Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A multi-objective fuzzy flexible job shop scheduling problem considering the maximization of processing quality

Jiarui Li[a] and Zailin Guan[a*]

[a]*School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, P.R. China*

C H R O N I C L E

Article history:
Received August 24 2023
Received in Revised Format October 18 2023
Accepted December 29 2023
Available online December 29 2023

Keywords:
Fuzzy flexible job shop scheduling problem
Multi-objective optimization
Spider monkey optimization algorithm
Aircraft shaft parts manufacturing systems

A B S T R A C T

This paper analyzes practical production characteristics, including customer's stringent quality requirements and uncertain processing time in aircraft shaft parts manufacturing. Considering the above characteristics, we propose a multi-objective fuzzy aircraft shaft parts production scheduling problem considering the maximization of production quality. We define this problem as a multi-objective fuzzy flexible job shop scheduling problem (MO-fFJSP) with fuzzy processing time. To address this problem, we developed an improved multi-objective spider monkey optimization (IMOSMO) algorithm. IMOSMO integrates strategies such as genetic operators, variable neighborhood search and Pareto optimization theory on the framework of the conventional Spider Monkey Optimization (SMO) framework and discretize the continuous SMO algorithm to solve MO-fFJSP. To enhance the efficiency of the algorithm, we further adjust the sequence of the local leader learning phase and the global leader learning phase within the proposed IMOSMO framework. We conduct a comparative analysis between the performance of IMOSMO and NSGA-II using 28 cases of varying scales. The computational results demonstrate the superiority of our algorithm over NSGA-II in terms of both solution diversity and quality. Moreover, the performance of the proposed algorithm upgrades as the problem scale increases.

1. Introduction

Aviation spare parts manufacturers need to improve their capacity to adapt to market demands and emphasize customer satisfaction in production arrangements due to the diverse changes in market demands and the swift development of the aerospace industry. Aerospace shaft components are critical parts within aviation spare parts, The efficiency and quality of their production significantly impact the delivery time and overall lifespan of aerospace products. Therefore, optimizing the scheduling of the aerospace shaft production workshop is of paramount importance.

The aerospace shaft production workshop is characterized by: a) a diverse range of products; b) distinct production processes for different shaft types; c) a multitude of intricate processing procedures; d) the non-unique allocation of machines to processing tasks. Consequently, the production of the aerospace shaft is a typical flexible job shop scheduling problem (FJSP).

Since the FJSP is a common NP-hard problem in discrete manufacturing systems, there has been an increase in related research recently (Sassi et al., 2022). Additionally, considering that practical scheduling problems often involve multiple optimization objectives, an increasing number of academics have studied the multi-objective flexible job shop scheduling problem (MO-FJSP) in-depth. Unlike single-objective optimization, MO-FJSP entails trade-offs between multiple objectives, such as minimizing manufacturing time, delays, tardiness, flow time, machine idle time, and so on (Caldeira et al., 2020). In aerospace shaft production workshops, different customers have varying demands for the quality of workpiece processing. To enhance customer satisfaction, when dealing with crucial orders that demand high processing quality, priority is often given to allocating optimal resources (machines with superior processing quality). However, this practice can result in specific

* Corresponding author Tel: +86-150-7112-0374
E-mail zlguan@hust.edu.cn (Z. Guan)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2024 Growing Science Ltd.
doi: 10.5267/j.ijiec.2023.12.011

resources becoming bottlenecks, leading to unnecessary accumulation of work-in-progress and disruptions in production, consequently affecting the production cycle. Therefore, the trade-off between processing time and processing quality should be taken into account. Currently, research on the MO-FJSP with the objective of maximizing processing quality is limited (Kong et al., 2013).

In the FJSP, the assumption of precise production processing time and fixed due dates is common (Pan et al., 2021). However, in actual manufacturing systems, due to human factors, setup times, and other reasons, the processing times of operations may be flexible, and due dates are not strictly fixed, making it difficult to define them as precise values. Consequently, several scholars have proposed the Fuzzy Flexible Job Shop Scheduling Problem (fFJSP). The fFJSP represents processing times or due dates as fuzzy values, which could be Triangular Fuzzy Numbers (TFNs) (Pan et al., 2021; García Gómez et al., 2023), Interval Numbers (INs) (Li et al., 2019; Han et al., 2016), or random numbers following specific distributions (Joo et al., 2018). The Flexible Job Shop Scheduling Problem with fuzzy values of processing time (FJSP-FPT) has received a lot of attention recently. To solve the FJSP-FPT, Xu et al. (2018) presented a flower pollination algorithm to reduce the fuzzy makespan. Gao et al. (2016) modified the artificial bee colony method while also optimizing the maximum makespan and maximum resource load. Lin et al. (2019) solved the FJSP-FPT using a unique approach called hybrid multi-verse optimization (HMVO).

Taking the above requirements into consideration, this study proposes the Multi-objective Fuzzy Flexible Job Shop Scheduling Problem (MO-fFJSP) considering the maximization of processing quality. In MO-fFJSP, each job has its own processing quality priority. Jobs with higher priority require allocation to high- precision/high-quality processing machines as much as possible. For the same job, downstream operations should be assigned to high-precision/high-quality processing machines whenever possible, as downstream operations tend to have higher failure costs. By quantifying the above rules, the objective can be formulated as the maximization of total processing quality. As mentioned earlier, the production cycle should be considered along with the processing quality. Therefore, the objectives of this problem are minimizing the maximum makespan while concurrently maximizing the total processing quality.

