Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities

Zhaohong Jia [a,b,*], Jianhai Yan [a,b], Joseph Y.T. Leung [c], Kai Li [d], Huaping Chen [e]

[a] *Key Lab of Intelligent Computing and Signal Processing of Ministry of Education Anhui University, Hefei, Anhui, 230039, PR China*
[b] *School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, PR China*
[c] *Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA*
[d] *School of Management, Hefei University of Technology, Hefei, Anhui, 230009, PR China*
[e] *School of Management, University of Science and Technology of China, Hefei, Anhui, 230026, PR China*

HIGHLIGHTS

- We consider fuzzy scheduling on parallel batch machines with arbitrary capacities.
- The triangular fuzzy number (TFN) and the optimistic coefficient are used.
- A fuzzy ant colony optimization algorithm is proposed to address the problem.
- Computational results exhibit the superiority of the proposed algorithm.

ARTICLE INFO

Article history:
Received 9 April 2018
Received in revised form 27 October 2018
Accepted 12 November 2018
Available online 29 November 2018

Keywords:
Parallel batch processing machines
Non-identical machine capacities
Fuzzy job processing time
Fuzzy ant colony optimization algorithm

ABSTRACT

We study the problem of scheduling on parallel batch processing machines with different capacities under a fuzzy environment to minimize the makespan. The jobs have non-identical sizes and fuzzy processing times. After constructing a mathematical model of the problem, we propose a fuzzy ant colony optimization (FACO) algorithm. Based on the machine capacity constraint, two candidate job lists are adopted to select the jobs for building the batches. Moreover, based on the unoccupied space of the solution, heuristic information is designed for each candidate list to guide the ants. In addition, a fuzzy local optimization algorithm is incorporated to improve the solution quality. Finally, the proposed algorithm is compared with several state-of-the-art algorithms through extensive simulated experiments and statistical tests. The comparative results indicate that the proposed algorithm can find better solutions within reasonable time than all the other compared algorithms.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

Batch processing machines (BPMs) have been widely applied in real applications, such as circuit testing, metal processing, port cargo handling, ship scheduling at navigation locks, airport scheduling and other related fields [1]. The batch scheduling problems (BSPs) originate from the semiconductor manufacturing industry. In the final testing phase of semiconductor production, chips are placed on a plate, which is then put into a furnace for high-temperature calcination. Compared with the other stages, this process takes more time so that it has become a bottleneck of the semiconductor manufacturing [2]. Therefore, it is vital to improve the efficiency of the burning process in production. In the model of batch scheduling, the jobs, the batches and the machines represent the chips, the plates, and the furnaces, respectively. Firstly, the jobs are grouped into batches according to the machine capacity constraint; secondly, the batches are processed on the machines. Once a batch begins to be processed, it cannot be interrupted during this period, and other jobs are not allowed to be inserted into the batch until the process is finished.

In recent years, many researchers have studied the BSPs under the deterministic environments. Nevertheless, the real production systems are often affected by the factors related to the environment during the processing. For example, the proficiency of the operators is different from each other; the machine malfunctions often make the processing time uncertain. As a result, a lot of uncertainties are generated in real production. Furthermore, the accurate processing information is generally hard to be obtained in advance. Therefore, researchers have to take these ambiguous factors into consideration to make the studied problem much closer to real applications. Thus, the fuzzy scheduling has become one of the hot spots in the field of scheduling. In the 1970s, fuzzy mathematics programming was applied to the classic scheduling [3,4].

* Corresponding author at: School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, PR China.
E-mail address: zhjia@mail.ustc.edu.cn (Z. Jia).

https://doi.org/10.1016/j.asoc.2018.11.027
1568-4946/© 2018 Elsevier B.V. All rights reserved.

Different from the studies on scheduling on one single machine in fuzzy environment [5–9], this paper focuses on fuzzy scheduling on multiple machines. In addition, most fuzzy scheduling problems is characterized as each job being able to be processed on any machine [10–13]. Nevertheless, we are concerned with the jobs being unable to be processed on all machines with non-identical capacities due to the machine capacity constraint. Furthermore, a lot of researchers have taken the fuzzy scheduling on classical parallel machines into consideration [14–17]. However, a more complicated problem of fuzzy scheduling on parallel BPMs is investigated in this study.

In this paper, we extend the deterministic BSP on parallel BPMs under a more realistic environment. That is, the job processing time is fuzzy and represented by the triangular fuzzy number. In addition, the job sizes are arbitrary, as well as the machine capacities. The optimization objective is to minimize the makespan.

The rest of this paper is organized as following: Section 2 briefly introduces the related work; Section 3 describes the studied problem; Section 4 presents a fuzzy ant colony optimization (FACO) algorithm to address the studied problem; Section 5 discusses the comparative experiments and analyzes the experimental results; some concluding remarks are drawn in Section 6.

2. Related work

The research on the BSPs begins with a single BPM with the deterministic constraints. Uzsoy [18] firstly investigated the problem of minimizing the makespan of scheduling jobs with non-identical sizes (SSBN) on one single BPM. Uzsoy proved the problem NP-hard and proposed several heuristics to address it. To minimize the makespan for scheduling the jobs with dynamic arrivals on one single BPM, Lee [19] proposed four heuristics and a local search method according to different ways of delaying the processing of the batches. According to the simulation results, some of the heuristics showed good average performance.

With the development of intelligent optimization algorithms, researchers have begun to apply these algorithms to the BSPs. Melouk et al. [20] used the simulated annealing (SA) algorithm to solve the problem of minimizing the makespan with non-identical-size jobs. The simulated results demonstrated that the proposed algorithm is superior to the commercial software CPLEX. Based on different encoding methods, Kashan et al. [21] provided two hybrid genetic algorithms (GAs) to schedule the jobs with non-identical sizes on one single BPM to minimize the makespan and the maximum tardiness, simultaneously. Chung et al. [22] tackled a two-stage hybrid flowshop scheduling problem with one single BPM in the first stage and one single classic machine in the second stage. Considering one single BPM, Pei et al. [23] developed a novel hybrid algorithm of gravitational search with tabu search to minimize the makespan under the constraint that the total resource consumption does not exceed a given limit. Computational experiments showed the effectiveness and the efficiency of the proposed algorithm.

Due to the fact that there are often multiple processing equipments in real production, scheduling on parallel BPMs has attracted more attention. Wang and Chou [24] proposed a mixed integer model to formulate the problem of scheduling the jobs with non-identical sizes and unequal release time on parallel BPMs. One dynamic programming algorithm and a meta-heuristic based on SA and GA are proposed. The experimental results verified the good performance of the proposed meta-heuristic. Jia et al. [25] considered scheduling a set of jobs with unequal release times and different sizes on parallel BPMs with non-identical capacities and put forward two meta-heuristics based on ant colony optimization (ACO) to minimize the makespan. The simulation results demonstrated that one of the proposed algorithms outperformed all the other compared ones. Arroyo and Leung [26] studied the scheduling of the jobs with different sizes and ready times on unrelated parallel BPMs, and proposed an iterated greedy (IG) algorithm to minimize the maximum completion time. The comparative experiments on randomly generated instances verified the effectiveness of the IG algorithm.

At present, most research on scheduling problems focuses on the deterministic cases. But there are generally many uncertainties in real production, which has made the fuzzy scheduling closer to the practical application. The fuzzy scheduling problems have aroused the interests of the researchers. The intelligent optimization algorithms have also exerted the advantages of solving the complex fuzzy scheduling problems. Mok et al. [27] proposed a fuzzy scheme to blur the static-standard fabric-cutting scheduling problem, and developed a genetic optimization algorithm to search for the fault-tolerant schedules. Balin [11] considered the scheduling on parallel machines with fuzzy processing time and proposed a genetic algorithm (GA) to minimize the fuzzy makespan. Then, Balin [12] extended to study the problem of the machines with different processing velocities. The author provided a GA to minimize the makepan and compared it with the longest processing time (LPT) rule. And the simulation results showed that the GA outperformed the LPT rule. Alcan and Basligil [28] proposed a machine-code-based GA to minimize the makespan of the non-identical machine scheduling problem with fuzzy job processing time. Yeh et al. [14] gave a GA and an SA to solve the parallel machine scheduling problem with trapezoidal processing time, respectively. They took the learning effect in scheduling into consideration to minimize the makespan. The simulated results showed that the SA was superior to the GA.

For the parallel machine scheduling problem with fuzzy triangular processing time and fuzzy trapezoidal due dates, Engin and Gözen [29] applied GA and SA to optimize three objectives, respectively, including minimizing the fuzzy makespan, maximizing the average agreement index and maximizing the minimum agreement index. Molla-Alizadeh-Zavardehi et al. [30] considered the problem of minimizing the total weighted tardiness penalties of jobs on one single BPM with fuzzy due dates. Based on the due time of the relevant fuzzy sets, two hybrid meta-heuristics were developed. Yimer and Demirli [31] investigated the two-stage flowshop scheduling with parallel BPMs in the mass customizing furniture scheduling industry. They presented a fuzzy mixed integer programming model and a GA to address the scheduling problem of customer orders.

As one of the effective swarm intelligence algorithms, ACO was first proposed by Dorigo et al. [32–35]. In the past few years, ACO has been successfully applied to scheduling problems. Xu et al. [36] established a mixed integer programming model and presented an ACO algorithm to minimize the makespan on a single BPM with different job release times. They proposed algorithm is verified to be superior to CPLEX and GA according to the simulation results. Cheng et al. [37] considered an integrated scheduling problem of production and distribution for manufacturers and proposed an improved ACO to minimize the total cost of production and distribution. Furthermore, many researchers have applied ACO to the scheduling problem [38–42], recently.

Moreover, ACO has been widely applied to the fuzzy scheduling problem. Cheng and Chen [9] studied the problem of scheduling jobs with fuzzy processing times and different sizes on one single BPM. For minimizing the fuzzy makespan, they proposed an improved ACO with a stochastic selection mechanism of the metropolis criterion to overcome the premature convergence. Liao and Su [17] dealt with scheduling on parallel machines with the trapezoidal processing time, the trapezoidal release time and the trapezoidal sequencing dependent setup time. A hybrid ACO incorporated with a fuzzy number ranking method was designed to minimize the makespan.

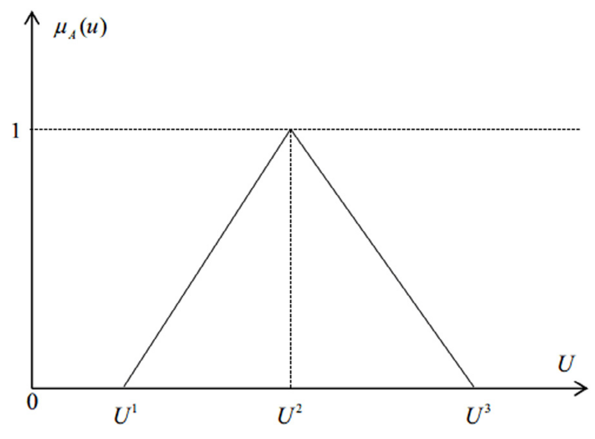

Fig. 1. Triangular fuzzy number (TFN)

In addition, SA [43], PSO [13,16], cooperative co-evolutionary algorithm [44], bee colony algorithm [45,46], imperialist competitive algorithm [30], fuzzy linguistic method [47], sensitivity analysis [48], and mixed integer fuzzy programming approach [49] are also applied to the fuzzy scheduling problems.

Most of the above research on fuzzy scheduling is based on the classic machine environment. Therefore, the fuzzy BSPs with fuzzy job processing time and different sizes on parallel BPMs with non-identical capacities is considered in this paper.

