APEM journal
Advances in Production Engineering & Management
Volume 13 | Number 1 | March 2013 | pp 44–56
https://doi.org/10.14743/apem2018.1.272
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Solving fuzzy flexible job shop scheduling problem based on fuzzy satisfaction rate and differential evolution

Ma, D.Y.[a,b], **He, C.H.**[c], **Wang, S.Q.**[d], **Han, X.M.**[b], **Shi, X.H.**[d,e,*]

[a]Business School, Jilin University, Changchun, P.R. China
[b]School of Computer Science and Engineering, Changchun University of Technology, Changchun, P.R. China
[c]School of Electronics Engineering and Computer Science, Peking University, Beijing, P.R. China
[d]Key Laboratory of Symbol Computation and Knowledge Engineering, Ministry of Education, Changchun, P.R. China
[e]College of Computer Science and Technology, Jilin University, Changchun, P.R. China

ABSTRACT

Focused on a variety of JSSP considered flexibility and fuzziness, namely the fuzzy flexibility JSSP (FfJSSP), a novel method based on fuzzy satisfaction rate and differential evolution (DE) algorithm is proposed in this paper. In the method, the fuzzy membership functions' parameters are determined according to normal distribution for maximum satisfaction rate calculation. Then a DE algorithm is proposed by well designing the coding for the problem and extending the related operators on the coding. A local exploring search for operation and machine parts of the coding is also introduced to improve the performance of the method. Experimental results show that our proposed method is effective compared with other five popular existed methods. Comparisons between different mutation and crossover strategies are also performed. Numerical results show that the proposed method could be applied to real FfJSSP problems.

ARTICLE INFO

Keywords:
Job shop scheduling problem (JSSP);
Fuzzy flexible JSSP (FfJSSP);
Differential evolution algorithm;
Normal distribution;
Local search

**Corresponding author:*
shixh@jlu.edu.cn
(Shi, X.H.)

Article history:
Received 10 February 2018
Revised 25 February 2018
Accepted 1 March 2018

1. Introduction

Job shop scheduling problem (JSSP) is one of the best known combinatorial optimization problems, and has received tremendous attention in different fields. Lots of real problems could be modelled into JSSP, such as cargo port scheduling [1], health care activity scheduling [2], electric vehicle charging station arrangement [3], and so on. In conventional JSSP, each operation could access to only one machine. But in flexible JSSP (fJSSP), it could access to multiple machines, which makes fJSSP more complex than JSSP. In recent years, fJSSP is attracting more and more attentions in the related fields. The existing methods for fJSSP could be divided into two categories: evolution algorithm and non-evolution algorithm. For non-evolution algorithms, Wu and Weng proposed a multi-agent scheduling method with earliness and tardiness objectives in flexible job shops [4]. Sahin *et al*. also developed a multi-agent based system to simultaneous schedule flexible machine groups and material handling system working under a manufacturing dynamic environment [5]. Zhang *et al*. developed a dynamic game theory based two-layer scheduling method to reduce makespan, the total workload of machines and energy consumption to achieve real-time multi-objective flexible job shop scheduling [6]. While the evolution algorithm based methods were a lot more than non-evolution algorithms. For example, Li and Pan presented a novel discrete artificial bee colony algorithm for solving the multi-objective flexible job

shop scheduling problem with maintenance activities in 2014 [7]. Gao proposed a hybrid genetic and variable neighborhood descent algorithm [8]. Li and Gao [9] and Zhang *et al*. [10] hybridized the genetic algorithm (GA) and tabu search (TS) and both developed a GA and TS based algorithm genetic algorithm for flexible job shop. Focused on sequence flexibility, Huang *et al*. proposed an improved GA method to minimize the makespan [11]. By defining the objective function as the weighted combination of the maximum of the completion time (makespan) and the mean of earliness and tardiness, Gao *et al*. proposed a discrete harmony search (DHS) algorithm to solve FJSP [12]. Li and Pan adopted an effective discrete chemical-reaction optimization to solve the multi-objective fJSSP [13], and Ozgüven *et al*. proposed a concept of removing invalid nodes before modeling, and considering a new problem with sequence dependent setup times [14]. Latest developments is that Yuan and Xu proposed a hybrid differential evolution (HDE) algorithms to minimize the makespan [15], and Xu *et al*. proposed an effective teaching-learning based optimization algorithm to solve the fJSSP with fuzzy processing time [16]. Focused on the machine part manufacturing, Supsomboon and Vajasuvimon presented a simulation model which is beneficial to assist in performance improving [17].

Fuzzy job shop scheduling problem (FJSSP) is also an extension of conventional JSSP in which the uncertain parameters of practical manufacturing are considered. For the fuzzy description of problem, many researchers represented fuzzy processing time by fuzzy triples, for example, Mehrabad *et al*. described the fuzzy delivery time with fuzzy number [18], Lei [19] and Kilic [20] constructed the objective function according to the intersected area of membership function and customer's requirements. Usually, FJSSP is solved by fuzzy theory combined with computational intelligence methods rather than traditional methods. For instance, Lei developed an achieve Pareto-optimality with particle swarm algorithm [19], Niu *et al*. proposed a fuzzy particle swarm algorithm [21], Kilic adopted the artificial ant colony algorithm to solve this problem [20], Zhang *et al*. put forward a generalized IFS to process the additive operation and to compare the operation between two IFSs [22], and Gao *et al*. proposed a discrete harmony search (DHS) algorithm for FJSSP [23]. By using multiobjective evolutionary algorithm combined with a novel dominance-based tabu search method to optimize both the expected makespan and the predictive robustness of the fuzzy schedule, José Palacios *et al*. developed a robust multiobjective optimization algorithm for fuzzy job shop scheduling problem [24].

