Complex & Intelligent Systems
https://doi.org/10.1007/s40747-024-01374-7

ORIGINAL ARTICLE

Flexible job-shop scheduling problem with parallel batch machines based on an enhanced multi-population genetic algorithm

Lirui Xue[1] · Shinan Zhao[1] · Amin Mahmoudi[2] · Mohammad Reza Feylizadeh[3]

Received: 6 July 2023 / Accepted: 21 January 2024

Abstract

The flexible job-shop scheduling problem (FJSP) with parallel batch processing machine (PBM) is one of those long-standing issues that needs cutting-edge approaches. It is a recent extension of standard flexible job shop scheduling problems. Despite their wide application and prevalence in practical production, it seems that current research on these types of combinatorial optimization problems remains limited and uninvestigated. More specifically, existing research mainly concentrates on the flow shop scenarios in parallel batch machines for job shop scheduling but few literature emphasis on the flexible job shop integration in these contexts. To directly address the above mentioned problems, this paper establishes an optimization model considering parallel batch processing machines, aiming to minimize the maximum completion time in operating and production environments. The proposed solution merges variable neighborhood search with multi-population genetic algorithms, conducting a neighborhood search on the elite population to reduce the likelihood of falling into local optima. Subsequently, its applicability was evaluated in computational experiments using real production scenarios from a partnering enterprise and extended datasets. The findings from the analyses indicate that the enhanced algorithm can decrease the objective value by as much as 15% compared to other standard algorithms. Importantly, the proposed approach effectively resolves flexible job shop scheduling problems involving parallel batch processing machines. The contribution of the research is providing substantial theoretical support for enterprise production scheduling.

Keywords Parallel batch-processing machines · Variable neighborhood search · Multi-population genetic algorithms

Introduction

"Production scheduling" is an integral component of production management, impacting nearly all enterprises in the manufacturing context. In 1954, renowned American scholar Johnson published the inaugural research paper on scheduling problems, focusing on a small-scale assembly line model with minimum total elapsed time [1]. Since then, research on production scheduling has evolved and become one of the most extensively studied topics in optimization problems [2]. At the dawn of the twentieth century, Vieira et al. discussed the importance of scheduling in the performance and productivity of manufacturing [3]. The authors introduced an algorithm for rescheduling manufacturing systems in response to abrupt changes and unexpected disruptions during the production process. Mahmoodjanloo et al. highlighted reconfigurable machine tools (RMTs) for a group of machines to satisfy manufacturing and production requirements. Authors applied RMTs to production scheduling under the FJSP context [4].

Various factors increase the complexity of these problems: (1) Building the foundation and structure of standard job-shop scheduling problems, and (2) multiple selectable devices and flexibility in employing pieces of machine for each operation [5, 6].

✉ Shinan Zhao
shinan89@just.edu.cn

Lirui Xue
liruiray@foxmail.com

Amin Mahmoudi
mahmoudi@seu.edu.cn

Mohammad Reza Feylizadeh
feylizadeh@iaushiraz.ac.ir

1 School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China

2 Department of Construction and Real Estate, School of Civil Engineering, Southeast University, Nanjing 210096, China

3 Department of Industrial Engineering, Shiraz Branch, Islamic Azad University, Shiraz, Iran

Published online: 02 March 2024

The flexible job-shop scheduling problem with parallel batch machines is an extension of the standard flexible job-shop scheduling problem (FJSP). It integrates a parallel batch processing machine to handle multiple tasks at the same time and completes them simultaneously. The application of parallel batch machines is inherently linked to specific production processes. For instance, in semiconductor manufacturing, several single-piece production processes are engaged, such as photolithography, metallization, and batch processing (i.e., wet oxidation and etching).

In metal casting companies, the production process is typically divided into several key stages. The initial stage involves batch production processes such as forging and furnace operations. Subsequent stages focus on single-piece production, where machining is critical for final shaping and molding. Jobs await processing in batches during the furnace stage. These queued jobs are batched together and processed collectively during this phase. Conversely, the single-piece machining stage aligns more closely with standard flexible production paradigms. As a result, the scheduling dilemma intertwined with these stages presents considerable implications for industries, especially those in semiconductor manufacturing and casting.

The flexible job shop scheduling problem with parallel batch processing machines has attracted widespread attention in the industry. However, most studies focus on the standard flexible job-shop scheduling problem (FJSP) and there is a lack of relevant research to address the FJSP considering parallel batch processing machines. The nuanced coordination and integration requirements evident across multiple processing stages in these scenarios can be regarded as practical illustrations of the flexible job-shop scheduling problem with parallel batch machines. Such instances are increasingly being acknowledged as pervasive challenges in production scheduling processes, underscoring their profound significance in ongoing research endeavors.

In addressing the flexible job shop scheduling problem incorporating parallel batch processing machines, this paper establishes a model that extends the standard flexible job shop problem (FJSP). To achieve this goal, we introduce a batch allocation strategy based on immediate predecessor operations, and devise a solution that merges two well-known algorithms: variable neighborhood search (VNS) algorithm and multi-swarm genetic algorithm. Furthermore, the enhancements made to these methods and algorithms are empirically tested using real-world data from an enterprise. Computational experiments show that the proposed approach effectively solves the scheduling problem in a flexible job shop with parallel batch processing machines. Compared to standard algorithms, the improved algorithm has increased performance by 15%. These achievements offer considerable theoretical backing for practical production scheduling in enterprises.

The main contributions of this study are listed below:

(1) Introduction of comprehensive scheduling of parallel batch processing machines in a complex flexible job shop environment.

The introduction of parallel batch processing machines in a flexible job shop environment leads to the establishment of an optimization model for the flexible job shop scheduling problem involving these machines. Most existing studies consider the issue of parallel batch processing machines in an assembly line environment.

(2) Improvement in the combination of variable neighborhood search algorithms and multi-population genetic algorithms.

Previous research primarily combined variable neighborhood search (VNS) algorithms with standard genetic algorithms. This paper is the first to combine VNS with multi-population genetic algorithms, aiming to solve the flexible job shop scheduling problem with parallel batch processing machines. Concurrently, this hybrid algorithm introduces the Hamming distance initialization method and an effective immigration method.

(3) The proposition of the immediate preceding operation batching method.

This method assigns batches according to the end time of immediate preceding operations within compatible job clusters, reducing the resource waste of parallel batch processing machines.

The remainder of this article is organized as follows: Research background and previous studies introduces previous work. Problem Description and Optimization Model presents the model of the flexible job shop scheduling problem with parallel batch processing machines. Solving Method introduces the methods used for solving the model and parts of optimization. Numerical studies validates the approach through case studies, and the final section concludes.

Research background and previous studies

Flexible job shop scheduling problem

The flexible job shop scheduling problem (FJSP) is a type of combinatorial optimization problem with practical application, first proposed by Brucker and Schlie [7]. The classic job shop scheduling problem involves scheduling a group of jobs, each with a designated machine. The FJSP becomes a more complicated problem by adding multiple selectable devices [8].

FJSP has been proven to be an NP-hard problem, and there is extensive research on how to solve it [9]. The most common solution methods include precise algorithms and intelligent algorithms. Precise algorithms include branch and

bound methods, linear programming, etc. The advantage of these methods is that they can provide the exact optimal solution. However, when the scale of the research object is large, or the research problem is complex, especially when the number of devices increases to a certain extent, the calculation process becomes difficult [10]. Therefore, these methods are usually used for specific problems.

Due to the limitations of precise algorithms, solutions to FJSP are mostly sought using intelligent optimization algorithms such as genetic algorithms and particle swarm optimization to obtain approximate optimal solutions. Nouiri et al. proposed an efficient distributed particle swarm optimization algorithm to solve the FJSP problem to minimize the maximum completion time [11]. The authors also validated the effectiveness of the algorithm through multiple standard datasets. Gao et al. focused on scheduling problems in remanufacturing engineering [12]. They used a two-stage artificial bee colony algorithm to solve the FJSP problem, focusing on the flexible job rescheduling problem with new job insertion. Marichelvam et al. improved the standard solution of the particle swarm algorithm by combining scheduling rule heuristic algorithms and then combined the variable neighborhood search algorithm with the particle swarm algorithm to solve the scheduling solution with the least time consumption [13]. A solution was proposed by Jin et al. to solve a flexible job shop scheduling problem with uncertain processing order and non-parallel batch processors [5]. To minimize the total flow time to the greatest extent, Nagano et al. proposed an efficient constructive heuristic method [14]. Furthermore, an enhanced genetic algorithm was developed based on an enhanced selection method to reduce the possibility of early population maturity [15].

To sum up this sub-section, many studies focus on the accuracy of the algorithm by improving selection, crossover, etc. They also established different optimization models for various objectives in the flexible job shop context.

FJSP with parallel batch machines

Batch machines can be divided into parallel batch machines and serial batch machines. Serial batch machines operate on a first-come and first-served principle, so that they can execute production tasks one by one and can only process one job unit at a time. Parallel batch machines, in contrast, can handle multiple jobs simultaneously, and the job processed in the same batch have the same start and end times.