3. Problem description

3.1. Fuzzy operations

To be simple, the triangular fuzzy number (TFN) is adopted to represent the fuzzy parameters in this paper. For example, the TFN of $\widetilde{T}$ is defined as $\widetilde{T} = (T^1, T^2, T^3)$, indicating that the value of $\widetilde{T}$ is in interval $[T^1, T^3]$, and its most likely value is T^2.

Suppose that A is a fuzzy set on Ω, for $x \in \Omega$, $\mu_A(x) \in [0, 1]$ is a fuzzy membership of x. Mapping $(\Omega \to [0, 1], x \to \mu_A(x))$ is a membership function of A. The fuzzy set A can be expressed as: $A = \{(x, \mu_A(x)), x \in \Omega\}$ [50]. When x_1 and x_2 belong to A, where $\mu_A(x_1) > \mu_A(x_2)$, it means that x_1 belongs to A to a greater extent than x_2 to A. In this paper, the TFN is used to represent the fuzzy processing time of each job. The TFN can be represented by a triplet $\widetilde{U} = (U^1, U^2, U^3)$ and the membership function $\mu_A(U)$ is defined as Eq. (1) [51].

$$\mu_A(U) = \begin{cases} \dfrac{U - U^1}{U^2 - U^1}, & U^1 \le U < U^2; \\ \dfrac{U - U^3}{U^2 - U^3}, & U^2 \le U \le U^3; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Therefore, a TFN $\widetilde{U} = (U^1, U^2, U^3)$ can be represented as Fig. 1, which clearly portrays the relationship between triangular fuzzy numbers and membership function. Functions both $[U^1, U^2] \to [0, 1]$ and $[U^2, U^3] \to [0, 1]$ are continuous.

Suppose there are two fuzzy numbers $\widetilde{U} = (U^1, U^2, U^3)$, $\widetilde{V} = (V^1, V^2, V^3)$, and a real number e. The following operations are defined for the TFNs.

(1) Addition operation (+)

$$\widetilde{U} + \widetilde{V} = (U^1 + V^1, U^2 + V^2, V^3 + V^3) \tag{2}$$

(2) Subtraction operation (-)

$$\widetilde{U} - \widetilde{V} = \begin{cases} 0, & \text{if } \widetilde{U} = \widetilde{V}; \\ (U^1 - V^3, U^2 - V^2, U^3 - V^1), & \text{otherwise.} \end{cases} \tag{3}$$

(3) Multiplication operation ($\times$)

$$e \times \widetilde{U} = (e \times U^1, e \times U^2, e \times U^3) \tag{4}$$

(4) Comparison operation ($>$ or $<$)
(i). If $(U^1+2U^2+U^3)/4 > (<)(V^1+2V^2+V^3)/4$, then $\widetilde{U} > (<)\widetilde{V}$.
(ii). If $(U^1 + 2U^2 + U^3)/4 = (V^1 + 2V^2 + V^3)/4$, then
If $U^2 > (<)V^2$, then $\widetilde{U} > (<)\widetilde{V}$.
(iii). If $U^2 = V^2$, then
If $(U^3 - U^1) > (<)(V^3 - V^1)$, then $\widetilde{U} > (<)\widetilde{V}$.
(5) Approximate maximum operation ($\vee$)
Suppose $\mu_{\widetilde{U}}(x)$ and $\mu_{\widetilde{V}}(y)$ are the membership functions of $\widetilde{U}$ and $\widetilde{V}$, respectively. The membership function of $\widetilde{U} \vee \widetilde{V}$, denoted by $\mu_{\widetilde{U}\vee\widetilde{V}}(z)$, is defined as Eq. (5) [52].

$$\mu_{\widetilde{U}\vee\widetilde{V}}(z) = \sup_{Z=x\vee y} \min(\mu_{\widetilde{U}}(x), \mu_{\widetilde{V}}(y)) \tag{5}$$

There are two methods to calculate the maximum of two TFNs [53]. The S method is defined as Eq. (6).

$$\widetilde{U} \vee \widetilde{V} \approx (U^1 \vee V^1, U^2 \vee V^2, U^3 \vee V^3) \tag{6}$$

The L method is defined as Eq. (7).

$$\widetilde{U} \vee \widetilde{V} \approx \begin{cases} \widetilde{U}, & \text{if } \widetilde{U} > \widetilde{V}; \\ \widetilde{V}, & \text{otherwise.} \end{cases} \tag{7}$$

Take $\widetilde{U}$ and $\widetilde{V}$ for an example. The instance of $\widetilde{U}$ and $\widetilde{V}$, as well as the results of $\widetilde{U} \vee \widetilde{V}$ and $\mu_{\widetilde{U}\vee\widetilde{V}}(z)$ by L and S methods, are illustrated in Fig. 2. Based on L method, the approximate maximum of the two fuzzy numbers is either $\widetilde{U}$ or $\widetilde{V}$. However, based on S method, the approximate maximum is a triple composed of $\widetilde{U}$ and $\widetilde{V}$. In contrast, the accurate fuzzy maximum derived from the extension principle is also provided in Fig. 2.

3.2. Preliminaries

According to the three-parameter denotation [54], the studied problem can be denoted by $P_m|p\text{-}batch, \widetilde{p_j}, s_j, Z_i, M_j(inclusive)|\widetilde{C_{\max}}$. A set of jobs with fuzzy processing time $\widetilde{p_j}$, non-identical sizes s_j are to be scheduled on a set of m parallel-batch processing machines, denoted by M. Each BPM M_i has its capacity Z_i. Some jobs cannot be processed on some machines whose capacities are smaller than the sizes of the jobs. That is, the machines that can process job J_j is a subset of M, denoted by M_j. The objective is to minimize the fuzzy makespan $\widetilde{C_{\max}}$.

In a deterministic BSP, the processing time of a batch equals to the maximum processing time of jobs in the batch, which is applicable to the fuzzy BSP. Hence, in the studied problem, the processing times of the jobs being fuzzy lead to the processing times of the batches, as well as those of the machines, being also fuzzy. Thus, the TFNs of $\widetilde{p_j}$, $\widetilde{P_{ki}}$, $\widetilde{C_i}$, and $\widetilde{C_{\max}}$ are defined as $\widetilde{p_j} = (p_j^1, p_j^2, p_j^3)$, $\widetilde{P_{ki}} = (P_{ki}^1, P_{ki}^2, P_{ki}^3)$, $\widetilde{C_i} = (C_i^1, C_i^2, C_i^3)$, and $\widetilde{C_{\max}} = (C_{\max}^1, C_{\max}^2, C_{\max}^3)$, respectively.

The assumptions of the problem are as follows:

(1) There are n jobs in a jobs set $J = \{J_1, J_2, \ldots, J_n\}$. Each job has its fuzzy processing time $\widetilde{p_j} = (p_j^1, p_j^2, p_j^3)$, where $0 < p_j^1 \le p_j^2 \le p_j^3$, representing that the job processing time is in $[p_j^1, p_j^3]$, and the most likely processing time is p_j^2. Each job has a non-identical size s_j, which is assumed not larger than the maximum machine capacity. All jobs are assumed arriving at time zero.

(2) There is a machine set $M = \{M_1, M_2, \ldots M_m\}$ of m BPMs with unequal capacities. The capacity of machine M_i is denoted by Z_i. All the jobs are grouped into batches which are scheduled on the m BPMs.

(3) Each job can only be grouped into one batch. The generated batch set is denoted by B. Batch B_{ki} denotes the kth batch

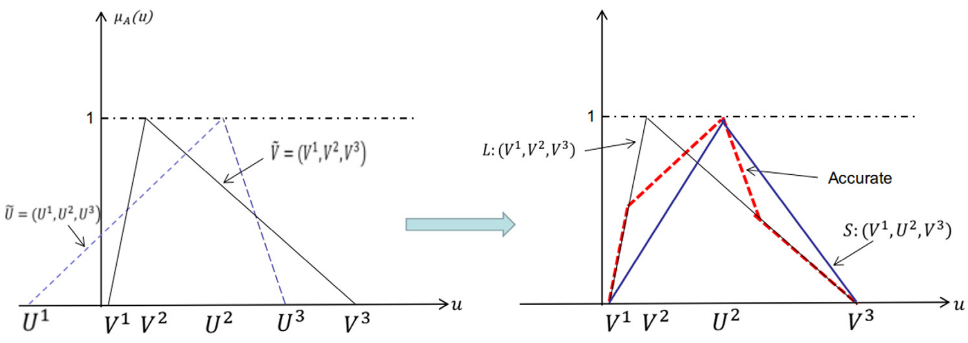

Fig. 2. Triangular fuzzy numbers and a comparison of approximate maximums and the accurate maximum.

scheduled on machine M_i. The size of each batch $B_{ki}(k = 1, 2, \dots, d; i = 1, 2, \dots, m)$ equals to the sum of the sizes of all the jobs in B_{ki}. The processing time of B_{ki}, denoted by $\widetilde{P_{ki}} = (P^1_{ki}, P^2_{ki}, P^3_{ki})$. Each batch is only scheduled on one machine. Once one batch begins to be processed, it cannot be interrupted. That is, no any other job can be inserted into the batch and any job in this batch cannot be removed until the processing is finished.

(4) The size of each batch cannot exceed the capacity of the machine that processes the batch. Since the sizes of some jobs are larger than the capacities of some machines, the jobs that can be processed by one machine is a subset of J.

(5) The completion time of machine M_i, denoted by $\widetilde{C_i}$, is equal to the sum of the processing times of all the batches on M_i, i.e., the completion time of the last batch on M_i. The makespan is the maximum completion time of all the machines, that is, $\widetilde{C_{\max}} = \max^m_{i=1}\{\widetilde{C_i}\}$.

The decision variable X_{jki} is defined as:

$$X_{jki} = \begin{cases} 1, & \text{if job } J_j \text{ is grouped into } B_{ki} \text{ on } M_i; \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where X_{jki} is a binary decision variable. $X_{jki} = 1$, if job J_j is grouped into the kth batch on machine M_i; otherwise, $X_{jki} = 0$.

Based on the above assumptions and definitions of variables, the mathematical model of the problem is established as follows:

$$\text{Minimize} \quad \widetilde{C_{\max}} \tag{9}$$

$$s.t. \sum_{k=1}^{d}\sum_{i=1}^{m} X_{jki} = 1, \forall J_j \in J \tag{10}$$

$$\sum_{J_j \in J} s_j X_{jki} \le Z_i, \forall B_{ki} \in B, M_i \in M \tag{11}$$

$$\widetilde{P_{ki}} \ge \widetilde{p_j} X_{jki}, \forall J_j \in J, B_{ki} \in B \tag{12}$$

$$\widetilde{C_{\max}} \ge \sum_{B_{ki} \in B} \widetilde{P_{ki}}, \forall M_i \in M \tag{13}$$

Formula (9) indicates the objective of the problem, i.e, the minimization of the makespan. Constraint (10) ensures that each job can only be grouped into one batch, which can only be processed on one machine. Constraint (11) ensures that the total sizes of the jobs in each batch does not exceed the capacity of the machine where the batch is scheduled. Constraint (12) defines the processing time of the batches. Constraint (13) indicates that the maximum completion time is not less than the completion time of each machine.

