Information Sciences 269 (2014) 142–158

Contents lists available at ScienceDirect

Information Sciences

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

Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects

Wei-Chang Yeh [a], Peng-Jen Lai [b], Wen-Chiung Lee [c], Mei-Chi Chuang [a,*]

[a] Integration and Collaboration Laboratory, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, P.O. Box 24-60, Hsinchu 300, Taiwan, ROC
[b] Department of Mathematics, National Kaohsiung Normal University, Kaohsiung 824, Taiwan, ROC
[c] Department of Statistics, Feng Chia University, Taichung, Taiwan

ARTICLE INFO

Article history:
Received 18 September 2012
Received in revised form 23 September 2013
Accepted 19 October 2013
Available online 6 November 2013

Keywords:
Scheduling
Learning effect
Parallel-machine
Makespan
Fuzzy number
Possibility measure

ABSTRACT

This paper addresses parallel machine scheduling with learning effects. The objective is to minimize the makespan. To satisfy reality, we consider the processing times as fuzzy numbers. To the best of our knowledge, scheduling with learning effects and fuzzy processing times on parallel machines has never been studied. The possibility measure will be used to rank the fuzzy numbers. Two heuristic algorithms, the simulated annealing algorithm and the genetic algorithm, are proposed. Computational experiments have been conducted to evaluate their performance.

1. Introduction

Traditionally, job processing times are assumed to be fixed and known over the entire process. In reality, the job processing times in many situations are not fully known in advance, and the efficiency of the production line improves continuously as firms produce more of a product and gain knowledge or experience. As a result, the production time of a given product is shorter if it is processed later. This phenomenon is known in literature as the "learning effect". Bachman and Janiak [2], Biskup [6], and Janiak and Rudek [19] have provided comprehensive reviews of scheduling models and problems with learning effects. For more recent papers, readers can refer to Lai and Lee [23], Lee [26], Zhang et al. [53], Cheng et al. [8], Kuo et al. [21], Lee and Wu [28], Yin et al. [47], Yin et al. [48], Rudek [38], Wang et al. [43], and Cheng et al. [9].

Most of the research on scheduling with learning effects focuses on the single machine environment and studies on parallel machines are relatively unexplored. Pinedo [36] noted that a bank of machines in parallel provides a setting that is important from both a theoretical and a practical point of view. From a theoretical viewpoint, a bank of machines in parallel is a generalization of the single machine and a special case of the flexible flowshop. From a practical point of view, a bank of machines in parallel is important because the occurrence of resources in parallel is very common in the real world. Techniques for machines in parallel are often used in decomposition procedures for multistage systems. In addition, many practical job shop and open shop scheduling problems can be treated as parallel machine scheduling problems under certain

* Corresponding author. Tel.: +886 3 5742443; fax: +886 3 572 2204.
E-mail address: d9734801@oz.nthu.edu.tw (M.-C. Chuang).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.10.023

constraints [13,15,31,4]. The parallel machine environment exists not only in manufacturing but also has applications in the service sector or in information systems, i.e., parallel information processing and distributed computing.

To the best of our knowledge, Mosheiov [33] was the first author to study the learning effect on parallel machines. He showed that the total completion time problem on parallel identical machines is polynomially solvable under Biskup's [5] position-based learning effect model. Eren [11] studied a bi-criterion identical parallel machines scheduling problem with a learning effect of setup times and removal times. The objective was to minimize the weighted sum of total completion time and total tardiness. He provided a mathematical programming model to solve problems with up to 15 jobs and five machines and three heuristic approaches to solve problems with large numbers of jobs. Toksari and Guner [42] considered a parallel machine earliness/tardiness (ET) scheduling problem with different penalties based on the effects of position-based learning and linear or nonlinear deterioration. They showed that the optimal solution for the ET scheduling problem under effects of learning and deterioration is a V-shaped schedule under certain agreeable conditions. Furthermore, they developed a mathematical model, an algorithm and a lower bound procedure for problems with large numbers of jobs. Okołowski and Gawiejnowicz [34] considered a parallel-machine makespan problem under the general DeJong's learning curve. For the NP-hard problem, they proposed two exact algorithms, a sequential branch-and-bound algorithm and a parallel branch-and-bound algorithm. Hsu et al. [16] investigated an unrelated parallel-machine scheduling problem with past-sequence-dependent setup time and learning effects. They derived a polynomial time solution for the total completion time problem. Kuo et al. [22] studied unrelated parallel-machine scheduling problems with setup time and learning effects. The setup time is proportional to the length of the jobs already processed, and the objectives are to minimize the total absolute deviation of job completion times and the total load on all machines. They showed that both problems are polynomially solvable. Yang et al. [46] considered the parallel-machine scheduling problem with aging effects and multiple-maintenance activities. The objective was to find the optimal maintenance frequencies, the optimal positions of the maintenance activities, and the optimal job sequences. They provided an efficient algorithm to solve the problem when the maintenance frequencies on the machines were given.

In the literature on scheduling with the learning effect, it is assumed that the job processing time is a known constant which will be shortened due to the learning effect. However, there are situations in which the job processing times might not be fully known in advance. In many practical industrial applications, the data are usually recorded or collected under the influence of some unexpected noise. As a result, the processing time of a job cannot be measured precisely, and fuzzy set theory provides a convenient alternative framework for modeling real-world systems mathematically. Balin [4] noted several advantages of fuzzy numbers. For instance, fuzzy theory provides an efficient way to model imprecision [1]. Using fuzzy set theory decreases the computational complexity of the scheduling problem compared to the stochastic-probability theory [3]. Fuzzy theory enables the use of fuzzy rules in heuristic algorithms [39].

In practical industrial applications, the data are usually recorded or collected under the influence of unexpected noise, so the processing time of a job often cannot be measured precisely. In this event, perhaps a more reasonable statement is that "the processing time is approximately 2 h". The better way to model the phrase "approximately 2 h" would be to invoke a fuzzy number $\tilde{2}$. In this paper, we shall use the possibility and necessity frameworks and ranking concept proposed by Dubois and Prade [10], thus enabling us to optimize the objective functions as fuzzy makespans or total weighted fuzzy completion times.

Fuzzy scheduling problems have been explored for more than two decades. Though the possibility and necessity concept was proposed early by Dubois and Prade [10], the history of applying the ranking method based on possibility or necessity measure to fuzzy scheduling problems is shorter. To the best of our knowledge, Itoh and Ishii [18] made an early attempt to use the possibility measure to propose the concept of $\lambda - P$ tardiness. Chanas and Kasperski [7] later investigated the single machine fuzzy scheduling problem based on possibility and necessity indices. We refer to Lai and Wu [24] for details of other works. Peng and Liu [35] proposed a hybrid intelligent algorithm to solve three parallel machine scheduling problems with fuzzy processing times. Tavakkoli-Moghaddam et al. [41] proposed a fuzzy multi-objective linear programming problem to minimize the total weighted tardiness and makespan simultaneously in a fuzzy environment. Balin [4] proposed a genetic algorithm for minimizing the makespan of parallel machines where processing times are fuzzy. Huang et al. [17] considered a fuzzy time-dependent project scheduling problem and proposed three fuzzy programming models to address different situations of decision–making encountered in practice.

In nature, many factors are often imprecise or uncertain in real-world scheduling problems. Fuzzy set theory is suitable to address the uncertainties, and it offers a convenient framework for modeling real-world production systems mathematically. Moreover, the actual job processing time may be shortened due to the repetition of similar tasks. The classification of related work is given in Table 1. To best of our knowledge, the fuzzy set theory and the learning effect have never been studied simultaneously, especially in the parallel machine environment. In this paper, we study a parallel machine scheduling problem with learning effects and fuzzy processing times. The objective is to minimize the fuzzy makespan. We adopted

Table 1
Literature review about the learning effect and fuzzy number.

	Single machine	Parallel machine	Others
Learning effect	[2,5,8,9,19,23,38,47]	[11,16,22,33,34] [42,46]	[6,12,21,26,28,43,48,53]
Fuzzy number	[7,18,41]	[1,4,35]	[3,17,24,25,39]

the possibility frameworks proposed by Dubois and Prade [10], a ranking method based on possibility measure for fuzzy scheduling problems. We assume that the fuzzy processing time is in a trapezoidal shape, from which we derived an analytic recursive formula to calculate fuzzy completion times. Finally, we proposed a simulated annealing algorithm and genetic algorithm to solve the problem. The rest of the paper is organized as follows. In Section 2, we present basic terminologies of fuzzy numbers. In Section 3, the problem is described, and the method for calculating fuzzy completion times is introduced. In Section 4, heuristic algorithms are proposed to search for near-optimal solutions. In Section 5, numerical examples are provided. The conclusion is presented in the final section.

