Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(3): 4838–4864.
DOI: 10.3934/mbe.2023224
Received: 18 November 2022
Revised: 18 December 2022
Accepted: 21 December 2022
Published: 04 January 2023

Research article

Multi-stage hybrid evolutionary algorithm for multiobjective distributed fuzzy flow-shop scheduling problem

Wenqiang Zhang[1,*], Xiaoxiao Zhang[1], Xinchang Hao[2], Mitsuo Gen[3], Guohui Zhang[4] and Weidong Yang[5]

[1] College of Information Science and Engineering, Henan University of Technology, China

[2] School of Art and Design, Changzhou Institute of Technology, China

[3] Fuzzy Logic Systems Institute, Tokyo University of Science, Japan

[4] School of Management Engineering, Zhengzhou University of Aeronautics, China

[5] Henan Key Laboratory of Grain Photoelectric Detection and Control, Henan University of Technology, China

* **Correspondence:** Email: zhangwq@haut.edu.cn.

Abstract: In the current global cooperative production mode, the distributed fuzzy flow-shop scheduling problem (DFFSP) has attracted much attention because it takes the uncertain factors in the actual flow-shop scheduling problem into account. This paper investigates a multi-stage hybrid evolutionary algorithm with sequence difference-based differential evolution (MSHEA-SDDE) for the minimization of fuzzy completion time and fuzzy total flow time. MSHEA-SDDE balances the convergence and distribution performance of the algorithm at different stages. In the first stage, the hybrid sampling strategy makes the population rapidly converge toward the Pareto front (PF) in multiple directions. In the second stage, the sequence difference-based differential evolution (SDDE) is used to speed up the convergence speed to improve the convergence performance. In the last stage, the evolutional direction of SDDE is changed to guide individuals to search the local area of the PF, thereby further improving the convergence and distribution performance. The results of experiments show that the performance of MSHEA-SDDE is superior to the classical comparison algorithms in terms of solving the DFFSP.

Keywords: hybrid evolutionary algorithms; sequence difference; multi-stage strategies; distributed fuzzy flow-shop scheduling; multiobjective optimization

1. Introduction

In the context of economic globalization, the emergence of global cooperation models and the advancement of science and technology have led to changes in the traditional manufacturing environment. The manufacture of large-scale products is no longer being carried out by local factories alone, but by a variety of different enterprises around the world. These multiple factories with supply and demand relationships form a distributed manufacturing system, and as collaborative manufacturing among companies becomes more common, distributed manufacturing has become a common production mode [1].

The distributed flow-shop scheduling problem (DFSP) is a variant of the flow-shop scheduling problem (FSP). In the case of the DFSP, the jobs to be processed should be allocated to the appropriate shop first. Second, a reasonable sequence of jobs should be carried out in the shops. The objective is to solve the optimal job sequence under certain constraints, so as to achieve efficient resource allocation and utilization [2]. The DFSP considers the realistic constraints of multiple factories. Compared with a single factory, the DFSP is more in line with the current distributed manufacturing environment, and it is a hot topic in the field of production and manufacturing.

In real-life production scheduling, unexpected situations such as sudden machine breakdowns and power interruptions may occur. These uncertainties can affect the accuracy of scheduling and eventually cause time parameters such as production scheduling to become a fuzzy value. So the originally deterministic parameters are unable to accurately describe the actual production scheduling problem. Therefore, it is important to take uncertainty into account in shop scheduling problems. The fuzzy flow-shop scheduling problem (FFSP) combines fuzzy theory with the FSP. Compared with the traditional production scheduling problem, the FFSP is more consistent with the actual production situation and has more practical research significance.

However, with the gradual expansion of production scale and the increasing complexity of the production environment, more and more objectives need to be optimized, and the shop scheduling problem has become a complex multiobjective optimization problem. The multiobjective distributed fuzzy flow-shop scheduling problem (MoDFFSP) considers both the distributed manufacturing environment and the uncertain fuzzy production time under the conditions of multiobjective optimization. Therefore, the research on this problem has important practical significance.

Today, there are more and more methods for dealing with scheduling optimization problems. Traditional mathematical methods, such as enumeration and the branch and bound method, are suitable for small and simple problems due to their high computational complexity. Therefore, heuristics and meta-heuristics (such as the genetic algorithm (GA), simulated annealing, tabu search, particle swarm optimization, differential evolution (DE), etc.) are used to solve complex combinatorial optimization problems [3]. Compared with traditional methods, meta-heuristics affords great improvements in solution time and solution quality, and it can be used to deal with shop scheduling problems.

GAs constitute a typical type of evolutionary algorithm and a heuristic intelligent algorithm proposed earlier. The algorithm has good robustness and an adaptive random search mechanism, making it suitable for solving various combinatorial optimization problems in reality. At the same time, it does not require continuous or differentiable optimization objects, and it is suitable for combinatorial optimization problems in non-convex spaces. GAs have been widely used to solve production scheduling problems. However, the disadvantage is that when dealing with more complex optimization problems, the GA will have insufficient convergence and local search capabilities to obtain high-quality and diverse solutions.

DE represents an algorithm that updates individuals based on the differences between individuals. It also has good robustness, a simple structure, easy implementation and strong global convergence ability. DE does not need to use the characteristic information of the problem, and it is suitable for solving some optimization problems in complex environments that cannot be solved by using conventional mathematical programming methods. It can effectively deal with the DFFSP. However, the disadvantage of DE is that it readily falls into the local optimum value of the objective function, especially when the dimension of the problem increases; the effect of DE is not satisfactory.

Each classical algorithm has its unique advantages; inevitably there are some disadvantages, and no single algorithm can solve all problems [4]. At present, a variety of different evolutionary algorithms are complementary and being combined to form a new hybrid evolutionary algorithm, which combines the advantages of different algorithms and complements the defects of a single algorithm, thereby enhancing the performance of the algorithm and better handling the multiobjective scheduling optimization problem [5].

In reality, the most complex problems are composed of many constraints and multiple conflicting objectives. When dealing with such problems, these objectives need to be considered at the same time, and as good as possible to achieve the best effect under certain conditions. However, although most of the existing algorithms consider convergence and distribution performance simultaneously in the optimization process, in the actual algorithm execution process, the individual evolutional direction in the solution space is different in different periods of the population during the evolution process. There are few adjustment strategies for different stages, which cannot accurately regulate different stages. Therefore, it is necessary to propose a multi-stage hybrid evolutionary algorithm with sequence difference-based differential evolution (MSHEA-SDDE).