3.3. Fuzzy calculation

In real applications, the factors, such as equipments and operators, often cause uncertainties in processing, which complicates the problems. Therefore, it is important to handle such ambiguities in the problem model. One effective way is to use the fuzzy calculation. In this paper, the fuzzy processing time of job J_j, $\widetilde{p_j}$ is expressed as a triplet (p^1_j, p^2_j, p^3_j) and $\mu(\widetilde{p_j})$ is the membership function of $\widetilde{p_j}$. Suppose $J_1, J_2, \dots, J_l$ are grouped into B_{ki}. According to the definition of the batch processing time in the studied problem, the S method is adopted. That is, the fuzzy batch processing times are calculated by Eq. (14).

$$\widetilde{P_{ki}} = \widetilde{p}_1 \vee \widetilde{p}_2 \vee \dots \widetilde{p}_l \approx (p^1_1 \vee \dots \vee p^1_l, p^2_1 \vee \dots \vee p^2_l, p^3_1 \vee \dots \vee p^3_l) \tag{14}$$

Suppose that three jobs which are J_1, J_2, J_3 are assigned to batch B_{11}, the fuzzy processing times of these jobs are $\widetilde{p}_1 = (2, 5, 7)$, $\widetilde{p}_2 = (4, 6, 8)$ and $\widetilde{p}_3 = (1, 3, 9)$, respectively. Thus, the processing time of B_{11}, i.e., $\widetilde{P_{11}}$, is (4,6,9), according to Eq. (14).

In this study, the fuzzy makespan $\widetilde{C_{\max}}$ is calculated according to Eq. (15).

$$\widetilde{C_{\max}} = \vee^m_{i=1}\{\widetilde{C_i}\} \tag{15}$$

where $\widetilde{C_{\max}}$ is obtained by running the comparing operation defined in Eq. (7) for $m-1$ times. Moreover, $\widetilde{C_i}$ is calculated by Eq. (16).

$$\begin{aligned}\widetilde{C_i} &= (\widetilde{P_{1i}} + \widetilde{P_{2i}} + \dots + \widetilde{P_{bi}}) \\ &= (P^1_{1i} + \dots + P^1_{bi}, P^2_{1i} + \dots + P^2_{bi}, P^3_{1i} + \dots + P^3_{bi})\end{aligned} \tag{16}$$

where $\widetilde{P_{ki}}$ is the fuzzy processing time of batch B_{ki}.

In order to provide the effective reference to the production practice, we adopt the optimistic coefficient [55] to calculate the integral. The value of the optimistic coefficient $\omega \in (0, 1)$ represents the conditions in real manufacturing. A larger ω indicates that the conditions in real production is better. $\omega = 1$ represents that the system is ideal [9]. The full integral value of the makespan, denoted by $V^\omega(\widetilde{C_{\max}})$, is a convex combination of $V_L(\widetilde{C_{\max}})$ and $V_R(\widetilde{C_{\max}})$ values through ω [56]. $V_L(\widetilde{C_{\max}})$ and $V_R(\widetilde{C_{\max}})$ denote the left and the right integrals, respectively, which are defined as follows:

$$\begin{aligned}V_L(\widetilde{C_{\max}}) &= \int_0^1 \mu_L^{-1}(\alpha)d\alpha \\ &= \int_0^1 [C^1_{\max} + (C^2_{\max} - C^1_{\max})\alpha]d\alpha = \frac{1}{2}(C^1_{\max} + C^2_{\max})\end{aligned} \tag{17}$$

$$V_R(\widetilde{C_{\max}}) = \int_0^1 \mu_R^{-1}(\alpha)d\alpha$$

Table 1
The parameters of the jobs.

J_j	s_j	(p_j^1, p_j^2, p_j^3)
1	4	(27.2,31.0,31.6)
2	10	(32.6,37.0,38.6)
3	6	(39.9,41.0,47.6)
4	11	(22.7,27.0,30.9)
5	4	(14.8,15.0,16.6)
6	24	(35.7,42.0,50.6)
7	4	(36.3,41.0,46.0)
8	2	(38.3,44.0,47.7)
9	3	(36.4,41.0,48.9)
10	2	(13.1,14.0,16.4)

$$= \int_0^1 [C_{\max}^3 + (C_{\max}^2 - C_{\max}^3)\alpha]d\alpha = \frac{1}{2}(C_{\max}^2 + C_{\max}^3) \tag{18}$$

where μ_L and μ_R denote the left and the right membership functions, respectively. μ_L^{-1} and μ_R^{-1} denote their inverse functions, respectively. $V_L(\widetilde{C_{\max}})$ and $V_R(\widetilde{C_{\max}})$ are used to reflect the optimistic and the pessimistic viewpoints of the decision maker on the makespan, respectively [57]. Note that they denote the medians in ranges $[C_{\max}^1, C_{\max}^2]$ and $[C_{\max}^2, C_{\max}^3]$, respectively.

$\widetilde{C_{\max}}$ is translated into $C_{\max}$ as follows:

$$\begin{aligned} C_{\max} &= V^{\omega}(\widetilde{C_{\max}}) = \omega V_L(\widetilde{C_{\max}}) + (1-\omega)V_R(\widetilde{C_{\max}}) \\ &= \tfrac{1}{2}\omega(C_{\max}^1 + C_{\max}^2) + \tfrac{1}{2}(1-\omega)(C_{\max}^2 + C_{\max}^3) \\ &= \tfrac{\omega C_{\max}^1 + C_{\max}^2 + (1-\omega)C_{\max}^3}{2} \end{aligned} \tag{19}$$

4. Fuzzy ACO algorithm

As one of the swarm-based intelligence algorithms, the ACO is inspired by the foraging behavior of real ants [58]. The ACO algorithm has the advantages, such as the distributed computing, the self-organizing, the positive feedback, so that it can deal with the complex combinatorial optimization problems [59]. The ACO has been mainly used to solve the deterministic scheduling problems. However, little research is related to applying the ACO to the fuzzy BSPs. In this study, the ACO algorithm with the optimistic coefficient ω is used to tackle the problem $P_m|p\text{-}batch, \widetilde{p_j}, s_j, Z_i, M_j|\widetilde{C_{\max}}$.

In the proposed FACO algorithm, the optimistic coefficient ω is introduced to deal with the fuzzy processing time to make the algorithm more practicable and stable in a fuzzy scenario. In order to improve the search efficiency, two candidate lists are incorporated to narrow the search space. Additionally, a fuzzy local optimization strategy is adopted to further improve the solution quality.

4.1. Encoding

The coding of the solution is one of the key issues of applying the ACO algorithm. In this paper, the coding method is based on the batches, that is, the solutions include the scheduling of the batches on the machines. Each solution is encoded into a $2 \times n$ vector directly. The numbers in each column denotes the scheduling of each job. The first row denotes the indexes of the batches that contain the jobs in the corresponding column. And the numbers in the second row represent the indexes of the machines that process the corresponding jobs. Suppose there are two BPMs, whose capacities are 10 and 25, respectively. The information of the jobs to be processed is shown in Table 1.

An example of the encoded solution is shown in Table 2. The column j represents that job J_j is assigned to the kth batch on machine i. The first row shows the batch indexes and the second row shows the machine indexes, respectively. According to Table 2, the 10 jobs are grouped into five batches.

Table 2
An example of solution code.

J_j	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
k	1	1	1	1	1	2	2	3	2	2
i	1	2	1	2	2	2	1	2	1	1

4.2. Pheromone trails update

The pheromone trail γ_{jki} is defined as the desirability of assigning job J_j into batch B_{ki} as shown in Eq. (20).

$$\gamma_{jki} = \frac{\sum_{J_x \in B_{ki}} \varphi_{xj}}{|B_{ki}|} \tag{20}$$

where φ_{xj} represents the desirability of grouping jobs J_x and J_j in the same batch, the initialization value of φ_{xj} is set to 0.1 according to the preliminary experiments, and $|B_{ki}|$ denotes the number of jobs in batch B_{ki}.

The purpose of updating the pheromone trail is to modulate the search direction of the ant colony according to the previous experiences and improve the exploration and exploitation abilities of the algorithm. In the proposed ACO algorithm, the global optimal solution, that is, the best solution from the first iteration to the current iteration, is used to update the pheromone.

Let $\phi_{xj}(t)$ denote the expectation value of jobs J_x and J_j being grouped into the same batch in the tth iteration, and $m_{xj}(t)$ denote the frequency of J_x and J_j being batched together in the tth iteration. The pheromone trail is updated according to Eq. (21).

$$\phi_{xj}(t+1) = (1-\rho) * \phi_{xj}(t) + \Delta\phi_{xj}(t) * m_{xj}(t) \tag{21}$$

where $\rho \in (0, 1)$, the pheromone evaporation rate, is used to control the evaporation rate of the pheromone trails and prevent the infinite accumulation of the pheromone trails. If J_x and J_j are not grouped into the same batch in the tth iteration, then $\Delta\phi_{xj}(t) = 0$; otherwise, $\Delta\phi_{xj}(t) = Q/C_{\max}^*(t)$, where Q is an input parameter and $C_{\max}^*(t)$ denotes the makespan of the global best solution in the current iteration. Note that the makespan of the global solution is also a TFN, i.e., $\widetilde{C_{\max}^*}(t)$, although $C_{\max}^*(t)$ used in Eq. (21) is a deterministic value. And, $C_{\max}^*(t)$ is the value of $\widetilde{C_{\max}^*}(t)$ after calculated with ω.

4.3. Heuristic information

In the studied problem, each job has two characters, i.e., the size and the processing time. Due to the machines having limited and non-identical capacities, different assignments of jobs significantly influence the processing time, as well as the total number, of the generated batches, which leads to the solutions with different quality. That is to say, the makespan is determined by both the job sizes and the jobs processing times in the studied problem. Thus, the heuristic information is based on the relationship between the candidate jobs and the unoccupied space of the current batch and the machine. The non-identical job sizes often lead to the batches being not fully occupied. The unoccupied space of the solution, i.e., the wasted space, can be formulated as Eq. (22). Each component in $\widetilde{C_{\max}} = (C_{\max}^1, C_{\max}^2, C_{\max}^3$ is denoted by $C_{\max}^x$ ($x = 1, 2, 3$). Since the job processing times are fuzzy, the wasted space is also fuzzy. The value of WS obtained by using the optimistic coefficient ω is defined as Eq. (23).

$$WS^x = (\sum_{i=1}^{m} Z_i) * C_{\max}^x - \sum_{J_j \in J} s_j * p_j^x, (x = 1, 2, 3) \tag{22}$$

$$WS = (\omega WS^1 + WS^2 + (1-\omega)WS^3)/2 \tag{23}$$

According to Eq. (22), it can be found that decreasing the unoccupied space in the solution is favorable to minimizing the makespan. And thus, in this problem, the heuristic information is based on the relationship between the candidate jobs and the wasted space (WS) of the current batches and the machines. Specifically, the fuzzy heuristic information $\widetilde{\eta_{jki}} = (\eta^1_{jki}, \eta^2_{jki}, \eta^3_{jki})$ is defined as Eq. (24). And, the value of η_{jki} obtained with the optimistic coefficient ω is defined in Eq. (25).

$$\widetilde{\eta_{jki}} = \begin{cases} Z_i(\widetilde{P_{ki}} - \widetilde{p_j} \vee \widetilde{P_{ki}}) + s_j\widetilde{p_j}, & \text{if } WS_i > WS_i^*; \\ (1,1,1), & \text{if } WS_i \le WS_i^*; \end{cases} \tag{24}$$

$$\eta_{jki} = (\omega\eta^1_{jki} + \eta^2_{jki} + (1-\omega)\eta^3_{jki})/2 \tag{25}$$

where WS_i and WS_i^* represent the wasted space of M_i and B_{ki} before and after job J_j is added into the current batch B_{ki}. Suppose there are two BPMs, whose capacities are $Z_1 = 10$ and $Z_2 = 25$, respectively. The information of the jobs to be processed and encoded are shown in Tables 1 and 2. The initialization value of WS for each batch is set to zero. For current batch B_{11}, if J_1 is put into, $WS^x_{(x=1,2,3)} =$ (163.2,186,189.6) according to Eq. (22), and $WS^* = 178.56$ with $\omega = 0.7$ by Eq. (23). Due to $WS^* > WS$, $\widetilde{\eta_{111}} = (1,1,1)$. In the case of J_2 being put into batch B_{12}, $WS^x_{(x=1,2,3)} = (706.2,801,838.6)$, and $WS^* = 773.46$, which is larger than WS. Therefore, $\widetilde{\eta_{212}} = (1,1,1)$. Then, when J_3 is placed into B_{11}, $WS^x_{(x=1,2,3)} = (671.5,655,804)$, and $WS^* = 683.125$, which is less than the previous WS. Then, $\widetilde{\eta_{311}} =$ (399,410,476), according to Eq. (24).