Many studies on batch processing machines mainly focus on how to batch and schedule batches. Fowler et al. summarized research on parallel batch processing, focusing on the scheduling within parallel machines and batching issues between compatible and incompatible job clusters [16]. However, they paid limited attention to the flexible job sequencing issues among non-parallel machines. Wu et al. discussed scheduling problems of progressively deteriorating parallel machines under uncertain conditions, but their focus remained primarily on the parallel machines [17]. A scheduling method for parallel batch processing machines with varying capacities was investigated by Jia et al. using triangular fuzzy numbers under uncertain conditions [18]. Their emphasis was on the uncertainties of the manufacturing environment and the differing processing capabilities and capacities of parallel batch processing. Zhang et al. investigated the issue of parallel batch processing machines in the context of cloud manufacturing, with a particular emphasis on parallel machines [19]. Moreover, a collaborative scheduling method was developed by Yuan et al. for a two-stage casting production line, with a primary focus on the coordination between the melting and casting processes and mechanical processing [20]. Regarding flexible production, most current research concentrates on flexible flow shop scheduling problems with parallel batch machines. For instance, Costa et al. [21] and Wang et al. [22] optimized batching in flexible flow shops using optimization algorithms. Furthermore, Vega et al. investigated a kind of unrelated parallel machine scheduling problem [23].

There has been little discussion to date on the flexible job shop scheduling problem with parallel batch machines. Ham introduced a scheduling problem for the semiconductor industry with varying priorities and parallel batch machines [24]. In this research, they proposed a mixed-integer model that focuses on solving priority and compatible job cluster issues. Song et al. introduced a method for solving these types of problems using a discrete particle swarm algorithm [25].

Although there are few studies on this problem, the flexible job shop scheduling problem with parallel batch machines still exists in real-world enterprise production. Therefore, this paper is looking forward to addressing this complex problem.

Problem description and optimization model

This section starts with the problem description in problem description, which is "the scheduling problem of a flexible job shop with existing parallel batch machines." Then, detailed comparisons and differences between this problem and existing production scheduling and flexible production will be provided. In the following sections, a planning model for this problem will be established on the common flexible job shop scheduling model.

Problem description

The flexible job shop scheduling problem with parallel batch processing machines inherits all the complexities of the flexible job shop scheduling problem [8]. To be more specific, some operations use parallel batch processing machines as

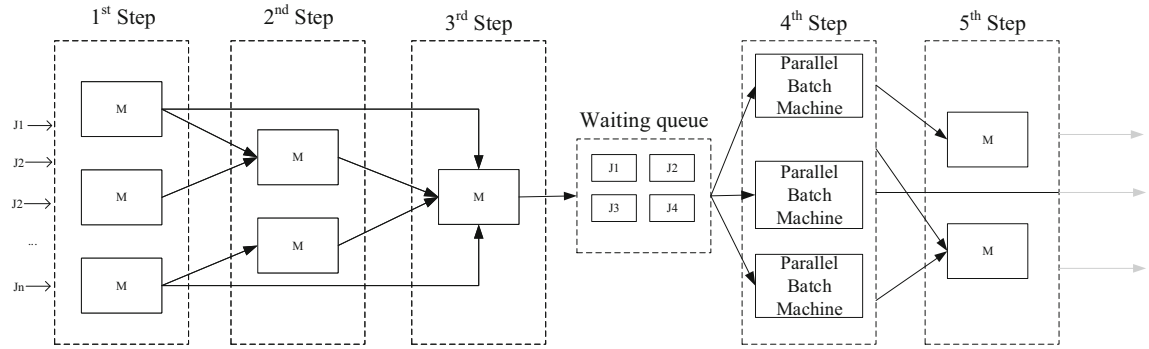


Fig. 1 A schematic flowchart of the production process in a flexible job shop with parallel batch machines. M = (Machines), Jn = (Job n)

machines or resources, which can process multiple jobs simultaneously within their capacity. The operations in the same batch on the parallel batch processing machine have the same start and end processing times. The parallel batch processing machine can only run one batch of operations in one processing cycle. The goal is also to obtain a superior production sequence and machine allocation, which raises the issue of batch allocation. Simplifying this process can yield potential outcome, as shown in the flowchart (Fig. 1). This flowchart reflects the production situation of the flexible job shop with parallel batch processors. In actual production, there are usually multiple operations using multiple parallel batch processing machines. Therefore, the schematic flowchart can reflect a part of the situation.

Optimization model

Assumptions and symbols

To simplify the problem, the scheduling problem considered in this paper is based on the following basic assumptions:

Assumption 1 Any job can be processed at time zero.

Assumption 2 The batch processing time is the maximum processing time of the jobs included in the batch. Jobs cannot be added or removed once batch processing begins.

Assumption 3 There is no intersection between single-processing machine operations and parallel-processing machine operations.

Assumption 4 Transit times between job locations are neglected.

Assumption 5 There are no constraints regarding the order/sequence of processing and operations of different jobs. But there are constraints regarding the order of processing between operations of the same job.

In this paper, the definitions of various symbols are listed in Table 1.

Table 1 Various symbols used in this research

Symbols	Definition
$i,j = 1,2,\ldots, n$	Operations
$k,l = 1,2,\ldots, n$	Jobs
$m,n = 1,2,\ldots, n$	Machines
$b = 1,2,\ldots,$ n	Batches
O_{ki}	The ith operation of the kth job
S_{ki}/E_{ki}	Start/end time of O_{ki}
T_{kim}	Processing time of O_{ki} on mth machine
B_{nb}	Set of bth batch operations on parallel processing machine n
U	Set of operations with parallel batch processing machines
C_{m}	Processing capacity of parallel processing machines
M_{ki}	Set of optional machines for O_{ki}
D_{ki}	Demand for processing capacity ($\forall$i $\in$ U)
H_{ki}	Equal to 1 if O_{ki} is a parallel machine, otherwise it is 0
Y_{kijl}	Equal to 1 if O_{ki} is a precedent operation of O_{kj}, otherwise it is 0
Z_1/Z_2	Auxiliary variables, S_{ki}/E_{ki} multiplied by Y_{kij}
Z_3	Auxiliary variables, D_{ki} multiplied by P_{kibn}

Decision variables:

$$X_{kim} = \begin{cases} 1, \ O_{ki} \in m \\ 0, \text{ Others} \end{cases} \tag{1}$$

$$P_{kibn} = \begin{cases} 1, \ O_{ki} \text{ processed on batch b of parallel } n \\ 0, \text{ Others} \end{cases} \tag{2}$$

Mathematical model

The optimization goal sought in this paper is to minimize the maximum completion time. That is, the most common

MakeSpan in scheduling problems, so that the production process is completed as quickly as possible, which aligns with the actual needs of enterprise production [26–28]. According to the problem description, there are three types of constraints, namely (1) operation constraints: the same type of job must complete the immediately preceding operation before starting the next one; (2) machine processing capacity constraints: non-parallel machines can only process one job at the same time, and the number of jobs processed by parallel machines at the same time cannot exceed its parallel capacity; (3) batch constraints: i.e., the same job can only be in one parallel machine batch, and the jobs in the same batch in the parallel machine operations must start and end processing at the same time.

In summary, the following mathematical model expressions can be obtained:

$$\text{Min}(\text{MakeSpan}) = \text{Min}(\text{Max}(E_{ki})) \tag{3}$$

S.T.

$$Z_1 - Z_2 > 0 \tag{4}$$

$$Y_{ij} + Y_{ji} = 1 \forall k,\ i,\ j \tag{5}$$

$$S_{ki} \geq 0,\ E_{ki} > 0 \forall i,\ j \tag{6}$$

$$Z_3 \leq C_m \tag{7}$$

$$E_{ki} = S_{ki} + T_{ki} \forall k,\ i \tag{8}$$

$$\sum_i X_{kim} = 1 \forall k,\ i,\ m \tag{9}$$

$$S_{ki} = S_{lj},\ E_{kj} = E_{lj} \forall i,\ j \in B_{mb} \tag{10}$$

$$P_{kiam} + P_{kibm} + P_{kian} + P_{kibn} = 1 \forall a \neq b,\ m \neq n,\ k,\ i \in U \tag{11}$$

Equation 3 signifies the objective function, which aims to minimize the duration between initiating production and completing all tasks, commonly called MakeSpan. Equations 4 and 5 serve as processing order constraints, ensuring that the same job gets produced according to the required operations without repetition; Eq. 6 implies a time constraint; Eq. 7 tackles the processing capacity constraint of parallel batch processing machines; Eq. 8 constitutes a processing time constraint; Eq. 9 conveys that a process can be performed on only one machine; Eq. 10 represents a batch processing time constraint, indicating that operations in the same batch have the same processing time; Eq. 11 stands as a batch constraint, the same operation can only be arranged in one batch. This model has undergone a linearization process and three auxiliary variables Z1–Z3 are used for linearization. More details can be found in Appendix A.