Fuzzy flexible job shop scheduling problem (FfJSSP) is the combination of FJSSP and fJSSP. Comparing fJSSP and FJSSP, the studies of FfJSSP are quite few. Kacem *et al*. proposed a pareto-optimality approach for FfJSSP with hybridization of evolutionary algorithms and fuzzy logic [25]. Zheng *et al*. solved the multi-objective FfJSSP with swarm-based neighborhood search algorithm [26]. Lei solved the FfJSSP with the objective of minimizing fuzzy scheduling time with co-evolutionary algorithm [27]. Gao *et al*. developed an artificial bee colony algorithm or flexible job-shop scheduling problem with fuzzy processing time [28]. In the existing methods, the membership functions are mostly determined by experience in FfJSSP or even in FJSSP. In order to minimize the influence of human factors, Azadeh *et al*. adopted random probability normal distribution, got the membership function with historical data, and solved multi-product assembly shop problems with fuzzy simulation [29].

In this paper, focused on FfJSSP, a differential evolution algorithm is proposed based on fuzzy satisfaction rate. Firstly, according to the experience data we construct the fuzzy triple membership functions of machine operation following the normal probability distribution. Then the objective function is defined by fuzzified machine processing time and costumer's expected time. At last, a novel Differential evolution (DE) algorithm is developed for scheduling process by defining a new representation of scheduling and corresponding operations. We also introduce a local search process to improve the performance of the method. To test the effectiveness of our proposed method, it is applied to two experiments for comparison. Numerical results show that our method is effective and practical.

2. Fuzzy flexible job shop scheduling problem

Compared with JSSP, one operation in FfJSSP problem could be processed on multiple machines rather than only on one machine. On the other hand, the processing time of an operation is fuzzy but not crisp, as well as the costumer's requirements. Therefore, FfJSSP is more complex than JSSP. The fuzziness of the problem is always described by fuzzy theory, for example, using fuzzy 3-tuples to represent processing times and fuzzy 4-tuples to represent costumer's requirements.

Supposed we have n jobs, m machines, denote J_i as Job i ($I \in [1, n]$), M_i as Machine i ($I \in [1, m]$), Num_i as the number of operations of J_i, $Q_{i,j}$ as the j-th operation of J_i, $S_{i,j}$ as the beginning time of $Q_{i,j}$, $C_{i,j}$ as the completing time of $Q_{i,j}$, $T_{i,j,k}$ as the processing time of $Q_{i,j}$ processed on machine k, $Q_{i,j,k}$ as the average $T_{i,j,k}$, $\sigma_{i,j,k}$ as the standard deviation of $T_{i,j,k}$. Then FfJSSP should satisfy some constraints as follows:

- Each operation of any jobs must be processed once and only once

$$\sum_{i=1}^{n} \sum_{j=1}^{Num} \sum_{k=1}^{m} X_{i,j,k} = \sum_{i=1}^{n} Num_i \tag{1}$$

  Here $X_{i,j,k}$ is the indicator to represent whether $Q_{i,j}$ is processed on machine k, namely that

$$X_{i,j,k} = \begin{cases} 1 & \text{if } O_{i,j} \text{ is processed on machine } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

- One operation must be processed after its precedent operations. Namely, for two operations of J_i, $O_{i,j}$, and $O_{i,k}$ ($j < k$), Eq. 3 must be satisfied:

$$S_{i,k} > C_{i,j} \ (j < k) \tag{3}$$

- All operations are non-preemptive. It means that once a machine starts processing an operation, the next operation to be processed on this machine must wait until the present operation completes, namely,

$$S_{l,m,k} \geq C_{i,j,k} \left(S_{l,m,k} \geq S_{i,j,k} \right) \tag{4}$$

- The processing time of every operation being processed on every machine agrees with normal distribution:

$$T_{i,j,k} \sim N\left(Q_{i,j,k}, \sigma_{i,j,k} \right) \tag{5}$$

  Generally, the objective function to be optimized for FfJSSP could be divided into two classes, namely, the mono-objective function and multi-objective function:

- Mono-objective
  - Minimizing the makespan [27]

$$f = \min(C_{max}) \tag{6}$$

    C_{max} is the makespan.
  - Fuzzy scheduling based on E/T penalty [30]
  - Scheduling based on satisfaction rate [26], which is adopted in this paper and discussed in detail in Section 3.
- Multi-objective [25]
  - Integrating multiple objectives using weights

$$f = \sum_{i=1}^{l} (w_i \times f_i) \tag{7}$$

    where f_i is a certain objective and w_i is the corresponding weights.
  - Integrating objectives using Pareto optimality.

3. Fuzzy satisfaction rate based on normal distribution

In practice, the manufacturing system is fuzzy because of its changing environment conditions. For example, a breakdown of machine might lead to an uncertain delivery time. At the same

time, the costumer's requirements are often fuzzified, e.g., the costumers are more likely to provide a fuzzified time interval for production delivery but not a crisp time point. Hence, more and more researchers focused on the study of so call fuzzy flexible job shop scheduling problem. There are two kinds of objective functions for FfJSSP, one is mono-objective function and the other is multi-objective function. Zheng *et al*. defined the concept of satisfaction rate to measure the effectiveness of the scheduling [26]. However, in [26] and most other studies before, the fuzzy numbers used to describe the final solution are generated by trial and error, which are heavily relied on the experience and human factor. Azadeh *et al*. solved multi-product assembly shop problems by applying normal distribution of historical data to the membership function [29]. Similarly in this paper, we exploit the historical data, i.e. processing time and assuming it obeys normal distribution to determine the fuzzy numbers. In this way, the influence of human factor on evaluation of effectiveness of manufacturing is decreased. And then, we combined it with the satisfaction rate to evaluate the effectiveness of scheduling. To describe our method clearly, we firstly introduce the satisfaction rate concept in Section 3.1 as stated bellow.

3.1 Satisfaction rate

The membership function of fuzzy set can be described by fuzzy numbers. In this problem the triangle fuzzy number and trapezoidal fuzzy number are applied to represent processing time and costumer requirement, respectively. Triangle fuzzy number can be described by a triple (L, M, R) and the trapezoidal one can be represented by a 4-tuple (d_1, d_2, d_3, d_4). Denote the membership as μ, these two membership functions are described as Eqs. 8 and 9 [26]:

$$\mu = \begin{cases} 0 & x < L \\ 2(x-L)/(R-L) & L \le x < (R+L)/2 \\ 2(R-x)/(R-L) & (R+L)/2 \le x \le R \\ 0 & x > R \end{cases} \tag{8}$$

$$\mu = \begin{cases} 0 & x < d_1 \\ (x-d_1)/(d_2-d_1) & d_1 \le x < d_2 \\ 1 & d_2 \le x < d_3 \\ (d_4-x)/(d_4-d_3) & d_3 \le x \le d_4 \\ 0 & x > d_4 \end{cases} \tag{9}$$

where L and R are the lower and upper bounds of confidence interval of machine processing time, d_1 is the most optimistic time, d_2 and d_3 are the lower and upper bounds of the acceptable time interval, and d_4 is the most pessimistic time, respectively.

