, respectively.

3.2.3. Population size

The population size plays an important role in the computational process of GA. For a large population size, it is easier to obtain a better solution, but it consumes more time. One schedule by using each method to generate an initial population and other members are created by using an interchange mutation operator [46]. This procedure will be repeated until the number of the members is equal to population size. In a preliminary trial, the population size N is set at 60 in our computational experiment.

3.2.4. Fitness function

In order to mimic the natural process of the survival of the fittest, the fitness function assigns to each member of the population a value reflecting their relative superiority or inferiority [36]. Our objective is to minimize the total weighted completion time. The fitness function of the strings can be calculated as follows:

$$f(S_i(v)) = \max_{1 \le l \le N}\left\{\sum_{j=1}^{n} w_j C_j(S_l(v))\right\} - \sum_{j=1}^{n} w_j C_j(S_i(v)),$$

where $S_i(v)$is the ith string chromosome in the vth generation, $\sum_{j=1}^{n} w_j C_j(S_i(v))$ is the total weighted completion time of $S_i(v)$, and $f(S_i(v))$ is the fitness function of $S_i(v)$. Therefore, the probability, $P(S_i(v))$, of selection for a schedule is to ensure that the probability of selection for a sequence with lower value of the objective function is higher. Here $P(S_i(v))$ can be calculated as follows:

$$P(S_i(v)) = f(S_i(v)) \Big/ \sum_{l=1}^{N} f(S_l(v)).$$

This is also the criterion used for the selection of parents for the reproduction of children.

3.2.5. Crossover

Crossover is an operation to generate a new offspring from two parents. This study uses linear order crossover (LOX) method which is developed by Falkenauer and Bouffouix [47], and is one of the better performers among the others [48,44]. In a pilot study, in order to protect the best schedule which has the minimum total weighted completion time at each generation, we transfer this schedule to the next population with no change. This operation enables us to choose the higher crossover with the crossover rate $P_c=1$.

3.2.6. Mutation

Mutation is another main operator of GA. It is used to prevent premature and fall into local optimum. Such an operation can be viewed as a transition from a current solution to its neighborhood solution in a local search algorithm. In this study, the mutation rates (P_m) are set at 0.05 based on our preliminary experiment.

3.2.7. Selection

It is a procedure to select offspring from parents to the next generation. In our study, the population sizes are fixed at 60 from generation to generation. Excluding the best schedule which has the minimum total weighted completion time, the rest of the offsprings are generated from the parent chromosomes by the roulette wheel method.

3.2.8. Termination

The proposed GA's are terminated after 30 generations in our preliminary experiment.

4. Computational experiment

A computational experiment was conducted to evaluate the efficiency of the branch-and-bound algorithm, and the accuracy of the genetic algorithms. The algorithms were coded in Fortran and run on Compaq Visual Fortran version 6.6 on a Intel(R) Core(TM)2 Quad CPU 2.66 GHz with 4 GB RAM on Windows Vista. The experimental design follows Chu's [49] framework. The normal 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 job-size and λ is a control variable. Four different sets of problem instances were generated by giving λ the values 1.0, 1.5, 2.0, and 3.0.

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. For the five genetic and their corresponding initial heuristic algorithms, the mean and the maximum error percentages were recorded, where the error percentage was calculated as

$$(GA_i - TC^*)/TC^* * 100\%,$$

and

$$(GA_{0i}-TC^*)/TC^**100\%, \quad i=1,2,\ldots,5,$$

where $GA_i(GA_{0i})$ is the total weighted completion time obtained from the genetic (initial heuristic) algorithm and TC the total weighted completion time of the optimal schedule. The computational times of the heuristic algorithms were not recorded since they were finished within a second.

The computational experiment consisted of two parts. In the first part of the experiment, two job sizes ($n=12$ and 15) and three different values of learning effect ($a=1.001$, 1.01, and 1.1) were tested in the branch-and-bound algorithm. The same sets of instances were used to test the performance of the branch-and-bound and the genetic heuristic algorithms. As a consequence, 48 experimental situations were examined. A set of 100 instances were randomly generated for each situation. In order to explore the impact of the weights brought up on the proposed algorithms, the weights were from a uniform distributions U(1,50) and U(1,100), respectively. The algorithms were set to skip to the next set of data if the number of nodes exceeded 10^9. The instances with number of nodes less than 10^9 were denoted as solvable instances (SI). The results are presented in Tables 1–3. Table 1 indicated that most number of nodes in the instances with smaller values of λ is larger than that in those with larger values of λ when fixed learning rate, the trend becomes clear when job size is up to 15. This was due to the fact that the performances of the LB_2 and some properties involved with ready times became weaker. It can be found that no clear trend of the learning effect existed in Table 1. It can be observed that fixed $n=15$, there were 310 cases and 301 cases in which the branch-and-bound algorithm could solve all the problems optimally larger than 10^9 nodes in Table 1. This implied that the instances with smaller values of weights are slightly difficult than those with larger ones.

As to the performance of the proposed GA algorithms, out of the 48 evaluations, the performances of proposed genetic heuristics were not affected as the learning rate, or the release times, or the number of jobs, or the values of λ varied. The maximum mean error percentages of GA_1, GA_2, GA_3, GA_4, and GA_5 were 0.079%, 0.105%, 0.105%, 0.115%, 0.781% in Table 2, while 0.062%, 0.062%, 0.062%, 0.074%, 0.789% in Table 3, respectively. The worst cases of the corresponding algorithms were 2.180%, 2.180%, 2.180%, 2.180%, and 11.755% in Table 2, while 1.398%, 1.398%, 1.398%, 2.872%, and 8.970% in Table 3, respectively. The performances of the front four genetic algorithms are better than the last one.

In the second part of the experiment, the proposed heuristic algorithms were tested with four different numbers of jobs: $n=20$, 40, 60, and 80 to further analyze the performance of the proposed heuristic algorithms for large job-sized problems. The mean execution time and the mean relative deviance percentage were reported for each heuristic. The relative deviance percentage (RDP) was calculated as

$$(GA_i-GA^*)/GA^**100\%,$$

and

$$(GA_{0i}-GA^*)/GA^**100\%, \quad i=1,2,\ldots,5,$$

where $GA_i(GA_{0i})$ is the value of the objective function generated by the ith heuristic (its initial), and $GA^*=\min\{GA_i, \quad i=1,2,\ldots,5\}$ is the smallest value of the objective function obtained from the heuristics. The results are reported in Tables 4 and 5.