Solving method

The flexible job shop scheduling problem with parallel batch machines, an extension of the flexible job shop scheduling problem that includes parallel batch machines, is classified as an NP-Hard problem. Various solutions have been proposed for such NP-hard problems, including deterministic algorithms such as branch, bound, and backtracking [29]. Nevertheless, these methods may result in exponential computation time, rendering them impractical and inefficient for large-scale problems. As a heuristic algorithm, the Genetic Algorithm simulates the evolution process of biological species in nature, demonstrating commendable optimization capabilities, and therefore is widely utilized [30]. However, standard genetic algorithms still suffer from various issues, such as premature convergence [31–33]. To address this, we propose an enhanced multi-population Genetic Algorithm in this paper, combining variable neighborhood search (VNS) information for optimal individual selection. We enhance inter-population diversity through a design method based on Hamming distance similarity and introduce an "effective immigrant" method to boost the algorithm's performance.

Construction of fitness function

Fitness refers to measuring the quality of solutions in the current population, corresponding to the current scheduling problem, which serves as the basis for selection and evolution. In the context of production scheduling problems, a widely used approach for establishing fitness involves the utilization of the reciprocal of the objective function, namely MakeSpan, as demonstrated in Eq. 12.

$$s = \frac{1}{S} \tag{12}$$

According to Eq. 12, as the difference in fitness within the population decreases during the iterative process, it can lead to a situation where individuals with better fitness are less likely to be retained with a higher probability. Therefore, changes are made to the fitness function to increase the diversity within the population, thereby reducing the occurrence of such phenomena. The fitness function is altered to map its original values to the range [0,1], thereby amplifying its differences, as shown in Eq. 13. Here, *s'* represents the fitness value of the current individual, while Min(*S*) and Max(S), respectively, represent the minimum and maximum fitness

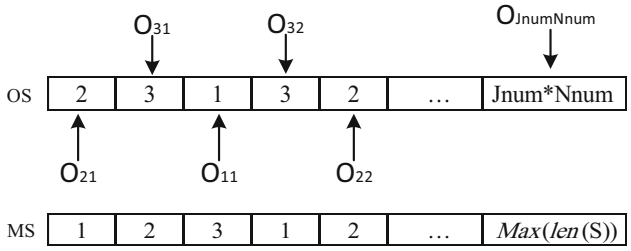


Fig. 2 Encoding method linking codes with processing operations

of the current population.

$$s = \frac{s\prime - \mathrm{Min}(S)}{\mathrm{Max}(S) - \mathrm{Min}(S)} \tag{13}$$

Chromosome encoding and decoding

Considering the research problem, the entire algorithm optimization process can be seen as solving two problems: the sequence of job processing and the allocation of machines. The grouping of jobs in the computational model is managed according to the completion time of the immediately preceding operation. A dual-layer encoding method is employed for both, assuming there are J_{num} jobs awaiting processing, And each job must undergo N_{num} operations. Different operations correspond to the same or different machine groups S as shown in Fig. 2. In this figure, the operation sequence (OS) layer employs a job number encoding scheme, where pending jobs are sequentially assigned distinct identifiers based on the order that they are to be processed. If the same identifier appears multiple times, then it represents different operations for this corresponding job. For instance, when a job with the identifier '2' appears for the first time in the encoding sequence, it represents operation O_{21}; if it is the second appearance, it refers to operation O_{22}. On the machine sequence (MS) layer, real-number encoding is further utilized. Its range is [1, max(len(S))], which corresponds to the maximum number of available machines in each machine group. In the process of decoding, these values are then translated to the actual machine identifiers.

Decoding is one of the topics studied in this paper. For problems involving parallel batch processing machines, a decoding method based on queuing and insertion mechanisms has been designed. The specific decoding steps are as follows:

Step 1: Read the job encoding and machine in order, and determine whether the operation corresponds to a parallel batch processing machine. If so, place it in the parallel batch processing queue; otherwise, place it in the regular queue.

Step 2: Place non-parallel batch processing in the processing queue according to the machine encoding. When reading the machine encoding, find the machine number in a cyclic manner. For example, if the machine layer encoding is k, and the current operation has m optional machine, choose the kth optional machine when $k < m$, and choose the $k - m + 1$th machine when $k \geq m$).

Step 3: Merge the operations that include parallel batch processing according to the processing order of the immediately preceding operation, and allocate the machine in batches according to the capacity of the parallel batch processing machine.

To explain this process in detail, let us take an example of 4 jobs, 4 machines, and 3 operations (the machine numbers are 1–3, and number 1 is a parallel batch processing machine with a processing capacity of 20 units). The encoding is the operation sequence layer (OS) [1, 3, 4, 2, 2, 4, 3, 1, 1, 2, 3, 4], and the machine sequence layer [3, 3, 2, 1, 2, 3, 4, 4, 1, 3, 4, 2]. Table 2 presents the process information of this illustrative example, including four jobs awaiting processing, their respective operations, available equipment, and requirements for parallel machine processing capabilities.

According to the process information and following steps 1–3 for decoding, we obtain the decoding result as shown in Fig. 3. The processing order, except for the parallel batch processing machine numbered 1, is from left to right for other operations, i.e., their processing order.

Table 2 A simple example of the decoding method

Job	Operation	Available machines	Requirements
1	1	2/4	N/A
	2	1	10 units
	3	2/3	N/A
2	1	3/4	N/A
	2	2	N/A
	3	1	10 units
3	1	2/3/4	N/A
	2	1	10 units
	3	2/4	N/A
4	1	2/3	N/A
	2	1	10 units
	3	3/4	N/A

Fitness evaluation

In the model for fitness computation in genetic algorithms, it is necessary to provide feedback on the corresponding fitness according to the information of the operation layer and the machine layer to carry out appropriate batch allocation work for operations with parallel machines. To obtain the fitness situation of the current solution, a calculation method

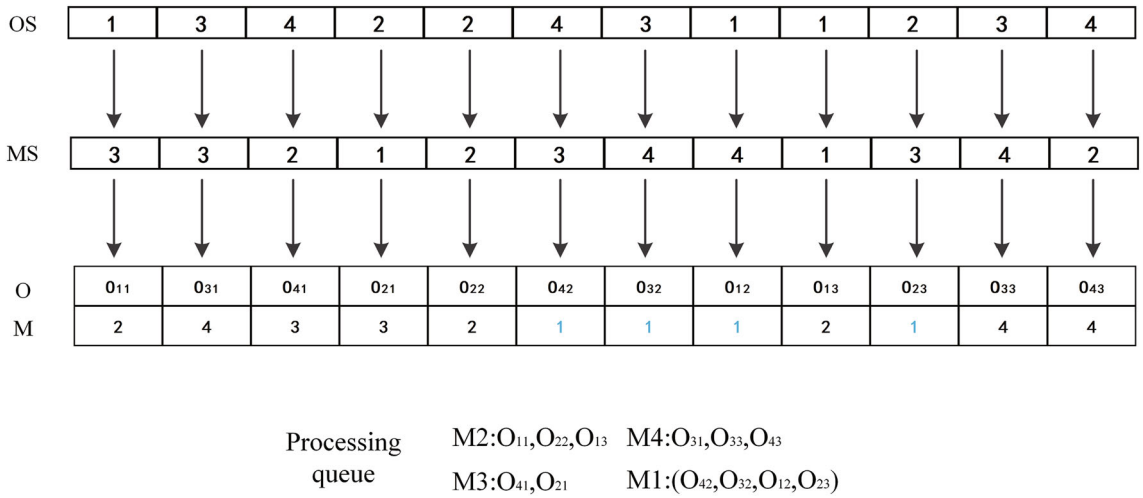


Fig. 3 A example of decoding process

simulating the production process is adopted here, and the production process is simulated according to the decoding result. This process first reads the operation layer and machine layer in order, then determines the current machine available time and the end time of the immediately preceding operation of the current operation (if any), takes the larger of the two values, and adds the processing time required for the current operation to obtain the start and end time of the operation processing, and updates the available time of the machine.

For parallel machines, this paper uses the "immediate preceding operation batching method to batch and group tasks. The method of immediate preceding operation batching first marks the encoding of the operation processed by the parallel batch processing machine (as marked in blue in Fig. 3), and generates a waiting queue. When processing to the marked operation machine, it first determines whether adding this operation reaches the upper limit of the parallel machine capacity and is not greater than the upper limit of the capacity. If so, after adding this operation to the waiting queue, all operations in the queue are simulated for the processing process. Otherwise, check whether there are parallel machine operations in future. If there are, put them into the waiting queue; otherwise, start processing immediately.

Enhanced algorithm

Considering the fact that the standard genetic algorithm (SGA) is prone to premature convergence, this paper proposes an enhanced multi-population genetic algorithm (MPGA) calculation method. This method enhances the search capability of the algorithm by initializing with Hamming distance and incorporating the idea of variable neighborhood search. An effective immigrant operator is introduced at the machine layer. The overall process of the enhanced algorithm can be seen in Fig. 4.