Because the makespan of a batch of jobs is fuzzy resulting from the fuzzy processing time, the membership functions of the makespan of job and the costumer requirement can be put in the same coordinate system, Fig. 1. Denote S_T as the area modelled by the triangular membership function, S_D the area modelled by the triangular membership function and S_E the intersected area of the two membership functions, respectively.

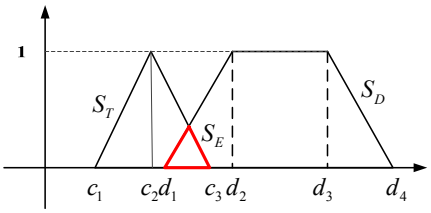

Fig. 1 Membership functions of the job makespan and the costumer requirement

Then, the satisfaction rate is defined as

$$fitness = \frac{S_E}{S_T} \tag{10}$$

3.2 Parameter estimation based on normal distribution

In the existing methods, the parameters of confidence interval of machine processing time, namely the lower and upper bounds L and R, are generated by experience. To avoid the influence of human factor, a normal distribution based method is developed in this section. Because there are many factors affecting the exactly machine processing time, we assume it obeys the normal distribution. Then we have

$$f(T_{i,j,k}) = Normal(Q_{i,j,k}, \sigma_{i,j,k}) = \frac{1}{\sigma_{i,j,k}\sqrt{2\pi}} exp\left(-\frac{(T_{i,j,k} - Q_{i,j,k})}{2\sigma_{i,j,k}^2}\right) \tag{11}$$

Therefore we could determine the parameters from the observation sample data according to statistical theory which is described step by step as follows.

1) Computation of average processing time. For machine M_i, the average processing time $Q_{i,j,k}$ of a specific operation and a specific job can be calculated by

$$Q_{i,j,k} = \frac{1}{L_{i,j,k}} \sum_{l=1}^{L_{i,j,k}} P_{i,j,k,l}\,,\ (1 \le i \le n,\ 1 \le j \le Num_i, 1 \le k \le m) \tag{12}$$

Here, $L_{i,j,k}$ represents the record number of processing time of the j-th operation of job i processed on machine k, $P_{i,j,k,l}$ processing time of the j-th operation of job i processed on machine k of the l-th record, respectively.

2) Computation of the standard deviation. For the standard deviation of $T_{i,j,k}$, it could be calculated according to

$$\sigma_{i,j,k} = \sqrt{\frac{1}{L_{i,j,k}} \sum_{l=1}^{L_{i,j,k}} \left(P_{i,j,k,l} - Q_{i,j,k}\right)^2}\ ,\ (1 \le i \le n,\ 1 \le j \le Num_i, 1 \le k \le m) \tag{13}$$

3) Determination of the confidence interval. For a confidence value α, the confidence interval of $T_{i,j,k}$ could be calculated by

$$[\theta_1(\alpha), \theta_2(\alpha)] = \left[Q_{i,j,k} - \frac{Z_{\alpha/2}\sigma_{i,j,k}}{\sqrt{L_{i,j,k}}}, Q_{i,j,k} + \frac{Z_{\alpha/2}\sigma_{i,j,k}}{\sqrt{L_{i,j,k}}}\right] \tag{14}$$

where $\theta_1(\alpha)$ and $\theta_1(\alpha)$ are the lower bound and upper bound of the confidence interval of $T_{i,j,k}$ on confidence level of $1-\alpha$, and $-Z_{\alpha/2}$ and $Z_{\alpha/2}$ are the lower bound and upper bound of the confidence interval of $N(0, 1)$, which are easy to obtain by checking from table.

Compared Eq. 8 and Eq. 14, the lower and upper bounds of confidence interval of machine processing time L and R are set as

$$L = Q_{i,j,k} - \frac{Z_{\alpha/2}\sigma_{i,j,k}}{\sqrt{L_{i,j,k}}} \tag{15}$$

$$R = Q_{i,j,k} + \frac{Z_{\alpha/2}\sigma_{i,j,k}}{\sqrt{L_{i,j,k}}} \tag{16}$$

4. Scheduling method based on differential evolution algorithm

Differential evolution algorithm (DE) is a novel evolution algorithm, which is firstly proposed by Rainer, S and Price K. to solve continuous space optimization problem [31]. It is a swarm-based global parallel evolutionary algorithm of which individuals in the population can evolve simultaneously. DE can generate new individual variation with straightforward operations, and then generate the new population with greedy comparison strategies. Compared with other evolution algorithms, DE has obvious advantages, reproducing the population based global search ability, real-numbered coding, and straightforward operations. In addition, it has a special memory ability, which makes DE able to track the current evolution situation and adjust the searching strategies. In spite of the intensive research, mostly, DE is used to solve non-flexible job shop scheduling problem. In this paper, a novel scheduling method is developed to apply DE to flexible job shop scheduling problem by well defining operation code and machine code, which are both coded with real numbers. The basic differential evolution algorithm will be described in Section 4.1.

4.1 Differential evolution algorithm

Similar as genetic algorithm (GA), differential evolution (DE) algorithm also performs selection, crossover, and mutation operations. While in DE, differential idea is introduced into evolution operations and it is coded in real number. Denote the *i*-th individual in the *G*-th generation as

$$X_i^G = \left[X_{i1}^G, X_{i2}^G, \cdots, X_{iD}^G\right], (i = 1,2, \cdots, NP) \tag{17}$$

where $x_{ij}^G \in [lower, upper]$ (j = 1,2,...,D), which is called gene, *lower* and *upper* are the predefined lower bound and upper bound of the real code, D is the dimension of the problem, and NP is the size of population, respectively.