It was observed that there are no effects on the performance of the heuristics as the values of n, a, or λ change. In addition, there is a significant difference between the performances of the first four genetic algorithms and the last one, since out of 96 cases, the overall numbers of the five GA algorithms with the smallest relative deviance percentage are 45, 49, 58, 53, and 0. Thus, it is recommended to use the GA_3 algorithm since it has both accuracy and the smallest RDP.

5. Conclusions

In this paper, we studied a single-machine total weighted completion time problem with the sum of processing times based

Table 1
The performance of the branch-and-bound algorithm.

n	a	λ	$w_i\sim$U(1,50)					$w_i\sim$U(1,100)				
			Node		cpu time		IS	Node		cpu time		IS
			Mean	Max	Mean	Max		Mean	Max	Mean	Max	
12	1.001	1.0	3,866,435	19,551,999	44.0088	220.9910	100	4,386,571	66,908,154	50.6195	806.8059	100
		1.5	1,377,717	7,572,901	15.3780	86.0347	100	2,369,710	35,077,777	27.0907	429.6113	100
		2.0	1,012,102	6,109,061	11.2146	71.8232	100	1,076,218	6,196,749	11.9450	72.2285	100
		3.0	681,823	8,715,088	7.4838	101.4014	100	657,416	8,062,312	7.1175	91.8691	100
	1.01	1.0	3,888,810	27175,103	44.2647	303.4531	100	4,531,436	42,958,200	51.7438	512.8691	100
		1.5	1,646,365	7,756,922	18.3378	89.6699	100	1,618,838	17,397,411	18.1738	200.6641	100
		2.0	1,026,144	10,851,068	11.3974	127.2812	100	1,229,180	11,777,213	13.7855	140.0410	100
		3.0	581,458	6,260,564	6.4338	73.1172	100	543,935	8,544,526	5.9530	100.3555	100
	1.1	1.0	4,117,798	40,010,486	47.3702	468.9707	100	5,832,865	61,965,613	67.4349	751.1445	100
		1.5	2,060,567	25,803,877	23.1923	303.7812	100	2,508,799	42,831,641	28.3886	524.6797	100
		2.0	985,771	12,343,486	10.9171	142.3203	100	1,295,681	8,875,516	14.3764	103.0078	100
		3.0	590,941	4,972,768	6.4296	56.9414	100	755,804	9,285,691	8.3277	101.0234	100
15	1.001	1.0	441,464,709	944,209,945	6670.0688	14481.5000	40	469,157,287	974,041,689	7058.3670	14756.6300	45
		1.5	291,018,139	812,446,653	4325.6592	12154.0625	66	379,665,183	996,429,704	5793.6420	15282.0000	68
		2.0	232,345,002	980,866,987	3443.7859	14842.3750	83	221,837,016	802,598,069	3311.6470	12770.9400	88
		3.0	143,005,609	990,174,764	2124.1665	16203.2969	94	137,613,767	953,880,820	2057.0370	14696.5000	94
	1.01	1.0	447,635,316	983,551,272	6732.9824	15356.8438	42	572,086,668	981,987,974	8580.9990	14431.8800	38
		1.5	271,074,736	993,164,439	4023.9661	15198.9375	72	266,268,047	984,601,771	4015.3350	15268.5800	72
		2.0	230,421,547	982,253,427	3438.6475	15314.7432	81	197,706,639	898,261,592	2955.0820	13841.1300	84
		3.0	129,391,248	747,376,425	1900.0920	11489.5156	97	131,059,083	982,491,961	1975.4490	14155.3900	94
	1.1	1.0	465,320,612	935,009,272	7052.9468	14340.4375	46	457,586,896	990,634,536	6890.0370	15040.7700	47
		1.5	291,061,045	974,461,884	4415.4800	15446.7188	81	282,359,975	989,850,608	4222.6400	15402.5900	82
		2.0	196,188,671	959,374,600	2927.0066	14319.4375	94	230,136,216	964,745,225	3441.7750	14999.5000	92
		3.0	135,889,272	864,225,071	2048.2971	13790.5000	94	146,382,607	958,114,382	2214.1330	14904.0200	95

Table 2
The performance of the proposed heuristic algorithms at $w_i \sim U(1,50)$.