To solve the proposed MO-fFJSP, we design an Improved Multi-objective Spider Monkey Optimization (IMOSMO) algorithm. The conventional SMO is typically designed to solve continuous optimization problems with a single objective. In this study, we introduce genetic operators, variable neighborhood search, and other strategies into the conventional SMO framework to replace the original position updating strategy. This discretizes the continuous SMO algorithm for solving the combinatorial optimization problem. We further integrate Pareto optimization theory and apply the algorithm to the optimization problems with multi objectives. To enhance the algorithm's efficiency and practicability, this study modifies the execution sequence of the local leader learning phase and the global leader learning phase in the multi-objective discrete SMO algorithm framework. The superior performance of the proposed IMOSMO algorithm in terms of algorithmic variety and solution efficacy has been verified by comparing it to the standard NSGA-II algorithm using test problems of various scales.

The remainder of this paper is structured as follows: Section 2 discusses the problem description and TFN operational rules. Section 3 introduces the IMOSMO, which is developed for the MO-fFJSP. Section 4 contains experimental results and comparison between proposed algorithm and conventional NSGA-II algorithm. Section 5 concludes with some findings.

2. MO-fFJSP considering the maximization of processing quality

2.1 Problem description

The MO-fFJSP considering the maximization of processing quality is defined as follows: M machines in the workshop are used to process N jobs. Each job n needs to undergo a fixed sequence of operations $O_{nk}(k = 1,2, \dots, k_n)$. The machine m processes operation O_{nk} is selected from the appropriate set of optional machines $S_{nk} \in S_m$. The machines within set S_{nk} are classified into different levels based on their processing capabilities in terms of quality and precision. Machines with heightened processing capabilities are attributed higher levels, corresponding to larger parameter values A_m. Similarly, the jobs to be processed in the workshop receive varied degrees of priority based on customer demand or the production characteristics of the products themselves. The more critical the workpiece, the greater the assigned criticality parameter, denoted as A_n. Furthermore, the criticality parameters for different operations of a given job might exhibit variability. For a given job n, the downstream operations tend to incur elevated costs in case of processing failures, therefore requiring a higher guarantee of processing quality. To simulate this practical production characteristic, this paper introduces a specific criticality parameter, denoted as A_{nk}, for each operation k of any job n. $A_{nk} = A_n + \alpha \cdot k$, where α is referred to as the criticality factor, indicating the degree to which the processing criticality of operation k is affected by the sequence of processing.

To simulate the inherent characteristic of processing time uncertainty in practical manufacturing systems, this study employs a triangular fuzzy processing time representation for operation O_{nk} on machine m, given by $t_{nkm} = (t^1_{nkm}, t^2_{nkm}, t^3_{nkm})$. Here, t^1_{nkm} and t^3_{nkm} represent the minimum and maximum processing times respectively, while t^2_{nkm} represents the most likely processing time. Therefore, the completion time C_{nk} of operation O_{nk} in this scheduling problem can be expressed as $C_{nk} = (C^1_{nk}, C^2_{nk}, C^3_{nk})$. The following assumptions underpin the scheduling problem discussed in this paper:

(1) Each operation must be completed on only one machine, and the process cannot be interrupted.
(2) No more than one operation can be processed on a single machine at any one time.
(3) The processing sequence and processing times are known in advance, ignoring setup time.
(4) The job criticality parameter A_n and machine processing level A_m are known and fixed.
(5) At time zero, all machines are idle, and all jobs are released.

The notations are listed in the table below.

Notation	Description
n, i	job ($n = 1,2, \dots, N, i = 1,2, \dots, N$)
m	machine ($m = 1,2, \dots, M$)
k, j	operation ($k = 1,2, \dots, k_n, j = 1,2, \dots, k_n$)
M	the total number of machines
N	the total number of jobs
k_n	the total number of operations for job n
O_{nk}	the k^{th} operation of job n
t_{nkm}	the processing time on machine m for operation O_{nk}
S_{nk}	the set of optional machines that operate O_{nk}
A_m	the processing level of machine m
A_n	the criticality of job n
α	the criticality factor
A_{nk}	the criticality of operation O_{nk}, $A_{nk} = A_n + \alpha \cdot k$
x_{nkm}	$x_{nkm} = \begin{cases} 1, \text{operation } O_{nk} \text{ is processed on machine } m \\ 0, \text{otherwise} \end{cases}$
Y_{ijnk}^m	$Y_{ijnk}^m = \begin{cases} 1, \text{if } O_{ij} \text{is the immediate preceding process of } O_{nk} \text{ on machine } m \\ -1, \text{if } O_{ij} \text{is the immediate post process of } O_{nk} \text{ on machine } m \\ 0, \text{otherwise} \end{cases}$
C_{nk}	the completion time of operation O_{nk}, $C_{nk} = (C_{nk}^1, C_{nk}^2, C_{nk}^3)$

The mathematical model of the MO-fFJSP studied in this paper is as follows:

$$\min\ f_1 = \sum_{1 \le n \le N} \max\{C_n\} \tag{1}$$

$$\max\ f_2 = \sum_{n=1}^{N} \sum_{k=1}^{k_n} \sum_{m \in S_{nk}} (x_{nkm} \cdot A_m \cdot A_{nk}) \tag{2}$$

s.t.