In each generation, the new population is generated by mutation and crossover operations first, and then selection operation is adopted to optimize the population.

Mutation

Mutation is used to maintain the diversity of population. There are many mutation strategies in the existing methods. Table 1 shows 5 most popular strategies. Taking the first one as an example, the mutation operation is performed according to

$$V_i^{G+1} = x_{r1}^G + \gamma \times \left(x_{r2}^G - x_{r3}^G\right) \tag{18}$$

where V_i^{G+1} is the new individual after mutation, $r1$, $r2$, and $r3$ are three different randomly selected number within the set {1,...,i –1, i +1,..., NP}, and γ is the scaling factor. If any components of V_i^{G+1} exceed the boundary, they should be revised by

$$V_{i,k}^{G+1} = \begin{cases} lower + \frac{V_{i,k}^{G+1} - upper}{2} & if\ (V_{i,k}^{G+1} > upper) \\ upper - \frac{lower - V_{i,k}^{G+1}}{2} & if\ (V_{i,k}^{G+1} < lower) \end{cases} \tag{19}$$

where $V_{i,k}^{G+1}$ represents the k-th component (gene) of V_i^{G+1}.

Table 1 Evolution strategies of DE

Strategies	Formula
rand/1	$V_i^{G+1} = x_{r1}^G + \gamma \times (x_{r2}^G - x_{r3}^G)$
best/1	$V_i^{G+1} = x_{best}^G + \gamma \times (x_{r2}^G - x_{r3}^G)$
rand-to-best/1	$V_i^{G+1} = x_i^G + \lambda \times \left(x_{best}^G - x_i^G\right) + \gamma \times (x_{r2}^G - x_{r3}^G)$
best/2	$V_i^{G+1} = x_{best}^G + \gamma \times (x_{r1}^G + x_{r2}^G - x_{r3}^G - x_{r4}^G)$
rand/2	$V_i^{G+1} = x_{r5}^G + \gamma \times (x_{r1}^G + x_{r2}^G - x_{r3}^G - x_{r4}^G)$

Crossover

Crossover is performed based on the results of mutation. It can be classified into binomial crossover and exponential crossover types, which are described by Eqs. 20 and 21, respectively.

$$U_{i,j}^{G+1} = \begin{cases} V_{i,j}^{G+1} & \text{if } (rand \le CR \text{ or } j = rand(1,D)) \\ X_{i,j}^G & \text{otherwise} \end{cases} \tag{20}$$

$$U_{i,j}^{G+1} = \begin{cases} V_{i,j}^{G+1} & if\ (rand \le CR \text{ and } j < D) \\ X_{i,j}^G & \text{otherwise} \end{cases} \tag{21}$$

where $U_{i,j}^{G+1}$ is the j-th component of the new individual U_i^{G+1} after crossover, *rand* is a random real number within [0,1], and CR is the crossover rate. It could be found that in binomial crossover at least one gene in V_i^{G+1} will be included, so the diversity of the population is improved.

Selection

Selection is defined by

$$X_i^{G+1} = \begin{cases} U_i^{G+1} & if\ (f\left(U_i^{G+1}\right) > f(X_i^G)) \\ X_i^G & \text{otherwise} \end{cases} \tag{22}$$

where f (*x*) is the objective function, X_i^{G+1} represents the newly generated individual in generation $G + 1$.

4.2 Differential evolution algorithm

Because DE uses real number for coding, it could not be applied to the scheduling problem directly. Qian *et al*. adopted a random key based Largest-Order-Value (L-O-V) rule to convert the continuous values of individuals in DE to job permutations [32, 33]. To the best of our knowledge, DE based methods are all applied to solve the JSSP problem, there are no related works for FfJSSP, and even for fJSSP. In this paper, by well defining the machine code and its operations to handle the flexibility, we extend Qian's method to the FfJSSP problems. Also a local search process is introduced in our proposed method.

Coding and decoding methods

Because in FfJSSP an operation could be processed on different machines, we design the code of a potential solution with two parts, namely the operation part and machine part (Fig. 2). For a FfJSSP problem with *n* jobs, denote m_i as the operation number of *i*-th job, then the total number of operations is $D = \sum_{i=1}^{n} m_i$, so the length of operation part and that of machine part both are *D*. Denote CO_i = ($CO_{i,1}$, $CO_{i,2}$,..., $CO_{i,D}$) as the operation part of the *i*-th individual and CM_i = ($CM_{i,1}$, $CM_{i,2}$,..., $CM_{i,D}$) as the machine part, each element of which is coded by a real number within [*lower*, *upper*]. Next the decoding methods of these two parts will be described.

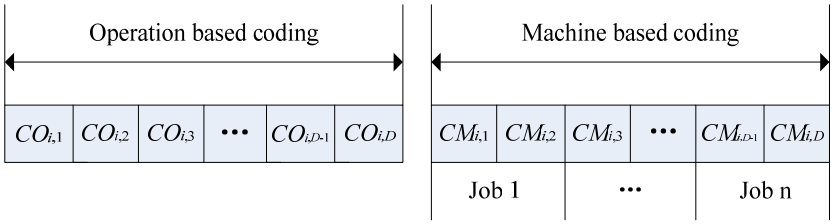

Fig. 2 Coding schedule

Decoding of operation part

Sort the elements of CO_i in a descending order, which is denoted as SO_i = ($SO_{i,1}$, $SO_{i,2}$,..., $SO_{i,D}$), where $SO_{i,p} \geq SO_{i,q}$ ($p<q$). Then we define an index vector for CO_i, denoting as $IndexO_i$ = ($indexo_{i,1}$, $indexo_{i,2}$,..., $indexo_{i,D}$), where $indexo_{i,k}$ = arg($SO_{i,j} = CO_{i,k}$). Define $m_0 = 0$, then the index number will be converted into job number according to

$$\pi_{i,j} = k, \text{ if } \sum_{s=1}^{k-1} m_s < \mu_{i,j} \leq \sum_{s=1}^{k} m_s \tag{23}$$

Here $\pi_{i,j}$ represents the corresponding job number of the operation $CO_{i,j}$. After obtained the index sequence of CO_i, it is easy to get the final job operation scheduling sequence: if $\pi_{i,j}$ is the *q*-th integer number *p*, it means that the *j*-th operation in the scheduling sequence should be the *q*-th operation of job *p*.