4.4. Candidate lists

In the studied problem, there are some jobs whose sizes are larger than the capacities of some machines. Therefore, two candidate lists, based on the machine capacities and the idle space of the batches, are generated in order to narrow the search space, respectively. The first candidate list L^1_i consists of the jobs that satisfy the capacity constraint of the current machine M_i, which is defined in Eq. (26).

$$L^1_i = \{J_j | s_j \le Z_i\} \tag{26}$$

The second candidate list L^2_{ki} includes the jobs whose sizes are less than the remaining capacity of batch B_{ki} on the machine M_i, as defined in Eq. (27).

$$L^2_{ki} = \{J_j | s_j \le (Z_i - \sum_{J_l \in B_{ki}} s_l)\} \tag{27}$$

4.5. Solution construction

In order to describe the method of building the solution, a $1 \times n$ visiting list VL is introduced to record the processing status of the jobs. The initial value of VL is [1, 2, 3.....n], indicating that all the jobs have not been scheduled. Once one job J_j is scheduled, the value at the corresponding position, i.e., jth element in VL, is changed into zero. $VL = [0, 0, 0....0]$ means that all the jobs have been scheduled. Suppose there are n_a ants in the colony. In each iteration, every ant generates one solution according to Algorithm 1, i.e., procedure SC, with the inputs of job set J and machine set M. The procedure SC is described as follows.

Algorithm 1: The SC Procedure.

```
1: $VL = [1, 2, 3....n]$./* Initialization.*/
2: while $VL \ne [0, 0, 0....0]$ do
3:    Sort the machines according to their fuzzy completion time. $\widetilde{C}_1 \le \widetilde{C}_2 \le \ldots \le \widetilde{C}_m$.
4:    Generate $L^1_i$ for all the machines according to Eq. (26).
5:    A new batch is built for the first machine with $L^1_i \ne \emptyset$. Firstly, an empty batch is built on the machine, and a job is randomly selected from $L^1_i$; secondly, update $VL$.
6:    Generate $L^2_{jki}$ for the current batch according to Eq. (27).
7:    While $L^2_{jki} \ne \emptyset$, job $J_j$ is selected from $L^2_{jki}$ and added into $B_{ki}$ according to probability $p_{jki}$, as defined in Eq. (28);
         $$p_{jki} = \begin{cases} \frac{(\gamma_{jki})^\alpha (\eta_{jki})^\beta}{\sum_{J_x \in L^2_{jki}} (\gamma_{jki})^\alpha (\eta_{jki})^\beta}, J_x \in L^2_{jki}; \\ 0, \text{otherwise}; \end{cases} \quad (28)$$
8:    Update $VL$ and $L^2_{jki}$.
9: end while
10: Output the schedule and stop.
```

4.6. Description of the proposed algorithm

For each constructed solution, a fuzzy local optimization (FLO) strategy is adopted to improve the solution quality. Firstly, choose the two machines with the largest and the smallest fuzzy completion time, denoted by M_a and M_b, respectively. The fuzzy completion time of the two machines are denoted by $\widetilde{C_a}$ and $\widetilde{C_b}$, respectively. Secondly, for each batch on M_a, if there is a single job with the longest processing time, denoted by J_j, in the current batch, and J_j satisfies ($\widetilde{p_j} + \widetilde{C_b} < \widetilde{C_a}$&&$s_j \le Z_b$), then move J_j from M_a to the first batch on M_b. Specifically, if there is a non-empty batch that can accommodate J_j, then insert J_j into this batch; otherwise, construct a new empty batch on M_b and insert J_j in the empty batch. Then, update $\widetilde{C_a}$ and $\widetilde{C_b}$. Repeat the above procedure until the fuzzy completion time of the two machines does not change. The FLO strategy is described in detail as Algorithm 2.

Algorithm 2: The FLO Strategy.

```
1: $flag \leftarrow 1$;
2: while $flag = 1$ do
3:    Find the two machines, denoted by $M_a$ and $M_b$, with the largest and the smallest fuzzy completion time, denoted by $\widetilde{C_a}$ and $\widetilde{C_b}$, in the constructed solutions, respectively;
4:    for each batch on $M_a$ do
5:       if only one job $J_j$ has the longest processing time and satisfies ($\widetilde{p_j} + \widetilde{C_b} < \widetilde{C_a}$&&$s_j \le Z_b$) then
6:          if there is a batch on $M_b$ that can accommodate $J_j$ then
7:             Move $J_j$ from $M_a$ to the batch on $M_b$ ;
8:          else
9:             Construct a new empty batch on $M_b$ and insert $J_j$ in this batch;
10:         end if
11:         Update $\widetilde{C_a}$ and $\widetilde{C_b}$;
12:      end if
13:   end for
14:   if the two machine indexes, i.e., $a$ and $b$, are no longer changed then
15:      $flag \leftarrow 0$;
16:   end if
17: end while
18: Output the updated solution and stop.
```

According to the above discussion, the fuzzy ant colony (FACO) algorithm employs n_a ants to construct the solution by calling procedure SC (Algorithm 1) for $T_{\max}$ iterations, where n_a and $T_{\max}$ are input parameters. In each iteration, the solution constructed by each ant is input into the FLO (Algorithm 2) to be improved, and the global-best solution is updated. The global best solution in the last iteration is output from the FACO as the final solution. The proposed FACO algorithm is described in detail as Algorithm 3.

Algorithm 3: The FACO Algorithm.

```
1:  Initialization parameters: $T_{\max}$, $\rho$, $\alpha$ and $\beta$, $n_a$, $\omega$;
2:  t←0; /* The index of the iteration. */
3:  Initialize the pheromone matrix;
4:  while $t < T_{\max}$ do
5:      $a \leftarrow 1$; /* The index of the ant. */
6:      while $(a \le n_a)$ do
7:          Call SC (Algorithm 1) to construct the solution;
8:          Update the pheromone trails according to Eq. (30);
9:          Call FLO (Algorithm 2) to improve the solution;
10:         Update the global best solution;
11:         a←a+1;
12:     end while
13:     t←t+1;
14: end while
15: Output the global best solution and stop.
```

5. Computational experiments

In order to verify the performance of the proposed FACO algorithm, the random testing instances are generated. Extensive comparison is executed between the proposed algorithm and several existing algorithms.

5.1. Experimental design

In the comparative experiments, job sets with eight different job numbers, where $n = \{90, 108, 126, 144, 162, 180, 300, 500\}$ are generated according to the method in Melouk et al. [20]. Moreover, each job set contains 10 testing instances. The fuzzy processing time is extended from the deterministic processing time as follows. Firstly, the processing time p_j^2 of each job J_j is uniformly distributed in [8,48], then the processing time p_j^1 and p_j^3 are set to $p_j^1 = p_j^2 - rand(0, 0.2p_j^2)$ and $p_j^3 = p_j^2 + rand(0, 0.2p_j^2)$, respectively. As a result, the fuzzy processing time, a TFN (p_j^1, p_j^2, p_j^3), of each job J_j is formed with the three values.

The total number of machines is set to 10 [25]. Since in real situation, the large-capacity machines generally cost more than the small-capacity ones, the number of the large-capacity machines is set to be less than that of the small-capacity machines. Specifically, there are three machine capacities, i.e., 10, 25, and 65, denoted by Z^1, Z^2, and Z^3, respectively. The numbers of the machines with the three capacities are set to be five, three and two, respectively.

Three types of job sizes are considered for each n, and the three corresponding job subsets are denoted by J^1,J^2 and J^3, respectively. The jobs in J^1 can be processed on any machine. The jobs in J^2 can be processed on the machines whose capacities are larger than 10. And, the jobs in J^3 can be processed on machines whose capacities are larger than 25. Furthermore, the job numbers in the three job sets, from J^1 to J^3, are set to be $2n/3$, $2n/9$ and $n/9$, respectively, i.e, $|J^1| > |J^2| > |J^3|$, indicating that the number of the small-sized jobs is much more than that of the larger-sized jobs. In this way, the distribution of the jobs with different sizes is consistent with that of the machines with non-identical capacities.

The job size, one key parameter of the experimental settings, is generally generated by the uniform distribution. But if the job sizes are generated according to the uniform distribution, the probability of obtaining a small-sized job is as same as that of a large-sized job. As a result, the number of jobs in a batch is relatively small. Moreover, the number of the large-sized jobs in one batch is less than that of the small-sized jobs in one single batch. Thus, it will make the problem relatively simple. Therefore,

Table 3
Parameters settings.

Factors	Categories and values
Machine capacities	$Z^1 = 10, Z^2 = 25, Z^3 = 65$
Machine numbers	$m_1 = 5, m_2 = 3, m_3 = 2$
Job numbers	$n \in \{90, 108, 126, 144, 162, 180, 300, 500\}$
Job processing time	$p_j^2 \sim U[8, 48]$
	$p_j^1 = p_j^2 - random(0, 0.2p_j^2)$
	$p_j^3 = p_j^2 + random(0, 0.2p_j^2)$
Job sizes	$s_j \sim P(\lambda_i)$

a different distribution is used to generate job sizes. That is, the job sizes are generated according to the Poisson distribution [60]. $\{s_j \mid J_j \in J^1\} \sim P(\lambda_1)$, $\{s_j \mid J_j \in J^2\} \sim P(\lambda_2)$ and $\{s_j \mid J_j \in J^3\} \sim P(\lambda_3)$, where $\lambda_c = Z^c/2$ $(c = 1, 2, 3)$. Specifically, $\lambda_1 = 5$, $\lambda_2 = 12.5$ and $\lambda_3 = 32.5$. As a result, the number of the small-sized jobs is more than that of the large-sized jobs for each instance group. Furthermore, the bounds of the job sizes should be compatible with the machine capacities. To generate the sizes of the jobs in J^c $(c = 1, 2, 3)$, the value of s_j is defined as Eq. (29).

$$s_j = \begin{cases} Z^{c-1}, & s_j < Z^{c-1}, \\ s_j, & Z^{c-1} \le s_j \le Z^c, \\ Z^c, & s_j > Z^c, \end{cases} \tag{29}$$

where $Z^0 = 1$. In addition, to make sure there are enough the small-sized jobs to fill in the large-capacity machines, we randomly select 70% of the jobs from interval $(Z^{c-1}, Z^c/2]$ and 30% from interval $(Z^c/2, Z^c]$ to constitute the final testing job sets.

Table 3 shows the parameter setting for the experiments.