$$\sum_{m=1}^{M} x_{nkm} = 1, \quad \forall n, k \tag{3}$$

$$\mathrm{C}_{nk} - \mathrm{C}_{n(k-1)} \ge \sum\nolimits_{m=1}^{M} (x_{nkm} \cdot \mathrm{t}_{nkm}), \quad \forall n; \mathrm{k} > 1 \tag{4}$$

$$\begin{cases} C_{nk} - C_{ij} \ge \mathrm{t}_{nkm}, & Y_{ijnk}^m = 1, n \ge i, k > j \\ C_{ij} - C_{nk} \ge \mathrm{t}_{ijm}, & Y_{ijnk}^m = -1, n \ge i, k > j \end{cases}, \forall m, \mathrm{i}, \mathrm{j}, \mathrm{n}, \mathrm{k} \tag{5}$$

$$x_{nkm} \in \{0,1\}, \forall n, k, m \tag{6}$$

$$Y_{ijnk}^m \in \{1,0,-1\}, \forall m, i, j, n, k \tag{7}$$

$$C_{nk} > 0, \forall n, k \tag{8}$$

The objective functions corresponding to equations (1) and (2) represent the minimization of maximum completion time and the maximization of job processing quality. Constraint (3) defines that each operation can only be completed on one machine.

Constraint (4) ensures that jobs are processed according to a fixed operation sequence. Constraint (5) states that a machine can only perform one operation at a time. Constraints (6)-(8) restrict the numerical type of decision variables.

2.2 Operational rules of TFN

This section establishes the operational rules (addition, maximization, and comparison) for TFN to standardize the computation process of solving MO-fFJSP. The addition operator calculates the completion time for each operation, the maximization operator determines the start time for each operation, and the comparison operator compares the maximum completion time.

For triangular fuzzy processing times $t^1 = (t_1^1, t_2^1, t_3^1)$ and $t^2 = (t_1^2, t_2^2, t_3^2)$, their addition operation is defined as follows:

$$t^1 + t^2 = (t_1^1 + t_1^2, t_2^1 + t_2^2, t_3^1 + t_3^2) \tag{9}$$

The maximization operation is defined as follows:

$$\mathrm{t}^{max} = (\max(t_1^1, t_1^2), \max(t_2^1, t_2^2), \max(t_3^1, t_3^2)) \tag{10}$$

Due to the imprecise nature of fuzzy processing times, the comparison operation between two fuzzy numbers is complex. For the triangular fuzzy processing time used in this study, the most accurate comparison method is to compare the ratio of the overlapping region between two fuzzy numbers. However, calculating the overlap ratio rapidly is challenging in practice. Hence, some studies have simplified the comparison operation of fuzzy numbers. A comparison method for a set of fuzzy numbers proposed by Wang et al. (2013) has been frequently referenced in recent years. However, this method only uses a few simplified fixed formulations for comparison and cannot adjust the fuzzy number comparison parameters according to the manager's preference, limiting its practical application in production. Therefore, we propose a straightforward and adaptable weighted comparison approach for comparing the sizes of two TFNs. This comparison method considers the size relationship between two TFNs from two aspects: the center value and the spread.

For triangular fuzzy processing times $t^1 = (t_1^1, t_2^1, t_3^1)$ and $t^2 = (t_1^2, t_2^2, t_3^2)$, t_2^1 and t_2^2 are center values, while $t_3^1 - t_1^1$ and $t_3^2 - t_1^2$ are the spread of the fuzzy numbers. We define the comparison operation as follows:

$$t_2^1 - t_2^2 = \rho * (t_2^1 - t_2^2) + (1 - \rho) * [(t_3^1 - t_3^2) + (t_1^1 - t_1^2)] \tag{11}$$

Here, ρ represents the decision-maker's emphasis on the center values of the fuzzy numbers. A higher value of ρ implies that the differences in center values between the two fuzzy numbers are more important. The selection of the parameter ρ holds the potential to impact the outcomes of the comparison between the two TFNs. In practical production scenarios, if there is a high probability that the fuzzy processing times fall near the center values or if the manager considers the differences in center values significant, it is recommended to set $\rho = 1$. Conversely, ρ can be set to 2 or 3.

3. IMOSMO for solving MO-fFJSP

The SMO algorithm is a population-based intelligent optimization algorithm that models spider monkeys' distinctive fission-fusion social structure (FFSS). FFSS is characterized by periodically splitting a large population into multiple smaller groups, which are then merged back into a larger population. By separating the group into smaller groups, this social structure successfully decreases direct competition among group members. The SMO method is suitable for tackling complex flexible job-shop scheduling problems because it can successfully balance the trade-off between exploration and exploitation (Bansal et al., 2014).

The conventional SMO was developed to address single-objective continuous optimization problems (Bansal et al., 2014). In this paper, genetic operators and variable neighborhood search are introduced into the SMO framework, and combined with Pareto optimization theory, an improved multi-objective spider monkey optimization algorithm (IMOSMO) is devised to address the MO-fFJSP.