The main contributions of this study can be summarized as follows:

1) The hybrid sampling strategy is used as the global search strategy, and the sequence difference-based differential evolution (SDDE) is used as the local search strategy. The combination of these two strategies balances the convergence and distribution performance of the algorithm at different stages.

2) The enhanced SDDE strategy is introduced to guide the evolution of poor individuals to better individuals, taking into account the different evolutional directions of the individuals in the solution space at different stages in the evolutionary process.

3) The algorithm is divided into three stages according to different periods of iteration, and different SDDE strategies are used in different stages. In the first stage, the hybrid sampling strategy makes the population rapidly converge toward the Pareto front (PF) in multiple directions. In the second stage, SDDE_1 is designed to accelerate the convergence and distribution speed of the algorithm. In the last stage, SDDE_2 is employed to guide individuals to search the local area of the PF, thereby further improving the convergence and distribution performance.

4) The triangular fuzzy number (TFN) represents the fuzzy processing time of the job, a method of bivector representation of job and factory is designed for the DFFSP for encoding and decoding and genetic operators are also designed.

The proposed MSHEA-SDDE is compared with a non-dominated sorting genetic algorithm (NSGA-II) [6], an improved-strength Pareto evolutionary algorithm (SPEA2) [7], a multiobjective evolutionary algorithm based on decomposition (MOEA/D) [8], a fast multiobjective hybrid evolutionary algorithm (MOHEA) [9] and a hybrid multiobjective evolutionary algorithm with differential evolution (HMOEA-DE) [10]. The results show that MSHEA-SDDE not only has excellent convergence performance, but can also guarantee that the obtained solutions have good distribution performance and diversity.

The remaining sections of this paper are organized as follows. Section 2 reviews the relevant research work on the MoDFFSP in recent years. Section 3 presents the mathematical model of the MoDFFSP. A detailed description of the proposed MSHEA-SDDE and the main strategies are shown in Section 4. Experimental results and analysis are given in Section 5. Section 6 provides a summary of the full paper as a conclusion, along with the future work.

2. Related work

In recent years, due to various uncertainties in real-world manufacturing systems, fuzzy theory has been widely used in fuzzy scheduling to describe uncertainty and a great deal of research has been done on the FSP; however, to the best of our knowledge, research on the DFFSP is still in its infancy, although it has received considerable attention from both academics and practitioners. Next, a review of closely related work is carried out.

As the real production environment becomes more complex and uncertain, there are increasing studies on the FFSP using fuzzy theory. In order to deal with a special structured n-job of three machine flow-shop scheduling with piecewise quadratic fuzzy processing times. Khalifa et al. developed a fuzzy approach with the help of a heuristic algorithm to minimize the rental cost of machines given the specified rental cost [11].

A trapezoidal membership function is used to represent the processing times, and an exact algorithm is proposed to achieve that minimization of the total waiting time of jobs [12]. Vinoba and Selvamalar presented triangular, trapezoidal and octagonal fuzzy numbers to solve the FFSP, and a branch and bound algorithm was adopted, which is modified to a fuzzy scenario with the objective of minimizing the makespan [13]. Li et al. considered the assembly DFFSP; an imperialist competitive algorithm with empire cooperation was developed to minimize the fuzzy makespan [14]. The above are all about the single-objective FFSP. In reality, multiple objectives need to be considered simultaneously, but the literature on solving the MoDFFSP is very limited. However, it is very common as a variant of the DFSP. Moreover, compared with the non-distributed environment, the research on distributed shop scheduling problems has great application value in the field of production.

Xi and Lei applied the integration of reinforcement learning and meta-heuristics to solve the two-stage distributed hybrid flow-shop scheduling problem (DHFSP) with fuzzy processing time [15]. They constructed a novel Q-learning-based teaching-learning based optimization to minimize the makespan. Li et al. proposed an improved brainstorming optimization algorithm for the type-two fuzzy distributed hybrid flow-shop problem (FDHFSP) with bivariate volume representations [16]. Cai and Lei studied the energy-efficient DHFSP under fuzzy processing time and proposed a collaborative restructuring frog-hopping algorithm with the objective of simultaneously optimizing the fuzzy maximum completion time, total consistency indicator and fuzzy total energy consumption [17]. Wang and Li studied the FDHFSP and proposed a shuffled frog-leaping algorithm with the collaboration of multiple search strategies to optimize multiple objectives simultaneously [18]. Cai et al. designed a collaborative variable search to optimize the total agreement index and fuzzy makespan simultaneously; seven neighborhood structures and two global search operators were used in two cooperated variable search parts to generate high quality solutions [19].

For the distributed permutation flow-shop scheduling problem (DPFSP), Ying et al. investigated the no-idle DPFSP with the objective of minimizing the makespan, they also presented an iterated reference greedy algorithm for effectively solving this problem [20]. Baysal et al. proposed an artificial bee colony (ABC) algorithm for the multiobjective distributed fuzzy permutation flow-shop scheduling problem (DFPFSP) [21]. Lin et al. proposed a backtracking search hyper-heuristic algorithm to solve the assembly DPFSP, and the backtracking search algorithm was employed as the high-level strategy to manipulate the low-level heuristics to operate in the solution space [22]. Baysal et al. developed an ABC algorithm to solve the multiobjective DFPFSP, where the process times and due dates of the jobs are considered as triangular and trapezoidal fuzzy numbers for the DFPFSP [23]. Wang et al. presented a fuzzy logic-based hybrid estimation of distribution algorithm (FL-HEDA) to address the DPFSP under the condition of machine breakdown with the makespan criterion [24]. The FL-HEDA hybridizes the probabilistic model of estimation of distribution algorithm with crossover and mutation operators of a GA to produce new offspring.

Recent years, more variants of the DFSP have been proposed in the literature by combining some constraints or multiobjectivity. In consideration of the uncertainty of manufacturing systems, Shao et al. investigated a distributed fuzzy blocking flow-shop scheduling problem in which there are multiple homogeneous factories and each one is set as a flow-shop with no intermediate buffers between any consecutive machines [25]. Aiming at the energy-aware distributed flow-shop with flexible assembly scheduling problem, a cooperative memetic algorithm with feedback was proposed by Wang and Wang to minimize the total tardiness and energy consumption [26]. Lu et al. studied an energy-efficient DPFSP with heterogeneous factories with makespan and total energy consumption as the objectives, including a hybrid multiobjective optimization algorithm, which integrates the iterated greedy; an efficient local search is designed to solve it [27].