5.2. Parameter settings

There are several parameters in the proposed FACO algorithm, and the setting of parameters has an influence on the search behaviors and the convergence speed of the algorithm. Thus, to obtain better performance, it is desirable to determine suitable values for these parameters firstly [61]. The parameters that affect the FACO algorithm include n_a, $T_{\max}$, α, β, ρ, and ω. To balance between solution quality, convergence speed and computation time, some of the parameters are set, according to the suggestions in [62], to be $n_a = 20$, $T_{\max} = 200$.

To determine the values of the other parameters, two instances from each of the six instance groups are randomly selected to form the preliminary testing instance set. In the preliminary experiments, the FACO algorithm with different values of ρ, Q, and the initial value of pheromone trails φ_{xj} are run on the testing instance set, respectively. And the average results on the testing set are calculated. The FACO algorithm obtains the best results when $\rho = 0.5$, $Q = n$, and the initial value of φ_{xj} equals to 0.1. We adopt this setting in the later experiments. And, $\omega = 0.7$ in the experiments, representing the optimistic attitude of a manager. To determine the weights of the pheromone trails and the heuristic information, different combinations of α and β from intervals (0,1) and [1,10] are selected, respectively. The FACO algorithm is run with different combinations of the values of α and β on the testing set, keeping the other parameters unchanged. The definitions of the pheromone trails and the heuristic information in this paper indicate that the former is less than one and the latter is larger than one. Therefore, we only consider the case with $\alpha \le \beta$. Specifically, 9 combinatorial values of α and β are tested in total. The experimental results are shown in Fig. 3, where the x-axis represents the combinational values of α and β, and the y-axis indicates the average objective values on the 16 testing instances.

As can be seen from Fig. 3, under the fuzzy environment, when $\alpha = 1$ and $\beta > 1$, the average objective value is relatively larger.

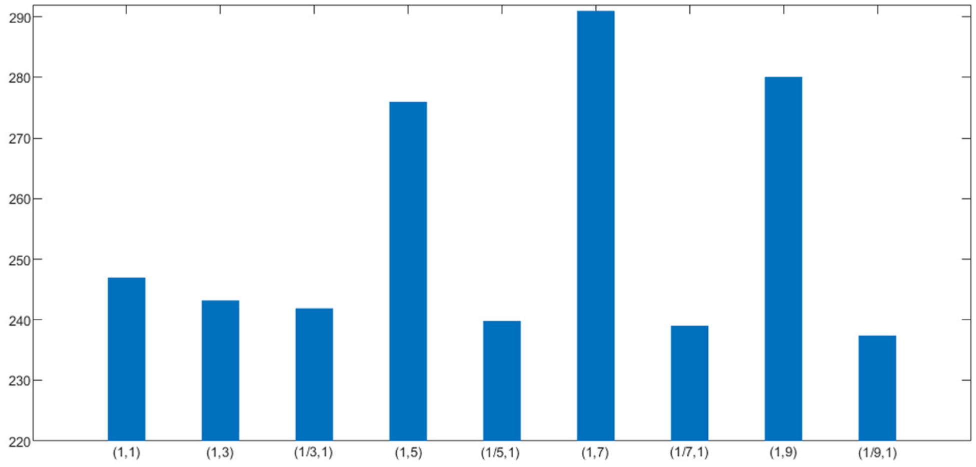

Fig. 3. Experimental results of different combinations of parameters.

Moreover, the average makespan is smaller when $\alpha < 1$ and $\beta = 1$. Furthermore, the average makespan is the least when $(\alpha, \beta) = (1/9, 1)$, indicating that the FACO algorithm is able to obtain the best solution quality with this value combination of α and β. Hence, $\alpha = 1/9$ and $\beta = 1$ are adopted in the following experiments.

5.3. *Experimental results*

In order to verify the performance of the proposed algorithm, two existing algorithms are implemented and compared with, that is, the PSO in Damodaran et al. [63] and the GA in Wang and Chou [24]. However, the size of each job in the problem [24] does not exceed the minimum capacity of the machines, which is different from our problem. As a result, the GA in Wang and Chou [24] cannot directly applied to our problem. Therefore, when generating the chromosome for the schedule of the jobs, the size of each batch needs to be compared with the machine capacity. If the total size of the jobs in some batch is larger than the capacity of the machine that processes the batch, then the job with the largest index is moved out and inserted into the first machine that can accommodate it, one by one, until the machine capacity constraint is satisfied. In this way, a feasible solution is obtained. All the other steps of the GA are the same as those in the original paper. In the problem in Damodaran et al. [63], some jobs cannot be processed on some machines with smaller capacities, and the arrive time of each job is zero, which is similar to our problem. Thus, the PSO can be applied to our problem. To make a fair comparison, we also determine the values of the parameters for the GA and the PSO through a series of preliminary testing experiments, so that the two algorithm can demonstrate the best performance on the studied problem. The method of determining parameter values and the testing instance set are all the same as those of determining the parameter values for the FACO in Section 5.2. Moreover, according to the experimental results, the better parameter settings of the GA and the PSO are all the same as those in the original papers. Specifically, the parameters of the GA are set as follows: both the population size and the maximum iteration are set to 200; the percentages of population for reproduction, crossover, mutation, and immigration are set as 0.1, 0.7, 0.1, and 0.1, respectively. According to the experimental results, the parameter values of the PSO are set as: the maximum iteration and the number of particles are set to 200 and 20, respectively; the inertia weight, the cognitive parameter and the social parameter, are set by 0.6, 2, 2, respectively. At the same time, the optimistic coefficient ω is introduced in both the GA and the PSO to deal with the fuzzy processing time. In addition, to verify the effectiveness of the local optimization, the FACO algorithm without the local optimization, denoted by the UFACO, is also employed as a compared algorithm.

To compare the algorithm performance more objectively and accurately, the FACO, the UFACO, the PSO and the GA, are run 30 times on each instance to obtain the average results, respectively. All the algorithms are programmed in C++ programming language and run on a PC with Intel 3.20 GHz and 8 GB RAM.

The relative distance between the solution and the lower bound is calculated to evaluate the solution quality. The method to compute the lower bound is provided in the Appendix. As defined in Eq. (30), R_A denotes the percentage of the relative distance between the objective value obtained by algorithm A and the lower bound *FLB* for each instance.

$$R_A = (C_{\max}^A/FLB - 1) \times 100 \tag{30}$$

where $C_{\max}^A$ is the makespan obtained by algorithm A, *FLB* represents the corresponding lower bound of the makespan. The smaller the value of R_A, the closer the objective value of algorithm A to the lower bound, meaning that the solution quality of algorithm A is better. The average distance of algorithm A to the lower bound, denoted by $\overline{R}_A$, is the average on the 30 runs for each instance.

The experimental results of different job sets are shown in Tables 4–11. In each table, the first column indicates the instance indexes of each instance group, and the second column shows the values of the lower bounds obtained according to the method in the Appendix for each instance. The $\overline{R}$, average run time $\overline{t}$ and the standard deviation *STD* on the thirty runs are used to measure the performance of the algorithms. In addition, the last row of each table records the average values on each column. Furthermore, the best values of $\overline{R}$ on each testing instance are all shown in bold.

From Tables 4–9, it can be seen that the $\overline{R}$ values of the two ACO-based algorithms, the FACO and the UFACO, are far superior to those of the GA and the PSO. Specifically, the average $\overline{R}$ values of the two ACO-based algorithms are not more than 19, while the average $\overline{R}$ value of the PSO and the GA are more than 33 and 68, respectively. It indicates that the solution quality of the ACO-based algorithms outperforms that of the other two algorithms. Moreover, the FACO algorithm always obtains the minimum average value of $\overline{R}$ on each instance group whatever the job number or the machine number is. That is to say, the solutions found by the FACO are closer to the lower bounds than those of any other compared algorithm.

The PSO algorithm requires the least run time to find the solution. However, the solution quality of the PSO is worse than those of the two ACO-based algorithms. Moreover, it can be seen that the FACO spends a litter more calculation time than the UFACO

Table 4
Comparative results for instances with 90 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	92	**19.13**	3.42	1.07	20.11	3.38	1.51	69.67	2.24	5.26	35.33	**0.52**	1.78
2	92	**24.46**	3.40	0.71	25.76	3.38	1.16	76.41	2.22	7.63	37.83	**0.54**	2.15
3	103	**16.80**	3.37	1.16	17.67	3.34	1.55	62.23	2.16	6.12	31.84	**0.57**	2.25
4	110	**12.91**	3.25	1.14	14.27	3.23	1.42	63.55	2.19	9.26	30.27	**0.54**	1.25
5	104	**16.44**	3.26	1.20	17.02	3.24	1.16	63.17	2.21	7.20	32.98	**0.52**	1.34
6	95	**16.63**	3.42	1.14	17.68	3.37	1.40	76.74	2.22	3.98	35.16	**0.53**	1.26
7	94	**23.62**	3.38	1.55	24.04	3.34	1.07	76.60	2.17	3.94	35.53	**0.53**	0.97
8	100	**16.70**	3.35	0.48	17.90	3.31	0.74	63.80	2.26	6.36	30.10	**0.53**	1.10
9	91	**17.58**	3.34	0.94	18.35	3.29	1.06	81.76	2.18	5.23	40.11	**0.52**	1.51
10	105	**15.33**	3.31	1.37	15.90	3.27	1.64	65.14	2.21	3.63	31.81	**0.53**	1.71
Avg	98.6	**17.96**	3.35	1.08	18.87	3.31	1.27	69.91	2.21	5.86	34.10	**0.53**	1.53

Table 5
Comparative results for instances with 108 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	113	**16.37**	5.31	1.27	16.99	5.26	1.48	70.89	3.06	9.01	36.46	**0.68**	2.10
2	109	**16.97**	5.11	1.27	17.52	5.06	1.45	71.19	3.06	4.25	37.25	**0.67**	1.17
3	113	**15.75**	5.19	1.14	16.37	5.15	0.85	70.27	3.13	9.67	34.60	**0.66**	1.85
4	108	**14.91**	5.14	0.88	16.30	5.10	1.51	67.22	3.22	4.99	36.20	**0.67**	1.29
5	116	**14.57**	5.27	0.99	15.78	5.20	1.25	72.07	3.13	6.00	35.86	**0.69**	2.41
6	111	**17.84**	5.40	1.55	19.19	5.34	1.34	75.32	3.08	7.18	34.68	**0.67**	2.17
7	123	**14.15**	5.15	1.17	14.55	5.00	1.60	65.61	3.08	7.39	32.60	**0.67**	1.29
8	108	**16.48**	5.10	1.14	17.96	5.00	0.97	72.41	3.08	3.88	38.89	**0.72**	2.21
9	107	**16.07**	5.22	1.40	17.57	5.19	1.75	75.14	3.14	7.65	43.46	**0.66**	1.84
10	108	**14.91**	5.14	0.88	16.30	5.10	1.51	67.22	3.22	4.99	36.20	**0.67**	1.29
Avg	111.6	**15.80**	5.20	1.17	16.85	5.14	1.37	70.73	3.12	6.50	36.62	**0.68**	1.76