Initialization method based on hamming distance

In the algorithm used in this paper, a multiple-group initialization method based on Hamming distance similarity search with individual pre-selection assignment is adopted to enhance the search capability. Hamming distance is a calculation method to measure the degree of difference between two equal-length strings, which was first proposed by Richard Hamming, an American mathematician and the founder of information theory, in the 1950s. Since then, it has been widely used in many fields and domains. The Hamming distance between two individuals is calculated by first treating each chromosome as a positional sequence. A heterogeneity operation is then performed, yielding a sum. This sum is subtracted from the total length, resulting in the Hamming distance. In Eq. 14, 'n' represents the number of chromosomes, and 'SNi' stands for the value of the chromosome at the 'ith' position.

$$\text{Length(Sequence)} - \left(\sum_{i}^{n}(S1_i \oplus S2_i)\right) \tag{14}$$

The multi-population initialization method based on Hamming distance similarity calculation first generates a relatively large population $N(N \gg n)$ in a random manner, takes out individuals in order as multi-dimensional vectors δ, and then calculates δ with the remaining individuals $\gamma(\gamma \in \text{N}, \gamma \notin \delta)$ separately. The one with the lowest similarity is placed in another population, and so on, forming multiple populations. This process is shown in Fig. 5.

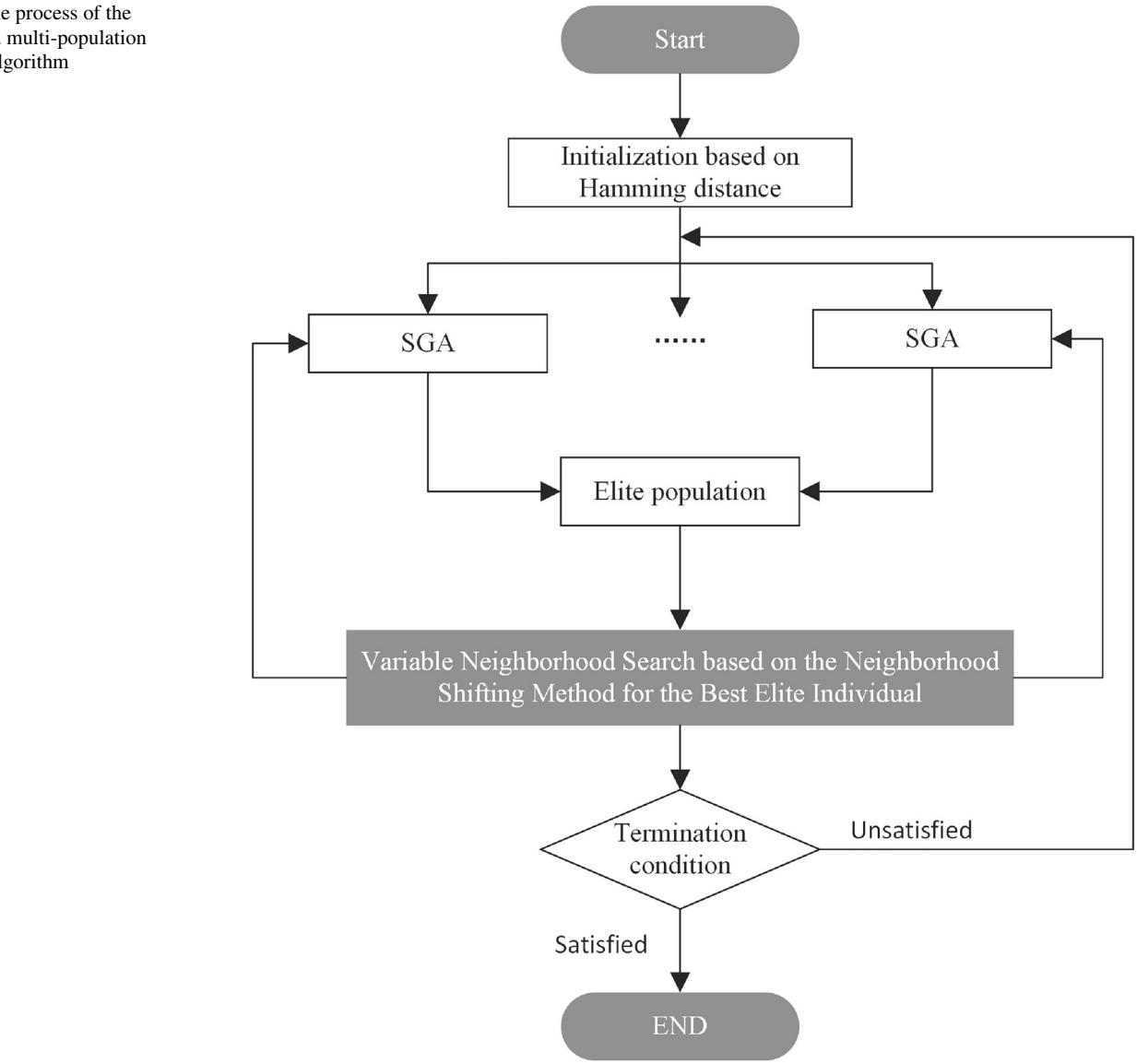


Fig. 4 The process of the enhanced multi-population genetic algorithm

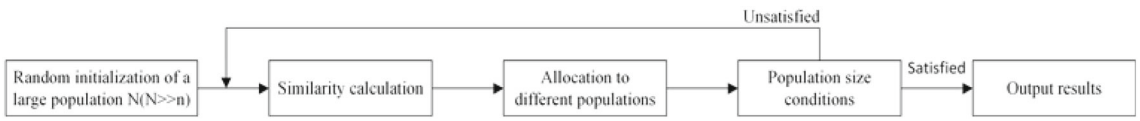


Fig. 5 The multi-population initialization method based on Hamming distance

Crossover, mutation, and selection method

In terms of selection methods, the "roulette wheel" is used to select individuals in the population. The principle of roulette wheel selection is that the probability of an individual being selected is related to its fitness. The better the fitness of the individual, the greater the probability of being selected, as shown in Eq. 14. In the Eq. 15, $f(x_i)$ represents the fitness of the 'i' individual, n represents the number of all individuals in the population, and pi represents the probability of the 'i' individual being selected.

$$P_i = \frac{f(x_i)}{\sum_n^j f(x_j)} \tag{15}$$

Regarding the crossover method, the operation layer uses POX [34, 35]. The POX crossover method works by randomly dividing the encoding set, J, into two groups: J1 and J2. It then places chromosomes from each group into corresponding positions to generate new individuals, C1 and C2.

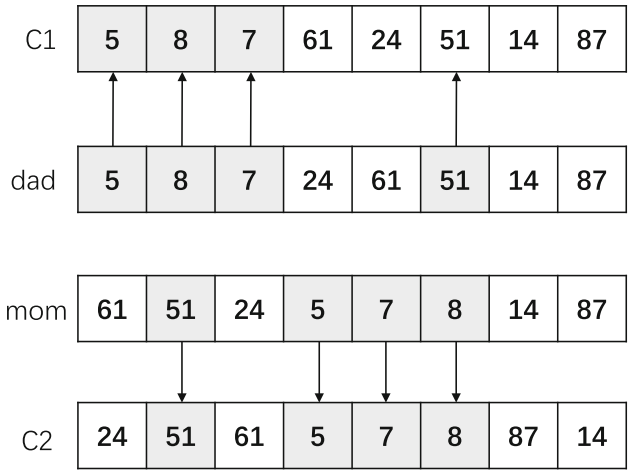


Fig. 6 POX crossover

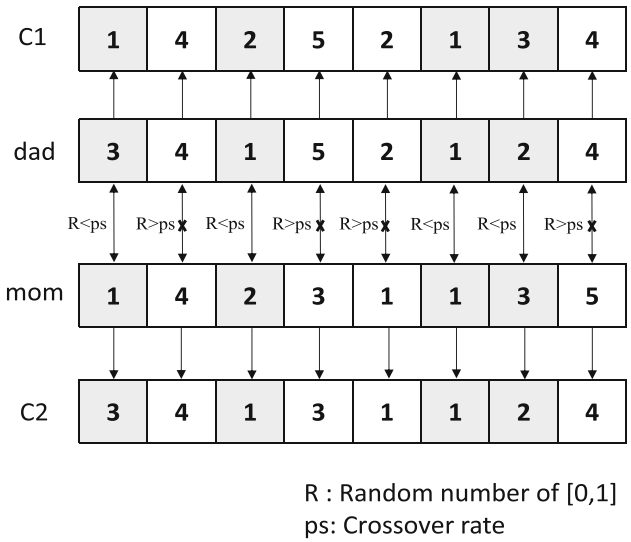


Fig. 7 Uniform crossover

Their fitness is calculated, and the one with higher fitness is taken as the result of the crossover. Figure 6 shows an example of a crossover, where the J1 set is {5, 8, 7, 51}, and the J2 set is {24, 61, 14, 87}. The machine layer uses a uniform crossover operator. Figure 7 is an example of uniform crossover, which is accomplished by calculating probabilities one by one. Figure 8 is an example of a mutation. Mutation uses swap mutation, where positions are randomly selected for exchange. The operation layer and the machine layer use the same mutation operator.