2. Preliminaries

In this section, we introduce a number of basic prerequisites for fuzzy numbers. For more details on the topic of fuzzy sets theory, we refer readers to Zimmermann [54] and Lai and Wu [24]. The fuzzy subset $\tilde{a}$ of R is defined by a function $\xi_{\tilde{a}}:R \to [0,1]$, which is called a membership function. The α-level set $\tilde{a}$, denoted by $\tilde{a}_\alpha = \{x \in R : \xi_{\tilde{a}}(x) \geqslant \alpha\}$ for all $\alpha \in (0,1]$. The 0-level set $\tilde{a}_0$ is defined as the closure of the set $\{x \in R : \xi_{\tilde{a}}(x) \geqslant \alpha\}$, i.e., $\tilde{a}_0 = cl(\{x \in R : \xi_{\tilde{a}}(x) \geqslant 0\}) = cl(\cup_{\alpha>0}\tilde{a}_\alpha)$.

The fuzzy subset $\tilde{a}$ of R is said to be a fuzzy number if the following conditions are satisfied:

(i) $\tilde{a}$ is normal, i.e., there exists an $x \in R$ such that $\xi_{\tilde{a}}(x) = 1$.
(ii) ξ_a is quasi-concave, i.e., $\xi_a(tx + (1-t)y) \geqslant \min\{\xi_a(x), \xi_a(y)\}$ for $t \in [0,1]$.
(iii) ξ_a is upper semi-continuous, i.e., $\{x \in R : \xi_{\tilde{a}}(x) \geqslant \alpha\}$ is a closed subset of R for each $\alpha \in (0,1]$.
(iv) The 0-level set $\tilde{a}_0$ is a closed and bounded subset of R.

Because $\tilde{a}_\alpha \subset \tilde{a}_0$ for each $\alpha \in (0,1]$, condition (iv) implies that the α-level sets $\tilde{a}$ are bounded subsets of R for all $\alpha \in (0,1]$. In addition, condition (ii) implies that the α-level set $\tilde{a}$ is a convex subset of R. We denote the set of all fuzzy numbers as $F(R)$. From conditions (i)–(iv), we see that if $\tilde{a} \in F(R)$, then the α-level set $\tilde{a}$ is a closed, bounded and convex subset of R, i.e., a closed interval in R. Therefore, we denote it as $\tilde{a}_\alpha = [\tilde{a}_\alpha^L, \tilde{a}_\alpha^U]$.

Let "$\odot$" be a binary operation $\oplus$, and Θ or $\otimes$ be between two fuzzy numbers $\tilde{a}$ and $\tilde{b}$. Using the Extension Principle [49–51], the membership function of $\tilde{a} \odot \tilde{b}$ is defined by

$$\xi_{\tilde{a}\odot\tilde{b}}(z) = \sup_{x \circ y = z} \min\{\xi_{\tilde{a}}(x), \xi_{\tilde{b}}(y)\} \tag{1}$$

for $\odot = \oplus$, Θ or $\otimes$ correspond to $\circ = +$, $-$ or $\times$. Using the Extension Principle again, the membership function of the maximum $\max\{\tilde{a}, \tilde{b}\}$ of $\tilde{a}$ and $\tilde{b}$ is defined by

$$\xi_{\max\{\tilde{a},\tilde{b}\}}(z) = \sup_{\max\{x,y\}=z} \min\{\xi_{\tilde{a}}(x), \xi_{\tilde{b}}(y)\} \tag{2}$$

Thus, we have the following well-known results (e.g., see Chanas and Kasperski [7]).

Proposition 1. *Let $\tilde{a}$ and $\tilde{b}$ be two fuzzy numbers. Then $\tilde{a} \oplus \tilde{b}$, $\tilde{a}\Theta\tilde{b}$, and $\tilde{a} \otimes \tilde{b}$ are also fuzzy numbers. Furthermore, we have*

$$(\tilde{a} \oplus \tilde{b})_\alpha = \left[\tilde{a}_\alpha^L + \tilde{b}_\alpha^L, \tilde{a}_\alpha^U + \tilde{b}_\alpha^U\right]$$

$$(\tilde{a}\Theta\tilde{b})_\alpha = \left[\tilde{a}_\alpha^L - \tilde{b}_\alpha^L, \tilde{a}_\alpha^U - \tilde{b}_\alpha^U\right]$$

$$(\tilde{a} \otimes \tilde{b})_\alpha = \left[\min\left\{\tilde{a}_\alpha^L\tilde{b}_\alpha^L, \tilde{a}_\alpha^L\tilde{b}_\alpha^U, \tilde{a}_\alpha^U\tilde{b}_\alpha^L, \tilde{a}_\alpha^U\tilde{b}_\alpha^U\right\}, \max\left\{\tilde{a}_\alpha^L\tilde{b}_\alpha^L, \tilde{a}_\alpha^L\tilde{b}_\alpha^U, \tilde{a}_\alpha^U\tilde{b}_\alpha^L, \tilde{a}_\alpha^U\tilde{b}_\alpha^U\right\}\right]$$

$$(\max\{\tilde{a}, \tilde{b}\})_\alpha = \left[\max\left\{\tilde{a}_\alpha^L\tilde{b}_\alpha^L\right\}, \max\left\{\tilde{a}_\alpha^U\tilde{b}_\alpha^U\right\}\right]$$

The trapezoidal fuzzy numbers are considered frequently for the application of the fuzzy number theory; hence, we assume that the fuzzy processing time is in a trapezoidal shape in this paper. We define a trapezoidal fuzzy number $\tilde{a}$ as (a^L, a_1, a_2, a^U), and the membership function of $\tilde{a}$ is given by

$$\xi_{\tilde{a}}(r) = \begin{cases} (r - a^L)/(a_1 - a^L) & \text{if } a^L \leqslant r \leqslant a_1, \\ 1 & \text{if } a_1 \leqslant r \leqslant a_2, \\ (a^U - r)/(a^U - a_2) & \text{if } a_2 \leqslant r \leqslant a^U, \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

We can see that $\tilde{a}_\alpha^L = (1-\alpha)a^L + \alpha a_1$ and $\tilde{a}_\alpha^U = (1-\alpha)a^U + \alpha a_2$.

Proposition 2. *Let $\tilde{a} = (a^L, a_1, a_2, a^U)$ and $\tilde{b} = (b^L, b_1, b_2, b^U)$ be two trapezoidal fuzzy numbers. It is known that $\tilde{a} \oplus \tilde{b}$ is also a trapezoidal fuzzy number. Moreover, we have*

$$\tilde{a} \oplus \tilde{b} = (a^L + b^L, a_1 + b_1, a_2 + b_2, a^U + b^U) \tag{4}$$

Now, we introduce the ranking concept based on the possibility indices proposed by Dubois and Prade [10]. Let $\tilde{a}$ and $\tilde{b}$ be two fuzzy numbers. Dubois and Prade [10] defined the following measure

$$Poss(\tilde{a} \succeq \tilde{b}) = \sup_{\{u,v;u \geqslant v\}} \min\{\xi_{\tilde{a}}(u), \xi_{\tilde{b}}(v)\} \tag{5}$$

If the value $Poss(\tilde{a} \succeq \tilde{b})$ is large, it means that the extent of the possibility of $\tilde{a}$ being larger than $\tilde{b}$ is greater. Otherwise, if the value $Poss(\tilde{a} \succeq \tilde{b})$ is small, then the strength of believing $\tilde{a}$ is larger than $\tilde{b}$ is weak. Therefore, if this possibility measure is considered as the degree of the fuzzy minimum makespan, we are going to minimize the following objective function

$$\min_{\pi \in \Pi} f(\pi) = Poss(\widetilde{C}_{\max} \succeq \tilde{c}_p) \tag{6}$$

where $\tilde{c}_p$ is a pre-determined fuzzy number used to measure the extent of the fuzzy minimum makespan and can be regarded as the reference value, Π is the set of all schedules and π is one of the schedules in Π. First, we need to calculate the $\widetilde{C}_j$ of every machine j, and calculate $Poss(\widetilde{C}_j \succeq \tilde{c}_p)$ of every machine j. We select the largest value among all the possibility measures as the $Poss(\widetilde{C}_{\max} \succeq \tilde{c}_p)$.

We use the following exact formulas proposed by Lai and Wu [24] to simplify the coding for calculating the possibility measure.