Table 6
Comparative results for instances with 126 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	132	**13.64**	7.62	0.94	15.00	7.56	0.92	66.59	4.23	6.98	33.56	**0.89**	2.00
2	123	**13.74**	7.33	0.99	14.39	7.31	1.06	68.29	4.24	5.68	37.72	**0.89**	1.90
3	130	**15.15**	7.45	1.42	16.54	7.38	1.43	67.85	4.22	6.60	36.00	**0.88**	1.23
4	132	**13.18**	7.32	1.17	14.24	7.28	1.32	69.39	4.28	5.52	34.55	**0.89**	1.65
5	126	**14.37**	7.54	0.74	15.08	7.43	1.25	72.86	4.21	5.27	36.75	**0.90**	2.45
6	137	**13.28**	6.96	1.40	14.74	6.93	1.48	71.90	4.18	7.74	34.96	**0.92**	1.73
7	131	**15.57**	7.26	1.58	17.33	6.98	2.26	68.86	4.15	6.97	37.40	**0.91**	2.00
8	133	**14.06**	7.70	1.77	15.11	7.36	1.52	68.50	4.22	8.85	33.61	**0.90**	1.34
9	133	**15.41**	7.47	1.72	16.47	7.37	1.10	71.13	4.15	12.13	36.84	**0.91**	1.89
10	129	**13.72**	7.10	1.57	15.04	7.04	1.26	70.70	4.24	7.84	33.33	**0.90**	1.70
Avg	130.6	**14.21**	7.38	1.33	15.39	7.26	1.36	69.61	4.21	7.36	35.47	**0.90**	1.79

Table 7
Comparative results for instances with 144 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	155	**13.35**	10.00	1.06	13.74	9.71	1.57	72.32	5.50	7.17	34.97	**1.10**	2.20
2	145	**12.41**	10.28	1.33	13.17	9.93	1.52	71.45	5.62	6.93	36.69	**1.08**	1.69
3	153	**13.14**	10.26	1.60	13.46	9.99	1.84	64.77	5.72	8.39	31.37	**1.08**	1.63
4	156	**11.86**	10.10	1.58	12.44	9.77	1.65	67.82	5.46	8.90	32.76	**1.11**	1.60
5	159	**14.53**	10.44	1.91	15.03	10.06	2.08	71.57	5.41	6.76	31.89	**1.09**	2.00
6	156	**12.82**	10.07	1.25	13.59	9.68	1.32	70.71	5.45	7.56	31.92	**1.10**	1.69
7	159	**13.71**	9.71	0.92	14.97	9.34	1.93	66.60	5.45	11.09	33.65	**1.10**	3.34
8	160	**15.31**	10.28	2.07	15.81	9.97	2.11	66.50	5.53	7.60	32.25	**1.12**	2.22
9	155	**14.00**	9.68	1.42	15.03	9.41	2.54	65.55	5.43	9.42	30.90	**1.13**	1.73
10	151	**13.05**	10.08	0.95	13.91	9.77	1.05	74.04	5.53	9.03	32.91	**1.09**	1.95
Avg	154.9	**13.42**	10.09	1.41	14.12	9.76	1.76	69.13	5.51	8.29	32.93	**1.10**	2.00

due to the incorporated local optimization strategy. Moreover, the average run time of the FACO is less than 17 s for the instances with 180 jobs and 10 machines, which is reasonable. Although the GA takes less than 10 s to find the solutions to instances with 180 jobs on the average, its average $\bar{R}$ is over 68 percents.

In terms of the *STD*, it can be seen from Table 4 that the FACO defeats the other compared algorithms, indicating that the solutions found by the FACO are most stable among the four algorithms. As in Tables 5–9, the *STD* values of the two ACO-based algorithms are better than those of the GA and the PSO in every instance, with the exception of four out of 50 instances totally. It shows that the

Table 8
Comparative results for instances with 162 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	180	**14.06**	12.85	0.82	14.89	12.73	0.92	65.44	7.14	6.39	31.28	**1.31**	1.77
2	169	**13.79**	13.02	0.95	14.62	12.77	0.48	68.99	7.19	7.99	34.67	**1.30**	2.50
3	169	**12.07**	12.72	1.58	13.31	12.59	1.35	71.18	6.91	6.52	34.79	**1.35**	1.40
4	171	**13.80**	13.66	1.35	14.39	13.25	1.58	71.70	6.77	10.30	34.85	**1.33**	3.10
5	161	**13.85**	12.90	1.32	14.41	12.88	2.21	69.19	6.88	9.42	34.53	**1.32**	2.12
6	179	**13.13**	12.32	1.58	14.53	12.16	1.70	69.44	6.86	9.74	32.79	**1.31**	2.41
7	171	**12.40**	13.26	1.03	13.04	13.05	2.21	69.53	6.83	13.03	34.09	**1.30**	1.77
8	168	**13.81**	13.45	1.48	14.70	13.39	1.49	71.79	6.88	12.75	37.20	**1.28**	1.72
9	177	**12.88**	12.95	1.25	13.95	12.84	2.57	66.55	6.91	8.31	31.24	**1.32**	1.25
10	170	**13.47**	12.98	1.20	14.06	12.71	2.23	71.06	6.80	8.02	36.59	**1.33**	1.99
Avg	171.50	**13.33**	13.01	1.26	14.19	12.84	1.68	69.49	6.92	9.25	34.21	**1.32**	2.00

Table 9
Comparative results for instances with 180 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	193	**11.55**	17.31	1.42	12.18	17.24	2.59	70.26	8.83	10.88	32.38	**1.50**	2.42
2	199	**19.15**	16.68	1.60	19.50	16.58	3.16	73.82	8.95	11.83	32.26	**1.51**	2.25
3	185	**14.11**	17.36	0.74	14.59	17.33	1.89	72.70	8.97	10.17	35.89	**1.48**	2.50
4	199	**12.46**	16.58	1.40	13.02	16.23	1.52	67.04	8.80	11.42	32.36	**1.50**	1.65
5	205	13.51	17.35	1.34	**13.41**	16.69	2.89	65.80	8.75	9.54	29.02	**1.49**	1.35
6	180	**14.11**	17.66	2.91	14.72	16.90	1.58	68.89	8.71	9.50	34.33	**1.47**	2.62
7	187	**12.35**	17.59	1.66	12.62	17.48	1.90	65.03	8.57	5.46	33.53	**1.49**	2.41
8	199	**11.66**	15.80	1.03	12.66	15.72	1.62	70.20	8.63	9.50	31.96	**1.54**	2.55
9	184	**10.92**	17.40	0.88	11.47	17.28	1.20	70.87	9.17	10.07	34.89	**1.54**	2.20
10	189	**13.23**	16.31	0.82	13.97	16.28	1.58	64.18	8.76	9.98	33.92	**1.51**	1.29
Avg	192	**13.31**	17.00	1.38	13.81	16.77	1.99	68.88	8.81	9.83	33.06	**1.50**	2.12

Table 10
Comparative results for instances with 300 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	323	**9.94**	22.57	1.84	10.65	21.47	2.24	71.70	13.95	14.97	32.80	**1.50**	2.49
2	312	**10.29**	22.40	1.45	10.65	21.63	2.00	70.43	12.67	17.08	33.71	**1.42**	2.83
3	322	**12.75**	22.43	2.35	13.14	20.66	3.46	74.14	12.82	17.34	33.44	**1.43**	3.02
4	321	**11.10**	23.40	2.47	11.77	20.96	2.86	69.85	11.88	14.20	31.71	**1.41**	2.31
5	329	**11.27**	21.46	2.41	11.78	20.21	3.28	69.79	12.25	13.68	32.17	**1.43**	2.51
6	324	**9.71**	21.76	1.50	10.10	20.62	1.88	68.80	12.48	13.89	33.65	**1.51**	2.53
7	327	**8.95**	21.21	1.87	9.50	21.75	1.92	64.63	12.44	14.40	29.14	**1.69**	2.10
8	303	**11.21**	22.95	2.51	12.07	22.62	2.39	72.28	11.57	16.07	35.49	**1.42**	2.54
9	328	**10.56**	23.17	3.12	10.79	22.31	2.74	71.42	12.20	13.74	32.17	**1.45**	3.58
10	307	**12.20**	23.74	2.22	12.89	22.23	2.40	69.54	11.40	16.98	34.42	**1.43**	3.01
Avg	319.60	**10.80**	22.51	2.18	11.34	21.45	2.52	70.26	12.37	15.23	32.87	**1.47**	2.69

Table 11
Comparative results for instances with 500 jobs.

Ins. no.	*FLB*	FACO			UFACO			GA			PSO		
		$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*	$\bar{R}$	$\bar{t}$	*STD*
1	507	**9.44**	66.60	2.80	9.77	65.26	3.57	73.23	36.16	26.35	33.98	**3.40**	2.91
2	533	**8.32**	65.01	2.06	8.41	64.40	2.57	70.66	44.70	24.87	32.87	**3.23**	2.34
3	527	**8.32**	74.05	2.29	8.46	67.11	2.33	68.08	43.04	28.47	33.42	**3.62**	3.43
4	521	**8.57**	67.86	2.74	8.69	65.46	2.89	72.56	33.14	24.40	35.18	**3.23**	2.39
5	547	**8.59**	64.18	3.34	9.09	63.98	2.56	68.57	31.94	22.31	30.22	**3.34**	2.34
6	527	**10.32**	77.06	3.64	10.20	65.04	4.34	70.48	31.89	21.44	33.85	**3.45**	2.65
7	512	**8.76**	74.38	2.50	8.87	64.51	3.12	73.93	31.95	23.75	35.36	**3.28**	3.15
8	547	**13.19**	77.20	5.02	13.31	65.23	5.82	74.61	31.89	23.46	32.22	**4.15**	3.74
9	526	**8.05**	74.34	2.02	8.51	64.53	2.27	73.85	32.11	20.51	34.07	**3.64**	3.81
10	518	**8.26**	83.37	2.82	8.78	66.57	3.14	69.23	32.02	25.25	32.70	**3.33**	2.93
Avg	526.50	**9.18**	72.40	2.92	9.41	65.21	3.26	71.52	34.88	24.08	33.39	**3.47**	2.97

solution construction method used in the ACO-based algorithms is effective and efficient. Moreover, the average *STD* values of the FACO on each instance group are less than any of the other algorithms. In addition, the GA shows the worst *STD* results among all the algorithms, as well as the solution quality.

Since the difference in the average run time between the FACO and the UFACO is less than one second, the time spent on the local optimization strategy is negligible. Furthermore, both the average ratios and the average *STD* values of the FACO are better than those of the UFACO. It demonstrates that the local optimization of the FACO significantly improves the solution quality of the algorithm.

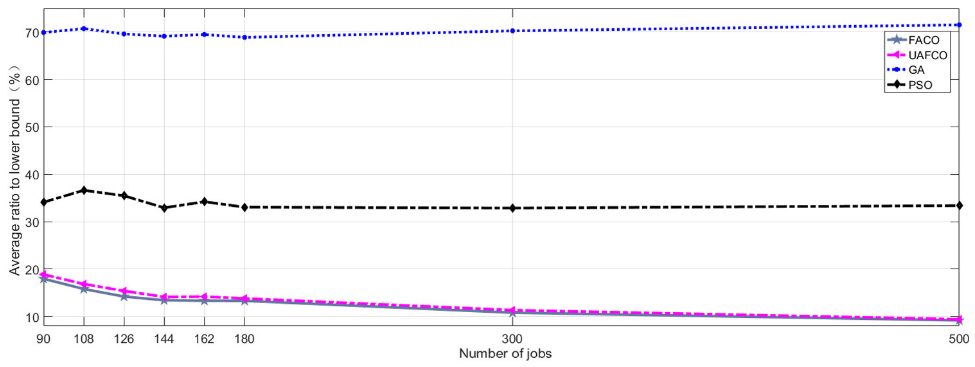

Fig. 4. The comparison of average performance.