3.1 Encoding and decoding

The key to the flexible job shop scheduling problem is to optimize two subproblems: machine selection and operation sequencing. Therefore, this study adopts a two-layer coding method. As shown in Figure 1, the first layer represents the selection of machines, where "3-2-4" are the machines selected for each of the three processes of job 1, "2-1-1-2" are the machines selected for each of the four processes of job 2, "3-1" are the machines selected for each of the two processes of job 3, and "1-2-4" are the machines selected for each of the three processes of job 4. The second layer represents the sequence of operations, where the first "4" in the sequence indicates the first processing operation of job 4, the second "4" indicates the second processing operation of job 4, the third "4" indicates the third processing operation of job 4, and so on. When these

two layers are integrated, it becomes evident that the first processing operation of job 3 will be performed on machine 3. The decoding method used is the insertion-based greedy decoding algorithm (Chaoyong et al., 2007), which ensures that an active schedule is obtained after decoding.

Machine coding	3	2	4	2	1	1	2	3	1	1	2	4
Operation coding	3	4	4	2	3	1	2	1	2	1	2	4

Fig. 1. Case of Encoding Scheme

3.2 Population initialization

Randomly generate an initial population. Generate a machine coding sequence by selecting one machine from the candidate machines set for operation k of job m randomly. Generate an operation coding sequence randomly. To ensure fair comparison in the comparative experiments in Section 4, different algorithms use completely identical settings for the initial population selection for the same instance.

3.3 Local leader phase

A crossover operator is introduced to replace the strategies for location updates within the conventional SMO algorithm in this phase. At the machine coding layer, a multi-point crossover operator is used, while at the operation coding layer, a precedence operation crossover (POX) operator is used. The crossover operator for machine coding layer works as shown in Figure 2: first, generate a random sequence $rand\_seq$ consisting of 1 and 2 of the same length as the machine coding layer. The positions where $rand\_seq$ has 2 are indexed as $index2$. Then, the segments in the machine coding layers of $SM_1$and SM_2 at the positions indexed by $index2$ are swapped, resulting in two new positions, SM_{new1} and SM_{new2}. The crossover operator for operation coding layer works as shown in Figure 3: first, the jobs are separated into two distinct sets, S_1 and S_2, randomly. Then, in SM_{new1}, the segments that correspond to jobs in S_1 are preserved, while the remaining segments are filled in order by the segments from SM_2 that correspond to jobs in S_2. In SM_{new2}, the segments that correspond to jobs in S_2 are preserved, while the remaining segments are filled in order by the segments from SM_1 that correspond to jobs in S_1.

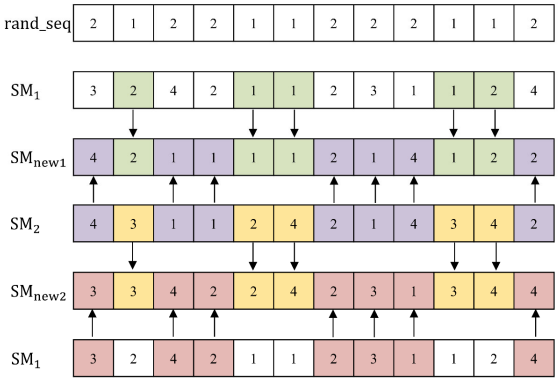

Fig. 2. Crossover operator for machine layer

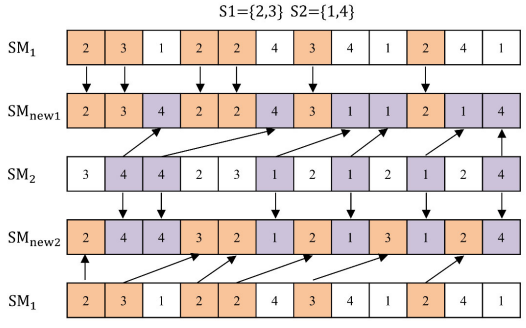

Fig. 3. Crossover operator for operation layer

The algorithm for LLP is shown in algorithm 1.

Algorithm 1 LLP
for $k = 1: group\_num$ // for each group
  Get k^{th} group $group_k$ and its size $group\_size_k$;
  for $i = 1: 2: group\_size_k - 1$
    if $rand \geq pr$
      Perform cross operations on SM_i and SM_{i+1} in $group_k$;
    end if
  end for
end for

3.4 Global leader phase

This phase introduces the mutation operator to replace the position updating strategy in the global leader phase of the conventional SMO. Specifically, a single-point mutation operator is used at the machine coding layer and an insertion mutation operator is applied at the operation coding layer. The mutation process at the machine coding layer involves randomly selecting a position within the sequence of machine coding and then replacing the chosen machine with another machine from the set of candidate machines for that operation. On the other hand, the mutation process at the operation coding layer is to randomly select a position within the sequence of operation coding and insert the operation at that position into another randomly selected position distinct from the current one. In addition to this, the strategy of updating the spider monkey position based on probability $prob$ in the conventional SMO is hardly applicable to multi-objective problems, therefore, in this paper, we introduce a binary tournament selection strategy to improve the algorithm and make it more applicable in solving multi-objective problems.

The algorithm for GLP is shown in algorithm 2.