Proposition 3. *Let $\tilde{a} = (a^L, a_1, a_2, a^U)$ and $\tilde{b} = (b^L, b_1, b_2, b^U)$ be two trapezoidal fuzzy numbers. Then, the following statements hold.*

(i) *If $a_2 \geqslant b_1$ then $Poss(\tilde{a} \succeq \tilde{b}) = 1$.*
(ii) *If $a_2 < b_1$ and $(b_1 - b^L) + (a^U - a_2) \neq 0$ then*

$$Poss(\tilde{a} \succeq \tilde{b}) = \max\left\{\frac{a^U - b^L}{(b_1 - b^L) + (a^U - a_2)}, 0\right\}$$

(iii) *If $a_2 < b_1$ and $(b_1 - b^L) + (a^U - a_2) = 0$ then $Poss(\tilde{a} \succeq \tilde{b}) = 0$.*

3. Problem formulation

The problem under study is given as follows. There are n jobs to be scheduled on m uniform parallel machines. All of the jobs are available for processing at time zero and can be processed on any machine j. Each machine can process one job at a time, and there is no precedence relation between jobs. For job i, the processing times are given as trapezoidal fuzzy numbers and denoted as $\tilde{p}_i = \left(p_i^L, p_i^{(1)}, p_i^{(2)}, p_i^U\right)$. The speed of machine j is denoted as s_j. Due to the learning effect [5], the actual processing time of job i scheduled in the rth position on machine j is

$$_j\tilde{p}_{ir} = \tilde{p}_i r^a / s_j \tag{7}$$

where $\tilde{p}_i$ denotes the processing time of job i, r denotes the position, $a \leqslant 0$ is the learning index, and s_j denotes the speed of machine j. The objective is to find a schedule in which the fuzzy makespan is minimized.

The fuzzy completion time $\widetilde{C}_j$ of every machine can be obtained through the summation of the fuzzy processing times multiplied by the inverse of s_j.

$$\widetilde{C}_j = \left(\sum_{i=1}^{n_j} \tilde{p}_{J_i r}\right)\frac{1}{s_j} = \left(\sum_{i=1}^{n_j} \tilde{p}_{J_i} r^a\right)\frac{1}{s_j} \quad \text{for} \quad j = 1, \ldots, m. \tag{8}$$

The fuzzy completion times $\widetilde{C}_j$ will be trapezoidal fuzzy numbers, which are denoted by

$$\widetilde{C}_j = \left(C_j^L, C_j^{(1)}, C_j^{(2)}, C_j^U\right) \tag{9}$$

where C_j^* can be calculated according to the following formulas:

$$C_j^K = \left(\sum_{i=1}^{n_j} \tilde{p}_{J_i}^K r^a\right)\frac{1}{s_j} \quad \text{for} \quad j = 1, \ldots, m \text{ and } K = L, (1), (2), U. \tag{10}$$

We explain how to calculate the fuzzy makespan and possibility measure in the following example. A problem with three jobs and three machines is considered in Table 2. Because we assume that the processing time is a trapezoidal fuzzy number, $\tilde{c}_p$ can be regarded as $(0, c_p, c_p, \infty)$, and c_p can be regarded as the total of the p_i^U of all jobs. The problem, therefore, has twenty-seven combinations. If we assume that job 1 is assigned to machine 1, job 3 is assigned to machine 2, and job 2 is assigned to machine 3, we can obtain the fuzzy completion time and the possibility measure as shown in Table 3. Finally, the optimal solution for this problem is given in Table 4.

4. Algorithms

The makespan problem under study is NP hard even for the deterministic processing time case. Thus, it is difficult to achieve an optimal solution with traditional optimization methods [4]. Many heuristic methods, such as Simulated Annealing (SA), Tabu Search, Genetic Algorithm (GA), Particle Swarm Optimization, Neural Network, and others, have been proposed for combination optimization problems. Among these novel soft computing methods, SA and GA have been well documented in the literature and have been applied to a wide variety of optimization problems [4]. In this paper, we will utilize these two methods for the problem under study.

4.1. Simulated annealing algorithm

Simulated annealing (SA) algorithm has become one of the most popular meta-heuristic methods and has been applied widely to solve many combinatorial optimization problems since the introduction by Kirkpatrick et al. [20]. SA has the advantage of avoiding being trapped in a local optimal. For the recent excellent performance of SA, we refer readers to Figielska [12], Lee et al. [27], Ribeiro et al. [37], and Mosadegh et al. [32]. SA begins with an initial solution. At each iteration, SA takes a solution from the neighborhood solutions of the current solution. If the candidate solution is better than the current one, the candidate solution is accepted as the current solution; otherwise, SA calculates a probability value to determine whether SA will accept the candidate solution. The worst solutions with a certain probability can therefore escape from a local optimal. SA utilizes a parameter called temperature to determine the probability value. The probability of accepting a worse solution is larger in the beginning, and the probability decreases in later iterations when the temperature decreases. The procedures of the SA implemented to the problem are outlined below.

4.1.1. Solution representation

For a problem with n jobs and m machines, the length of each solution is n. The solution is represented by a string of real numbers. The integer part of each real number is generated from $(1, m)$ and a fraction part is generated from $(0, 1)$. The integer part is interpreted as the machine assignment. The job sequence for each machine is obtained by sorting the fractional

Table 2
Numerical example.

Learning index a	$a = -0.322$
The reference value $\tilde{c}_p$	$(0, 245, 245, \infty)$
The speed of machine s_j	Machine 1: 2; Machine 2: 1.5; Machine 3: 1
Fuzzy processing time $\tilde{p}_i$	Job 1: (15, 31, 52, 82) Job 2: (23, 41, 58, 82) Job 3: (24, 25, 62, 81)

Table 3
Results of numerical example.

$\tilde{c}_i$	$\tilde{c}_1 = (15, 31, 52, 82) \times \frac{1}{2} = (7.50, 15.50, 26.00, 41.00)$
	$\tilde{c}_2 = (24, 25, 62, 81) \times \frac{1}{1.5} = (16.00, 16.67, 41.33, 54.00)$
	$\tilde{c}_3 = (23, 41, 58, 82) \times \frac{1}{1} = (23, 41, 58, 82)$
Poss	$Poss(\tilde{c}_1 \succeq \tilde{c}_p) = \frac{(41-0)}{(245-0+41-26)} = 0.1577$
	$Poss(\tilde{c}_2 \succeq \tilde{c}_p) = \frac{(54-0)}{(245-0+54-41.33)} = 0.2096$
	$Poss(\tilde{c}_3 \succeq \tilde{c}_p) = \frac{(82-0)}{(245-0+82-58)} = 0.3048$

Table 4
The best result of the example.

The best sequence	Machine 1: Job 1 → Job 2; Machine 2: Job 3
The best possibility measure	0.2737

parts. For example, a solution of a problem with 6 jobs and 3 machines is shown in Fig. 1. The schedule corresponding to this solution is jobs 3 and 6 assigned to machine 1, jobs 1 and 4 assigned to machine 2, jobs 2 and 5 assigned to machine 3. As shown in Fig. 2, sorting the genes for machine 1 in ascending order results in the job sequence 6 → 3, the job sequence 1 → 4 for machine 2, and the job sequence 5 → 2 for machine 3.

4.1.2. Generation of the initial solution

Generate an initial solution at random.

4.1.3. Calculation of the objective value

First, we calculate the fuzzy completion time of the last job on each machine according to Eq. (8). Second, we calculate the objective value according to Eq. (6).

4.1.4. Neighborhood generation

Given a solution, the neighbor of the current solution can be obtained by the method shown in Fig. 3. We can obtain a neighbor of the solution by choosing four positions at random and can alter their current value according to upper and lower bounds. In small and large job-sized problems, we also choose four positions to alter their values.

We calculate the objective value of the candidate solution. Let Z_{cur} be the value of the objective function for the current solution and Z_{new} be the value of the objective function for the new candidate. If $Z_{new} \leqslant Z_{cur}$, then the current candidate is accepted as the current solution. If $Z_{new} > Z_{cur}$, then the new sequence is accepted with probability

$$\text{Prob}_{accept} = \exp\left(\frac{-(Z_{new} - Z_{cur})}{T_t}\right) \tag{11}$$

where t is the iteration number and T_t is the current temperature.

4.1.5. Update the temperature

SA starts with an initial temperature $T_1 = T_{\max}$, and the temperature decreases according to Eq. (12) as proposed by Maulik and Mukhopadhyay [30]

$$T_t = T_{\max} 0.9^t \tag{12}$$

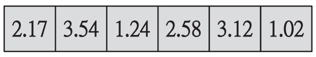

Fig. 1. Chromosome representation.

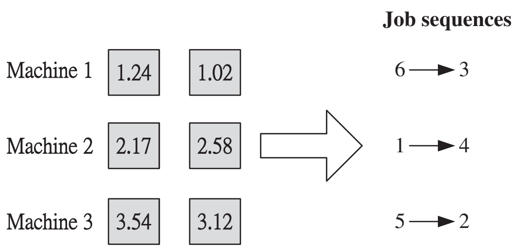

Fig. 2. Job sequences.