Effective immigrant operator

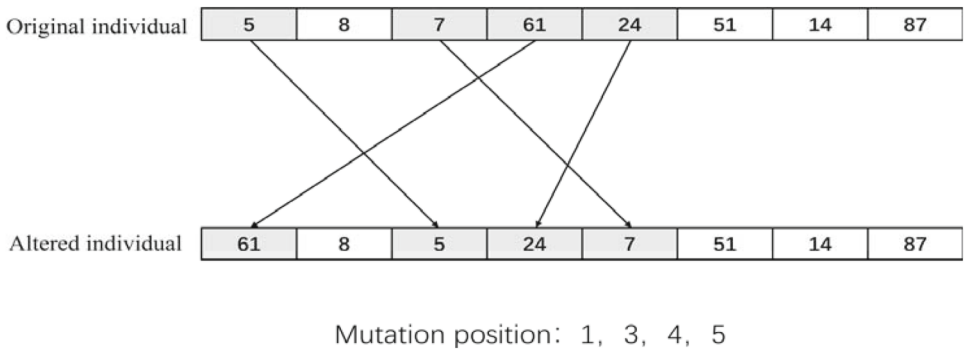
The immigrant operator in a standard multi-population genetic algorithm typically replaces the worst individual in one population directly with the best individual from another population. However, due to the complexity of the flexible job shop scheduling problem (FJSP), there are two parts of the encoding, operation, and machine that need to be migrated. How to avoid information loss caused by different layer migrations or to prevent the population from evolving in a worse direction is a problem that needs to be solved when applying MPGA to FJSP. This paper adopts a method of pre-validation of machine layer encoding for information exchange. This method mainly verifies and matches the individual to be migrated with the operation layer of the other population before the machine layer individual migration, confirms that it can bring improvement to the population, then assigns the genes of the better individual to the worse individual. The steps of this process are as follows:

Step 1—Search: In one population, identify the chromosome with the poorest fitness, termed as the "worst individual," and in another population, identify the chromosome with the optimal fitness, termed as the "best machine." These two are chosen as targets for migration.

Step 2-Pre-validation: Discard the corresponding machine individual of the worse chromosome, use the best fitness machine layer chromosome in another population as the operation layer of this operation layer, and calculate the fitness.

Step 3-Effective Immigration: Determine whether immigration can improve the original population. If it can, accept the new immigrant; otherwise, reject this immigration behavior.

Fig. 8 Mutation



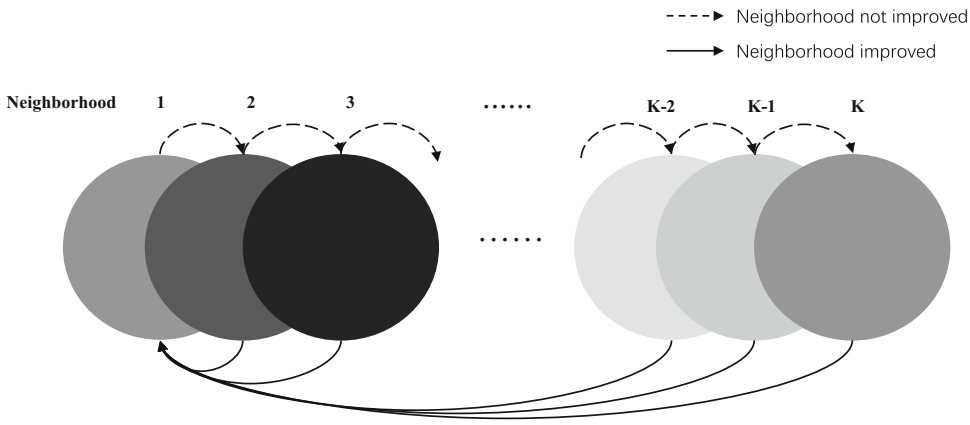


Fig. 9 Variable neighborhood search

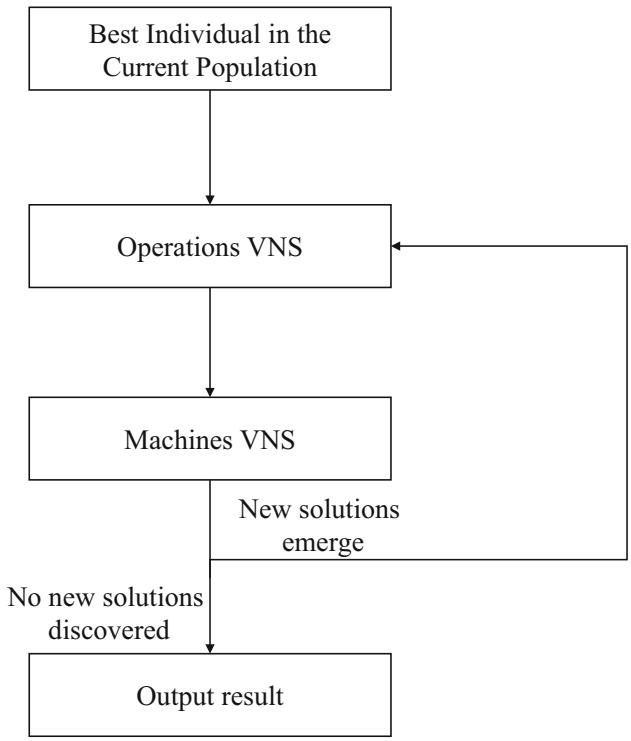


Fig. 10 Best individual variable neighborhood search

This method can reduce the worse genes brought by the immigration operation and avoid the possibility of the population moving in a worse direction due to the machine layer switch.

Best individual variable neighborhood search strategy

To further reduce the likelihood of the genetic algorithm falling into local optima, this paper introduces the concept of variable neighborhood search to perform secondary optimization on the best individual in the elite population. Variable neighborhood search can gradually improve the advantages of the current solution by searching in different neighborhoods. Combined with MPGA, it can search within a larger range, enhancing the solution's quality. The principle of the variable neighborhood search concept is shown in Fig. 9. First, perturbation is used for searching within the local neighborhood. When no better solution can be found within the local neighborhood, it moves to the next neighborhood. When the best solution is found within the local neighborhood, it returns to the initial neighborhood.

It is worth noting that here, only the best individuals in each generation and each population are searched, reducing the computational demand. This strategy first performs a variable neighborhood search on the operation layer. After the search is completed, the machine layer is searched without changing it. This process is repeated until no better solutions can be found in both the operation layer and the machine layer, as shown in Fig. 10.

In the MPGA approach, the problem is transformed using real number encoding, with the neighborhood structure being discrete. This paper introduces a method of generating different neighborhoods through parallel chromosome shifting, thereby expanding the search range. The neighborhood structure is depicted in Fig. 11.

After defining the neighborhood structure, it is necessary to design a perturbation method to search within the current neighborhood. Based on the neighborhood structure designed in this study, the perturbation method of the neighborhood adopts a non-equal exchange method. The operation level and the machine level use the same neighborhood structure and perturbation strategy, the difference lies in their different encodings. To begin with, the first chromosome in the current neighborhood is selected as the exchange target 1, then a different chromosome from the queue is randomly selected for exchange. Figure 12 shows this process using a partial chromosome as an example.

The pseudocode for the entire Best Individual Variable Neighborhood Search algorithm is shown in Algorithm BVNS.

```
Input : Best individual in operation layer as BO
Input : Best individual in machine layer as BE
Output: new BO
Output: new BE
Function VNS(BO,BE):
WHILE Changeop==1 AND ChangeEqu==1
 Start =0;
  WHILE Start<length(BO) do:
   TempBO = Shaking(BO,Start)   /* Proceed according to Shaking method */
   newF = Calculate Fitness(TempBO,BE)
   IF newF IS LESS THAN the original fitness
     BO =tempBO       /* Accepted new BO */
     Start =1;        /* Reset the search queue */
     Changeup=1;
   ELSE
     TempBO = BO;       /* Rejected new BO */
     Start = Start +1 ;    /*Move to the next */
    END
   Start=0;
  END
  WHILE Start<length(BE) do:
  /* Apply the same method for searching in the machine layer */
  TempBE = Shaking(BE,Start)
  newFbe = Calculate Fitness(BO,TempBE);
   IF newFbe is LESS THAN the newF fitness
     BE =tempBE       /* Accepted new BE*/
      Start =1;        /* Reset the search queue */
      ChangeEqu=1;
  ELSE
     TempBE = BE;       /* Rejected new BE */
     Start = Start +1 ;    /*Move to the next */
    END
RETURN newBO=BO , newBE =BE;
END
```

Algorithm BVNS: variable neighborhood search for the best individual

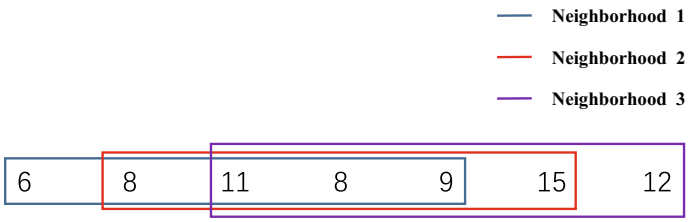


Fig. 11 Neighborhood structure

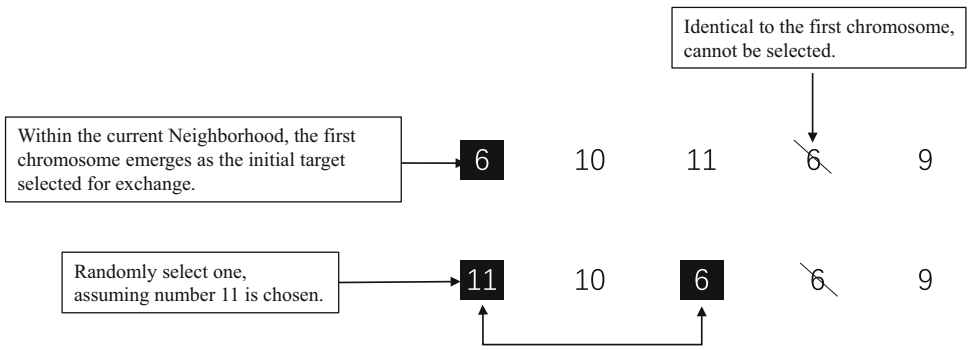


Fig. 12 Perturbation method

Numerical studies

Due to a lack of canonical instances addressing the flexible job-shop scheduling problem incorporating parallel batch processing machines, the open-source dataset in [36] is purposefully used for the standard flexible job-shop scheduling (MK01-MK10). This dataset is a recognized benchmark and spans various sizes and complexities of flexible job-shop scheduling scenarios. In terms of the methodology in [37], we augmented this dataset to ensure alignment with our research objectives.