n	a	λ	Error percentage										Error percentage									
			GA_{01}		GA_{02}		GA_{03}		GA_{04}		GA_{05}		GA_1		GA_2		GA_3		GA_4		GA_5	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	1.001	1.0	0.059	1.121	0.056	1.121	0.046	1.121	0.092	3.284	2.777	27.501	0.030	1.121	0.027	1.121	0.027	1.121	0.037	1.204	0.625	6.192
		1.5	0.035	1.203	0.035	1.203	0.032	1.203	0.030	1.203	1.287	16.007	0.003	0.299	0.003	0.299	0.003	0.299	0.003	0.299	0.259	3.309
		2.0	0.004	0.245	0.004	0.245	0.004	0.245	0.007	0.322	0.854	6.183	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.217	3.589
		3.0	0.001	0.080	0.001	0.080	0.001	0.080	0.001	0.080	0.424	5.775	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.128	1.496
	1.01	1.0	0.067	1.601	0.063	1.601	0.065	1.601	0.068	1.161	2.541	15.759	0.030	0.842	0.030	0.842	0.029	0.842	0.047	1.161	0.507	7.624
		1.5	0.052	2.180	0.052	2.180	0.057	2.180	0.053	2.180	1.649	9.324	0.040	2.180	0.040	2.180	0.040	2.180	0.032	2.180	0.371	5.902
		2.0	0.014	0.492	0.014	0.492	0.014	0.492	0.017	1.120	1.076	14.836	0.008	0.492	0.008	0.492	0.008	0.492	0.003	0.252	0.230	6.306
		3.0	0.003	0.181	0.003	0.181	0.004	0.181	0.008	0.186	0.440	5.214	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.161	0.103	1.495
	1.1	1.0	0.085	1.912	0.085	1.912	0.085	1.912	0.070	1.509	2.952	33.217	0.059	1.912	0.059	1.912	0.059	1.912	0.042	1.478	0.781	11.755
		1.5	0.065	1.369	0.065	1.369	0.049	1.369	0.052	0.935	1.154	9.515	0.033	1.367	0.033	1.367	0.028	1.367	0.015	0.429	0.209	4.453
		2.0	0.047	1.109	0.047	1.109	0.041	1.109	0.041	1.109	1.333	9.372	0.023	1.109	0.023	1.109	0.019	1.109	0.019	1.109	0.299	5.129
		3.0	0.007	0.505	0.007	0.505	0.007	0.505	0.008	0.505	0.704	7.295	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.117	4.071
15	1.001	1.0	0.090	1.201	0.118	1.201	0.118	1.201	0.141	1.201	2.275	9.411	0.078	1.130	0.105	1.130	0.105	1.130	0.115	1.130	0.732	3.689
		1.5	0.043	1.254	0.043	1.254	0.043	1.254	0.057	1.254	1.611	9.917	0.022	1.254	0.022	1.254	0.022	1.254	0.026	1.254	0.660	5.916
		2.0	0.018	0.429	0.018	0.429	0.018	0.429	0.030	0.429	1.008	11.993	0.006	0.137	0.006	0.137	0.006	0.137	0.011	0.292	0.354	2.888
		3.0	0.000	0.038	0.000	0.038	0.000	0.038	0.005	0.244	0.637	8.892	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.151	0.285	3.678
	1.01	1.0	0.154	2.632	0.154	2.632	0.154	2.632	0.096	2.632	1.876	14.263	0.079	1.759	0.079	1.759	0.079	1.759	0.033	1.051	0.622	6.319
		1.5	0.024	0.420	0.024	0.420	0.020	0.420	0.046	0.897	0.970	5.086	0.021	0.420	0.021	0.420	0.019	0.420	0.031	0.897	0.205	3.400
		2.0	0.024	0.808	0.024	0.808	0.014	0.736	0.031	1.055	0.939	6.110	0.008	0.351	0.008	0.351	0.008	0.351	0.010	0.351	0.368	3.262
		3.0	0.009	0.340	0.009	0.340	0.009	0.340	0.012	0.340	0.635	7.260	0.006	0.340	0.006	0.340	0.006	0.340	0.009	0.340	0.213	3.914
	1.1	1.0	0.080	1.389	0.080	1.389	0.076	1.389	0.065	1.389	1.938	20.669	0.060	1.389	0.060	1.389	0.060	1.389	0.030	1.389	0.421	3.766
		1.5	0.045	0.925	0.045	0.925	0.043	0.925	0.063	1.505	1.370	7.388	0.016	0.464	0.016	0.464	0.016	0.464	0.029	0.701	0.406	4.379
		2.0	0.007	0.260	0.007	0.260	0.008	0.260	0.011	0.367	0.742	3.984	0.005	0.260	0.005	0.260	0.005	0.260	0.005	0.260	0.286	2.191
		3.0	0.009	0.363	0.005	0.209	0.005	0.209	0.007	0.248	0.528	4.384	0.006	0.363	0.003	0.209	0.003	0.209	0.003	0.209	0.177	2.190

Table 3
The performance of the proposed heuristic algorithms at $w_i \sim U(1,100)$

n	a	λ	Error percentage										Error percentage									
			GA_{01}		GA_{02}		GA_{03}		GA_{04}		GA_{05}		GA_1		GA_2		GA_3		GA_4		GA_5	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	1.001	1.0	0.083	1.398	0.083	1.398	0.083	1.398	0.099	1.398	2.598	17.499	0.062	1.398	0.062	1.398	0.062	1.398	0.062	1.398	0.789	8.970
		1.5	0.045	1.788	0.045	1.788	0.047	1.788	0.054	1.788	2.059	11.725	0.015	0.917	0.015	0.917	0.015	0.917	0.016	0.917	0.527	6.600
		2.0	0.024	1.306	0.024	1.306	0.024	1.306	0.027	1.306	1.010	5.455	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.081	0.253	2.292
		3.0	0.011	0.483	0.011	0.483	0.011	0.483	0.011	0.483	0.520	3.958	0.002	0.220	0.002	0.220	0.002	0.220	0.002	0.220	0.083	1.286
	1.01	1.0	0.082	2.397	0.082	2.397	0.083	2.397	0.151	6.669	2.511	16.644	0.044	1.234	0.044	1.234	0.045	1.234	0.074	2.872	0.503	7.441
		1.5	0.050	2.969	0.050	2.969	0.051	2.969	0.059	2.969	1.450	15.331	0.001	0.071	0.001	0.071	0.000	0.008	0.000	0.008	0.419	4.038
		2.0	0.033	1.286	0.033	1.286	0.033	1.286	0.027	0.872	1.027	6.931	0.027	1.286	0.027	1.286	0.027	1.286	0.017	0.669	0.255	3.263
		3.0	0.002	0.221	0.002	0.221	0.002	0.221	0.001	0.071	0.650	7.975	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.071	0.113	2.171
	1.1	1.0	0.066	1.336	0.066	1.336	0.051	1.081	0.070	1.400	3.075	20.137	0.036	1.336	0.036	1.336	0.023	0.743	0.023	0.743	0.793	8.556
		1.5	0.042	1.371	0.042	1.371	0.042	1.371	0.046	1.371	1.769	10.954	0.024	1.371	0.024	1.371	0.024	1.371	0.019	1.371	0.472	4.233
		2.0	0.020	1.101	0.020	1.101	0.020	1.101	0.022	1.101	0.906	9.526	0.003	0.207	0.003	0.207	0.003	0.207	0.003	0.207	0.216	4.049
		3.0	0.012	0.542	0.012	0.542	0.012	0.542	0.008	0.571	0.663	4.773	0.001	0.106	0.001	0.106	0.001	0.106	0.001	0.099	0.162	2.092
15	1.001	1.0	0.069	1.014	0.069	1.014	0.069	1.014	0.058	1.014	2.267	9.446	0.030	0.856	0.030	0.856	0.030	0.856	0.030	0.856	0.594	3.940
		1.5	0.014	0.257	0.013	0.257	0.013	0.257	0.013	0.257	1.435	7.090	0.002	0.095	0.002	0.095	0.002	0.095	0.002	0.095	0.540	6.029
		2.0	0.007	0.275	0.005	0.275	0.005	0.275	0.005	0.275	1.357	9.177	0.004	0.234	0.002	0.071	0.002	0.071	0.002	0.071	0.490	3.777
		3.0	0.001	0.123	0.001	0.123	0.001	0.123	0.005	0.230	0.484	4.322	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.230	0.203	2.060
	1.01	1.0	0.067	0.707	0.067	0.707	0.067	0.707	0.091	0.707	1.994	10.859	0.037	0.707	0.037	0.707	0.037	0.707	0.037	0.707	0.432	3.988
		1.5	0.051	0.927	0.046	0.927	0.046	0.927	0.046	0.927	1.430	11.449	0.026	0.927	0.026	0.927	0.026	0.927	0.026	0.927	0.496	6.902
		2.0	0.004	0.110	0.004	0.110	0.003	0.110	0.003	0.110	0.831	7.428	0.002	0.110	0.002	0.110	0.002	0.110	0.002	0.110	0.316	4.681
		3.0	0.003	0.120	0.003	0.120	0.003	0.120	0.005	0.120	0.627	4.254	0.001	0.054	0.001	0.054	0.001	0.054	0.001	0.054	0.221	2.097
	1.1	1.0	0.058	1.135	0.058	1.135	0.046	0.615	0.045	0.546	1.959	9.985	0.017	0.546	0.017	0.546	0.017	0.546	0.020	0.546	0.708	4.055
		1.5	0.009	0.238	0.011	0.238	0.009	0.238	0.013	0.322	1.343	9.713	0.005	0.238	0.005	0.238	0.005	0.238	0.009	0.322	0.467	3.985
		2.0	0.016	0.628	0.016	0.628	0.016	0.628	0.019	0.628	0.832	6.691	0.003	0.313	0.003	0.313	0.003	0.313	0.007	0.313	0.272	3.202
		3.0	0.005	0.172	0.005	0.172	0.005	0.172	0.003	0.131	0.441	3.251	0.002	0.172	0.002	0.172	0.002	0.172	0.000	0.000	0.155	1.886