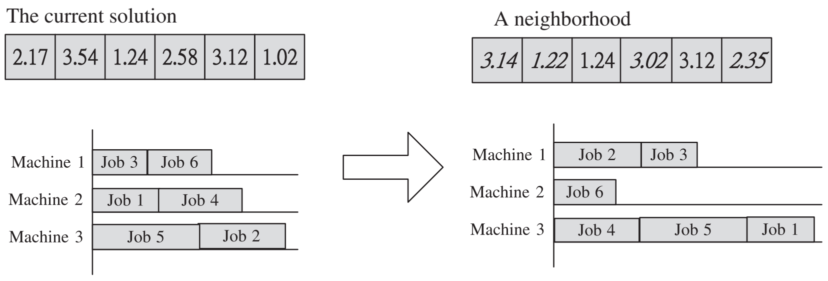

Fig. 3. Neighborhood structure.

4.1.6. Stopping rule

If the maximal generation is reached, then stop; otherwise, go to 4.1.3 to perform another iteration.

4.2. Genetic algorithms

The genetic algorithm (GA) has received considerable attention for its potential as an optimization technique for many complex problems. GA has been applied successfully in the area of industrial engineering, including scheduling and sequencing, reliability design, vehicle routing, facility layout and location, transportation, and other areas. We refer readers to Balin [4], Yan et al. [45], Zhang and Cheng [52], Xu and Zhou [44], and Lin [29].

The usual form of GA was described by Goldberg [14]. The genetic algorithm (GA) starts with an initial population of random solutions. Each chromosome in the population represents a solution to the problem. Chromosomes evolve through successive generations. In each generation, chromosomes are evaluated by a fitness value. Chromosomes with smaller fitness values have a higher probability of being selected. Offspring are produced through the crossover and mutation operations. The crossover operator merges two chromosomes to create offspring that possess certain features from their parents. The mutation operator produces a spontaneous random change in genes to prevent premature convergence. The procedures of the GA applied to the problem are outlined below.

4.2.1. Chromosome representation

The chromosome representation of GA is the same as SA.

4.2.2. Generation of initial population

Generate an initial population of N chromosomes at random.

4.2.3. Calculation of objective value

The fitness value of each chromosome is calculated as in Section 4.1.3.

4.2.4. Selection

We use the roulette wheel method to select the parent chromosomes i with probability p_i

$$p_i = f_i \Big/ \sum_{i=1}^{n} f_i \tag{13}$$

$$f_i = 1/fitness_i \tag{14}$$

where $fitness_i$ is the fitness value of chromosome i.

4.2.5. Crossover

In this study, we use the parameterized uniform crossover proposed by Spears and DeJong [40] that uniformly randomly selects the value of each gene from its parents' genes to form the offspring. As shown in Fig. 4, the resulting offspring would be (1.89, 3.54, 2.75, 2.58, 3.12, 2.07) if the 2nd, 4th, and 5th genes are randomly selected from parent 1, and the other genes are from parent 2.

4.2.6. Mutation

The mutation mechanism randomly selects a gene according to the mutation rate and alters its value. As shown in Fig. 5, the fourth gene is randomly selected and altered to 1.06.

4.2.7. Stopping rule

If the maximal generation is reached, then stop; otherwise, go to 4.2.3 to perform the next iteration.

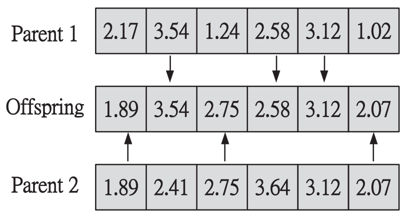

Fig. 4. Uniform crossover.

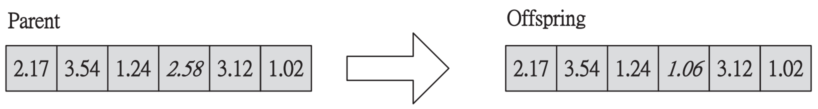

Fig. 5. Mutation mechanism.

5. Numerical examples

To evaluate the performance of the proposed algorithms, a computational experiment is conducted in this section. All of the algorithms are coded in C and run on Intel Core version i3 on a personal computer with 2.94 GHz CPU and 1.9 GB RAM in Windows XP.

The computational experiment consists of three parts. In the first part of the experiment, the number of jobs is set at two levels, namely, four and seven. The number of machines (m) is set at two levels, namely, two and three. All of the fuzzy processing times are assumed to be trapezoidal in shape. In addition, $p_i^L,\ p_i^{(1)},\ p_i^{(2)}$, and p_i^U are generated from four discrete uniform distributions, namely $U(15, 24)$, $U(25, 44)$, $U(45, 64)$ and $U(65, 84)$. The speeds of the machines (s_i) are generated from two continuous uniform distributions, namely, $U(1, 5)$ and $U(1, 10)$. Three patterns of learning effects are studied, they are $a = -0.152$, $a = -0.322$, and $a = -0.515$, respectively. A set of 100 instances is randomly generated for each situation. The same set of instances is used to test the performance of the proposed algorithms. As a result, 24 experimental cases are conducted, and a total of 2400 instances are tested. The parameter settings of two algorithms are given in Table 5, and we also use the same parameter settings in the second part of the experiment. The α-level sets of fuzzy completion times can be obtained using the formulas from Section 3. $\tilde{c}_p = (0, c_p, c_p, \infty)$ is a pre-determined trapezoidal fuzzy number used to measure the extent of the fuzzy minimum makespan, and we assume c_p is the total of the p_i^U of all jobs. The mean and the maximum error percentages were reported for each heuristic. For instance, the error percentage of SA is calculated as

Table 5
Parameter settings of SA and GA for small job-sized problems.

	GA	SA
Parameter setting	Iteration number is 100 Population size is 80 Mutation rate is 0.02	Iteration number is 4000 $T_{\max}$ is 100

Table 6
The mean and maximum error percentages for the SA and GA.

s_i	m	n	LE	SA			GA		
				Error percentages		Optimal count	Error percentages		Optimal count
				Mean	Max		Mean	Max	
U(1,5)	2	4	1	0.00	0.00	100	0.00	0.00	100
			2	0.00	0.00	100	0.00	0.00	100
			3	0.00	0.00	100	0.00	0.00	100
		7	1	0.02	0.96	92	0.04	0.57	48
			2	0.06	2.68	90	0.06	0.73	54
			3	0.03	0.64	90	0.09	0.62	38
	3	4	1	0.00	0.00	100	0.00	0.31	98
			2	0.00	0.00	100	0.00	0.00	100
			3	0.00	0.00	100	0.00	0.10	99
		7	1	0.04	1.15	92	0.21	1.80	46
			2	0.05	2.86	90	0.28	3.49	42
			3	0.07	1.15	84	0.29	2.22	36
U(1,10)	2	4	1	0.00	0.00	100	0.00	0.00	100
			2	0.00	0.00	100	0.00	0.00	100
			3	0.00	0.00	100	0.00	0.00	100
		7	1	0.10	4.27	94	0.03	0.29	48
			2	0.06	4.28	88	0.06	0.62	40
			3	0.05	2.73	91	0.08	0.77	41
	3	4	1	0.21	12.87	98	0.01	0.65	98
			2	0.00	0.00	100	0.00	0.00	100
			3	0.00	0.00	100	0.00	0.21	99
		7	1	0.04	2.44	91	0.13	2.27	42
			2	0.10	2.88	80	0.21	2.12	42
			3	0.06	2.45	89	0.26	1.88	31

$$(SA - OPT)/OPT \times 100\% \tag{15}$$

where *SA* is the value of the objective function generated by the simulated algorithm, and *OPT* is the value from the branch and bound algorithm. In addition, the numbers of times that the heuristics yield the optimal solutions are also reported. The results were presented in Table 6. For GA, the mean error percentage reaches 0.29, and the maximum error percentage reaches 2.27 in the worst case. GA only yields the optimal solution for 70.92% out of the 2400 instances. However, SA can yield the optimal solution for 94.54%. The mean error percentage reaches 0.21, and the maximum error percentage reaches 12.87 in the worst case. Although the maximum error percentage in the worst case when using SA is larger than the maximum error percentage of GA, the SA solutions are very close to the optimal solutions for most of the cases.