To ascertain the practical relevance and effectiveness of our proposed methodology in real-world industrial settings, we ventured beyond the confines of the modified dataset. We examined a production scenario from a leading enterprise specializing in heavy industrial foundry and forging. This conglomerate boasts an intricate production landscape, encompassing sectors like steelmaking, forging, non-ferrous metal casting, mechanical machining, heat treatment, and holistic machinery manufacturing. Throughout its production spectrum, myriad machine alternatives and parallel batch processing operations are evident, with several potentially qualifying as parallel batch processing activities. Indeed, many of their production pathways align with the nuances of the flexible job-shop scheduling quandary when integrated with parallel batch machinery. For our empirical analysis, we opted for an authentic production sequence of a specific product category (the procedural route is delineated in Appendix B). Our study incorporated two distinct product archetypes, with the fourth operation contingent on parallel batch processing machinery. Given the enterprise's operational modalities, the parallel machinery's throughput stands at four units per batch, cumulating to a production demand for 24 distinct products. This scenario served as our computational and analytical fulcrum. For a comprehensive insight into the expanded dataset and the enterprise's intrinsic production modalities, one may refer to Appendices B and C, respectively.

Experimental design

In the entirety of our numerical experimentation, we conducted computational analyses on both extended open-source instances and real-world industrial production cases. We made comparative evaluations of the following algorithms: the standard variable neighborhood multi-population genetic algorithm proposed in this paper, the standard genetic algorithm, the standard multi-population genetic algorithm, the enhanced multi-population genetic algorithm, as well as the particle swarm optimization (PSO) algorithm with its various variants, and the novel dragonfly algorithm [38]. Notably, both the particle swarm optimization and dragonfly algorithms are typically tailored for continuous domain problems, necessitating a discretization to be apt for the issue under our study [39]. To achieve this discretization, for the PSO and its variants and the dragonfly algorithm under comparison, we adopted the 'Random Key' [40] concept for encoding discretization. For the multi-population component, as an example, a dual population model was employed: with a population size set at 100, and a maximum iteration count of 500. The crossover and mutation rates for Population 1 were set at 0.6 and 0.2 respectively, while those for Population 2 were 0.8 and 0.05. The maximum iteration limit remained at 500.The program corresponding to the algorithm in this paper runs on

Table 3 Calculated results of all datasets

Dataset	Methods	Min	Max	Avg	SD
e-MK01	sGA	46	50	47.8	1.4
	MPGA	45	48	46.1	0.94
	PSO	58	63	58.9	1.51
	VNSGA	44	46	45.1	0.83
	QPSO	49	58	52.4	2.49
	GA-PSO	48	54	50.4	1.68
	DA	49	63	56.4	3.87
	eMPGA	40	46	43	1.48
e-MK02	sGA	41	44	42.4	1.11
	MPGA	35	39	36.6	1.17
	PSO	42	47	44.7	1.48
	VNSGA	38	40	38.5	0.67
	QPSO	39	46	43.4	2.53
	GA-PSO	40	44	41.9	1.24
	DA	45	51	49.4	2.12
	eMPGA	34	37	35.6	0.80
e-MK03	sGA	272	292	282	6.14
	MPGA	276	287	281.6	2.90
	PSO	309	355	335	13.99
	VNSGA	259	270	266.2	3.12
	QPSO	290	336	313.7	13.59
	GA-PSO	265	317	291.5	14.59
	DA	359	388	373.2	10.30
	eMPGA	247	253	250	2.00
e-MK04	sGA	81	91	87.7	2.60
	MPGA	78	96	81	2.60
	PSO	90	98	93.7	2.32
	VNSGA	82	86	84	1.41
	QPSO	90	98	93.3	2.79
	GA-PSO	81	92	87.4	3.29
	DA	99	106	102.8	1.98
	eMPGA	74	78	76.3	1.10
e-MK05	sGA	173	190	182	4.95
	MPGA	173	183	178.3	3.34
	PSO	186	218	194.7	9.10
	VNSGA	171	179	174.3	2.64
	QPSO	186	207	194.6	6.26
	GA-PSO	171	195	180.6	6.84
	DA	207	216	211.5	3.41
	eMPGA	164	172	167.7	2.41
e-MK06	sGA	150	162	158.3	3.83
	MPGA	115	127	118.6	4.05
	PSO	161	180	168.7	6.05
	VNSGA	161	180	131.6	3.72
	QPSO	154	180	164.4	8.22

Table 3 (continued)

Dataset	Methods	Min	Max	Avg	SD
	GA-PSO	138	164	146.3	7.41
	DA	187	199	191.6	4.05
	eMPGA	111	119	116.1	2.80
e-MK07	sGA	150	162	158.3	3.83
	MPGA	115	127	118.6	4.05
	PSO	161	180	168.7	6.05
	VNSGA	161	180	131.6	3.72
	QPSO	154	180	164.4	8.22
	GA-PSO	138	164	146.3	7.41
	DA	187	199	191.6	4.05
	eMPGA	111	119	116.1	2.80
e-MK08	sGA	489	518	506.8	9.04
	MPGA	493	533	513.3	13.07
	PSO	525	576	555.6	15.12
	VNSGA	487	497	493.1	3.47
	QPSO	528	570	543.3	11.77
	GA-PSO	488	533	509.2	11.94
	DA	586	603	592.6	6.15
	eMPGA	479	482	479.3	0.90
e-MK09	sGA	422	471	446.5	17.00
	MPGA	389	435	412.9	14.78
	PSO	466	528	497.2	18.83
	VNSGA	398	417	404.5	5.64
	QPSO	450	504	484	18.37
	GA-PSO	410	470	433.9	20.31
	DA	537	558	549	7.52
	eMPGA	360	379	371.8	5.54
e-MK10	sGA	395	417	405.4	6.96
	MPGA	326	357	345.9	8.23
	PSO	392	457	428	17.50
	VNSGA	350	363	355.1	3.93
	QPSO	412	445	427.9	10.08
	GA-PSO	358	397	377.9	13.36
	DA	467	492	479.4	8.30
	eMPGA	334	347	338.5	3.85

a cloud host based on KVM virtualization, with a configuration of 4vCPUs, 8 GB, and the operating system is 64-bit Windows Server 2019, Standard.

Results

This study incorporated the concept of variable neighborhood search (VNS) into the multi-population genetic algorithm, implementing a secondary local search on the elite individuals during the iterative process, which enhances the algorithm's search capabilities. Additionally, an enhanced initialization method was employed to increase the diversity of the initial population. To mitigate the potential influence of parameters, two different parameter sets were used in comparison with the standard genetic algorithm, and the optimal value from the two was selected for comparison. Each algorithm was run 10 times, with all encoding and decoding methods, other than the ones used for computation, adopted from methods discussed earlier in this study. The average values were taken from different runs.

Extended dataset comparative analysis

In our preliminary analysis, we delved into the extended dataset, which inherently draws from a comprehensive open-source benchmark encompassing a gamut of scenarios pertinent to flexible job-shop scheduling. Our augmentations to this dataset were meticulously aligned with the contours of our research premise.

Subsequent to exhaustive iterative simulations, we have distilled the holistic computational outcomes as presented in Table 2. Table 3 shows the detailed results of multiple iterations, including the maximum (Max), minimum (Min), average values (Avg), and the standard deviation across these iterations (SD). A salient observation that merits attention is the suboptimal performance of some nascent methodologies in specific computational evaluations. The potential underpinnings of such anomalies will be dissected in the final chapter of this discourse.

A perusal of the aggregated results underscores that the methodology we proffer exhibits a palpable edge in a preponderance of scenarios within the extended dataset.

Empirical analysis on real-world production sequences

The results delineated heretofore have been derived from analytical evaluations anchored in our meticulously augmented open-source dataset. To rigorously substantiate the efficacy and robustness of our propounded methodology within the milieu of authentic industrial manufacturing paradigms, we have embarked on a systematic scrutiny of two distinct product manufacturing sequences extant within the selected enterprise's operational spectrum. The aggregated mean outcomes, consequent to multiple iterative simulations, are illustrated in Fig. 13.