As an important energy-efficient scheduling problem, the energy-efficient DFSP plays an essential part in green manufacturing. In recent years, it has attracted much attention. The energy consumption and carbon-efficient characteristics were considered by Zhao et al. [28, 29]. They presented a hyper-heuristic algorithm with Q-learning to address the energy efficiency of the distributed blocking flow-shop scheduling problem (DBFSP). Moreover, a Pareto-based discrete Jaya algorithm was proposed to solve the carbon-efficient DBFSP with the criteria of total tardiness and total carbon emission. Pan et al. aimed to solve the distributed energy-efficient parallel machine scheduling problem by integrating factory assignment and machine assignment into an extended machine assignment to handle the coupled relations of subproblems [30]. Zhao et al. investigated an energy-efficient distributed no-wait flow-shop scheduling problem with sequence-dependent setup time for the minimization of makespan and total energy consumption and a cooperative meta-heuristic algorithm based on Q-learning was proposed to address it [31].

Different from the single-objective problem, multi-objective scheduling problems have high complexity. From the above literature, it can be seen that more and more scholars are using hybrid intelligent algorithms to deal with multiobjective optimization in different fields. Due to the shortcomings of a single algorithm, hybrid intelligent algorithms can combine the advantages of different algorithms to complement the defects of a single algorithm. Therefore it can better deal with more complex multiobjective optimization problems. Lu et al. investigated a DHFSP, a Pareto-based multi-objective hybrid iterated greedy algorithm was designed by integrating the merits of genetic operators and iterated greedy heuristics [32]. Considering the distributed manufacturing problem in an uncertain environment, Zheng et al. proposed a co-evolutionary algorithm with a specific problem strategy for the multiobjective fuzzy distributed hybrid assembly line scheduling problem with the objectives of optimizing fuzzy total tardiness and robustness through the use of a reasonable combination of estimation distribution algorithm and iterative greedy search [33]. Zhang et al. proposed a multiobjective particle swarm optimization with directional search (MoPSO-DS) to solve the DFPFSP [34]. MoPSO-DS divides the particle swarm into three subgroups, and three subgroups are biased in different regions of the PF. Then, particles are updated in the direction of the partiality.

Although the research on the above scheme uses a hybrid algorithm to improve the convergence and distribution performance of the algorithm, in the actual algorithm execution process, the evolutional direction of the individual in the solution space is different at different stages of the population evolutionary process. There are few adjustment strategies, and it is impossible to accurately control different stages. Moreover, the evolutionary algorithm has been proved to be one of the algorithms that can effectively solve multiobjective optimization problems. However, the performance research and applications of multiobjective evolutionary algorithms in uncertain environments are less. Therefore, this work combines a GA with a DE algorithm and a MSHEA-SDDE is proposed to study the MoDFFSP.

3. Problem formulation

3.1. Operations on fuzzy number

Among the fuzzy numbers, the TFN is a method of converting uncertain fuzzy linguistic variables into definite values. In the production scheduling problem, researchers mostly use the TFN to represent the fuzzy processing time of the job. That is, $\tilde{t} = (t_1, t_2, t_3)$, where t_1, t_2 and t_3 represent the optimal ideal processing time of the job, the processing time without accident and the processing time in the worst case respectively. The fuzzy processing time represented by the TFN is shown in Figure 1.

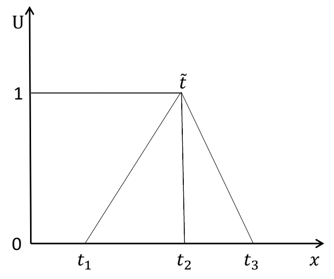

Figure 1. Processing time represented by the TFN.

The function definition formula of the TFN is as follows:

$$U(x) = \begin{cases} \frac{x-t_1}{t_2-t_1} & x \in [t_1, t_2] \\ \frac{t_3-x}{t_3-t_2} & x \in [t_2, t_3] \\ 0 & x < t_1, x > t_3 \end{cases} \tag{3.1}$$

In order to deal with the fuzzy processing time, arithmetic operations of fuzzy numbers are used, including addition, subtraction, maximum and ranking operations. In the study, the fuzzy maximum and ranking operations are used to determine the starting time and sequence of each job in the process, and the fuzzy addition and subtraction operations are used to determine the completion time of the jobs in the process.

For two TFNs $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$, the addition, subtraction, ranking and maximum operations are shown as follows [35].

1) Addition and subtraction operations

$A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$

$A - B = (a_1 - b_1, a_2 - b_2, a_3 - b_3)$

2) Ranking operation

The following criterions are adopted to rank two TFNs [36].

Step 1: Compare $F_1(A) = (a_1, 2a_2, a_3)/4$ [37]; if $F_1(A) > F_1(B)$, then $A > B$.

Step 2: If the F_1 value of A and B are the same, compare $F_2(A) = a_2$; if $F_2(A) > F_2(B)$, then $A > B$.

Step 3: If the F_1 and F_2 values of A and B are the same, compare $F_3(A) = a_3 - a_1$; if $F_3(A) > F_3(B)$, then $A > B$.

3) Maximum operation

If $A > B$, then $A \vee B = A$; otherwise $A \vee B = B$ [38].

3.2. Problem description

In the MoDFFSP, the objective is to select a suitable factory for the job and find a reasonable order of the job in each factory, which minimizes the fuzzy maximum completion time of the jobs in all factories and the fuzzy maximum total flow time of all jobs in all factories simultaneously. In this problem, each job can be assigned to any factory, and once the distribution of jobs between factories is determined, no jobs can be transferred to another factory. The notations of the MoDFFSP are defined as follows:

Indices

i: index of job, $(i = 1, 2, \ldots, n)$;

j: index of machine, $(j = 1, 2, \ldots, m)$;

k: index of factory, $(k = 1, 2, \ldots, f)$.

Parameters

n: the total number of jobs;

m: the total number of machines in each factory;

f: the total number of factories;

o_k: the total number of jobs allocated to the kth factory;

π^k: the job sequences in factory k, $\pi^k = \{\pi^k_1, \pi^k_2, \ldots, \pi^k_i, \ldots, \pi^k_{o_k}\}$;

$\tilde{t}^{\mathrm{P}}_{i,j,k}$: fuzzy processing time of job i on machine j in factory k.