In the second part of the experiment, the number of jobs is set at two levels, namely, 10 and 20. The number of machines (m) is set at two levels, namely, two and three. A set of 100 instances is randomly generated for each situation. The same set of instances is used to test the performance of the proposed algorithms. As a result, 24 experimental cases have been conducted, and a total of 2400 instances were tested. We use the relative deviations (Ratio1 and Ratio2) to evaluate their performance as shown in the following

$$\text{Ratio 1} = \frac{SA - \min(SA, GA)}{\min(SA, GA)} \times 100\% \tag{15}$$

$$\text{Ratio 2} = \frac{GA - \min(SA, GA)}{\min(SA, GA)} \times 100\% \tag{16}$$

where SA denotes the solution obtained by the simulated annealing algorithm, and GA denotes the mean and maximum of two ratios for two algorithms are summarized in Table 7. There are significant differences between the performance of SA and GA. The mean of Ratio 1 is 88.04%, which is better than Ratio 2, and the max of Ratio 1 is 51.95%, which is better than Ratio 2. Table 8 shows the number of times that SA or GA yields a better solution than the other. SA yields a better solution 89.25% of the time (2,142 out of 2400), while GA yields a better solution only 8.3% of the time (200 out of 2400) for small-sized problems. Moreover, for SA, the worst mean of Ratio 1 is only 0.0035, and the maximum of Ratio 1 is 0.0959. However, the mean and maximum of Ratio 2 is 0.0167 and 0.2350 for GA in the worst case. SA outperforms GA when n is 20, and learn-

Table 7
The computational results of SA and GA for small job-sized problems.

m	n	S_i	U(1,5)				U(1,10)			
			Ratio1		Ratio 2		Ratio1		Ratio 2	
		LE	Mean	max	Mean	Max	Mean	Max	Mean	Max
2	10	1	0.32	7.81	1.67	23.50	0.21	8.56	1.08	15.72
		2	0.17	5.23	1.12	21.35	0.08	6.07	1.20	10.94
		3	0.11	2.41	0.97	10.08	0.23	7.28	1.15	8.51
	20	1	0.18	4.12	0.88	9.70	0.03	2.00	0.64	6.01
		2	0.01	0.54	0.92	7.69	0.07	2.83	1.21	8.72
		3	0.00	0.26	1.18	4.95	0.01	0.69	1.31	4.42
3	10	1	0.07	2.55	1.20	16.62	0.35	9.59	1.03	11.40
		2	0.15	9.60	0.93	9.30	0.16	4.79	0.83	10.36
		3	0.23	6.17	0.61	2.37	0.06	1.77	0.90	11.32
	20	1	0.09	2.34	0.87	5.51	0.13	2.04	0.80	6.82
		2	0.09	3.59	1.04	8.47	0.02	1.44	1.00	4.87
		3	0.04	2.41	1.22	7.01	0.03	1.54	1.26	4.23

Table 8
The number of times that SA or GA yield better solution than another algorithm for small job-sized problems.

S_i	m	n	LE	SA	GA	S_i	m	n	LE	SA	GA
U(1,5)	2	10	1	82	16	U(1,10)	2	10	1	91	9
			2	85	12				2	97	3
			3	87	11				3	98	2
		20	1	84	7			20	1	91	9
			2	82	10				2	93	7
			3	81	15				3	97	3
	3	10	1	87	6		3	10	1	95	4
			2	92	5				2	94	6
			3	88	10				3	97	3
		20	1	80	16			20	1	86	13
			2	85	13				2	96	4
			3	79	12				3	95	4

ing index is −0.515, as shown in Fig. 6(b, f and h). As for the mean of Ratio 1, the relative deviation decreases when the number of jobs increases. Comparing the mean execution times, as in Table 9, with the unit of CPU time in seconds, the execution times of both the algorithms are longer as the number of jobs increases. In terms of the mean execution time, SA is 7.27% faster than GA.

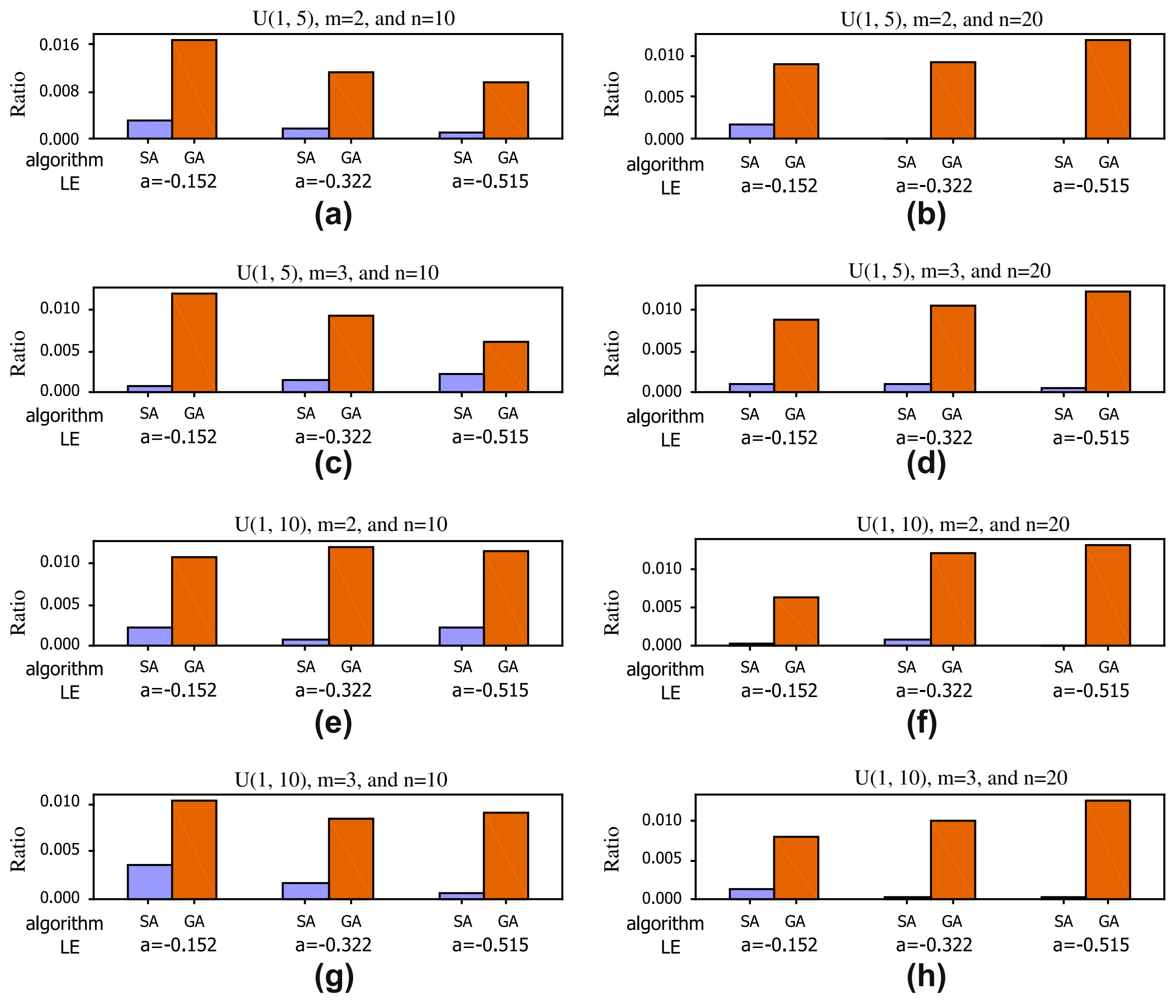

Fig. 6. Computational results of small job-sized problem for SA and GA.

Table 9
CPU time of SA and GA for small job-sized problems.

m	*n*	S_i	U(1,5)				U(1,10)			
			CPU time (SA)		CPU time (GA)		CPU time (SA)		CPU time (GA)	
		LE	Mean	Max	Mean	Max	Mean	Max	Mean	Max
2	10	1	0.09	0.13	0.10	0.13	0.09	0.11	0.10	0.13
		2	0.09	0.11	0.09	0.11	0.09	0.11	0.10	0.13
		3	0.09	0.11	0.09	0.11	0.09	0.13	0.10	0.13
	20	1	0.28	0.34	0.29	0.36	0.29	0.36	0.30	0.41
		2	0.27	0.38	0.28	0.36	0.28	0.38	0.30	0.39
		3	0.28	0.38	0.29	0.38	0.29	0.39	0.30	0.39
3	10	1	0.08	0.09	0.09	0.11	0.08	0.09	0.09	0.11
		2	0.08	0.09	0.08	0.11	0.08	0.11	0.09	0.11
		3	0.07	0.09	0.08	0.11	0.08	0.11	0.09	0.11
	20	1	0.22	0.27	0.24	0.28	0.23	0.33	0.25	0.30
		2	0.22	0.27	0.23	0.28	0.23	0.33	0.24	0.31
		3	0.22	0.30	0.23	0.28	0.23	0.34	0.24	0.31

To show the influence of the learning effect, Figs. 7 and 8 show the fitness values of SA and GA with respect to three learning effects. The fitness values of SA and GA are smaller when the learning effect is stronger. The fitness values decrease as the number of jobs increases.