The average results indicate that solving the problem directly with DA exhibits relatively poor performance, with a higher average value. The possible reasons for this will be discussed in the last section. The solution quality of the standard particle swarm optimization (PSO) is slightly superior to DA. The multi-population genetic algorithm (MPGA) also shows some improvement, but there remains a gap compared to the enhanced multi-population genetic algorithm (eMPGA) introduced in this study. This paper also explored the integration of VNS into sGA (GA + VNS) and the hybrid of genetic algorithm and particle swarm optimization, but found limited improvement, only slightly better than the standard GA.

The comparison of the results using different algorithms is presented in Table 4. These results suggest that the methods used in this study can effectively solve the scheduling problem of parallel batch machines in a flexible job-shop setting. Compared to other algorithms, the enhanced population genetic algorithm adopted in this research also presents some level of advancement, yielding superior results for this specific problem. Figure 14 depicts the Gantt chart of the scheduling scheme obtained by the enhanced algorithm, where M1–M9 represent different machines, with M9 being the parallel batch machine.

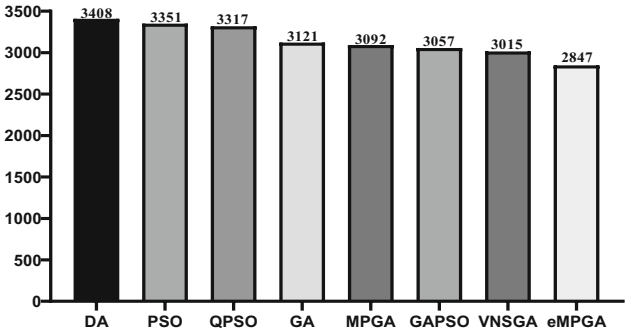


Fig. 13 A comparison of average values for different algorithms

Table 4 Comparison of the results using different algorithms

	Min	Max	Avg	SD
GA	3090	3197	3120.6	29.39
MPGA	3006	3214	3092	68.67
PSO	3250	3454	3351	56.54
VNSGA	2994	3044	3015.3	17.02
QPSO	3234	3464	3317.8	68.19
GA-PSO	2964	3182	3057	53.58
DA	3281	3487	3408.5	67.45
eMPGA	2802	2870	2846.6	21.89

Conclusion

As introduced earlier, many types of production, including semiconductor manufacturing and forging, can be regarded as the flexible job-shop scheduling problem with parallel batch machines. Despite the real-world prevalence of this problem in many enterprises, current research on flexible job shop scheduling with parallel batch-processing machines remains relatively sparse. In addition, the integration of smart technologies into all aspects of life is an important current trend [41].

The flexible job shop scheduling problem with parallel batch machines builds upon the foundation of flexible job shop scheduling problem by incorporating parallel machines. This necessitates attention to not only the allocation of equipment for different processes but also the batching issues of parallel machines, thereby increasing the complexity of problem-solving. This paper focuses on such problems, establishing an optimization model based on the flexible

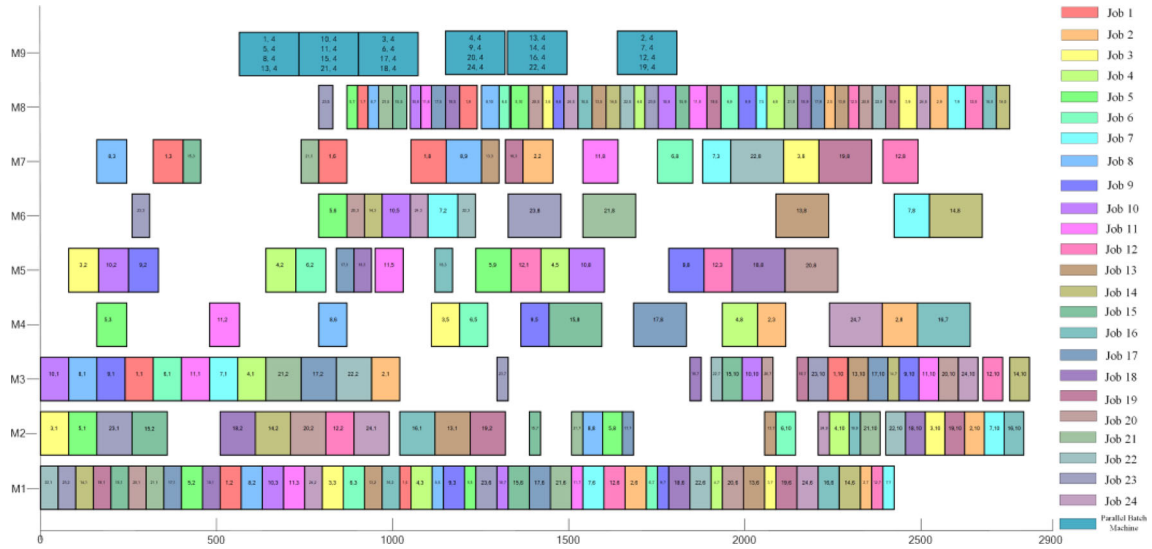


Fig. 14 The Gantt chart of the scheduling scheme via the enhanced algorithm

job shop scheduling problem, to minimize the maximum completion time, otherwise known as MakeSpan. Given the characteristics of these types of problems, we employed a batch assignment method based on the immediate predecessor operation, and designed an enhanced multi-swarm genetic algorithm combined with neighborhood search via initialization and migration modifications for problem-solving. Computational experiments were conducted using actual production cases from cooperative enterprises and an expanded dataset. The results show that the method proposed in this paper can effectively solve the flexible job shop scheduling problem with parallel batch-processing machines, and the enhanced algorithm demonstrates certain improvements compared to the standard algorithm. These findings contribute to enhancing the production efficiency of such enterprises, and the relevant research results are expected to be applied in practical enterprise operations.

In conclusion, there are several interesting phenomena uncovered in this research and many further studies can be carried out in future. Particularly, certain advanced algorithms such as the Dragonfly Algorithm can be intricate in application due to an expansive parameter setup. The added complexity of requiring specific transformations when working within discrete domains might potentially compromise their problem-solving effectiveness. Thus, to optimally apply these algorithms, it is very important to identify the most suitable parameter configurations and transformation techniques in future research. Additionally, there are other facets in this study that deserve comprehensive exploration. For instance, the initialization method used in this research requires a larger initial population for support, which increases the computational load and impacts efficiency. Furthermore, changes in migration methods may sometimes lead to an insufficient exchange of information among initial populations. In future, the transportation time of materials between different equipment could be taken into account in future. Last but not least, expanding the discussion on incompatibilities could enhance the applicability of our developed model.

Acknowledgements This research was funded by the National Natural Science Foundation (NSFC) of China (Grant Nos. 72001096, 72374088, 72101109, and 72001111), the "Belt and Road" Innovative Talents Ex-change Foreign Experts Project of China (Grant No. DL2023014010L), the Research Initiation Fund of Jiangsu University of Science and Technology (1042932005), and the Graduate Practice Innovation Program Project of Jiangsu Province (Grant No. SJCX22_1884).

Data availability The data supporting the findings of this study are available within the article and its supplementary materials.

Declarations

Conflict of interest The authors declare that they have no competing interests, financial or non-financial, that could influence the work reported in this paper.

Appendix A: Linearized transformation

To linearize the model, auxiliary variables Z1, Z2, and Z3 need to adhere to the following constraints:

$$Z_1 = S_{kj} \times Y_{kij} \tag{16}$$

$$Z_2 = E_{ki} \times Y_{kij} \tag{17}$$

$$Z_3 = D_{ki} \times P_{kibn} \tag{18}$$

S.T.

$$Z_1 \leq S_{kj_{max}} Y_{kij} \tag{19}$$

$$Z_1 \geq S_{kj_{min}} Y_{kij} \tag{20}$$

$$Z_1 \leq S_{kj} \tag{21}$$

$$Z_1 \geq S_{kj} - (1 - Y_{kij}) S_{kj_{max}} \tag{22}$$

$$Z_2 \leq E_{ki_{max}} Y_{kij} \tag{23}$$

$$Z_2 \geq S_{kj_{min}} Y_{kij} \tag{24}$$

$$Z_2 \leq E_{ki} \tag{25}$$

$$Z_2 \geq E_{ki} - (1 - Y_{kij}) E_{ki_{max}} \tag{26}$$

$$Z_3 \leq D_{ki_{max}} \times P_{kibm} \tag{27}$$

$$Z_3 \geq 0 \tag{28}$$

Appendix B: Detailed production process

- It should be noted that some processes have been changed for confidentiality purposes.
- PMP means Parallel Batch Machines processing.

Type of Jobs 1:

Operation	Processing time	Optional machine	Demand for PMP capacity
1	80	M2,M3	N/A
2	85	M4,M5,M6,M7	N/A
3	60	M1	N/A
4	160	M9	10 units
5	80	M4,M5,M6,M7	N/A
6	30	M8	N/A
7	32	M1	N/A
8	100	M4,M5,M6,M7	N/A
9	50	M8	N/A
10	56	M2,M3	N/A

Type of Jobs 2:

Operation	Processing time	Optional machine	Demand for PMP capacity
1	50	M1	N/A
2	100	M2,M3	N/A
3	50	M4,M5,M6,M7	N/A
4	160	M9	10 units
5	40	M8	N/A
6	60	M1	N/A
7	32	M3,M2	N/A
8	150	M4,M5,M6,M7	N/A
9	38	M8	N/A
10	56	M3,M2	N/A

Appendix C: Extended MK dataset

*PMP means Parallel Batch Machines processing.