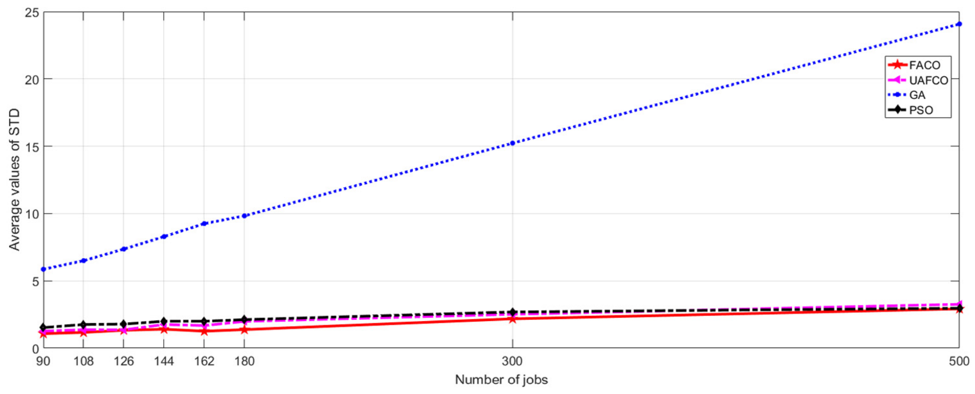

Fig. 5. The comparison of average *STD*.

The Tables 10 and 11 present the results obtained by the algorithms for instances with 300 and 500 jobs, it shows that large instances of the $\overline{R}$ values of the two ACO-based algorithms are far better than that of GA and PSO in each instance. At the same time, FACO beats UFACO in all instances. GA has the worst performance among all algorithms in terms of the average $\overline{R}$ values. In terms of the $\bar{t}$, the two ACO-based algorithms are in a reasonable extent. The *STD* values of the two ACO-based algorithms are better than other algorithms, with the exception of three in instances with 500 jobs. Furthermore, the average *STD* values of the FACO on each instance group are less than any of the other algorithms. Therefore, it can get conclusions that the FACO is superior to other algorithms in larger instances.

Figs. 4 and 5 show the average performance of solution quality and the average *STD* of all the algorithms on each instance group, respectively. In both figures, the *x*-axis represents the numbers of jobs and the *y*-axis represents the average $\overline{R}$ values and the average *STD*, respectively, on each instance group. It can be observed from Fig. 4 that the deviations between the solutions and the lower bounds of all the algorithms gradually decrease with the increase of job numbers. And the trend of the two ACO-based algorithms is more obvious. That is to say, the FACO and the UFACO can obtain better solutions with the number of jobs increasing. When the job number reaches 500, the average $\overline{R}$ of the FACO converges to about 9 percents. It can be seen from Fig. 5 that the average *STD* values of the GA sharply increase with the increase of job numbers, especially for large scale problems. The other three algorithms show similar tendency of *STD*. Specifically, all the values of average *STD* of algorithms FACO, UAFCO, and PSO are almost stable when the numbers of jobs increase. However, the average ratios of the PSO is much worse than those of the FACO and the UAFCO when the problem scales increase. In terms of the values of average *STD*, the FACO slightly outperforms the UAFCO on all the instance groups, indicating that the local optimization is effective.

The above experimental results have clearly demonstrated the better search ability of the FACO, not only the convergence performance but also the quality of the solutions.

5.4. Statistical analysis

In order to further verify the performance of the proposed algorithm, the FACO and the other comparative algorithms are compared by a non-parametric statistical analysis method, i.e., the Wilcoxon signed-rank test. The Wilcoxon signed-rank test can compare two related samples or matched samples without the distribution of the date to prove the effectiveness of the samples. Thus, it can be applied to test the significance of the algorithms [64].

In the experiments, 160 samples, i.e., 20 instances of each instance group are selected firstly. Then, the average objective values on the 30 runs of each sample are used as a measurement for the statistical analysis. Wilcoxon's test is conducted with a 5% significance level by using the statistical software package, i.e., Statistical Product and Service Solutions (SPSS). After performing Wilcoxon test on the solutions obtained on the instances will have three assessment values for each pair of two algorithms: the R^+ (the sum of ranks where the first algorithm outperforms the second one), the R^- (the sum of ranks for the opposite case), and $\rho - value$. The $\rho - value$ and z value are used to analyze the results [65]. The $\rho - value$ indicates the level of significance of the hypothesis test. If $\rho - value$ is less than θ (the significance level), the hypothesis is statistically significant with 100*(1-θ)% confidence level. If z value exceeds the critical values (from -1.96 to $+1.96$), the null hypothesis is rejected [65].

The nonparametric statistical analysis performed using the Wilcoxon signed-rank test is presented in Table 12. From the

Table 12
Wilcoxon's test results for solution quality considering different combinations of algorithms.

Algorithm pairs		Solution quality			
Algorithm1	Algorithm2	R^+	R^-	z	$\rho - value$
FACO	GA	12880.00	0.00	−10.972	0.00
FACO	PSO	12880.00	0.00	−10.972	0.00
FACO	UFACO	12880.00	0.00	−10.973	0.00
UFACO	GA	12880.00	0.00	−10.972	0.00
UFACO	PSO	12880.00	0.00	−10.972	0.00

results, it is obvious that the FACO outperforms the GA, the PSO and the UFACO with level of significance $\theta = 0.05$ since the $\rho - value$ of each of the pairs are below 0.05. Similarly, we can see UFACO outperforms GA and PSO with same level of significance. On the other hand, The z value exceeds the critical values. This indicates that the comparison are statistically significant.

6. Conclusions and future work

The fuzzy scheduling problem is closer to the real production than the deterministic scheduling problem. Therefore, in this paper, we investigate the problem of scheduling the jobs with fuzzy processing time and non-identical job sizes on parallel BPMs with non-identical capacities to minimize the makespan. A fuzzy ACO-based algorithm is proposed to solve the problem. According to the objective $\widetilde{C}_{\max}$, two candidate lists are incorporated into the algorithm to reduce the search space. The problem-oriented heuristic information is designed to guide the search. Finally, a local optimization strategy is used to improve the solution quality. In addition, the optimism coefficient is introduced to provide the decision values. The experimental results show that the proposed algorithm significantly outperforms the compared algorithms because the proposed algorithm is able to find better solutions with good convergence performance.

Future research can be extended to more complex manufacturing environments, such as non-identical job arrivals, different machine capacities, online approach to scheduling, unrelated parallel BPMs, different shop environments and the cloud environments. On one hand, there exists more uncertainty in cloud manufacturing than in classical manufacturing, such as fuzzy job arrival time, fuzzy setup time, and so on. On the other hand, the virtualization of manufacturing equipments and the distinct positions of orders bring new challenges for fuzzy scheduling. Additionally, the fuzzy scheduling problems with multiple objectives are also of great interests. We can also consider applying ACO to learning sequences, learning effect, and deterioration effect in a fuzzy scheduling problems.

Acknowledgments

The work of the first author is supported by the National Natural Science Foundation of China under grant 71601001, the Humanity and Social Science Youth Foundation of Ministry of Education of China under grant 15YJC630041, the Science Foundation of Anhui Province, China under grant 1608085MG154, the Natural Science Foundation of Anhui Provincial, China Education Department under grant KJ2015A062. The work of the last two authors are supported by the National Natural Science Foundation of China under grants 71871076 and 71671168, respectively.

Appendix A. Notations

Indexes	
J_j	the job index, $j = 1, 2\ldots, n$
B_{ki}	the batch index, $k = 1, 2\ldots, d$; $i = 1, 2\ldots, m$
M_i	the machine index, $i = 1, 2\ldots, m$
Parameters	
$\widetilde{p_j} = (p_j^1, p_j^2, p_j^3)$	the fuzzy processing time of job J_j
s_j	the size of job J_j
$\widetilde{P_{ki}} = (P_{ki}^1, P_{ki}^2, P_{ki}^3)$	the fuzzy processing time of the kth batch on machine M_i
S_{ki}	the size of the kth batch on machine M_i
$\widetilde{C_i} = (C_i^1, C_i^2, C_i^3)$	the fuzzy completion time of machine M_i
Z_i	the capacity of machine M_i
Decision variable	
X_{jki}	job J_j is grouped into batch B_{ki} or not
Objective	
$\widetilde{C}_{\max}$	the maximum fuzzy completion time

Appendix B. A fuzzy lower bound

As mentioned in the literature review, Uzsoy [18] has proved that SSBN is NP-hard. The studied problem is more complicated than the SSBN, and thus, is at least NP-hard. For the NP-hard problems, it is able to find the lower bound to evaluate the correctness, as well as the solution quality obtained by heuristics. Therefore, a method to calculate the fuzzy lower bound is provided.

The fuzzy lower bound can be obtained through relaxing the job sizes. To facilitate understanding, we assume that there are three different machine capacities denoted by Z^1, Z^2, and Z^3 ($Z^1 < Z^2 < Z^3$), and the corresponding machine numbers are m_1, m_2, and m_3, respectively. The generalization to more than three machine capacities is easy to see. Firstly, all the jobs are divided into three subsets, denoted by J^1, J^2, and J^3, according to the machine capacities, where $J = J^1 \cup J^2 \cup J^3$, $J^1 = \{J_j | s_j \leq Z^1\}$, $J^2 = \{J_j | Z^1 < s_j \leq Z^2\}$, $J^3 = \{J_j | Z^2 < s_j \leq Z^3\}$. Secondly, each job J_j is relaxed into s_j unit-size jobs whose processing time is $\widetilde{p_j}$. Thirdly, according to the dispatch rules, the unit-size jobs are grouped into batches which are scheduled on the machines. Finally, the fuzzy maximum completion time is computed as a lower bound of the studied problem.

When computing the lower bound, the fuzzy processing time of one batch B_{ki} equals to the maximum fuzzy processing time of all the jobs in this batch after the processing time of these jobs is compared by the "$\vee$" operation for $|B_{ki}| - 1$ times. Let $\widetilde{F_1} = (F_1^1, F_1^2, F_1^3)$, defined in Eq. (31), denote the maximum fuzzy job processing time among all the jobs in the job set J. As defined in Eq. (32), F_1 denotes the value obtained by using the optimistic coefficient ω.

$$\widetilde{F_1} = \{\widetilde{p_{j1}} \vee \widetilde{p_{j2}} | J_{j1}, J_{j2} \in J, j1 \neq j2\} \tag{31}$$

$$F_1 = \left\lceil (\omega * F_1^1 + F_1^2 + (1-\omega) F_1^3)/2 \right\rceil \tag{32}$$

Let $\widetilde{F_2} = (F_2^1, F_2^2, F_2^3)$ represents the maximum fuzzy completion time obtained by allocating the unit-size jobs in J^3 to the machines with capacity Z^3. $F_2^x(x = 1, 2, 3)$ is formulated as Eq. (33). As defined in Eq. (34), F_2 denotes the value obtained by using the optimistic coefficient ω.

$$F_2^x = (\sum_{J_j \in J^3} p_i^x * s_j)/(m_3 * Z^3), (x = 1, 2, 3) \tag{33}$$

$$F_2 = \lceil(\omega * F_2^1 + F_2^2 + (1-\omega)F_2^3)/2\rceil \tag{34}$$

$\widetilde{F_3} = (F_3^1, F_3^2, F_3^3)$ denotes the maximum fuzzy completion time obtained by scheduling the unit-size jobs in J^2 and J^3 on the machines with capacities Z^2 and Z^3, where $F_3^x(x = 1, 2, 3)$ is formulated as Eq. (35). F_3 defined in Eq. (36) denotes the value obtained by using the optimistic coefficient ω.

$$F_3^x = (\sum_{J_j \in J^2 \cup J^3} p_j^x * s_j)/\sum_{c=2}^{3}(m_i * Z^c), (x = 1, 2, 3) \tag{35}$$

$$F_3 = \lceil(\omega * F_3^1 + F_3^2 + (1-\omega)F_3^3)/2\rceil \tag{36}$$

After the unit-size jobs in J^1, J^2 and J^3, are assigned on the machines with capacities of Z^1, Z^2 and Z^3, the maximum fuzzy completion time, denoted by (F_4^1, F_4^2, F_4^3), is calculated as Eq. (37). F_4 in Eq. (38) denotes the value obtained by using the optimistic coefficient ω.