To evaluate the influence of the performance of SA and GA, the two-way analysis of variance (ANOVA) is performed at a significance level of 0.05. The results are shown in Tables 9 and 10, respectively. In Table 10, all the four main effect and the five of six interaction effects exhibit significant differences in the fitness value (except the relation between *s* and *m*). From Table 11, the relationship between *s* and *m* and the relationship between *s* and *a* do not show significant differences in the

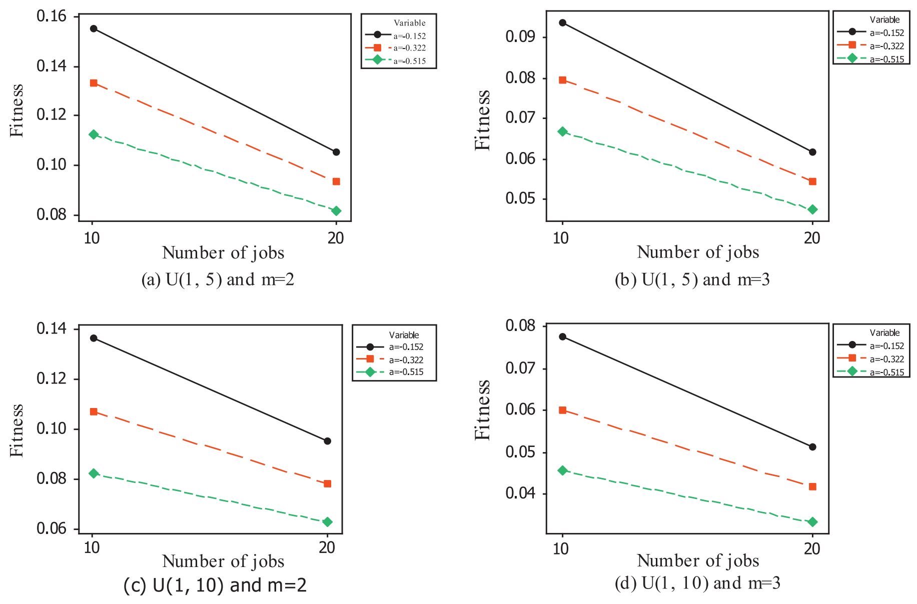

Fig. 7. Comparison of the fitness value of three learning indexes for SA in small-sized problem.

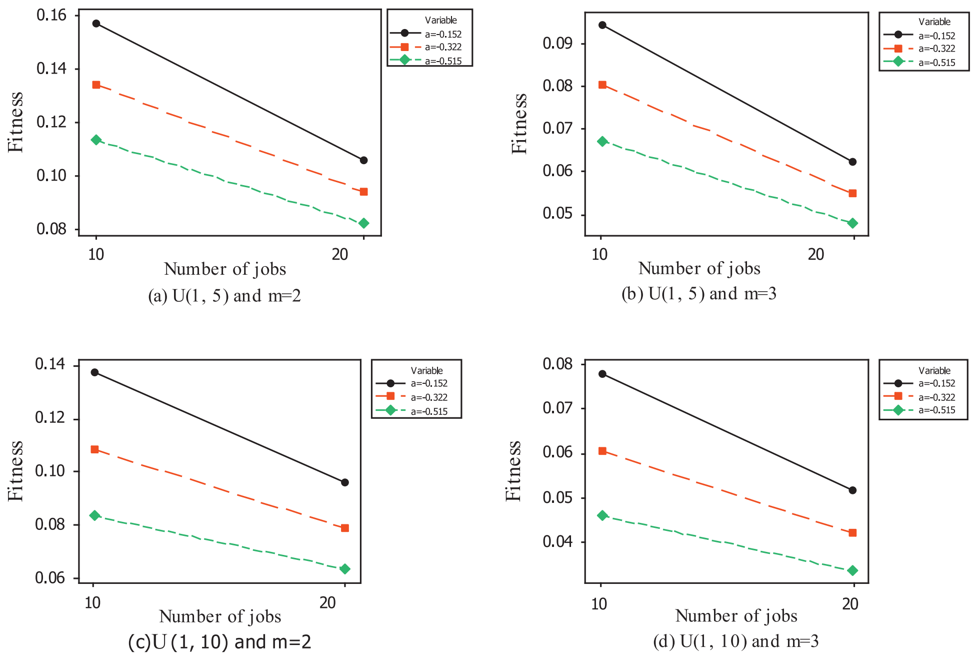

Fig. 8. Comparison of the fitness value of three learning indexes for GA in small-sized problem.

fitness value. Therefore, in the second part of the computational experiments, we still consider s, m, a, and n as the control factors.

In the third part of the experiment, the proposed heuristic algorithms are tested with four different numbers of jobs: $n = 100$, 200, 500 and 1000. The numbers of machines (m) are set at two levels, namely, 20 and 30. The settings of the processing times (p_j) are the same as those for the small job-sized problems, as above. The speeds of the machines (s_i) are generated from two continuous uniform distributions, namely, $U(1, 5)$ and $U(1, 10)$. We also consider three patterns of learning effects. A set of 100 instances is randomly generated for each situation. As a result, 48 experimental cases are conducted, and a total of 4800 instances are tested. The parameter settings for GA and SA are given in Table 12. The results for large job-sized problems are given in Tables 13–15.

Table 13 summarizes all of the experimental results, including two ratios for large job-sized problems. The mean value of Ratio 1 is still better than Ratio 2 for all of the experiments. In addition, the mean value of Ratio 2 increases as the number of jobs increases, the speed of machines or the number of machines increases. As illustrated in Table 14, SA outperforms GA 99.98% of the time. Moreover, the execution time of two algorithms increases when the number of jobs increases as shown in Table 15. Although the execution times of SA and GA are similar when the numbers of jobs are 500 and 1000, the results of SA are always better than GA.

As shown in Figs. 9 and 10, the fitness values of SA and GA are smaller when the learning effect is stronger. For SA, the fitness values decrease when the number of jobs increases. However, this is not the case for GA, as shown in Fig. 9(c and d).

As shown in Tables 16 and 17, all four of the main effects and five of six interaction effects exhibit significant differences in the fitness values, except the relationship between m and n for GA.

Table 10
Two-way ANOVA of the fitness value for SA.

Source	DF	MS	F	p-Value
s	1	0.0020000	593.847	0.000
m	1	0.0120000	0.004	0.000
n	1	0.0050000	0.002	0.000
a	2	0.0020000	587.023	0.000
$s * m$	1	0.0000245	7.679	0.022
$s * n$	1	0.0000000	33.030	0.000
$s * a$	2	0.0000270	8.462	0.009
$m * n$	1	0.0000000	79.681	0.000
$m * a$	2	0.0000000	36.239	0.000
$n * a$	2	0.0000000	45.457	0.000
Error	9	0.0000032		
Total	23			

Table 11
Two-way ANOVA of the fitness value for GA.

Source	DF	MS	F	p-value
s	1	0.0020000	626.367	0.000
m	1	0.0120000	0.004	0.000
n	1	0.0050000	0.002	0.000
a	2	0.0020000	619.003	0.000
$s * m$	1	0.0000263	8.565	0.017
$s * n$	1	0.0000000	36.895	0.000
$s * a$	2	0.0000224	7.301	0.013
$m * n$	1	0.0000000	85.371	0.000
$m * a$	2	0.0000000	40.021	0.000
$n * a$	2	0.0000000	47.918	0.000
Error	9	0.0000031		
Total	23			

Table 12
Parameter settings of SA and GA for large job-sized problems.

	GA	SA
Parameter setting	Iteration number is 300 Population size is 200 Mutation rate is 0.05	Iteration number is 30000 $T_{max} = 100$

Table 13
The computational results of SA and GA for large job-sized problems with 100 random instances for every suite.

m	n	S_i	U(1,5)				U(1,10)			
			Ratio1		Ratio2		Ratio1		Ratio2	
		LE	Mean	Max	Mean	Max	Mean	Max	Mean	Max
20	100	1	0	0	22	36	0	0	34	112
		2	0	0	25	46	0	0	39	115
		3	0	0	29	61	0	0	50	163
	200	1	0	0	33	74	0	0	57	149
		2	0	0	39	83	0	0	70	171
		3	0	0	49	115	0	0	90	176
	500	1	0	0	51	96	0	0	98	188
		2	0	0	60	110	0	0	119	244
		3	0	0	75	135	0	0	148	283
	1000	1	0	0	68	122	0	0	127	255
		2	0	0	80	139	0	0	150	279
		3	0	0	98	158	0	0	181	336
30	100	1	0	0	31	51	0	0	41	72
		2	0	0	33	54	0	0	47	91
		3	0	0	37	65	0	0	56	123
	200	1	0	0	42	70	0	0	62	131
		2	0	0	48	85	0	0	73	165
		3	0	0	56	108	0	0	88	190
	500	1	0	0	61	101	0	0	113	192
		2	0	0	69	107	0	0	134	226
		3	0	0	83	134	0	0	155	265
	1000	1	0	0	75	113	0	0	141	249
		2	0	0	85	139	0	0	163	293
		3	0	0	101	151	0	0	185	343

Table 14
The number of times that SA or GA yield better solution than another algorithm.