Table 4
The performance of the proposed heuristic algorithms for large-size jobs at $w_i \sim U(1,50)$

n	a	λ	Relative deviance percentage										Relative deviance percentage									
			GA_{01}		GA_{02}		GA_{03}		GA_{04}		GA_{05}		GA_1		GA_2		GA_3		GA_4		GA_5	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
20	1.001	1.0	0.023	0.855	0.025	0.855	0.025	0.855	0.031	0.523	3.456	17.218	0.017	0.855	0.018	0.855	0.018	0.855	0.018	0.523	1.949	14.468
		1.5	0.015	0.717	0.014	0.717	0.014	0.717	0.017	0.313	1.958	14.289	0.005	0.211	0.004	0.211	0.004	0.211	0.013	0.313	1.052	10.642
		2.0	0.022	0.635	0.022	0.635	0.020	0.635	0.018	0.635	1.270	8.011	0.006	0.592	0.006	0.592	0.006	0.592	0.001	0.080	0.639	4.990
		3.0	0.001	0.075	0.001	0.075	0.001	0.075	0.002	0.082	0.875	6.812	0.000	0.035	0.000	0.035	0.000	0.035	0.001	0.082	0.440	2.761
	1.01	1.0	0.050	2.003	0.052	2.003	0.053	2.003	0.044	0.950	3.334	12.368	0.035	1.511	0.037	1.511	0.030	1.511	0.015	0.331	1.683	10.616
		1.5	0.007	0.256	0.007	0.256	0.007	0.256	0.021	0.303	2.068	11.588	0.004	0.256	0.004	0.256	0.004	0.256	0.012	0.266	1.081	6.510
		2.0	0.007	0.222	0.007	0.222	0.007	0.222	0.007	0.222	0.979	5.897	0.005	0.222	0.005	0.222	0.005	0.222	0.004	0.222	0.513	2.458
		3.0	0.002	0.086	0.002	0.086	0.002	0.081	0.003	0.090	0.529	3.778	0.001	0.086	0.001	0.086	0.000	0.000	0.001	0.079	0.253	2.317
	1.1	1.0	0.054	1.755	0.054	1.755	0.054	1.755	0.060	1.755	2.744	14.823	0.025	0.945	0.025	0.945	0.025	0.945	0.032	0.945	1.508	9.280
		1.5	0.014	0.428	0.014	0.428	0.012	0.428	0.018	0.468	1.928	8.815	0.009	0.428	0.009	0.428	0.009	0.428	0.014	0.468	1.060	5.259
		2.0	0.013	0.496	0.011	0.496	0.015	0.496	0.023	0.728	1.159	9.512	0.006	0.496	0.006	0.496	0.008	0.496	0.010	0.496	0.613	5.886
		3.0	0.002	0.154	0.002	0.154	0.000	0.011	0.002	0.066	0.564	2.366	0.000	0.011	0.000	0.011	0.000	0.011	0.001	0.063	0.285	1.898
40	1.001	1.0	0.005	0.119	0.005	0.119	0.005	0.119	0.019	0.430	3.817	15.183	0.002	0.084	0.002	0.084	0.002	0.084	0.015	0.430	2.981	10.912
		1.5	0.010	0.151	0.009	0.151	0.007	0.136	0.010	0.136	2.312	10.358	0.006	0.151	0.005	0.151	0.002	0.106	0.005	0.101	1.841	8.211
		2.0	0.007	0.121	0.006	0.121	0.006	0.121	0.005	0.050	1.730	7.450	0.004	0.121	0.003	0.121	0.003	0.121	0.002	0.038	1.368	5.969
		3.0	0.002	0.106	0.001	0.079	0.001	0.079	0.002	0.060	0.913	3.372	0.001	0.043	0.001	0.043	0.000	0.040	0.002	0.060	0.688	3.091
	1.01	1.0	0.014	0.487	0.014	0.487	0.012	0.487	0.012	0.298	3.256	7.347	0.013	0.487	0.013	0.487	0.012	0.487	0.009	0.194	2.698	7.132
		1.5	0.005	0.087	0.005	0.087	0.005	0.087	0.007	0.089	2.179	12.388	0.003	0.087	0.003	0.087	0.003	0.087	0.005	0.089	1.744	10.552
		2.0	0.003	0.128	0.003	0.128	0.001	0.037	0.003	0.057	1.739	7.516	0.002	0.128	0.002	0.128	0.001	0.037	0.002	0.057	1.333	6.457
		3.0	0.001	0.039	0.001	0.039	0.001	0.039	0.001	0.028	0.828	5.326	0.001	0.039	0.001	0.039	0.001	0.039	0.001	0.028	0.662	4.737
	1.1	1.0	0.014	0.578	0.015	0.578	0.013	0.578	0.010	0.163	3.514	12.246	0.010	0.578	0.010	0.578	0.009	0.578	0.006	0.090	2.813	10.235
		1.5	0.004	0.096	0.004	0.096	0.003	0.096	0.007	0.212	2.411	12.401	0.003	0.096	0.003	0.096	0.002	0.096	0.005	0.212	1.923	8.021
		2.0	0.004	0.097	0.004	0.097	0.003	0.097	0.006	0.097	1.399	4.607	0.001	0.068	0.001	0.068	0.000	0.014	0.003	0.073	1.137	4.079