Algorithm 2 GLP
Calculate the fitness of each population and perform non-dominated sorting;
for $k = 1: group\_num$ // for each group
  Get k^{th} group $group_k$ and its size $group\_size_k$;
  Perform a binary tournament selection operation on $group_k$ to get $group_{new}$;
  for $i = 1: group\_size_k$
    if $rand < pr$
      Perform mutation operation on SM_i in $group_k$;
    end if
  end for
  Merge $group_k$ and $group_{new}$, and update $group_k$ using an elitism preservation strategy;
end for

3.5 Local leader learning phase

In the conventional SMO algorithm, the local leader learning phase commences following the completion of the global leader learning phase. However, if we use the same order when solving discrete optimization problems with multi objectives, it will result in a situation where both the entire population needs to undergo non-dominated sorting for updating the global leaders, and each subgroup needs to undergo separate non-dominated sorting for updating the local leaders. This would create an unnecessary computational burden. Hence, the proposed algorithm moves the local leader learning phase to occur before the global leader learning phase.

The algorithm for LLLP is shown in algorithm 3.

Algorithm 3 LLLP
for $k = 1: group\_num$ // for each group
  Get k^{th} group $group_k$ and the local leader of the k^{th} group $local\_leader_k$;
  perform non-dominated sorting on $group_k$;
  The individual with the highest non-dominance rank is selected as the set $candidate$;
  if $local\_leader_k$ is same as $candidate$
    The $local\_limit\_count_k$ is increased by 1;
  else
    Update the $local\_leader_k$ to $candidate$;
    Reset $\text{local\_limit\_count}_k$ to 0;
  end if
end for

3.6 Global leader learning phase

After the update of local leaders, perform non-dominated sorting on the current global leader along with the updated local leaders from all subgroups in order to update the global leader.

The algorithm for GLLP is shown in algorithm 4.

Algorithm 4 GLLP
Merge the $global\_leader$ with all $local\_leader$ into a set $candidate$;
Remove duplicate individuals from the $candidate$;
Perform non-dominated sorting on the $candidate$;
Update the $candidate$ to the individual with the highest non-domination rank;
if $global\_leader$ is same as $candidate$
  $global\_limit\_count$ is increased by 1;
else
  Update $global\_leader$ to $candidate$;
  Reset $global\_limit\_count_k$ to 0;
end if

3.7 Local leader decision phase

In this phase, a variable neighborhood search is introduced to replace the position updating strategy of the local leader decision stage in the conventional SMO algorithm. The machine coding layer's variable neighborhood search process (Wang et al., 2013) is as follows: a random integer I is generated from 1 to L, where L is half of the length of the machine coding layer. Select I distinct positions randomly from the machine coding layer sequence. For each selected position, a machine different from the one at that position, taken from the candidate machine set at that position, replaces the current machine (if there is only one candidate machine, the position is skipped).

The variable neighborhood search process for the operation coding layer involves the following steps: select two different jobs randomly. For these two jobs, identify all operations indexed as $index$ and their corresponding values $jobs$ within the operation coding layer. The $jobs$ are then randomly placed in the positions indexed by $index$. By repeating these two procedures N_{vns} times, a neighborhood is obtained. Finally using a greedy strategy to update the position after non-dominated ordering of the resultant neighborhood.

A case of variable neighborhood search is depicted in Fig. 4.

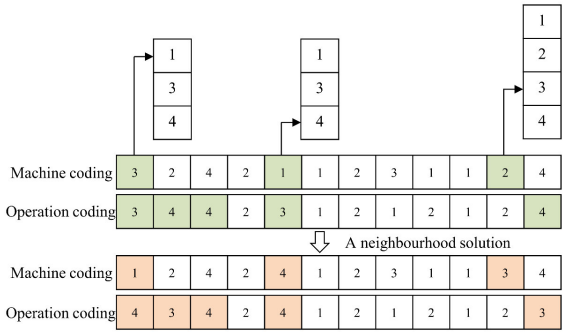

Fig. 4. Case of variable neighborhood search

The algorithm for LLDP is shown in algorithm 5.

Algorithm 5 LLDP
for $k = 1$: group_num // for each group
  if $local\_limit\_count_k > local\_leader\_limit$
    Reset $\text{local\_limit\_count}_k$ to 0;
    Get group_size_k;
    for $i = 1$: $group\_size_k$
      if $rand \geq pr$
        Updating SM_i of $group_k$ using random initialization method;
      else
        Updating SM_i of $group_k$ using variable neighborhood search algorithm;
      end if
    end for
    Calculate the fitness value of updated $group_k$;
    Perform non-dominated sorting on $group_k$;
    Update the $local\_leader_k$ of $group_k$;
    Remove duplicate individuals from the $local\_leader_k$;
end for

3.8 Global leader decision phase

The algorithm for GLDP is shown in algorithm 6.

Algorithm 6 GLDP
if $global\_limit\_count > global\_leader\_limit$
  Reset $global\_limit\_count$ to 0;
  if $group\_num < MG$ // MG is the given number of maximum group
    $group\_num = group\_num + 1$;
    Update $group\_num$ groups;
    Update $local\_leader$;
  else
    $group\_num = 1$;
    Update group to the whole population;
    Update $local\_leader$ to $global\_leader$;
  end if
  Reset local_limit_count to 0;
end if

4. Experiment results and algorithm performance analysis

This study designed 28 sets of MO-fFJSP test problems with varying scales for evaluating the performance of IMOSMO algorithm and conduct a comparation with the classical NSGA-II algorithm.