S_i	m	n	LE	SA	GA	S_i	m	n	LE	SA	GA
U(1,5)	20	100	1	100	0	U(1,10)	20	100	1	100	0
			2	100	0				2	100	0
			3	100	0				3	99	1
		200	1	100	0			200	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
		500	1	100	0			500	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
		1000	1	100	0			1000	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
	30	100	1	100	0		30	100	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
		200	1	100	0			200	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
		500	1	100	0			500	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0
		1000	1	100	0			1000	1	100	0
			2	100	0				2	100	0
			3	100	0				3	100	0

Table 15
CPU time of SA and GA for large job-sized problems.

m	*n*	S_i	U(1,5)				U(1,10)			
			CPU times (SA)		CPU times (GA)		CPU times (SA)		CPU times (GA)	
		LE	Mean	Max	Mean	Max	Mean	Max	Mean	Max
20	100	1	9.98	10.67	10.94	11.16	10.23	11.39	11.01	11.27
		2	9.70	10.50	10.67	11.00	10.09	11.41	10.67	10.92
		3	9.86	10.94	10.57	10.77	10.33	11.98	10.49	10.74
	200	1	27.98	29.97	28.75	29.30	28.99	33.19	28.70	29.55
		2	27.91	30.45	28.29	28.97	29.39	34.81	28.28	28.70
		3	28.94	31.88	28.25	29.08	31.07	36.03	28.15	28.81
	500	1	128.84	142.52	122.07	122.81	134.16	152.70	119.60	120.88
		2	133.19	149.97	121.15	121.88	139.53	158.11	118.61	119.66
		3	141.28	160.92	120.84	122.20	148.02	164.16	118.26	119.24
	1000	1	464.16	507.19	420.91	429.00	476.57	526.34	410.76	412.00
		2	484.62	534.19	416.72	417.86	497.33	549.97	409.38	410.89
		3	507.08	566.02	410.04	412.30	521.56	563.91	408.78	410.63
30	100	1	9.96	10.67	11.67	12.03	10.29	11.11	11.63	11.95
		2	9.83	10.48	11.39	11.58	10.08	11.00	11.35	11.70
		3	9.82	10.69	11.22	11.42	10.32	11.38	11.53	11.77
	200	1	25.42	26.88	27.88	28.64	26.35	29.00	28.03	28.55
		2	24.96	26.69	27.24	28.00	26.06	28.76	27.01	27.31
		3	25.67	27.88	27.00	27.52	26.59	29.81	26.81	27.22
	500	1	101.03	106.83	99.81	100.19	105.54	116.73	99.74	100.53
		2	103.08	109.76	98.43	99.28	108.11	121.59	98.32	99.20
		3	107.90	115.63	98.05	99.31	112.34	125.20	98.00	99.27
	1000	1	333.11	351.39	313.68	314.75	350.07	374.81	312.95	314.16
		2	343.88	363.86	312.57	317.17	360.69	382.34	311.72	315.28
		3	360.08	382.17	311.57	312.55	370.53	386.80	311.16	312.00

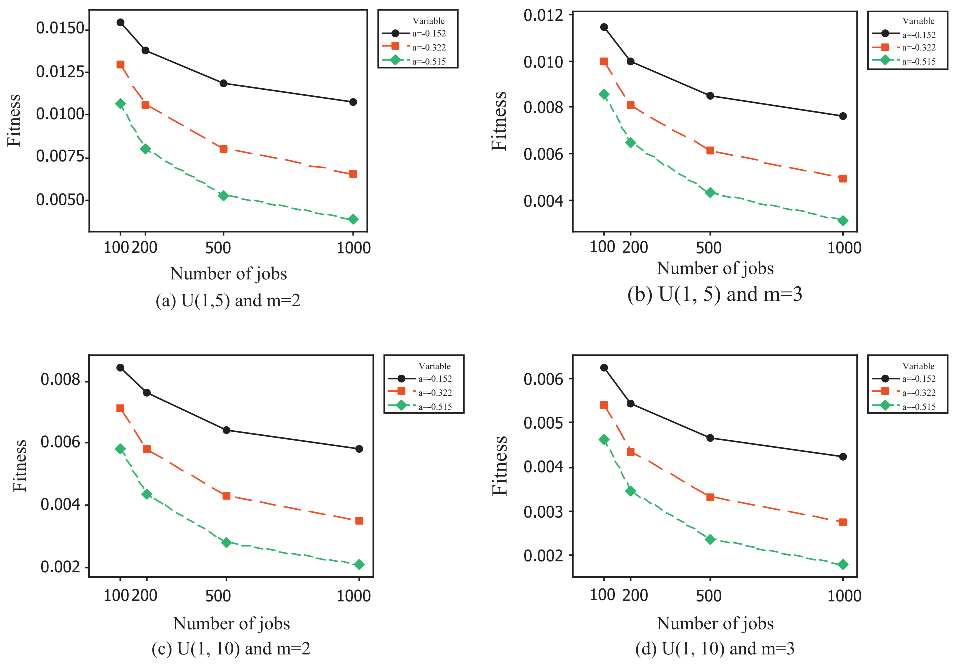

Fig. 9. Comparison of the fitness value of three learning indexes for SA in large-sized problem.

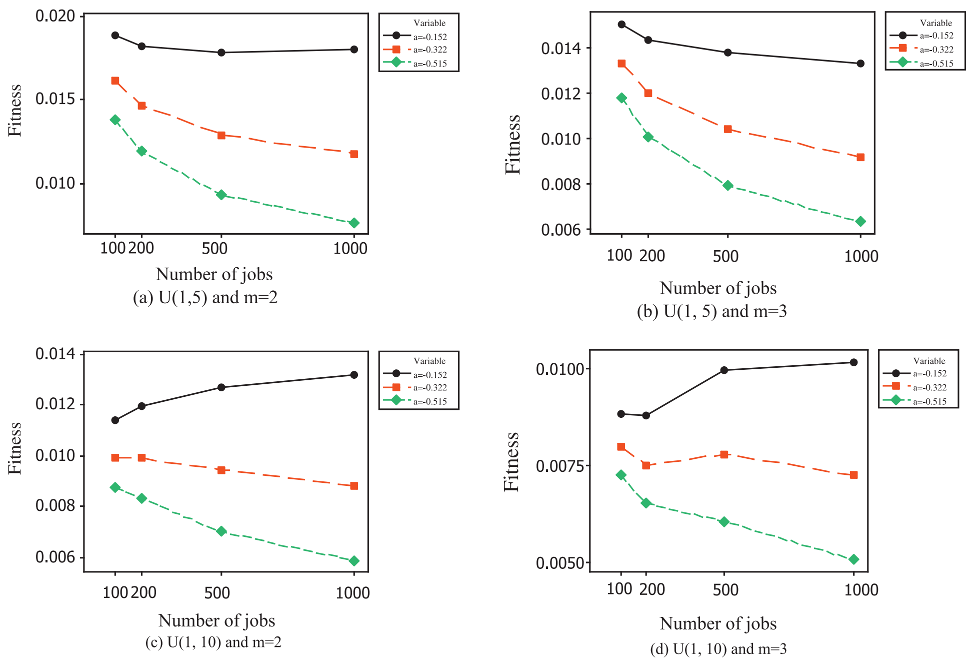

Fig. 10. Comparison of the fitness value of three learning indexes for GA in large-sized problem.

Table 16
Two-way ANOVA of the fitness value for SA for large size problems.

Source	DF	MS	*F*	*p*-Value
s	1	0.0000000	5051	0.000
m	1	0.0000402	1092	0.000
n	3	0.0000403	1094	0.000
a	2	0.0000574	1560	0.000
$s*m$	1	0.0000034	93.563	0.000
$s*n$	3	0.0000035	94.325	0.000
$s*a$	2	0.0000048	130.572	0.000
$m*n$	3	0.0000006	16.287	0.000
$m*a$	2	0.0000029	78.706	0.000
$n*a$	6	0.0000004	11.576	0.000
Error	23			
Total	47			

Table 17
Two-way ANOVA of the fitness value for GA for large size problems.

Source	DF	MS	*F*	*p*-Value
s	1	0.0000000	3615	0.000
m	1	0.0000712	1279	0.000
n	3	0.0000109	195.446	0.000
a	2	0.0000000	1940	0.000
$s*m$	1	0.0000020	36.019	0.000
$s*n$	3	0.0000062	111.435	0.000
$s*a$	2	0.0000053	95.096	0.000
$m*n$	3	0.0000000	1.659	0.204
$m*a$	2	0.0000042	75.683	0.000
$n*a$	6	0.0000038	68.652	0.000
Error	23	0.0000000		
Total	47			

6. Conclusions

The parallel machine scheduling problem has received important attention in the engineering field for its popularity in information systems and manufacturing. The impact of the learning effect on production procedure is cost saving, and the fuzzy set theory is suitable to address the uncertainties. Fuzzy set theory offers a convenient framework for modeling real-world production systems mathematically. A limited number of studies on parallel machine scheduling problem with fuzzy theory are available. However, the learning effect has not been considered in these studies. In this paper, we study the parallel machine scheduling problem with fuzzy processing times and learning effects. The objective is to minimize the fuzzy makespan based on the possibility measure. We proposed two algorithms, the simulated annealing algorithm and the genetic algorithm, to solve the scheduling problem.