Decision Variables

$\tilde{t}^{C}_{i,j,k}$: fuzzy completion time of job i on machine j in factory k;

Mathematical Model

$$\min \quad \tilde{C}_{max} = \max\{\tilde{t}^{C}_{\pi^k_{o_k},m,k}\}, k = 1, 2, \ldots, f \tag{3.2}$$

$$\min \quad FT = \max\{\textstyle\sum_{i=1}^{o_k} \tilde{t}^{C}_{i,m,k}\}, k = 1, 2, \ldots, f \tag{3.3}$$

$$\text{s.t.} \quad \tilde{t}^{C}_{\pi^k_1,1,k} = \tilde{t}^{P}_{\pi^k_1,1,k}, k = 1, 2, \ldots, f \tag{3.4}$$

$$\tilde{t}^{C}_{\pi^k_{i+1},1,k} = \tilde{t}^{C}_{\pi^k_i,1,k} + \tilde{t}^{P}_{\pi^k_{i+1},1,k}, \quad i = 1, 2, \ldots, n-1; k = 1, 2, \ldots, f \tag{3.5}$$

$$\tilde{t}^{C}_{\pi^k_1,j+1,k} = \tilde{t}^{C}_{\pi^k_1,j,k} + \tilde{t}^{P}_{\pi^k_1,j+1,k}, \quad j = 1, 2, \ldots, m-1; k = 1, 2, \ldots, f \tag{3.6}$$

$$\tilde{t}^{C}_{\pi^k_i,j,k} = \max\{\tilde{t}^{C}_{\pi^k_i,j-1,k}, \tilde{t}^{C}_{\pi^k_{i-1},j,k}\} + \tilde{t}^{P}_{\pi^k_i,j,k}, \quad i = 2, 3, \ldots, n; j = 2, 3, \ldots, m; k = 1, 2, \ldots, f \tag{3.7}$$

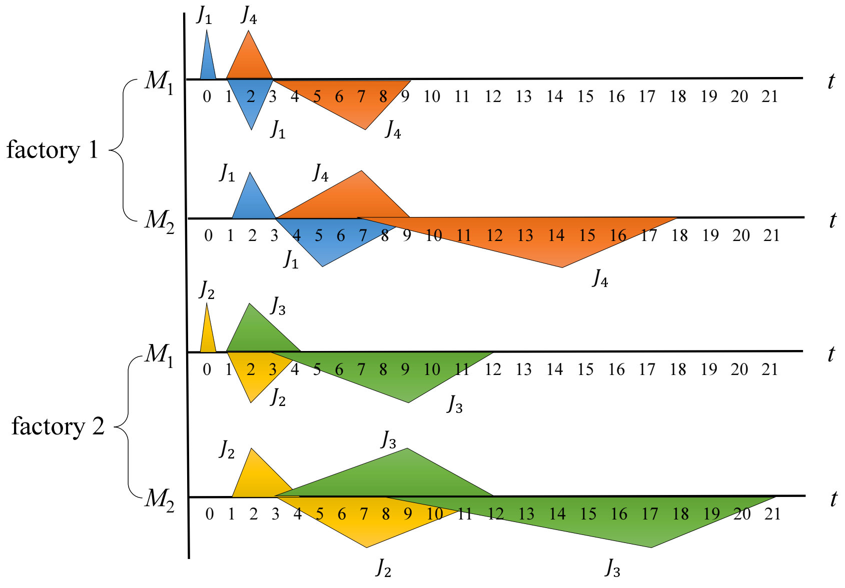

Figure 2. Gantt chart of example problem.

Two objective functions are used to evaluate the solution, where one is the minimization of the fuzzy maximum completion time of the jobs in all factories, which can be calculated by using Eq (3.2), and the other is the minimization of the fuzzy maximum total flow time of all jobs in all factories, which can be calculated by using Eq (3.3). Equation (3.4) expresses the initial value of calculating the fuzzy time. The first step of each job is executed on machine 1, so the fuzzy completion time of the jobs on the first machine within each factory can be calculated by using Eq (3.5). Since job 1 is the first job to be operated, it is the first to be processed on each machine, so the fuzzy completion time of job 1 on each machine can be calculated by using Eq (3.6). Equation (3.7) is used to calculate the fuzzy completion time for each of the remaining jobs for each process. Figure 2 shows the Gantt chart of the DFFSP, taking two factories and four jobs and two machines as an example. The sequence of jobs is (j_1, j_2, j_3, j_4), and the corresponding factories are (f_1, f_2, f_2, f_1). After the sequence of jobs is determined, the processing time arrangement mechanism of the jobs on the machine is different from the traditional precision time processing arrangement due to the use of the improved discrete sorting coding method.

Job 1: Machine 1: (1, 2, 3), Machine 2: (2, 3, 6)
Job 2: Machine 1: (1, 2, 4), Machine 2: (2, 5, 8)
Job 3: Machine 1: (2, 7, 8), Machine 2: (5, 8, 9)
Job 4: Machine 1: (2, 5, 6), Machine 2: (4, 7, 9)

Through the calculation of the above formula, the values of the two objectives of the DFFSP can be obtained. The fuzzy completion time of the jobs of factory 1 is (7, 14, 18), and that of factory 2 is (8, 17, 21); the longest fuzzy completion time of the jobs among all factories is used as the final fuzzy completion time, so the fuzzy completion time of the jobs of the DFFSP is (8, 17, 21). In addition, the fuzzy total flow time of the DFFSP is determined by the fuzzy completion time of all jobs in the factory. The total fuzzy flow time for factory 1 is (10, 19, 27), and that for factory 2 is (11, 24, 33); the fuzzy total flow time is the longest and final total flow time in all factories, so the total fuzzy flow time for the DFFSP is (11, 24, 33).

4. Proposed method

4.1. Overview of the MSHEA-SDDE

The proposed MSHEA-SDDE consists mainly of a hybrid evolutionary algorithm (HEA) with a hybrid sampling strategy and an SDDE local search strategy. The hybrid sampling strategy is divided into an edge region sampling strategy based on the vector-evaluated GA (VEGA) [39] and a central region sampling strategy based on the Pareto-dominating and dominated relationship-based fitness function (PDDR-FF) [40]. Among them, the edge region sampling strategy can select the individuals near to the edge region of the PF so as to better hold the individuals with good performance on the single objective. The central region sampling strategy uses the PDDR-FF to evaluate the dominance of individuals in the population, and thus selects individuals with a larger dominance area near to the central region of the PF. The combination of the two sampling strategies can help the MSHEA-SDDE to search for solutions in multiple regions of the PF, as well as improve the convergence and distribution performance of the algorithm. The multi-stage SDDE local search strategy is to guide individuals to search local areas directionally by comparing sequence differences between individuals according to different periods of algorithm iteration so as to improve the local search ability of the MSHEA-SDDE.