4.1 Test instances

The experiment divides the test instances into moderate-scale (denoted as Group A) and large-scale (denoted as Group B). Group A contains 12 (3*4) test instances and Group B contains 16 (4*4) test instances. Table 1 presents the characteristics of the instances.

Table 1
Characteristics of the instances

parameter	Group A	Group B
The number of jobs	{6,8,10}	{12,14,16,18}
The number of machines	{4,6,8,10}	{10,12,14,16}
The number of operations	[2,8]	[5,20]
Processing time	([3,4],[4,6],[5,8])	
Machine processing quality level	[1,4]	
Criticality of jobs	[1,5]， obey the normal distribution	
Critical factor	5/ average number of jobs	

4.2 Performance evaluation metrics

In this paper, we use the Inverted Generational Distance (IGD) (Czyżżak and Jaszkiewicz, 1998) and the Hypervolume (HV) (Zitzler & Thiele, 1999) indicators to evaluate algorithm performance. Specifically, IGD quantifies the average distance between all individuals in the true Pareto frontier and the obtained algorithmic Pareto frontier. A smaller IGD value signifies improved convergence and dispersion of the algorithm. On the other hand, HV measures the hypervolume enclosed by the Pareto frontier generated by the algorithm and a reference point. A larger HV value indicates superior overall algorithmic performance.

4.3 Parameter selection

The performance of an algorithm is closely tied to the selection of parameters; hence, parameter tuning holds paramount importance. The SMO algorithm involves four key parameters: LLL, GLL, MG, and pr (Bansal et al., 2014). In this study, LLL is set at twice the value of GLL, and the population size for all test instances is fixed at 300. A table of $L_{16}(4^3)$ orthogonal experimental designed for the three parameters GLL, MG, and pr is shown in Table 2. Among the test instances with equal numbers of jobs, one is randomly selected for parameter tuning experiment (total of 7 test instances).

Table 2
Parameter and level

Parameter	Level			
	1	2	3	4
GLL	10	20	30	40
MG	3	4	5	6
pr	0.1	0.2	0.3	0.4

Given the inherent stochastic nature of the algorithm, this study conducted 10 independent experiments for each parameter set. The performance metrics for each parameter set were the normalized average values of IGD and HV. Table 3 presents the orthogonal table and the corresponding performance index for the B11 instance within the set of 7 test instances. Meanwhile, the parameter response values for instance B11 are displayed in Table 4. Table 5 provides the parameter settings for all test instances.

Table 3
Orthogonal table and performance index

Experiment No.	Parameter			Performance index
	GLL	MG	pr	
1	1	1	3	0.6547
2	1	2	4	0.9023
3	1	3	2	0.6146
4	1	4	1	0.2126
5	2	1	4	0.3893
6	2	2	3	0.7717
7	2	3	2	0.2572
8	2	4	1	0.2400
9	3	1	2	0.2949
10	3	2	4	0.8058
11	3	3	3	0.6867
12	3	4	1	0.6876
13	4	1	1	0.6148
14	4	2	4	0.8403
15	4	3	2	0.5806
16	4	4	3	0.2201

Table 4
Response value for parameters

Level	Parameter		
	GLL	MG	pr
1	0.4394	0.3859	0.7447
2	**0.3280**	**0.0846**	0.5526
3	0.3750	0.5071	0.2001
4	0.4149	0.5798	**0.0601**

Table 5
Parameters setting for test instances

Instance	Parameter			
	GLL	MG	pr	t
A01-A04	20	4	0.4	90s
A05-A08	20	5	0.4	120s
A09-A12	20	4	0.4	150s
B01-B04	20	4	0.4	180s
B05-B08	20	4	0.4	210s
B09-B12	20	4	0.4	240s
B13-B16	10	4	0.4	270s

4.4 Experiment results

All the algorithms and experimental tests detailed in this study were executed through MATLAB 2019b, conducted on a computer with an “Intel Core i5-3450” CPU@3.1GHz and 12 GB RAM. Much like the parameter-tuning experiments, each test instance was iterated 10 times to ensure statistical robustness. Table 6 presents the experimental results of various algorithms after 10 iterations under the parameter settings shown in Table 5.

As depicted in Table 6, IMOSMO significantly outperforms NSGA-II in the number of victories in both IGD and HV performance metrics. In terms of mean performance, IMOSMO exhibits optimization superiority rates of 59.53% and 3.62% for the average IGD and HV metrics, respectively, in the large-scale test instances (Group B). These values are higher than the respective rates of 36.47% and 1.37% observed in the moderate-scale test instances (Group A). This suggests that

IMOSMO's algorithmic performance improves with increasing test instance scales. Regarding standard deviation, for the majority of test instances, IMOSMO exhibits lower standard deviations in both IGD and HV metrics compared to NSGA-II, implying that the stability of the IMOSMO algorithm surpasses that of NSGA-II.

Overall, IMOSMO outperforms NSGA-II notably in both moderate-scale and large-scale test instances, thereby confirming the efficacy of the IMOSMO algorithm. The impressive algorithmic performance of IMOSMO stems from its fission-fusion structure, which effectively addresses issues like local convergence and premature convergence, ensuring population diversity. Furthermore, the introduction of variable neighborhood search enhances the algorithm's local search capability and improves population quality.