Computational experiments were conducted to evaluate the performance and execution time of the heuristics under several different scenarios, and the impact of the learning effects is also discussed. From the computational results, SA is observed to outperform GA, and thus SA is recommended for this problem.

Extensions of the paper may consider different learning curves or fuzzy due dates. In addition, a new ranking method for fuzzy numbers is also an important issue for fuzzy scheduling problems and is worth further investigation.

Acknowledgement

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

References

[1] A. Anglani, Grieco, E. Guerriero, R. Musmanno, Robust scheduling of parallel machines with sequence-dependent set-up costs, European Journal of Operational Research 161 (2005) 704–720.
[2] A. Bachman, A. Janiak, Scheduling jobs with position-dependent processing times, Journal of the Operational Research Society 55 (2004) 257–264.
[3] J. Balasubramanian, I.E. Grossmann, Scheduling optimization under uncertainty—an alternative approach, Computers and Chemical Engineering 27 (2003) 469–490.
[4] S. Balin, Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation, Information Sciences 181 (2011) 3551–3569.
[5] D. Biskup, Single-machine scheduling with learning considerations, European Journal of Operational Research 115 (1999) 173–178.
[6] D. Biskup, A state-of-the-art review on scheduling with learning effect, European Journal of Operational Research 188 (2008) 315–329.
[7] S. Chanas, A. Kasperski, Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates, Engineering Applications of Artificial Intelligence 14 (2001) 377–386.
[8] T.C.E. Cheng, C.C. Wu, W.C. Lee, Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects, Information Science 178 (2008) 2476–2487.
[9] T.C.E. Cheng, W.H. Kuo, D.L. Yang, Scheduling with a position-weighted learning effect based on sum-of-logarithm-processing-times and job position, Information Sciences 221 (2013) 490–500.
[10] D. Dubois, H. Prade, Unfair coins and necessity measures: towards a possibilistic interpretation of histograms, Fuzzy Sets and Systems 10 (1983) 15–20.
[11] T. Eren, A bicriteria parallel machine scheduling with a learning effect of setup and removal times, Applied Mathematical Modelling 33 (2009) 1141–1150.
[12] E. Figielska, A genetic algorithm and a simulated annealing algorithm combined with generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources, Computers & Industrial Engineering 56 (2009) 142–151.
[13] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 5 (1986) 533–549.
[14] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[15] G.A. Süer, E. Báez, Z. Czajkiewicz, Minimizing the number of tardy jobs in identical machine scheduling, Computers & Industrial Engineering 25 (1993) 243–246.
[16] C.J. Hsu, W.H. Kuo, D.L. Yang, Unrelated parallel machine scheduling with past-sequence-dependent setup time and learning, Applied Mathematical Modelling 35 (2011) 1492–1496.
[17] W. Huang, S.K. Oh, W. Pedrycz, A fuzzy time-dependent project scheduling problem, Information Sciences 246 (2013) 100–114.
[18] T. Itoh, H. Ishii, Fuzzy due-date scheduling problem with fuzzy processing time, International Transactions in Operational Research 6 (1999) 639–647.
[19] A. Janiak, R. Rudek, A note on a makespan minimization problem with a multi-abilities learning effect, OMEGA – The International Journal of Management Science 38 (3–4) (2010) 213–217.
[20] S. Kirkpatrick, C.D.Jr. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[21] W.H. Kuo, C.J. Hsu, D.L. Yang, Worst-case and numerical analysis of heuristic algorithms for flowshop scheduling problems with a time-dependent learning effect, Information Science 184 (2012) 282–297.
[22] W.H. Kuo, C.J. Hsu, D.L. Yang, Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects, Computers & Industrial Engineering 61 (2011) 179–183.
[23] P.J. Lai, W.C. Lee, Single-machine scheduling with general sum-of- processing-time-based and position-based learning effects, OMEGA – The International Journal of Management Science 39 (2011) 467–471.
[24] P.J. Lai, H.C. Wu, Using genetic algorithms to solve fuzzy flow shop scheduling problems based on possibility and necessity measures, International Journal of Uncertainty, Fuzziness and Knowledge-based Systems 16 (2008) 409–433.
[25] P.J. Lai, H.C. Wu, Evaluate the fuzzy completion times in the fuzzy flow shop scheduling problems using the virus-evolutionary genetic algorithms, Applied Soft Computing 11 (2011) 4540–4550.
[26] W.C. Lee, Scheduling with general position-based learning curves, Information Sciences 181 (2011) 5515–5522.
[27] W.C. Lee, S.K. Chen, C.W. Chen, C.C. Wu, A two-machine flowshop problem with two agents, Computer & Operations Research 38 (2011) 98–104.
[28] W.C. Lee, C.C. Wu, Some single-machine and m-machine flowshop scheduling problems with learning considerations, Information Science 179 (2009) 3885–3892.
[29] C.H. Lin, A rough penalty genetic algorithm for constrained optimization, Information Sciences 241 (2013) 119–137.
[30] U. Maulik, A. Mukhopadhyay, Simulated annealing based automatic fuzzy clustering combined with ANN classification for analyzing microarray data, Computers & Operations Research 37 (2010) 1369–1380.

[31] L. Min, W. Cheng, Scheduling algorithm based on evolutionary computing in identical parallel machine production line, Robotics and Computer-Integrated Manufacturing 19 (2003) 401–407.
[32] H. Mosadegh, M. Zandieh, S.M.T. Fatemi Ghomi, Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines, Applied Soft Computing 12 (2012) 1359–1370.
[33] G. Mosheiov, Scheduling problems with a learning effect, European Journal of Operational Research 132 (2001) 687–693.
[34] D. Okołowski, S. Gawiejnowicz, Exact and heuristic algorithms for parallel-machine scheduling with DeJong's learning effect, Computers & Industrial Engineering 59 (2010) 272–279.
[35] J. Peng, B. Liu, Parallel machine scheduling models with fuzzy processing times, Information Sciences 166 (2004) 49–66.
[36] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, third ed., Prentice Hall, Upper Saddle River, NJ, 2008.
[37] G.M. Ribeiro, G.R. Mauri, L.A.N. Lorena, A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields, Computers & Industrial Engineering 60 (2011) 519–526.
[38] R. Rudek, The single processor total weighted completion time scheduling problem with the sum-of-processing-time based learning model, Information Sciences 199 (2012) 216–229.
[39] R. Slowinski, M. Hapke, Scheduling Under Fuzziness, Physica-Verlag Editions, New York, 2000.
[40] W.M. Spears, K.A. DeJong, On the virtues of parameterized uniform crossover, Proceedings of the Fourth International Conference on Genetic Algorithms (1991) 230–236.
[41] R. Tavakkoli-Moghaddam, B. Javadi, F. Jolai, A. Ghodratnama, The use of a fuzzy multi-objective linear programming for solving a multi-objective single-machine scheduling problem, Applied Soft Computing 10 (2010) 919–925.
[42] M.D. Toksari, E. Guner, Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration, Computers & Operations Research 36 (2009) 2394–2417.
[43] J.B. Wang, Y.B. Wu, P. Ji, A revision of some single-machine and *m*-machine flowshop scheduling problems with learning considerations, Information Sciences 190 (2012) 227–232.
[44] J. Xu, X. Zhou, A class of multi-objective expected value decision-making model with birandom coefficients and its application to flow shop scheduling problem, Information Science 179 (2009) 2997–3017.
[45] H.S. Yan, H.X. Wang, X.D. Zhang, Simultaneous batch splitting and scheduling on identical parallel production lines, Information Science 221 (2013) 501–519.
[46] D.L. Yang, T.C.E. Cheng, S.J. Yang, C.J. Hsu, Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities, Computers & Operations Research 39 (2012) 1458–1464.
[47] Y. Yin, D. Xu, X. Huang, Notes on some single-machine scheduling problems with general position-dependent and time-dependent learning effects, Information Science 181 (2011) 2209–2217.
[48] Y. Yin, D. Xu, K. Sun, H. Li, Some scheduling problems with general position-dependent and time-dependent learning effects, Information Science 179 (2009) 2416–2425.
[49] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—I, Information Sciences 8 (1975) 199–249.
[50] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—II, Information Sciences 8 (1975) 301–357.
[51] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—III, Information Sciences 9 (1975) 43–80.
[52] R. Zhang, W. Cheng, Bottleneck machine identification method based on constraint transformation for job shop scheduling with genetic algorithm, Information Science 188 (2012) 236–252.
[53] X.G. Zhang, G.L. Yan, W.Z. Huang, G.C. Tang, A note on machine scheduling with sum-of-logarithm-processing-time-based and position-based learning effects, Information Sciences 187 (2012) 298–304.
[54] H.J. Zimmermann, Fuzzy Sets Theory and its Applications, second ed., Kluwer Academic Publishers., New York, 1991.