During the execution of the MSHEA-SDDE, the evolutional directions of individuals in the solution space in different periods of the evolution of the population are different. In the early stages of the evolutionary process, the evolutional direction of individuals is mainly toward the true PF that is, when the optimization objective of the MSHEA-SDDE is to improve the convergence performance. In the middle and late stages of the evolutionary process, the individuals evolve toward further convergence to the true PF, while also exploring local regions of the PF. The idea of many hybrid algorithms is to enhance local search capabilities, but it is often easy to overlook the trade-off between convergence performance and distribution performance in the first and middle stages of the MSHEA-SDDE. Therefore, the proposed MSHEA-SDDE focuses on applying SDDE strategies with different evolutional directions to the MSHEA-SDDE in stages. At the same time, through different design strategies in the three stages, the convergence and distribution performance in the evolution of the MSHEA-SDDE are correspondingly improved, thereby improving the overall performance of the MSHEA-SDDE. The framework of MSHEA-SDDE is shown in Figure 3.

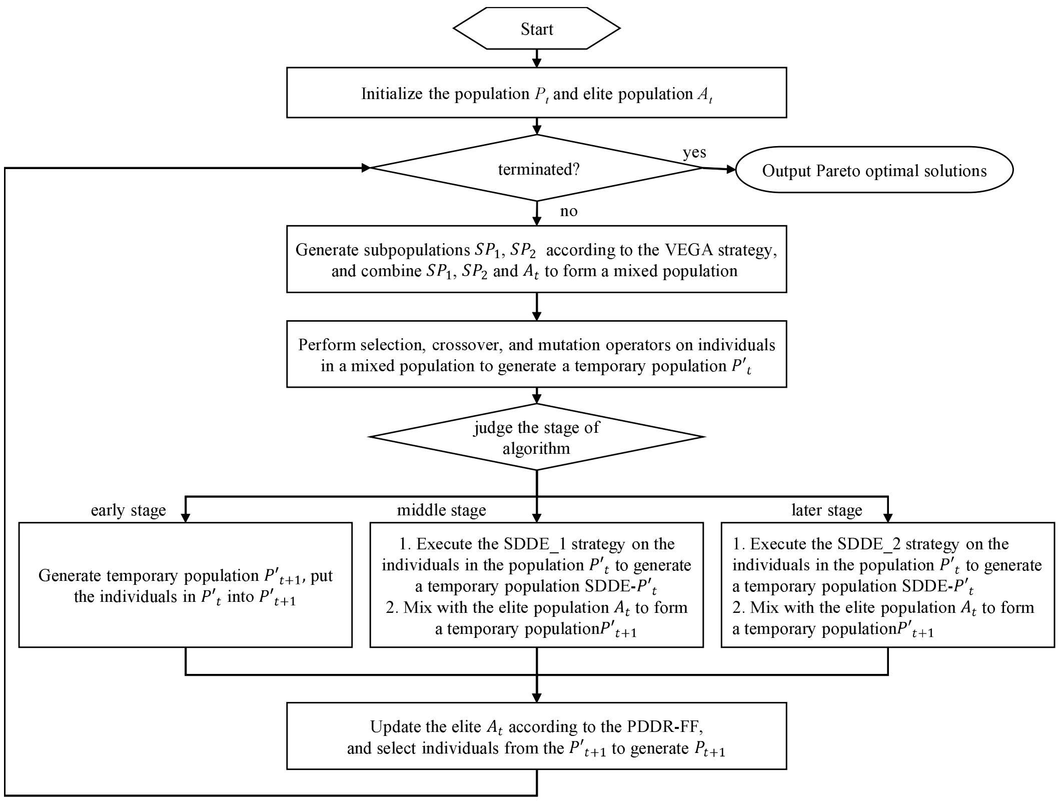
Start
Initialize the population P_t and elite population A_t
terminated?
yes
Output Pareto optimal solutions
no
Generate subpopulations SP_1, SP_2 according to the VEGA strategy, and combine SP_1, SP_2 and A_t to form a mixed population
Perform selection, crossover, and mutation operators on individuals in a mixed population to generate a temporary population P'_t
judge the stage of algorithm
early stage
middle stage
later stage
Generate temporary population P'_{t+1}, put the individuals in P'_t into P'_{t+1}
1. Execute the SDDE_1 strategy on the individuals in the population P'_t to generate a temporary population SDDE-P'_t 2. Mix with the elite population A_t to form a temporary populationP'_{t+1}
1. Execute the SDDE_2 strategy on the individuals in the population P'_t to generate a temporary population SDDE-P'_t 2. Mix with the elite population A_t to form a temporary populationP'_{t+1}
Update the elite A_t according to the PDDR-FF, and select individuals from the P'_{t+1} to generate P_{t+1}

Figure 3. Framework of MSHEA-SDDE.

4.2. Encoding and decoding

In this paper, the fuzzy processing time due to the job is represented by the TFN. Therefore, an improved discrete sorting encoding method is adopted in the experiment. The number of genes on the chromosome represents the sequence number of the job to be processed, while the real number representing the precise processing time on the gene is adjusted to a TFN represented by an array. Besides, in order to cope with the DFFSP in the multi-factory situation, we adopt a bivector to represent each individual in the problem space. Therefore, the encoding process consists of two parts, where the first vector represents the sequence of the jobs, and each gene in this vector is the job number and does not repeat; the sequence of the job in the solution can be obtained through the gene sequence on this vector [41]. The second part of the vector represents the factory sequence, and each gene in this vector corresponds to the factory number but can be repeated.

Figure 4 shows an example of the encoding for an individual with a chromosome divided into two parts: job sequence and the factory sequence. The first {5} on the job sequence {5, 4, 3, 2, 8, 6, 7, 1} represents job 5, while the second {4} represents job 4. In the factory sequence {1, 2, 3, 3, 2, 1, 1, 2}, the first {1} represents the factory number corresponding to job 1, the second {2} represents the factory number corresponding to job 2 and so on. Each job corresponds to a factory number, each job will be assigned to the corresponding factory for processing and each factory has an assigned sequence of jobs. This encoding method can intuitively express each job and its corresponding factory.

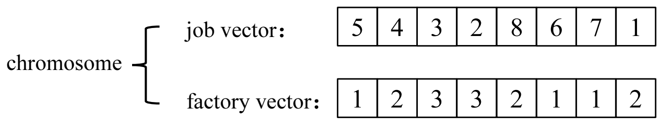

Figure 4. Example of bivector encoding.