Table 6
Results of experiments for algorithms

Instance	IMOSMO		NSGA-II	
	IGD	HV	IGD	HV
	Mean ± Standard Deviation			
A01	**0.0031±0.0015**	**0.5478±0.0044**	0.0032±0.0015	0.5475±0.0050
A02	0.0156±0.0086	0.4502±0.0245	0.0158±0.0084	**0.4509±0.0194**
A03	**0.0024±0.0022**	**0.4149±0.0090**	0.0036±0.0040	0.4144±0.0091
A04	**0.0016±0.0033**	**0.4278±0.0269**	0.0099±0.0061	0.4192±0.0226
A05	**0.0037±0.0024**	**0.5355±0.0109**	0.0133±0.0144	0.5190±0.0233
A06	**0.0045±0.0029**	**0.4809±0.0155**	0.0097±0.0056	0.4644±0.0243
A07	**0.0120±0.0145**	**0.3338±0.0391**	0.0124±0.0200	0.3337±0.0401
A08	**0.0021±0.0026**	**0.3613±0.0079**	0.0050±0.0067	0.3602±0.0079
A09	**0.0044±0.0015**	**0.5245±0.0106**	0.0052±0.0019	0.5179±0.0163
A10	**0.0069±0.0031**	**0.5233±0.0154**	0.0099±0.0051	0.5152±0.0181
A11	**0.0133±0.0179**	**0.4061±0.0433**	0.0169±0.0186	0.3979±0.0406
A12	**0.0062±0.0044**	**0.4751±0.0287**	0.0158±0.0082	0.4640±0.0283
B01	**0.0066±0.0060**	**0.5145±0.0182**	0.0169±0.0104	0.4916±0.0210
B02	**0.0026±0.0019**	**0.5697±0.0186**	0.0080±0.0044	0.5542±0.0184
B03	**0.0078±0.0098**	**0.5303±0.0345**	0.0099±0.0081	0.5203±0.0479
B04	**0.0154±0.0112**	**0.4424±0.0240**	0.0252±0.0105	0.4287±0.0369
B05	**0.0029±0.0015**	**0.5626±0.0284**	0.0151±0.0049	0.5430±0.0271
B06	**0.0048±0.0032**	**0.5507±0.0153**	0.0144±0.0081	0.5283±0.0246
B07	**0.0049±0.0067**	**0.5363±0.0174**	0.0267±0.0135	0.5137±0.0277
B08	**0.0151±0.0130**	**0.4497±0.0443**	0.0187±0.0144	0.4387±0.0325
B09	**0.0040±0.0040**	**0.5570±0.0279**	0.0166±0.0088	0.5316±0.0303
B10	**0.0022±0.0012**	**0.5046±0.0145**	0.0107±0.0060	0.4775±0.0208
B11	**0.0068±0.0052**	**0.5586±0.0117**	0.0151±0.0064	0.5430±0.0186
B12	**0.0049±0.0037**	**0.5197±0.0247**	0.0148±0.0089	0.5057±0.0242
B13	**0.0015±0.0014**	**0.5684±0.0072**	0.0100±0.0051	0.5525±0.0147
B14	**0.0028±0.0022**	**0.5536±0.0251**	0.0132±0.0063	0.5321±0.0287
B15	**0.0048±0.0046**	**0.5317±0.0323**	0.0159±0.0068	0.5101±0.0408
B16	**0.0069±0.0070**	**0.5397±0.0306**	0.0218±0.0113	0.5131±0.0268
Ratio	27/28	27/28	0/28	1/28

Fig. 5 depicts an illustrative example of the Pareto front for test instance A01. Due to the characteristics of maximizing triangular fuzzy numbers, the Pareto fronts for the minimum completion time, the most likely completion time, and the maximum completion time all contain individuals with dominance relationships. However, it should be noted that all solutions shown in the graph are non-dominated. For instance, solution S1 in the graph dominates solution S2 in terms of the minimum completion time, maximum completion time, and processing quality. Nevertheless, according to the triangular fuzzy number maximization operation proposed in this paper, S2 dominates S1 with respect to the completion time. As a result, solutions S1 and S2 are mutually non-dominated.

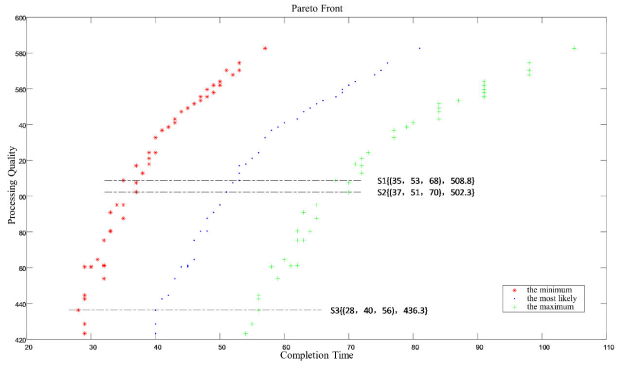

Fig. 5. Pareto front for A01

Fig. 6 presents a detailed scheduling Gantt chart for the non-dominated solution S3 from Fig. 5.