For clearly understand, a simple scheduling sequence is taken as an example to describe the method. The scheduling sequence has 3 jobs with 2, 2, and 3 operations, respectively. The initial individual is CO_i = (3.2, 1.4, 6.5, 3.7, 4.9, 7.9, 6.3). Then the sorted sequence CO_i, index sequence $IndexO_i$ and job sequence $\pi_{i,j}$ are listed in Table. 2. The result of job sequence is $\pi_{i,j}$ = (3, 3, 1, 3, 2, 1, 2), so the final operation schedule should be: (O_{31}, O_{32}, O_{11}, O_{33}, O_{21}, O_{12}, O_{22}).

Table 2 An example of decoding operation part

Dimension *j*	1	2	3	4	5	6	7
CO_{ij}	3.2	1.4	6.5	3.7	4.9	7.9	6.3
O_{ij}	7.9	6.5	6.3	4.9	3.7	3.2	1.4
Index O_{ij}	6	7	2	5	4	1	3
π_{ij}	3	3	1	3	2	1	2

Decoding of machine part

An element in machine part decoding represents which machine the corresponding operation is processed on. Different from the operation part, each element of machine part is corresponding to a FIXED operation, namely that the q-th operation of p-th job $O_{p,q}$ is represented by the k-th element of machine part, where

$$k = \sum_{i=1}^{p-1} m_i + q \tag{24}$$

To decode it, we firstly construct the available machine set for each operation process $O_{p,q}$ as $S_{p,q}$ ={machines that can process $O_{p,q}$}, and denote $|S_{p,q}|$ as the size of $S_{p,q}$, namely that the available machine number of operation $O_{p,q}$. Then the decoding method is shown as follows:

a) Calculate the size of interval [*lower*, *upper*]: $\Phi = upper - lower$.
b) For the position corresponding to the operation $O_{p,q}$ in the machine part, we divide the interval [*lower*, *upper*] into $|S_{p,q}|$ subintervals: $\varphi_k = \Phi / |S_{p,q}|$, where k is defined by Eq. 24.
c) For each individual code of machine part, the machine numbers of operations are determined by

$$\omega_{i,k} = \left\lfloor \frac{CM_{i,k}}{\varphi_k} \right\rfloor, (i = 1,2,\cdots,NP;\ k = 1,2,\cdots,D) \tag{24}$$

where i is the index number of individual in the population, k is the position in the code of machine part, $\omega_{i,k}$ represents the serial number of the dispatched machine in the corresponding set for $O_{p,q}$, and k, p, q satisfy Eq. 24. Then it is easy to find the dispatched machine of $O_{p,q}$ in the set of $S_{p,q}$. For example, there are 2 jobs with 2 and 4 operations, respectively, and there are 6 machines. So, D = 2 + 4 = 6. Besides, it is set that *lower* = 0, and *upper* = 10. The corresponding optional machine sets are $S_{1,1}$ = {1,3,5}, $S_{1,2}$ = {1,2,4,5}, $S_{2,1}$ = {2,4}, $S_{2,2}$ = {1,2,3,4,6}, $S_{2,3}$ = {4,6}, and $S_{2,3}$ = {2,3,4,5}. For individual CM_i = (4.4, 1.6, 1.2, 8.9, 0.3, 7.9), the calculation steps are shown in Table 3. Table 3 shows that the dispatched machines for the 6 operations are (3, 1, 2, 6, 4, 5). It means that the 1st and 2nd operations of job 1 are processed on machine 3 and 1 respectively; the 1st, 2nd, 3rd, and 4th operations of job 2 are processed on machine 2, 6, 4, and 5, respectively.

Table 3 An example of decoding machine part

Dimension j	1(O_{11})	2(O_{12})	3(O_{21})	4(O_{22})	5(O_{23})	6(O_{24})
CM_{ij}	4.4	1.6	1.2	8.9	0.3	7.9
φ_j	3.33	2.5	5	2	5	2.5
ω_{ij}	2	1	1	5	1	4
Corresponding machine	3	1	2	6	4	5

Local search option

In order to improve the search ability, local search process is usually performed within the general DE framework. For example, Zhang and Wu proposed a tree-based local exploration to solve JSSP problem [34]. In this paper, we use the idea in [34] to do the local search for our operation part of the code, and develop a novel local exploration which is applied to the machine part of the code. Next, they will be described concretely in this section.

Local search exploration for operation part

Defining a function swap (a, b) which means swapping a and b, then the pseudocode of this algorithm could be described by Fig. 3.

```
Select an operation individual α
For d = 1 to D
    Set P = {ϕ}
    For k = 1 to RandNum1
     Randomly select a position pair of α for swapping, gener-
        ating a new individual p_k.
      Add p_k into P.
    End For
    Denote the best one as η.
    For k = 1 to RandNum1
        For j = 1 to RandNum2
         Randomly select a position pair of p_k for swapping,
        generating a new individual p_{RandNum1×k+j}.
                Add p_{RandNum1×k+j} into P.
        End For
          Calculate the fitness of individuals in P. Keep the
          top RandNum1 individuals in the queue and discard
          the others.
        Update η.
    End For
End For
Output η.
```

Fig. 3 Pseudocode of local search exploration for operation part

Local search exploration for machine population

Corresponding to our designed machine code, a novel local exploration process is proposed in the paper. The pseudocode of the method is described by Fig. 4.

```
Set Φ = upper – lower
Select a machine individual β={b_1, b_2, ..., b_D}
For j = 1 to D
    Calculate p and q according to Eq. 24
    φ_k = Φ/ |S_{p,q}|
    For k = 1 to |S_{p,q}|
        c_k = b_j + k×φ_k
        Update c_k according to Eq. 20
    End For
    Use the best c_k replacing b_j.
End For
Output updated β.
```

Fig. 4 Pseudocode of local search exploration for machine part

4.3 The framework of the algorithm

To sum up, the framework of the algorithm is described as follows:

1) Initialize the scaling factor F, the crossover rate CR, the size of the population NP, and the total iteration time N; set the current iteration number $G = 0$; input the fuzzy number of the processing time.
2) Initialize the population (Including operation part CO, and machine part CM), and calculate the best fitness BF.
3) Perform the mutation operation on CO and get population V; perform the crossover operation on V and get population U; perform the selection operation on V and U to get the new population CO.
4) Apply the local exploration search operation to CO and update population.
5) Perform the mutation operation on CM and get population MV; perform the crossover operation on MV and get population MU; perform the selection operation on MV and MU to get the new population CM.
6) Apply the local exploration search operation to CM and update population.
7) Set $G = G + 1$, if $G > N$ output the best individual, otherwise go to Step 2.