		3.0	0.001	0.021	0.001	0.021	0.000	0.021	0.001	0.024	0.872	4.325	0.000	0.021	0.000	0.021	0.000	0.021	0.001	0.024	0.700	3.258
60	1.001	1.0	0.010	0.311	0.007	0.249	0.007	0.249	0.014	0.419	4.245	15.770	0.009	0.311	0.006	0.249	0.006	0.249	0.012	0.419	3.711	13.713
		1.5	0.009	0.156	0.009	0.156	0.008	0.156	0.006	0.150	2.641	6.845	0.007	0.156	0.007	0.156	0.006	0.156	0.004	0.053	2.315	6.500
		2.0	0.001	0.030	0.001	0.030	0.001	0.030	0.001	0.015	1.650	7.650	0.001	0.030	0.001	0.030	0.001	0.030	0.001	0.015	1.448	6.830
		3.0	0.002	0.037	0.001	0.029	0.001	0.029	0.001	0.029	1.083	5.485	0.001	0.037	0.001	0.027	0.001	0.027	0.000	0.004	0.909	5.459
	1.01	1.0	0.008	0.242	0.007	0.242	0.008	0.230	0.010	0.103	3.979	11.171	0.007	0.242	0.006	0.242	0.007	0.230	0.007	0.103	3.488	10.718
		1.5	0.003	0.080	0.003	0.080	0.003	0.049	0.004	0.041	2.396	6.537	0.003	0.080	0.003	0.080	0.002	0.049	0.004	0.041	2.136	5.901
		2.0	0.001	0.024	0.001	0.024	0.001	0.024	0.002	0.024	2.011	10.702	0.000	0.022	0.000	0.022	0.000	0.022	0.001	0.015	1.746	9.619
		3.0	0.001	0.023	0.001	0.023	0.001	0.016	0.001	0.016	1.038	3.973	0.001	0.023	0.001	0.023	0.000	0.011	0.000	0.007	0.878	2.991
	1.1	1.0	0.007	0.165	0.007	0.165	0.008	0.233	0.024	0.404	3.483	12.069	0.003	0.107	0.003	0.107	0.004	0.233	0.019	0.404	3.101	11.598
		1.5	0.003	0.105	0.003	0.105	0.005	0.170	0.004	0.122	2.419	9.187	0.002	0.074	0.002	0.074	0.004	0.170	0.002	0.022	2.086	8.018
		2.0	0.004	0.107	0.004	0.107	0.002	0.047	0.001	0.014	1.417	6.030	0.003	0.107	0.003	0.107	0.002	0.047	0.001	0.014	1.234	4.794
		3.0	0.002	0.034	0.002	0.034	0.001	0.034	0.001	0.012	1.048	6.163	0.001	0.024	0.001	0.024	0.001	0.024	0.000	0.008	0.892	4.593
80	1.001	1.0	0.009	0.201	0.010	0.201	0.010	0.208	0.010	0.237	4.218	11.809	0.006	0.182	0.007	0.182	0.007	0.208	0.008	0.237	3.847	10.681
		1.5	0.002	0.065	0.002	0.065	0.002	0.065	0.003	0.104	2.642	9.312	0.002	0.065	0.002	0.065	0.002	0.065	0.003	0.104	2.396	8.912
		2.0	0.001	0.034	0.001	0.034	0.001	0.034	0.001	0.032	1.875	12.963	0.001	0.034	0.001	0.034	0.001	0.034	0.001	0.032	1.687	11.611
		3.0	0.001	0.019	0.001	0.019	0.001	0.019	0.001	0.019	1.099	4.568	0.001	0.014	0.001	0.014	0.001	0.014	0.000	0.004	0.984	3.953
	1.01	1.0	0.004	0.123	0.004	0.123	0.004	0.123	0.005	0.063	3.754	9.301	0.004	0.123	0.004	0.123	0.003	0.123	0.004	0.063	3.403	8.828
		1.5	0.005	0.098	0.006	0.098	0.005	0.098	0.005	0.059	2.530	7.917	0.004	0.098	0.005	0.098	0.004	0.098	0.003	0.059	2.291	7.316
		2.0	0.002	0.038	0.002	0.038	0.002	0.038	0.001	0.028	1.883	8.158	0.001	0.020	0.001	0.020	0.001	0.012	0.001	0.008	1.689	7.308
		3.0	0.000	0.013	0.000	0.013	0.000	0.013	0.001	0.006	1.104	4.846	0.000	0.005	0.000	0.005	0.000	0.004	0.001	0.006	0.989	4.640
	1.1	1.0	0.008	0.184	0.008	0.184	0.008	0.184	0.008	0.128	3.815	13.681	0.007	0.184	0.007	0.184	0.007	0.184	0.007	0.128	3.509	12.971
		1.5	0.003	0.054	0.003	0.054	0.003	0.037	0.003	0.049	2.338	6.305	0.003	0.054	0.003	0.054	0.003	0.037	0.002	0.040	2.127	5.751
		2.0	0.001	0.017	0.001	0.017	0.001	0.017	0.001	0.009	1.721	6.663	0.001	0.017	0.001	0.017	0.001	0.017	0.001	0.009	1.565	5.978
		3.0	0.001	0.015	0.000	0.009	0.000	0.009	0.001	0.005	0.872	3.451	0.001	0.015	0.000	0.009	0.000	0.009	0.001	0.005	0.801	3.014