In the decoding process, the jobs are distributed to different factories for processing according to their factory sequence numbers. In decoding, both parts of the chromosome are traversed from left to right. First, the corresponding job of factory 1 is selected and put together as the internal distribution of the job of factory 1, and without changing the front and rear positions of the jobs, the preserved job order is the operation order of the internal job of factory 1. Then, operate factory 2 according to the steps of factory 1, pick out the parts of factory 2 and put them together without changing the order, complete the distribution of the parts of factory 2 and the sorting of the internal parts of factory 2, and so on. This encoding and decoding method can solve the problem of allocating jobs to factories, as well as the sorting problem of the job inside of the factory in the DFFSP. Figure 5 shows an example of decoding with eight jobs and three factories.

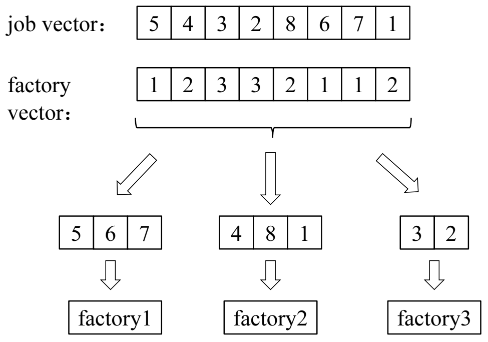

Figure 5. Example of bivector decoding.

4.3. Genetic operators

For the selection operator, a binary tournament selection is used as the selection operator. In the edge selection strategy, binary tournament selection is also used to generate sub-populations. Two individuals are randomly selected in the population and the function value of one of the optimization objectives in Eqs (4.2) and (4.3) is compared; then, the individual with better performance on the objective is selected and placed in the corresponding sub-population. In the central selection strategy, individuals are ranked according to their fitness values and the best performers are selected as the elite population. Using the binary tournament selection as the selection operator allows individuals with high fitness values to be retained under the condition of ensuring the diversity of solutions.

For the crossover operator, since this work adopts the encoding method of bivectors, different crossover operations are carried out for the two parts of a chromosome. First, for the job vector, the order crossover (OX) method is used. While ensuring the sufficient diversity of individuals in the population, the OX method also pays attention to the partially connected gene segments in the chromosome; thus, the gene sequences of some excellent chromosomes can be better retained in the offspring individuals. As for the factory vector, the effect of sequential crossover is not obvious because the factory sequence number can be repeated, so the single-point crossover is adopted in this part. The single point crossing method randomly generates an integer in the job sequence, and the random number is the mark position of the single point crossing. Figure 6 shows an example of a process that uses the crossover operator for two parent individuals.

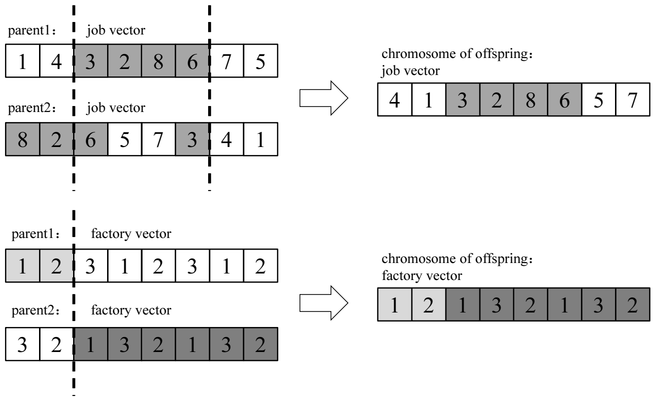

Figure 6. Example of a crossover operator.

As shown in Figure 6, in the job vector part, the parent chromosome 1 is {1, 4, 3, 2, 8, 6, 7, 5}, and the parent chromosome 2 is {8, 2, 6, 5, 7, 3, 4, 1}; there are two mark points in the chromosome. Mark points 1 and 2 on parent chromosome 1 with the gene fragments {3, 2, 8, 6}, which are retained in the same order and position on the offspring chromosome, and are not selected in parent chromosome 2 to avoid illegal chromosome solution. Then, in parent chromosome 2, unselected genes are put into the offspring chromosome in sequence after the gene locus of mark point 2. After the last gene of the parent chromosome is put into the offspring, the selection continues from the first gene of the parent chromosome until all alleles are selected.

In the factory vector part, the factory vector of the parent chromosome 1 is {1, 2, 3, 1, 2, 3, 1, 2} and the factory vector of the parent chromosome 2 is {3, 2, 1, 3, 2, 1, 3, 2}. Because of the single-point crossover method, only one marked position is set in the factory vector part. For example, the marked position in Figure 6 is 2. Then combine the genes on the 0 and 1 loci in the parent 1 factory vector with the genes from 2 to the maximum length of the chromosome in the parent 2 factory vector to generate the factory vector of the offspring chromosome {1, 2, 1, 3, 2, 1, 3, 2}.

In terms of mutation operators, two different mutation operators are also used. For the job vector, the exchange mutation method is adopted; the specific operation is to randomly select two gene positions on the vector and then exchange the gene information within the two positions. As for the factory vector, because the factory sequence number can be repeated, the effect of exchange variation is not obvious, so the single-point mutation method is adopted for this part. In this method, a gene locus is randomly selected in the factory sequence and the factory sequence number expressed on this gene locus is randomly changed to other factory sequence numbers. Figure 7 shows an example of a process that uses the mutation operator on an individual.

As shown in Figure 7, in the part of job vector, the parent chromosome is {1, 4, 3, 2, 8, 6, 7, 5}, and the two gene locus 2 and 5 are randomly selected. The exchanged genes are {3} and {6}, and the offspring chromosome after exchange is {1, 4, 6, 2, 8, 3, 7, 5}. In the factory vector, the factory vector of the parent chromosome is {1, 2, 3, 1, 2, 3, 1, 2}; the single point mutation is to randomly selects a gene locus; for example, the factory number on gene locus 2 is 3, and the gene on this locus is randomly changed to another factory number 1. Thus, the generated offspring chromosome factory vector is {1, 2, 1, 1, 2, 3, 1, 2}.

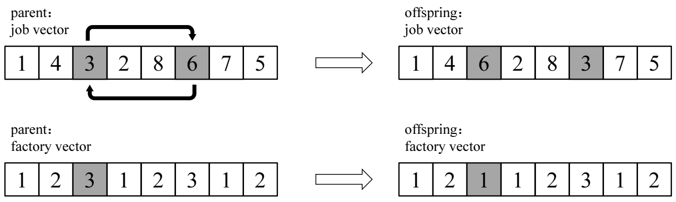

Figure 7. Example of mutation operator.