5. Results and discussion

To examine the effectiveness of the algorithm, it is better to implement it on some benchmark problems and compare the results with other existing methods. On the best of our knowledge, there is still no benchmark dataset for Fuzzy Flexible Job Shop Scheduling Problems (FfJSSP) at all. Therefore a Flexible Job Shop Scheduling Problem (fJSSP) is taken from Ref. [25] firstly, by which we could compare the results of our proposed method with other existing algorithms and examine its basic performance on fJSSP. To further evaluate our method on FfJSSP, we construct 30 problems for our second experiment, which could also be used as benchmark dataset for other researchers to study FfJSSP.

5.1 Case 1

The data set of first experiment is selected from Ref. [25]. There are 4 jobs in the problem, and the numbers of operation of jobs are 3, 3, 4 and 2, respectively. All operations are processed on 5 machines. The processing time is shown in Table 4. The result of our method is compared with five often used existing methods [35]. The result is listed in Table 5. In the table, C* = 11 is the optimal result of the problem. It can be seen that our proposed method could obtain the optimal result together with other 4 existing methods, while AL+CGA method only reaches 16, which are much more than others. Fig. 5 shows the Gantt chart of one feasible solution obtained by our method.

5.2 Case 2

Because there is no suitable data set for FfJSSP, we construct 30 FfJSSP problems to examine our proposed DE method. From the historical data, namely the historical processing time of different operations on different machines, the fuzzy numbers of membership function parameters are deduced according to normal distribution.

Table 4 Processing time

Jobs	Operations	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Table 5 Comparisons of different algorithms

$n \times m$	C*	AL+CGA	GEN ACE	KBACO	TSPCB	ABC	Proposed DE
4×5	11	16	11	11	11	11	11

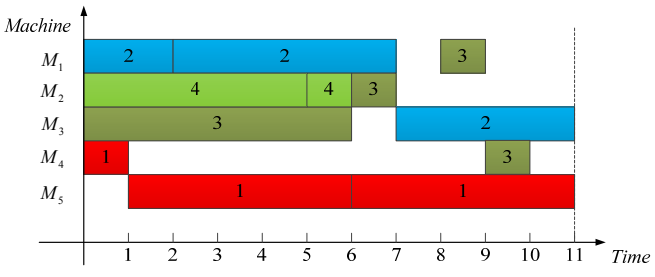

Fig. 5 Gantt chart of one feasible solution

The fuzzy numbers are needed to calculate the objective function in DE scheduling method. The 30 FfJSSP dataset together with their fuzzy numbers could be available on the following link *https://github.com/BrotherQi/APEM_DATA*, the scale of which ranges from 5×8 to 10×10, being more complicated than that of Case 1. Performing our proposed DE based scheduling method, 10 different strategies are executed for comparison, namely that 5 mutation strategies multiplying 2 crossover strategies described in Section 4.1. The results are shown in Table 6. In Table 6, the second column d_1, d_2, d_3, d_4 represent the fuzzy numbers of costumer's requirements, from the third to the 13th columns are the max satisfied rates of the ten strategies, in the order of rand/1 mutation + binomial crossover, best/1 mutation + binomial crossover, rand-to-best/1 mutation + binomial crossover, best/2 mutation + binomial crossover, rand/2 mutation + binomial crossover, rand/1 mutation + exponential crossover, best/1 mutation + exponential crossover, rand-to-best/1 mutation + exponential crossover, best/2 mutation + exponential crossover, and rand/2 mutation + exponential crossover. The results demonstrate our proposed method is applicable for FfJSSP. Comparing the average results of different strategies, it can be seen that those results are similar, the best one is 0.708 of the rand/2 mutation + binomial crossover strategy, and the worst one is 0.643 of rand-to-best/1 mutation + exponential crossover strategy. The average result of five binomial crossover strategies is 0.678, a little bigger than that of five exponential crossover strategies 0.662.