Table 5
The performance of the proposed heuristic algorithms for large-size jobs at $w_i \sim U(1,100)$

n	a	λ	Relative deviance percentage										Relative deviance percentage									
			GA_{01}		GA_{02}		GA_{03}		GA_{04}		GA_{05}		GA_1		GA_2		GA_3		GA_4		GA_5	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
20	1.001	1.0	0.050	1.898	0.052	1.898	0.053	1.898	0.046	1.139	3.202	12.910	0.042	1.898	0.043	1.898	0.043	1.898	0.032	1.139	1.664	9.696
		1.5	0.024	0.678	0.024	0.678	0.024	0.678	0.024	0.678	2.137	20.172	0.013	0.575	0.014	0.575	0.014	0.575	0.014	0.575	1.161	14.588
		2.0	0.003	0.144	0.003	0.144	0.003	0.144	0.006	0.144	1.275	10.503	0.000	0.008	0.000	0.000	0.000	0.000	0.002	0.084	0.652	7.157
		3.0	0.002	0.068	0.001	0.068	0.001	0.068	0.006	0.164	0.693	4.947	0.001	0.053	0.000	0.001	0.000	0.001	0.003	0.101	0.334	3.059
	1.01	1.0	0.039	0.530	0.041	0.530	0.041	0.530	0.037	0.530	4.032	33.313	0.018	0.530	0.020	0.530	0.018	0.530	0.014	0.530	2.409	28.590
		1.5	0.015	1.093	0.015	1.093	0.015	1.093	0.016	0.280	1.726	12.007	0.014	1.093	0.014	1.093	0.014	1.093	0.012	0.280	0.921	6.182
		2.0	0.016	0.801	0.016	0.801	0.018	0.801	0.016	0.801	1.296	8.821	0.007	0.335	0.007	0.335	0.009	0.335	0.006	0.335	0.605	5.511
		3.0	0.001	0.059	0.001	0.041	0.001	0.041	0.002	0.078	0.623	3.390	0.001	0.059	0.000	0.000	0.000	0.000	0.000	0.000	0.297	2.199
	1.1	1.0	0.038	1.311	0.037	1.311	0.043	1.311	0.043	1.311	2.683	17.732	0.010	0.412	0.010	0.412	0.016	1.041	0.016	1.041	1.459	11.537
		1.5	0.013	0.288	0.013	0.288	0.012	0.288	0.012	0.288	1.612	10.982	0.003	0.166	0.003	0.166	0.002	0.166	0.002	0.092	0.896	6.513
		2.0	0.014	0.655	0.014	0.655	0.014	0.655	0.017	0.655	1.185	6.684	0.008	0.655	0.008	0.655	0.008	0.655	0.010	0.655	0.584	4.302
		3.0	0.002	0.210	0.002	0.210	0.000	0.000	0.002	0.105	0.593	5.651	0.001	0.078	0.001	0.078	0.000	0.000	0.001	0.105	0.307	2.929
40	1.001	1.0	0.005	0.066	0.005	0.066	0.006	0.066	0.014	0.423	3.727	17.058	0.003	0.066	0.003	0.066	0.003	0.066	0.010	0.423	2.970	14.054
		1.5	0.009	0.160	0.009	0.160	0.007	0.138	0.008	0.138	2.319	13.293	0.005	0.160	0.005	0.160	0.003	0.078	0.003	0.050	1.824	10.482
		2.0	0.002	0.090	0.002	0.090	0.002	0.090	0.003	0.090	1.774	7.187	0.001	0.065	0.000	0.000	0.000	0.010	0.001	0.043	1.387	4.419
		3.0	0.002	0.107	0.002	0.107	0.001	0.080	0.000	0.007	0.946	3.342	0.001	0.067	0.001	0.067	0.000	0.040	0.000	0.007	0.712	3.068
	1.01	1.0	0.011	0.505	0.011	0.505	0.011	0.505	0.011	0.298	3.246	8.367	0.010	0.505	0.010	0.505	0.010	0.505	0.007	0.168	2.674	7.259
		1.5	0.005	0.173	0.005	0.159	0.005	0.159	0.006	0.159	2.112	11.185	0.004	0.173	0.002	0.094	0.002	0.094	0.003	0.094	1.666	9.510
		2.0	0.004	0.132	0.004	0.132	0.003	0.132	0.002	0.038	1.687	7.004	0.002	0.132	0.003	0.132	0.002	0.132	0.000	0.018	1.304	5.181
		3.0	0.000	0.026	0.000	0.011	0.000	0.011	0.001	0.026	0.887	5.171	0.000	0.026	0.000	0.000	0.000	0.000	0.001	0.026	0.703	4.667
	1.1	1.0	0.012	0.602	0.012	0.602	0.012	0.602	0.008	0.145	3.534	11.991	0.009	0.602	0.009	0.602	0.009	0.602	0.004	0.089	2.851	10.544
		1.5	0.004	0.102	0.004	0.102	0.004	0.057	0.003	0.057	2.394	8.687	0.003	0.102	0.003	0.102	0.002	0.050	0.001	0.029	1.947	8.520
		2.0	0.003	0.068	0.003	0.068	0.003	0.068	0.003	0.069	1.333	4.450	0.001	0.068	0.001	0.068	0.001	0.068	0.002	0.063	1.076	3.986