MK01:

Jobs	PMP operation	PMP time	Demand for PMP capacity
1	3	10	10 units
2	N/A	N/A	N/A
3	N/A	N/A	N/A
4	N/A	N/A	N/A
5	4	10	10 units
6	3	10	10 units
7	5	10	10 units
8	6	10	10 units
9	5	10	10 units
10	N/A	N/A	N/A

The rest of the description is based on the table above and the following data format is 'Job-parallel machine Operation':

MK03: 1-7,2-2,3-9,4-7,5-8,8-2,14-8,15-6
MK04: 1-8,2-3,3-3,4-2,5-5,6-8,7-4,8-5,9-10,10-3,11-415-2
MK05: 1-4,2-5,3-5,4-5,5-5,6-3,7-3,8-4,9-5,10-6,11-5,12-5,13-5,14-6,15-4
MK06: 4-4,5-14 MK07:5-5,8-4
MK08: 1-9,2-8,3-12,7-11,9-9,10-9,18-10,20-10
MK09: 1-2,2-5,3-6,4-6,5-6,6-5,7-5,8-5,9-5,10-5,11-5,12-5,13-5,14-9,15-9,16-13,17-9,18-10
MK10: 1-3,2-3,3-9,4-7,5-7,6-7,7-7,8-7,9-7,10-7,11-8,12-2,13-6,14-5,15-5,16-5,17-5,18-5,19-5,20-5

References

1. Johnson SM (1954) Optimal two- and three-stage production schedules with setup times included. Nav Res Logist 1:61–68. https://doi.org/10.1002/nav.3800010110
2. Da Col G, Teppan EC (2022) Industrial-size job shop scheduling with constraint programming. Oper Res Perspect 9:100249. https://doi.org/10.1016/j.orp.2022.100249
3. Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: a framework of strategies, policies, and methods. J Sched 6:39–62
4. Mahmoodjanloo M, Tavakkoli-Moghaddam R, Baboli A, Bozorgi-Amiri A (2020) Flexible job shop scheduling problem with reconfigurable machine tools: an improved differential evolution algorithm. Appl Soft Comput 94:106416. https://doi.org/10.1016/j.asoc.2020.106416
5. Jin L, Zhang C, Wen X et al (2021) A neutrosophic set-based TLBO algorithm for the flexible job-shop scheduling problem with routing flexibility and uncertain processing times. Complex Intell Syst 7:2833–2853. https://doi.org/10.1007/s40747-021-00461-3
6. Dauzère-Pérès S, Ding J, Shen L, Tamssaouet K (2023) The flexible job shop scheduling problem: a review. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2023.05.017
7. Brucker P, Schlie R (1990) Job-shop scheduling with multipurpose machines. Computing
8. Ham A (2017) Flexible job shop scheduling problem for parallel batch processing machine with compatible job families. Appl Math Model 45:551–562
9. Al Aqel G, Li X, Gao L (2019) A modified iterated greedy algorithm for flexible job shop scheduling problem. Chin J Mech Eng 32:21. https://doi.org/10.1186/s10033-019-0337-7
10. Panwalkar SS, Iskander W (1977) A survey of scheduling rules. Oper Res 25:45–61
11. Nouiri M, Bekrar A, Jemai A et al (2018) An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem J. Intell Manuf 29:603–615
12. Gao KZ, Suganthan PN, Chua TJ et al (2015) A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. Expert Syst Appl 42:7652–7663
13. Marichelvam M, Geetha M, Tosun Ö (2020) An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors—a case study. Comput Oper Res 114:104812
14. Nagano MS, Rossi FL, Martarelli NJ (2019) High-performing heuristics to minimize flowtime in no-idle permutation flowshop. Eng Optim 51:185–198
15. Teekeng W, Thammano A (2012) Modified genetic algorithm for flexible job-shop scheduling problems. Proc Comput Sci 12:122–128
16. Fowler JW, Mönch L (2022) A survey of scheduling with parallel batch (p-batch) processing. Eur J Oper Res 298:1–24. https://doi.org/10.1016/j.ejor.2021.06.012
17. Wu X, Guo P, Wang Y, Wang Y (2022) Decomposition approaches for parallel machine scheduling of step-deteriorating jobs to minimize total tardiness and energy consumption. Complex Intell Syst 8:1339–1354. https://doi.org/10.1007/s40747-021-00601-9
18. Jia Z, Yan J, Leung JYT et al (2019) Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. Appl Soft Comput 75:548–561. https://doi.org/10.1016/j.asoc.2018.11.027
19. Zhang H, Li K, Chu C, Jia Z (2022) Parallel batch processing machines scheduling in cloud manufacturing for minimizing total service completion time. Comput Oper Res 146:105899. https://doi.org/10.1016/j.cor.2022.105899
20. Yuan X, Yang Y, Tan W, Yin B (2021) Two-stage collaborative scheduling of casting production line based on hybrid parallel chaotic optimization algorithm J. Hunan Univ (Natural Sciences) 48:161–169. https://doi.org/10.16339/j.cnki.hdxbzkb.2021.10.019
21. Costa A, Cappadonna FA, Fichera S (2014) A novel genetic algorithm for the hybrid flow shop scheduling with parallel batching and eligibility constraints. Int J Adv Manuf Technol 75:833–847
22. Wang I-L, Yang T, Chang Y-B (2012) Scheduling two-stage hybrid flow shops with parallel batch, release time, and machine eligibility constraints. J Intell Manuf 23:2271–2280
23. De La Vega J, Moreno A, Morabito R, Munari P (2023) A robust optimization approach for the unrelated parallel machine scheduling problem. TOP 31:31–66. https://doi.org/10.1007/s11750-021-00621-1
24. Ham AM, Cakici E (2016) Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. Comput Ind Eng 102:160–165. https://doi.org/10.1016/j.cie.2016.11.001
25. Song L, Liu C, Shi H (2022) Discrete particle swarm algorithm with Q-learning for solving flexible job shop scheduling problem with parallel batch processing machine. J Phys Conf Ser 2303:012022. https://doi.org/10.1088/1742-6596/2303/1/012022

26. Ahmadian MM, Khatami M, Salehipour A, Cheng TCE (2021) Four decades of research on the open-shop scheduling problem to minimize the makespan. Eur J Oper Res 295:399–426. https://doi.org/10.1016/j.ejor.2021.03.026
27. Haddadzade M, Razfar MR, Zarandi MHF (2014) Integration of process planning and job shop scheduling with stochastic processing time. Int J Adv Manuf Technol 71:241–252. https://doi.org/10.1007/s00170-013-5469-9
28. Liu T-K, Chen Y-P, Chou J-H (2014) Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer. IEEE Access 2:1598–1606. https://doi.org/10.1109/ACCESS.2015.2388486
29. Introduction to branch and bound—data structures and algorithms tutorial. https://www.geeksforgeeks.org/introduction-to-branch-and-bound-data-structures-and-algorithms-tutorial/. Accessed 29 Mar 2023
30. Deep K, Thakur M (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188:895–911
31. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80:8091–8126. https://doi.org/10.1007/s11042-020-10139-6
32. Deep K, Thakur M (2007) A new mutation operator for real coded genetic algorithms. Appl Math Comput 193:211–230. https://doi.org/10.1016/j.amc.2007.03.046
33. Tian X, Liu X (2021) Improved hybrid heuristic algorithm inspired by tissue-like membrane system to solve job shop scheduling problem. Processes 9:219. https://doi.org/10.3390/pr9020219
34. Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35:3202–3212. https://doi.org/10.1016/j.cor.2007.02.014
35. Zhang C, Rao Y, Li P (2008) An effective hybrid genetic algorithm for the job shop scheduling problem. Int J Adv Manuf Technol 39:965–974. https://doi.org/10.1007/s00170-007-1354-8
36. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41:157–183. https://doi.org/10.1007/BF02023073
37. Liu R, Zhou L, Wang C et al (2020) Research on flexible job-shop scheduling problem with parallel batch processing machines. J Wuhan Univ Technol (Information & Management Engineering) 42:36–43
38. Meraihi Y, Ramdane-Cherif A, Acheli D, Mahseur M (2020) Dragonfly algorithm: a comprehensive review and applications. Neural Comput Appl 32:16625–16646. https://doi.org/10.1007/s00521-020-04866-y
39. Li J (2021) Research and application of scheduling optimization algorithms for fuzzy flexible job-shop. Master's thesis, Jiangnan University
40. Sun L, Lin L, Gen M, Li H (2019) A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling. IEEE Trans Fuzzy Syst 27:1008–1022
41. Hassan MA, Javed R, Farhatullah et al (2023) Intelligent transportation systems in smart city: a systematic survey. In: 2023 international conference on robotics and automation in industry (ICRAI), pp 1–9