Table 6 Results of Case 2

Data set	d_1,d_2,d_3,d_4	Strategies									
		1	2	3	4	5	6	7	8	9	10
1	300,350,390,440	0.620	0.572	0.570	0.680	**0.692**	0.521	0.541	0.680	0.632	0.434
2	330,370,400,430	0.845	0.934	0.940	0.904	**0.987**	0.934	0.958	0.706	0.894	0.935
3	300,330,380,410	0.653	0.660	**0.758**	0.662	0.599	0.448	0.729	0.685	0.497	0.486
4	70,77,85,93	0.546	0.590	0.556	0.596	0.649	0.653	0.550	0.688	**0.694**	0.629
5	70,75,83,93	0.460	0.524	0.395	0.400	0.551	**0.622**	0.585	0.460	0.487	0.504
6	78,83,89,98	0.431	0.425	0.595	0.612	0.624	0.486	0.856	0.504	**0.882**	0.622
7	110,120,125,145	0.719	0.816	0.755	0.710	0.763	0.690	**0.875**	0.717	0.591	0.585
8	100,110,120,145	0.483	0.484	0.528	0.340	0.575	0.468	0.493	0.484	0.491	**0.577**
9	90,100,105,125	0.645	0.707	0.669	0.669	0.739	0.733	**0.742**	0.646	0.711	0.715
10	55,60,65,75	0.880	0.890	0.911	0.925	0.854	0.888	0.856	0.898	0.813	**0.933**
11	60,65,70,85	0.918	0.925	0.916	0.907	0.905	0.880	0.909	**0.971**	0.932	0.929
12	70,75,80,100	0.669	0.726	0.654	0.699	0.718	0.681	0.713	0.682	0.666	**0.778**
13	95,100,105,130	0.639	0.615	**0.683**	0.579	0.656	0.631	0.662	0.646	0.576	0.668
14	110,115,118,135	0.496	0.506	0.446	0.437	**0.627**	0.511	0.544	0.476	0.335	0.309
15	105,110,120,135	0.695	0.637	**0.798**	0.679	0.659	0.698	0.629	0.647	0.712	0.605
16	60,65,70,85	0.789	0.784	0.781	0.767	0.844	0.736	0.733	0.824	**0.847**	0.801
17	85,90,95,110	0.772	0.708	0.815	0.834	0.877	**0.893**	0.796	0.749	0.712	0.734
18	70,75,80,95	**0.684**	0.569	0.591	0.538	0.539	0.620	0.675	0.515	0.591	0.486
19	120,125,130,150	0.492	0.415	0.535	0.528	0.594	0.558	**0.716**	0.490	0.504	0.340
20	130,135,140,160	0.898	0.942	0.887	0.918	0.929	0.924	0.871	0.866	0.943	**0.949**
21	120,125,130,150	0.476	0.255	0.467	0.281	0.505	0.472	0.568	**0.645**	0.358	0.511
22	140,145,150,170	0.675	0.620	0.749	0.744	0.740	0.713	0.715	**0.758**	0.575	0.719
23	135,140,150,165	0.772	0.708	0.815	0.834	0.877	**0.893**	0.796	0.749	0.712	0.734
24	125,130,135,155	0.676	**0.787**	0.669	0.723	0.708	0.703	0.693	0.672	0.640	0.694
25	220,230,240,255	0.582	0.739	0.702	0.738	0.701	**0.791**	0.577	0.569	0.674	0.651
26	240,245,250,265	0.661	0.726	0.787	0.787	0.669	0.510	0.544	0.416	**0.839**	0.776
27	225,230,235,255	**0.784**	0.667	0.647	0.648	0.661	0.615	0.715	0.615	0.525	0.484
28	145,150,155,170	0.608	0.648	0.687	0.668	0.607	**0.743**	0.617	0.459	0.637	0.643
29	165,170,175,200	0.621	**0.668**	0.650	0.626	0.635	0.602	0.538	0.528	0.536	0.498
30	175,180,185,210	0.712	0.568	0.610	0.708	**0.762**	0.467	0.719	0.550	0.605	0.662
Ave	-	0.663	0.661	0.686	0.671	**0.708**	0.669	0.697	0.643	0.654	0.646

5.3 Discussion

The experiment of Case 1 is simple, the result of which shows that our proposed method is no less than the existing methods, and obviously better than AL+CGA method. In Case 2, we construct 30 more complicated datasets, and apply our method to them with different combinations of mutation and crossover strategies. The results are really effective and show that our method could be applied to real fuzzy flexible JSSP problems. It could be found that there are no significant difference among different strategy combinations, especially between two different crossover strategies.

6. Conclusion

Focused on FfJSSP, this paper develops a novel scheduling method based on fuzzy satisfaction rate and differential evolution algorithm. In the method, fuzzy parameters are deduced based on normal distribution for the calculation for satisfaction rate firstly. And then, a differential evolution is proposed for scheduling. Firstly, the coding method is well designed for FfJSSP, and then mutation, crossover and selection operations are proposed corresponding to the coding method. Moreover, local exploration search process is introduced in the method to improve the performance. At last, our proposed method is applied to two experiments to test the performance. Numerical results show that the developed method is practical and effective.

Acknowledgement

The authors are grateful to the support of the National Natural Science Foundation of China (61373050), and the Science Technology Development Project from Jilin Province of China (20130101070JC, 20130206003SF).

References

[1] Tang, M., Gong, D., Liu, S., Zhang, H. (2016). Applying multi-phase particle swarm optimization to solve bulk cargo port scheduling problem, *Advances in Production Engineering & Management*, Vol. 11, No. 4, 299-310, doi: 10.14743/apem2016.4.228.

[2] Burdett, R.L., Kozan, E. (2018). An integrated approach for scheduling health care activities in a hospital, *European Journal of Operational Research*, Vol. 264, No. 2, 756-773, doi: 10.1016/j.ejor.2017.06.051.

[3] Tang, M., Gong, D., Liu, S., Lu, X. (2017). Finding key factors for electric vehicle charging station location: a simulation and ANOVA approach, *International Journal of Simulation Modelling*, Vol. 16, No. 3, 541-554, doi: 10.2507/IJSIMM16(3)CO15.

[4] Wu, Z.B., Weng, M.X. (2005). Multiagent scheduling method with earliness and tardiness objectives in flexible job shops, *IEEE Transactions on Man, and Cybernetics, Part B: Cybernetics*, Vol. 35, No. 2, 293-301, doi: 10.1109/TSMCB.2004.842412.

[5] Sahin, C., Demirtas, M., Erol, R., Baykasoğlu, A., Kaplanoğlu, V. (2017). A multi-agent based approach to dynamic scheduling with flexible processing capabilities, *Journal of Intelligent Manufacturing*, Vol. 28, No. 8, 1827-1845, doi: 10.1007/s10845-015-1069-x.

[6] Zhang, Y.F., Wang, J., Liu, Y. (2017). Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact, *Journal of Cleaner Production*, Vol. 167, 665-679, doi: 10.1016/j.jclepro.2017.08.068.

[7] Li, J.Q., Pan, Q.K., Tasgetiren, M.F. (2014). A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities, *Applied Mathematical Modelling*, Vol. 38, No. 3, 1111-1132, doi: 10.1016/j.apm.2013.07.038.

[8] Gao, J., Sun, L.Y., Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, *Computers & Operations Research*, Vol. 35, No. 9, 2892-2907, doi: 10.1016/j.cor.2007.01.001.

[9] Li, X.Y., Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem, *International Journal of Production Economics*, Vol. 174, 93-110, doi: 10.1016/j.ijpe.2016.01.016.

[10] Zhang, Q., Manier, H., Manier, M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times, *Computers & Operations Research*, Vol. 39, No. 7, 1713-1723, doi: 10.1016/j.cor.2011.10.007.