		3.0	0.000	0.015	0.000	0.015	0.000	0.015	0.000	0.015	0.865	3.247	0.000	0.005	0.000	0.005	0.000	0.002	0.000	0.010	0.656	2.262
60	1.001	1.0	0.004	0.172	0.004	0.172	0.004	0.172	0.005	0.054	3.904	12.667	0.003	0.172	0.003	0.172	0.003	0.172	0.003	0.031	3.495	11.719
		1.5	0.002	0.048	0.002	0.048	0.002	0.048	0.001	0.039	2.308	9.601	0.001	0.033	0.001	0.029	0.001	0.029	0.001	0.024	2.033	9.087
		2.0	0.001	0.035	0.001	0.035	0.001	0.029	0.001	0.016	1.697	5.570	0.001	0.029	0.001	0.029	0.001	0.029	0.000	0.006	1.474	4.936
		3.0	0.000	0.003	0.000	0.003	0.000	0.003	0.000	0.006	1.065	4.780	0.000	0.003	0.000	0.003	0.000	0.003	0.000	0.006	0.913	4.008
	1.01	1.0	0.007	0.081	0.007	0.110	0.005	0.095	0.006	0.095	3.925	11.937	0.003	0.081	0.003	0.110	0.002	0.095	0.003	0.095	3.509	10.545
		1.5	0.002	0.056	0.002	0.056	0.002	0.056	0.003	0.072	2.514	6.924	0.001	0.055	0.001	0.055	0.001	0.055	0.002	0.072	2.239	6.613
		2.0	0.001	0.024	0.001	0.024	0.001	0.024	0.001	0.024	1.667	5.838	0.001	0.020	0.001	0.020	0.001	0.020	0.000	0.016	1.437	5.581
		3.0	0.000	0.012	0.000	0.012	0.000	0.012	0.000	0.009	0.938	3.046	0.000	0.012	0.000	0.012	0.000	0.012	0.000	0.009	0.796	2.619
	1.1	1.0	0.007	0.309	0.008	0.309	0.009	0.309	0.012	0.309	3.671	10.401	0.001	0.035	0.002	0.147	0.003	0.147	0.006	0.191	3.282	10.267
		1.5	0.003	0.093	0.003	0.093	0.003	0.093	0.003	0.087	2.248	12.647	0.001	0.040	0.001	0.040	0.001	0.040	0.002	0.087	1.974	11.558
		2.0	0.002	0.058	0.002	0.058	0.002	0.058	0.001	0.041	1.427	6.922	0.001	0.046	0.001	0.046	0.001	0.046	0.001	0.017	1.249	6.215
		3.0	0.001	0.079	0.001	0.079	0.001	0.082	0.000	0.004	0.979	5.100	0.001	0.079	0.001	0.079	0.001	0.082	0.000	0.004	0.835	3.971
80	1.001	1.0	0.006	0.146	0.005	0.146	0.006	0.146	0.003	0.049	4.274	15.879	0.005	0.146	0.004	0.146	0.005	0.146	0.001	0.037	3.896	13.747
		1.5	0.002	0.070	0.002	0.070	0.002	0.070	0.002	0.070	2.866	13.038	0.001	0.016	0.001	0.016	0.001	0.050	0.001	0.050	2.581	11.932
		2.0	0.001	0.032	0.001	0.032	0.001	0.033	0.001	0.028	1.946	11.680	0.001	0.010	0.001	0.011	0.001	0.014	0.001	0.011	1.731	9.007
		3.0	0.000	0.010	0.000	0.010	0.000	0.010	0.000	0.010	1.122	4.917	0.000	0.007	0.000	0.007	0.000	0.006	0.000	0.002	1.010	4.502
	1.01	1.0	0.003	0.034	0.003	0.032	0.003	0.036	0.004	0.036	4.037	12.813	0.003	0.034	0.002	0.032	0.003	0.036	0.003	0.036	3.703	11.617
		1.5	0.001	0.025	0.001	0.025	0.001	0.025	0.001	0.024	2.484	10.856	0.001	0.025	0.001	0.025	0.001	0.025	0.001	0.024	2.286	10.505
		2.0	0.001	0.041	0.001	0.041	0.001	0.041	0.001	0.033	1.825	6.743	0.001	0.041	0.001	0.041	0.001	0.041	0.000	0.013	1.638	6.366
		3.0	0.000	0.016	0.000	0.016	0.000	0.016	0.001	0.016	1.007	3.874	0.000	0.007	0.000	0.007	0.000	0.007	0.000	0.005	0.916	3.580
	1.1	1.0	0.012	0.639	0.012	0.639	0.013	0.687	0.005	0.092	3.926	13.111	0.010	0.639	0.011	0.639	0.011	0.687	0.004	0.092	3.646	12.037
		1.5	0.002	0.065	0.002	0.065	0.002	0.065	0.001	0.030	2.358	11.001	0.001	0.065	0.001	0.065	0.001	0.065	0.001	0.012	2.177	9.948
		2.0	0.001	0.024	0.001	0.024	0.001	0.024	0.001	0.015	1.672	7.961	0.001	0.024	0.001	0.024	0.000	0.024	0.000	0.006	1.513	7.118
		3.0	0.001	0.015	0.001	0.015	0.001	0.015	0.000	0.011	0.940	3.108	0.000	0.015	0.000	0.015	0.000	0.015	0.000	0.002	0.848	2.912