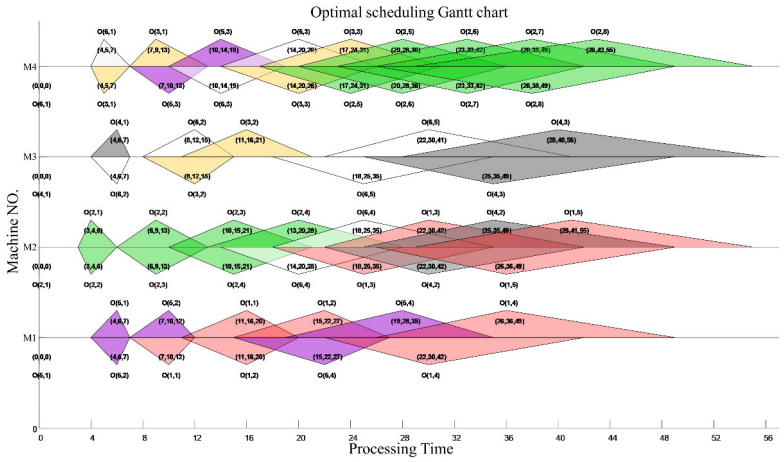

Fig. 6. The scheduling Gantt chart for S3

5. Conclusions

The fFJSP is a typical NP-hard problem and is a common category of production scheduling models in manufacturing systems. Therefore, this problem holds significant research significance and attracts substantial attention. Addressing the practical manufacturing characteristic of fuzzy processing times in the aerospace shaft production workshop, this paper introduces a fFJP with fuzzy processing times in the context of aerospace shaft manufacturing.

In practical production scenarios, scheduling problems often involve multiple objectives. Given the pronounced focus on customer satisfaction within aerospace shaft production workshops, this paper quantifies customer demand (termed as job quality levels) and processing resource capacities to construct a maximization objective for total processing quality. Alongside minimizing the maximum completion time, a MO-fFJSP is formulated to reflect the real-world aerospace shaft production context.

To tackle this problem, the IMOSMO algorithm is devised. The conventional SMO algorithm is traditionally applied to single-objective continuous optimization problems. This paper incorporates genetic operators and variable neighborhood search into the SMO framework. By integrating Pareto optimization theory, IMOSMO is tailored for solving discrete combinatorial optimization problems. To enhance computational efficiency, this paper also adjusts the execution sequence of the local leader learning phase and the global leader learning phase within the multi-objective discrete SMO algorithm framework, thus reducing the computational burden of individual evaluation and sorting.

To verify the algorithm's performance, two sets of test instances with different scales are designed: a large-scale problem set consisting of 16 instances and a moderate-scale problem set containing 12 instances. The classic NSGA-II algorithm is adopted as a benchmark for comparison, and experiments are conducted on all 28 instances. In terms of the number of victories in both IGD and HV metrics, the results show that IMOSMO surpasses NSGA-II. Through comparative analysis of algorithm performance, it is observed that IMOSMO's performance improves with larger problem scales, and its stability surpasses that of NSGA-II, thus confirming the efficacy of the proposed algorithm.

References

Bansal, J. C., Sharma, H., Jadon, S. S., & Clerc, M. (2014). Spider monkey optimization algorithm for numerical optimization. *Memetic computing*, *6*, 31-47.

Caldeira, R. H., Gnanavelbabu, A., & Vaidyanathan, T. (2020). An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption. *Computers & Industrial Engineering*, *149*, 106863.

Chaoyong, Z., Yunqing, R., Peigen, L., & Xinyu, S. (2007). Bilevel genetic algorithm for the flexible job-shop scheduling problem. *Journal of mechanical engineering*, *43*(4), 119-124.

Czyżżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of multi-criteria decision analysis*, *7*(1), 34-47.

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*, *65*, 52-67.

García Gómez, P., González-Rodríguez, I., & Vela, C. R. (2023). Enhanced memetic search for reducing energy consumption in fuzzy flexible job shops. *Integrated Computer-Aided Engineering*, *30*(2), 151-167.

Han, Y., Gong, D., Jin, Y., & Pan, Q. K. (2016). Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time. *Applied Soft Computing*, *42*, 229-245.

Joo, B. J., Shim, S. O., Chua, T. J., & Cai, T. X. (2018). Multi-level job scheduling under processing time uncertainty. *Computers & Industrial Engineering*, *120*, 480-487.

Kong, W., Ding, J., Chai, T., Zheng, X., & Yang, S. (2013, April). A multiobjective particle swarm optimization algorithm for load scheduling in electric smelting furnaces. In *2013 IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES)* (pp. 188-195). IEEE.

Li, X., Gao, L., Wang, W., Wang, C., & Wen, L. (2019). Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time. *Computers & Industrial Engineering*, *135*, 1036-1046.

Lin, J., Zhu, L., & Wang, Z. J. (2019). A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *127*, 1089-1100.

Pan, Z., Lei, D., & Wang, L. (2021). A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(8), 5295-5307.

Sassi, J., Alaya, I., Borne, P., & Tagina, M. (2022). A decomposition-based artificial bee colony algorithm for the multi-objective flexible jobshop scheduling problem. *Engineering Optimization*, *54*(3), 524-538.

Wang, L., Zhou, G., Xu, Y., & Liu, M. (2013). A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. *International Journal of Production Research*, *51*(12), 3593-3608.

Xu, W., Ji, Z., & Wang, Y. (2018). A flower pollination algorithm for flexible job shop scheduling with fuzzy processing time. *Modern Physics Letters B*, *32*(34n36), 1840113.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, *3*(4), 257-271.