[11] Huang, X.W., Zhao, X.Y., Ma, X.L. (2014). An improved genetic algorithm for job-shop scheduling problem with process sequence flexibility, *International Journal of Simulation Modelling*, Vol. 13, No. 4, 510-522, doi: 10.2507/IJSIMM13(4)CO20.

[12] Gao, K.Z., Suganthan, P.N., Pan, Q.K., Chua, T.J., Cai, T.X., Chong, C.S. (2016). Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives, *Journal of Intelligent Manufacturing*, Vol. 27, No. 2, 363-374, doi: 10.1007/s10845-014-0869-8.
[13] Li, J.Q., Pan, Q.K. (2012). Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity, *Applied Soft Computing*, Vol. 12, No. 9, 2896-2912, doi: 10.1016/j.asoc.2012.04.012.
[14] Özgüven, C., Yavuz, Y., Özbakır, L. (2012). Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times, *Applied Mathematical Modelling*, Vol. 36, No. 2, 846-858, doi: 10.1016/j.apm.2011.07.037.
[15] Yuan, Y., Xu, H. (2013). Flexible job shop scheduling using hybrid differential evolution algorithms, *Computers & Industrial engineering*, Vol. 65, No. 2, 246-260, doi: 10.1016/j.cie.2013.02.022.
[16] Xu, Y., Wang, L., Wang, S.Y., Liu, M. (2015). An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time, *Neurocomputing*, Vol. 148, 260-268, doi: 10.1016/j.neucom.2013.10.042.
[17] Supsomboon, S., Vajasuvimon, A. (2016). Simulation model for job shop production process improvement in machine parts manufacturing, *International Journal of Simulation Modelling*, Vol. 15, No. 4, 611-622, doi: 10.2507/IJSIMM15(4)3.352.
[18] Mehrabad, M.S., Pahlavani, A. (2009). A fuzzy multi-objective programming for scheduling of weighted jobs on a single machine, *The International Journal of Advanced Manufacturing Technology*, Vol. 45, No. 1-2, 122-139, doi: 10.1007/s00170-009-1947-5.
[19] Lei, D.M. (2008). Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems, *The International Journal of Advanced Manufacturing Technology*, Vol. 37, No. 1-2, 157-165, doi: 10.1007/s00170-007-0945-8.
[20] Kilic, S. (2007). Scheduling a fuzzy flowshop problem with flexible due dates using ant colony optimization, In: Giacobini, M. *et al*. (eds.), *Applications of Evolutionary Computing, EvoWorkshops 2007,* Vol. 4448, Springer-Verlag, Berlin, Heidelberg, 742-751, doi: 10.1007/978-3-540-71805-5_80.
[21] Niu, Q., Jiao, B., Gu, X.S. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time, *Applied Mathematics and Computation*, Vol. 205, No. 1, 148-158, doi: 10.1016/j.amc.2008.05.086.
[22] Zhang, X.G., Deng, Y., Chan, F.T.S, Xu, P.D., Mahadevan, S., Hu, Y. (2013). IFSJSP: A novel methodology for the job-shop scheduling problem based on intuitionistic fuzzy sets. *International Journal of Production Research*, Vol. 51, No. 17, 5100-5119, doi: 10.1080/00207543.2013.793425.
[23] Gao, K.Z., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F. (2015). An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time, *International Journal of Production Research*, Vol. 53, No.19, 5896-5911, doi: 10.1080/00207543.2015.1020174.
[24] Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J. (2017). Robust multiobjective optimisation for fuzzy job shop problems, *Applied Soft Computing*, Vol. 56, 604-616, doi: 10.1016/j.asoc.2016.07.004.
[25] Kacem, I., Hammadi, S., Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic, *Mathematics and Computers in Simulatio*n, Vol. 60, No. 3-5, 245-276, doi: 10.1016/S0378-4754(02)00019-8.
[26] Zheng, Y.L., Li ,Y.X., Lei, D.M. (2012). Multi-objective swarm-based neighborhood search for fuzzy flexible job shop scheduling, *The International Journal of Advanced Manufacturing Technology*, Vol. 60, No. 9-12, 1063-1069, doi: 10.1007/s00170-011-3646-2.
[27] Lei, D.M. (2012). Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling, *Applied Soft Computing*, Vol. 12, No. 8, 2237-2245, doi: 10.1016/j.asoc.2012.03.025.
[28] Gao, K.Z., Suganthan, P.N., Pan, Q.K., Chua, T.J., Chong, C.S., Cai, T.X. (2016). An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time, *Expert Systems with Applications*, Vol. 65, 52-67, doi: 10.1016/j.eswa.2016.07.046.
[29] Azadeh, A., Hatefi, S.M., Kor, H. (2012). Performance improvement of a multi product assembly shop by integrated fuzzy simulation approach, *Journal of Intelligent Manufacturing*, Vol. 23, No. 5, 1861-1883, doi: 10.1007/s10845-011-0501-0.
[30] Shadrokh, S., Kianfar, F., (2007). A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty, *European Journal of Operational Research*, Vol. 181, No. 1, 86-101, doi: 10.1016/j.ejor.2006.03.056.
[31] Storn, R., Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, Vol. 11, No. 4, 341-359, doi: 10.1023/A:1008202821328.
[32] Bean, J.C. (1994). Genetic algorithms and random keys for sequencing and optimization, *ORSA journal on computing*, Vol. 6, No. 2, 154-160, doi: 10.1287/ijoc.6.2.154.
[33] Qian, B., Wang, L., Huang, D.X., Wang, W.L., Wang, X. (2009). An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers, *Computers & Operations Research*, Vol. 36, No. 1, 209-233, doi: 10.1016/j.cor.2007.08.007.
[34] Zhang, R., Wu, C. (2011). A hybrid differential evolution and tree search algorithm for the job shop scheduling problem, *Mathematical Problems in Engineering*, Vol. 2011, Article ID: 390593, doi: 10.1155/2011/390593.
[35] Li, J.Q., Pan, Q.K., Gao, K.Z. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, *The International Journal of Advanced Manufacturing Technology*, Vol. 55, No. 9-12, 1159-1169, doi: 10.1007/s00170-010-3140-2.