4.4. Mixed selection strategy

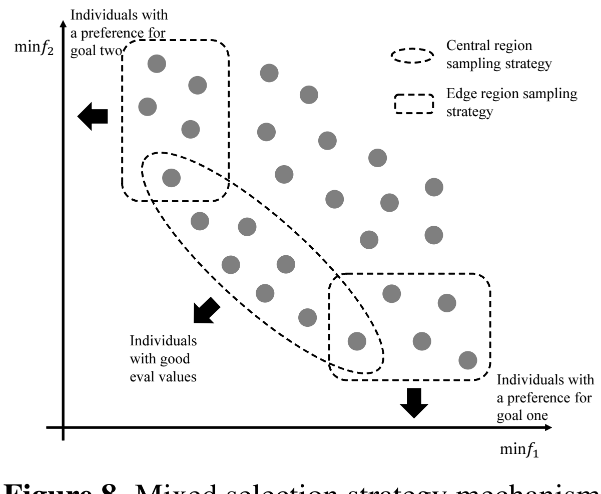

Figure 8. Mixed selection strategy mechanism.

In the proposed MSHEA-SDDE, the HEA uses a hybrid selection strategy as the global search strategy, which combines the edge selection strategy based on the VEGA and the center selection strategy based on the PDDR-FF. Among them, the edge selection strategy can select the individuals whose edge regions close to the PF are biased toward the two objectives so as to better save the individuals with better performance on the single objective. The center selection strategy uses the PDDR-FF to evaluate the dominant area of individuals in the solution space so as to select individuals with a large dominant area near to the center region of the PF. The following formula is used to calculate the PDDR-FF value of individual X_i.

$$\text{eval}(X_i) = q(X_i) + \frac{1}{p(X_i)+1}, i = 1, 2, \ldots, \ popSize \tag{4.1}$$

where $q(X_i)$ is the number of individuals dominating individual X_i, $p(X_i)$ is the number of individuals dominated by individual X_i and $popSize$ is the population size. It can be seen from the formula that when the *eval* value of an individual is smaller, it indicates that this individual has a large dominated area, the individual performs better and the *eval* value of non-dominated individuals is less than 1. Therefore, after obtaining the *eval* value of individuals and ranking it; the individuals with a small *eval* value are selected from small to large, which can filter out non-dominated individuals and individuals with a large dominated area.

The hybrid selection strategy formed by the combination of the two selection strategies can help the MSHEA-SDDE to search for solutions in multiple regions of the PF, as well as to improve the convergence and distribution performance of the MSHEA-SDDE. Figure 8 illustrates the main mechanism of the hybrid selection strategy.

4.5. Multi-stage SDDE strategy

To further improve the quality of the solution, a multi-stage SDDE strategy is investigated in this paper. SDDE is a local search strategy based on sequence differences between individuals in the population. The strategy searches the population generated by the hybrid strategy in the iterative process of the MSHEA-SDDE, which is used to strengthen the search ability of individuals in their surrounding areas so as to further improve the quality of the solution set. The multi-stage SDDE includes SDDE_1 and SDDE_2. According to the different stages of the MSHEA-SDDE evolutionary process, the SDDE strategy is combined with the HEA to improve the convergence and distribution performance.

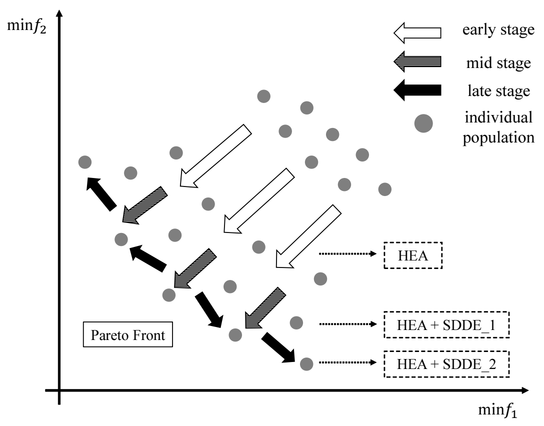

Figure 9. Process of MSHEA SDDE algorithm.

When the hybrid strategy in the HEA is used to evolve the population, the population can quickly converge to the PF in the early iteration of the MSHEA-SDDE. In the middle iteration of the MSHEA-SDDE, the population obtained by the hybrid strategy converges to the PF in the solution space. At this time, the SDDE_1 local search strategy can guide the population to further converge to the true PF. In the later iteration of the MSHEA-SDDE, the population has converged to near the true PF; at this time, the evolutional direction of the individual in the SDDE strategy is adjusted, and SDDE_1 is adjusted to SDDE_2 so that the MSHEA-SDDE guides the individual to explore the optimal solution in the single objective direction. Thus, the diversity of final solutions is improved and the distribution performance of the MSHEA-SDDE is improved.The algorithm process of the MSHEA-SDDE is shown in Figure 9.

The main process of SDDE_1 is divided into the following three steps.

Step 1: Three individuals S_1, S_2, S_3 are randomly selected from the population according to the PDDR-FF value, so that *eval* $(S_1) <$ *eval* $(S_2) <$ *eval* (S_3).

Step 2: An exchange sequence method is used to obtain the sequence differences between the better performing individuals S_1 and S_2.

Step 3: Obtain the sequence difference between S_1 and S_2 according to a certain scale ratio, and then act on the poorly performing individual S_3.

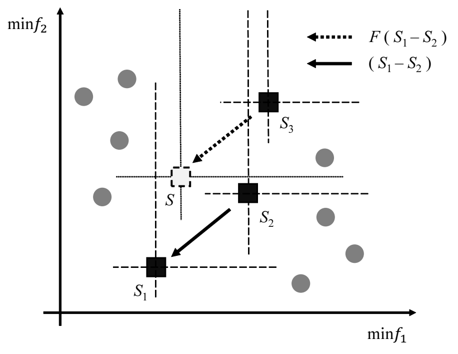

Figure 10. SDDE_1 main mechanisms.

Figure 10 shows the main mechanism of SDDE_1, and Eq (4.2) is used to represent the main process of SDDE_1.

$$\mathrm{S} = S_3 + F(S_1 - S_2) \tag{4.2}$$

where $(S_1 - S_2)$ represents the sequence difference between individuals S_1 and S_2, F denotes the scale ratio of individual sequence difference and S represents the new individual obtained by the SDDE strategy for individual S_3. The main process of the SDDE_1 strategy is to randomly select three different individuals in the population, use an exchange sequence to detect the difference between the two individuals with better performance and take a certain proportion of the difference to act on the individual with worse performance. The SDDE_1 strategy can guide the population to converge better to the PF and improve the convergence performance of the MSHEA-SDDE.