learning effect and release times. The contributions of this paper were that we developed a branch-and-bound algorithm incorporating with several dominances and two lower bounds to derive the optimal solution, and we also proposed a genetic 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 can solve the instances up to job numbers with less than or equal to 15. Moreover, the computational experiments also showed that the proposed GA_3 algorithm performed quite well.

Acknowledgements

We are grateful to the Editor and the referees for their constructive comments on an earlier version of our paper. This paper was supported by the NSCunder Grant number NSC 99-2221-E-035 -057 -MY3

References

[1] Heiser J, Render B. Operations Management. 5th ed. Englewood Cliffs, NJ: Prentice Hall; 1999.
[2] Russell R, Taylor III BW. Operations Management: multimedia version. 3rd ed. Upper Saddle River, NJ: Prentice Hall; 2000.
[3] Biskup D. Single-machine scheduling with learning considerations. European Journal of Operational Research 1999;115:173–8.
[4] Cheng TCE, Wang G. Single machine scheduling with learning effect considerations. Annals of Operations Research 2000;98:273–90.
[5] Janiak A, Rudek R. The learning effect: getting to the core of the problem. Information Processing Letters 2007;103:183–7.
[6] Bachman A, Janiak A. Scheduling jobs with position-dependent processing times. Journal of the Operational Research Society 2004;55:257–64.
[7] Wang JB. Single-machine scheduling problems with the effects of learning and deterioration. Omega 2007;35(4):397–402.
[8] Cheng TCE, Wu CC, Lee WC. Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects. Information Sciences 2008;178:2476–87.
[9] Wang JB, Ng CT, Cheng TCE, Liu LL. Single-machine scheduling with a time-dependent learning effect. International Journal of Production Economics 2008;111:802–11.
[10] Janiak A, Rudek R. A new approach to the learning effect: beyond the learning curve restrictions. Computers and Operations Research 2008;35:3727–36.
[11] Wang JB, Wang D, Wang LY, Lin L, Yin N, Wang WW. Single machine scheduling with exponential time-dependent learning effect and past-sequence-dependent setup times. Computers and Mathematics with Applications 2009;57:9–16.
[12] Wang JB. Single-machine scheduling with general learning functions. Computers and Mathematics with Applications 2008;56:1941–7.
[13] Wang JB. Single machine scheduling with past-sequence-dependent setup times and time-dependent learning effect. Computers & Industrial Engineering 2008;55(3):584–91.
[14] Sun L. Single-machine scheduling problems with deteriorating jobs and learning effects. Computers & Industrial Engineering 2009;57:843–6.
[15] Yin Y, Xu D, Sun K, Li H. Some scheduling problems with general position-dependent and time-dependent learning effects. Information Sciences 2009;179:2416–25.
[16] Wu CC, Lee WC. Single-machine and flowshop scheduling with a general learning effect model. Computers & Industrial Engineering 2009;56(4):1553–8.
[17] Toksari MD, Oron D, Güner E. Single machine scheduling problems under the effects of nonlinear deterioration and time-dependent learning. Mathematical and Computer Modelling 2009;50:401–6.
[18] Zhang X, Yan G. Machine scheduling problems with a general learning effect. Mathematical and Computer Modelling 2010;51:84–90.
[19] Huang X, Wang JB, Wang LY, Gao WJ, Wang XR. Single machine scheduling with time-dependent deterioration and exponential learning effect. Computers & Industrial Engineering 2010;58:58–63.
[20] Wang JB. Single machine scheduling with a time-dependent learning effect and deteriorating jobs. Journal of the Operational Research Society 2009;60:583–6.
[21] Toksari MD, Isleyen SK, Güner E, Baykoç ÖF. Assembly line balancing problem with deterioration tasks and learning effect. Expert Systems with Applications 2010;37:1223–8.
[22] Wang JB, Liu LL. Two-machine flow shop problem with effects of deterioration and learning. Computers & Industrial Engineering 2009;57:1114–21.
[23] Biskup D. A state-of-the-art review on scheduling with learning effect. European Journal of Operational Research 2008;188:315–29.
[24] Wang JB, Guo Q. A due-date assignment problem with learning effect and deteriorating jobs. Applied Mathematical Modelling 2010;34:309–13.
[25] Janiak A, Janiak WA, Rudek R, Wielgus A. Solution algorithms for the makespan minimization problem with the general learning model. Computers & Industrial Engineering 2009;56:1301–8.
[26] Wang JB. Single-machine scheduling with a sum-of-actual-processing-time-based learning effect. Journal of the Operational Research Society 2010;61:172–7.
[27] Wang JB, Sun L, Sun L. Single machine scheduling with exponential sum-of-logarithm-processing-times based learning effect. Applied Mathematical Modelling 2010;34:2813–9.
[28] Janiak A, Rudek R. Experience-based approach to scheduling problems with the learning effect. IEEE Transactions on Systems, Man, and Cybernetics—Part A 2009;39:344–57.
[29] Janiak A, Rudek R. A note on a makespan minimization problem with a multi-ability learning effect. Omega 2010;38:213–7.
[30] Lee WC, Wu CC, Hsu PH. A single-machine learning effect scheduling problem with release times. Omega: The International Journal of Management Science 2010;38:3–11.
[31] Eren T. Minimizing the total weighted completion time on a single machine scheduling with release dates and a learning effect. Applied Mathematics and Computation 2009;208:355–8.
[32] Lee WC, Wu CC, Liu MF. A single-machine bi-criterion learning scheduling problem with release times. Expert Systems with Applications 2009;36:10295–303.
[33] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1977;1:343–62.
[34] Hardy GH, Littlewood JE, Polya G. Inequalities. London: Cambridge University Press; 1967.
[35] Beasley D, Bull D, Martin RR. An overview of genetic algorithms, part 1: fundamentals. Journal of University Computing 1993;15:58–69.
[36] Iyer SK, Saxena BS. Improved genetic algorithm for the permutation flowshop scheduling problem. Computers and Operations Research 2004;31:593–606.
[37] Chen JS, Pan JCH, Lin CM. A hybrid genetic algorithm for the reentrant flowshop scheduling problem. Expert Systems with Applications 2008;34:570–7.
[38] Chou FD. An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem. Expert Systems with Applications 2009;36:3857–65.
[39] Ugur A. Path planning on a cuboid using genetic algorithms. Information Sciences 2008;178(16):3275–87.
[40] Darwin CR. On the Origin of Species through Natural Selection. London: John Murray; 1859.
[41] Holland J. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press; 1975.
[42] Essafi I, Matib Y, Dauzere-Peres S. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers and Operations Research 2008;35:2599–616.
[43] Etiler O, Toklu B, Atak M, Wilson J. A generic algorithm for flow shop scheduling problems. Journal of Operations Research Society 2004;55(8):830–5.
[44] Bean JC. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal of Computing 1994;6:154–60.
[45] Reeves C. Heuristics for scheduling a single machine subject to unequal job release times. European Journal of Operational Research 1995;80:397–403.
[46] Chen CL, Vempati VS, Aljaber N. An application of genetic algorithms for flow shop problems. European Journal of Operations Research 1995;80:389–96.
[47] Falkenauer E, Bouffoix S. A genetic algorithm for job shop. In: Proceedings of the 1991 IEEE International Conference on Robotics and Automation, 1991.
[48] Etiler O, Toklu B. Comparison of genetic crossover operators using in scheduling problems. Journal of the Institute of Technology, Gazi University, Turkey 2001;14:21–32.
[49] Chu CB. A branch-and-bound algorithm to minimize total flow time with unequal release dates. Naval Research Logistics 1992;39:859–75.