$$F_4^x = (\sum_{J_j \in J^1 \cup J^2 \cup J^3} p_j^x * s_j)/\sum_{c=1}^{3}(m_i * Z^c), (x = 1, 2, 3) \tag{37}$$

$$F_4 = \lceil(\omega * F_4^1 + F_4^2 + (1-\omega)F_4^3)/2\rceil \tag{38}$$

Based on the above discussion, the fuzzy lower bound is computed according to Eq. (39).

$$FLB = \max\{F_1, F_2, F_3, F_4\} \tag{39}$$

References

[1] M. Mathirajan, A.I. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, Int. J. Adv. Manuf. Technol. 29 (9–10) (2006) 990–1001.
[2] C.Y. Lee, R. Uzsoy, L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, Oper. Res. 40 (4) (1992) 764–775.
[3] H.J. Zimmermann, Description and optimization of fuzzy systems, Int. J. Gen. Syst. 2 (1) (1975) 209–215.
[4] H. Tanaka, T. Okuda, K. Asai, On fuzzy-mathematical programming, J. Cybern. 3 (4) (1974) 37–46.
[5] S. Chanas, A. Kasperski, Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates, Eng. Appl. Artif. Intell. 14 (3) (2001) 377–386.
[6] T. Itoh, H. Ishii, Fuzzy due-date scheduling problem with fuzzy processing time, Int. Trans. Oper. Res. 6 (6) (1999) 639–647.
[7] 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, Appl. Soft Comput. 10 (3) (2010) 919–925.
[8] X. Li, H. Ishii, T. Masuda, Single machine batch scheduling problem with fuzzy batch size, Comput. Ind. Eng. 62 (3) (2012) 688–692.
[9] B. Cheng, K. Li, B. Chen, Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization, J. Manuf. Syst. 29 (1) (2010) 29–34.
[10] J. Peng, Parallel machine scheduling models with fuzzy processing times, Inform. Sci. 166 (1) (2004) 49–66.
[11] S. Balin, Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation, Inform. Sci. 181 (17) (2011) 3551–3569.
[12] S. Balin, Non-identical parallel machine scheduling with fuzzy processing times using genetic algorithm and simulation, Int. J. Adv. Manuf. Technol. 61 (9–12) (2012) 1115–1127.
[13] S.A. Torabi, N. Sahebjamnia, S.A. Mansouri, M.A. Bajestani, A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem, Appl. Soft Comput. 13 (12) (2013) 4750–4762.
[14] W.C. Yeh, P.J. Lai, W.C. Lee, M.C. Chuang, Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects, Inform. Sci. 269 (2014) 142–158.
[15] M. Naderi-Beni, E. Ghobadian, S. Ebrahimnejad, R. Tavakkoli-Moghaddam, Fuzzy bi-objective formulation for a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times, Int. J. Prod. Res. 52 (19) (2014) 5799–5822.
[16] J. Behnamian, Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling, Int. J. Adv. Manuf. Technol. 75 (5–8) (2014) 883–895.
[17] T.W. Liao, P. Su, Parallel machine scheduling in fuzzy environment with hybrid ant colony optimization including a comparison of fuzzy number ranking methods in consideration of spread of fuzziness, Appl. Soft Comput. 56 (2017) 65–81.
[18] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes, Int. J. Prod. Res. 32 (7) (1994) 1615–1635.
[19] C.Y. Lee, Minimizing makespan on a single batch processing machine with dynamic job arrivals, Int. J. Prod. Res. 37 (1) (1999) 219–236.
[20] S. Melouk, P. Damodaran, P.Y. Chang, Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing, Int. J. Prod. Econ. 87 (2) (2004) 141–147.
[21] A.H. Kashan, B. Karimi, F. Jolai, An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes, Eng. Appl. Artif. Intell. 23 (6) (2010) 911–922.
[22] T.P. Chung, H. Sun, C.J. Liao, Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint, Comput. Ind. Eng. 113 (2017) 859–870.
[23] J. Pei, X. Liu, P.M. Pardalos, M. Kong, Single-machine scheduling with learning effect and resource-dependent processing times in the serial-batching production, Appl. Math. Model. (2017).
[24] H.M. Wang, F.D. Chou, Solving the parallel batch-processing machines with different release times, job sizes, and capacity limits by metaheuristics, Expert Syst. Appl. 37 (2) (2010) 1510–1521.
[25] Z.H. Jia, X.H. Li, J.Y.T. Leung, Minimizing makespan for arbitrary size jobs with release times on p-batch machines with arbitrary capacities, Future Gener. Comput. Syst. 67 (2017) 22–34.
[26] J.E.C. Arroyo, J.Y.T. Leung, An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times, Comput. Ind. Eng. 105 (2017) 84–100.
[27] P.Y. Mok, C.K. Kwong, W.K. Wong, Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory, European J. Oper. Res. 177 (3) (2007) 1876–1893.
[28] P. Alcan, H. BaşL1Gil, A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem, Adv. Eng. Softw. 45 (1) (2012) 272–280.
[29] O. Engin, S. Gözen, Parallel machine scheduling problems with fuzzy processing time and fuzzy duedate: an application in an engine valve manufacturing process, J. Mult.-Valued Logic Soft Comput. 15 (2–3) (2009) 107–123.
[30] S. Molla-Alizadeh-Zavardehi, R. Tavakkoli-Moghaddam, F.H. Lotfi, A modified imperialist competitive algorithm for scheduling single batch-processing machine with fuzzy due date, Int. J. Adv. Manuf. Technol. 85 (9–12) (2016) 2439–2458.
[31] A.D. Yimer, K. Demirli, Fuzzy scheduling of job orders in a two-stage flowshop with batch-processing machines, Internat. J. Approx. Reason. 50 (1) (2009) 117–137.
[32] M. Dorigo, V. Maniezzo, A. Colorni, V. Maniezzo, Positive feedback as a search strategy, 1991.
[33] M. Dorigo, Optimization, Learning and Natural Algorithms (Ph.D. thesis), Politecnico di Milano, 1992.
[34] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.
[35] M. Dorigo, T. Stützle, The ant colony optimization metaheuristic: Algorithms, applications, and advances, in: Handbook of Metaheuristics, 2003, pp. 250–285.
[36] R. Xu, H. Chen, X. Li, Makespan minimization on single batch-processing machine via ant colony optimization, Comput. Oper. Res. 39 (3) (2012) 582–593.
[37] B.Y. Cheng, J.Y.T. Leung, K. Li, Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method, Comput. Ind. Eng. 83 (2015) 217–225.
[38] B. Du, H.P. Chen, G.Q. Huang, H.D. Yang, Preference Vector Ant Colony System for Minimising Make-Span and Energy Consumption in a Hybrid Flow Shop, Springer, London, 2011, pp. 279–304.
[39] D.P. Mahato, R.S. Singh, Maximizing availability for task scheduling in on-demand computing-based transaction processing system using ant colony optimization, Concurr. Comput.: Pract. Exper. 30 (11) (2018) e4405.
[40] V. Riahi, M. Kazemi, A new hybrid ant colony algorithm for scheduling of no-wait flowshop, Oper. Res. 18 (1) (2018) 55–74.

[41] S. Zhang, T.N. Wong, Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning, J. Intell. Manuf. 29 (3) (2018) 585–601.
[42] B. Zhao, J. Gao, K. Chen, K. Guo, Two-generation pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines, J. Intell. Manuf. 29 (1) (2018) 93–108.
[43] C.C. Chyu, W.S. Chang, Optimizing fuzzy makespan and tardiness for unrelated parallel machine scheduling with archived metaheuristics, Int. J. Adv. Manuf. Technol. 57 (5–8) (2011) 763.
[44] J.J. Palacios, I. Gonzalez-Rodrguez, C.R. Vela, J. Puente, Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop, Fuzzy Sets and Systems 278 (2015) 81–97.
[45] K.Z. Gao, P.N. Suganthan, Q.K. Pan, T.J. Chua, C.S. Chong, T.X. Cai, An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time, Expert Syst. Appl. 65 (2016) 52–67.
[46] K.Z. Gao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, A. Sadollah, Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion, Knowl.-Based Syst. 109 (2016) 1–16.
[47] F. Geyik, K. Elibal, A linguistic approach to non-identical parallel processor scheduling with fuzzy processing times, Appl. Soft Comput. 55 (2017) 63–71.
[48] S. Petrovic, C. Fayad, D. Petrovic, Sensitivity analysis of a fuzzy multiobjective scheduling problem, Int. J. Prod. Res. 46 (12) (2008) 3327–3344.
[49] A.D. Yimer, K. Demirli, Minimizing weighted flowtime in a two-stage flow shop with fuzzy setup and processing times, in: Fuzzy Information Processing Society 2006. NAFIPS 2006. Annual meeting of the North American, vol. 1, no. 2 2006, June, pp. 708–713.
[50] R.E. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment, Manage. Sci. 17 (4) (1970) B–141.
[51] C. Wang, D. Wang, W.H. Ip, D.W. Yuen, The single machine ready time scheduling problem with fuzzy processing times, Fuzzy Sets and Systems 127 (2) (2002) 117–129.
[52] D. Lei, Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling, Appl. Soft Comput. 12 (8) (2012) 2237–2245.
[53] M. Sakawa, T. Mori, An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate, Comput. Ind. Eng. 36 (2) (1999) 325–341.
[54] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1979) 287–326.
[55] J. Wang, A fuzzy robust scheduling approach for product development projects, European J. Oper. Res. 152 (1) (2004) 180–194.
[56] Kim Kuk, S. Park Kyung, Ranking fuzzy numbers with index of optimism, Fuzzy Sets and Systems 35 (2) (1990) 143–150.
[57] T.S. Liou, M.J.J. Wang, Ranking fuzzy numbers with integral value, Fuzzy Sets and Systems 50 (3) (1992) 247–255.
[58] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, Boston, MA, 2011, pp. 36–39.
[59] E.M. Loiola, N.M.M. de Abreu, P.O. Boaventura-Netto, P. Hahn, T. Querido, A survey for the quadratic assignment problem, European J. Oper. Res. 176 (2) (2007) 657–690.
[60] J.Q. Wang, J.Y.T. Leung, Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan, Int. J. Prod. Econ. 156 (2014) 325–331.
[61] Z.H. Jia, J.Y.T. Leung, An improved meta-heuristic for makespan minimization of a single batch machine with non-identical job sizes, Comput. Oper. Res. 46 (2014) 49–58.
[62] Z.H. Jia, K. Li, J.Y.T. Leung, Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities, Int. J. Prod. Econ. 169 (2015) 1–10.
[63] P. Damodaran, D.A. Diyadawagamage, O. Ghrayeb, M.C. Vélez-Gallego, A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines, Int. J. Adv. Manuf. Technol. 58 (9–12) (2012) 1131–1140.
[64] A. Biswas, B. Biswas, Analyzing evolutionary optimization and community detection algorithms using regression line dominance, Inform. Sci. 396 (2017) 185–201.
[65] K. Khosravi, B.T. Pham, K. Chapi, A. Shirzadi, H. Shahabi, I. Revhaug, D.T. Bui, A comparative assessment of decision trees algorithms for flash flood susceptibility modeling at haraz watershed, northern Iran, Sci. Total Environ. 627 (2018) 744–755.