In the later iteration stage of the MSHEA-SDDE, the population has converged to the PF by using the hybrid sampling strategy and SDDE_1 local search strategy, and the individuals in the population are distributed in multiple regions of the PF. However, using the SDDE_1 strategy makes the individuals in the population evolve in the direction of a small PDDR-FF value, which may lead to an insufficient solution search for the edge and local areas of the PF. For this situation, we propose the SDDE_2 strategy, which adjusts the selection and evolutional direction of individuals in SDDE_1, in order to improve the search ability of the MSHEA-SDDE for the PF edge and local region. The operation process of SDDE_2 is as follows.

Step 1: According to the PDDR-FF value, three individuals S'_1, S'_2, S'_3 with a PDDR-FF value less than 1 are randomly selected from the population so that *eval* $(S'_1) <$ *eval* $(S'_2) <$ *eval* $(S'_3) < 1$.

Step 2: The exchange sequence method is used to obtain the sequence difference between the better performing individuals S'_1 and S'_2.

Step 3: Obtain the sequence difference between S'_1 and S'_2 according to a certain scale ratio, and then act on the individual S'_3 with a large PDDR-FF value.

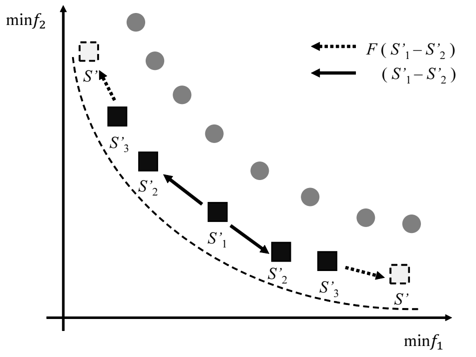

Figure 11. SDDE_2 mechanism.

Figure 11 shows the main mechanism of SDDE_2, and Eq (4.3) is used to represent the main process of SDDE_2.

$$S' = S'_3 + F(S'_1 - S'_2) \tag{4.3}$$

where $(S'_1 - S'_2)$ still represents the sequence difference between individuals S'_1 and S'_2, F represents the scale ratio of individual sequence difference, and S' represents the new individual obtained by individual S'_3 through the SDDE strategy. However, the difference between the SDDE_2 strategy and SDDE_1 strategy is that the PDDR-FF values of the three individuals selected in the first step of the SDDE_2 strategy are all less than 1. These three individuals are non-dominated solutions located on the PF. It also guides individuals with small PDDR-FF values to evolve in the direction of large PDDR-FF values. This adjustment method can make the selected individuals search the edge and local areas of the PF without reducing the convergence performance of the MSHEA-SDDE, allowing the MSHEA-SDDE to find the optimal solution in the two single objective directions in the non-dominated solution and improve the distribution of the final solution. The SDDE_2 strategy should be used in the late iteration of the MSHEA-SDDE. Using this strategy too early will lead to the elite individuals having smaller PDDR-FF values near the central region of the population during the iteration process, which may make the MSHEA-SDDE fall into the local optimal value. In the later stage of the MSHEA-SDDE, the SDDE_1 strategy is adjusted to the SDDE_2 strategy, which can improve the distribution performance while ensuring the convergence performance of the MSHEA-SDDE.

5. Experimental results

In this section, in order to verify the performance of the proposed MSHEA-SDDE, the experiments compare the proposed algorithm with NSGA-II [6], SPEA2 [7], MOEA/D [8], MOHEA [9] and HMOEA-DE [10]. In these comparison algorithms, for the algorithm with an elite population, the non-dominated solution in the elite population is taken as the final solution of the algorithm, and for the algorithm without an elite population, the non-dominated solution in the last generation solution set is taken as the final solution of the algorithm. The maximum number of iterations for all algorithms is set to 600, the population size is 100 and each algorithm is run 30 times on each problem to take the average for comparison. In order to evaluate the performance of the algorithm, we selected six multiobjective algorithm evaluation indicators such as generational distance (*GD*), inverted generational distance (*IGD*), hypervolume (*HV*), spacing (*SP*), spread (*Spread*) and coverage (*C*) to evaluate and compare the performance of the algorithm. If the significance result is "-", it means that the comparison algorithm is significantly weaker than the main algorithm; if the result is "*", it indicates that the difference is not significant; if the result is "+", it means that the comparison algorithm is better than the main algorithm. All experiments were run on the same machine in the same experimental environment; the machine was configured with an Intel Core i7-9750H CPU, 16 GB memory and a Windows 10 operating system. The experiment was implemented in JAVA language. The experimental results are presented in a table, where the bold font indicates the best performing value in that part.

In terms of data sets of the DFFSP, this experiment uses an improved data set based on the Taillard benchmark data set. Since standard time is adopted in the standard data set and the characteristics of fuzzy processing time are not considered, the improved method is to randomly adjust the standard time to a fuzzy time within a small range to meet the fuzzy processing time of the job requirement [42]. The specific adjustment operation refers to the idea in the literature [43]; according to the Eq (5.1), the standard time in the FSP data set is extended to the fuzzy time.

$$\begin{cases} \tilde{t_{ij}} = \left(round\left(\mathrm{t}^1_{\mathrm{i}j}\right), t_{ij}, round\left(\mathrm{t}^2_{\mathrm{i}j}\right)\right) \\ t^1_{\mathrm{i}j} \in \left[0.85t_{ij}, 0.94t_{ij}\right] \\ t^2_{\mathrm{i}j} \in \left[1.1t_{ij}, 1.19t_{ij}\right] \end{cases} \tag{5.1}$$

In the above formula, t_{ij} is the standard time in the data set of the FSP and $\tilde{t_{ij}}$ is the expanded fuzzy time used in this paper. Based on the fuzzy data set, we added additional factory parameters to form the DFFSP. Twelve different scales of data were used for testing, namely, five machines, 10 machines and 20 machines, corresponding to 20 jobs, 50 jobs, 100 jobs, 200 jobs and 500 jobs. Using data of different scales can verify the optimization effect of the algorithm, ranging from simple to complex processing situations.

5.1. Parameter settings

Different combinations of the crossover rate Cr and mutation rate Mr in MSHEA_SDDE and scale ratio F in SDDE were tested. Due to excessive experimental data, only the most valuable interval is shown here, and the ranges are as follows: $Cr \in \{0.6, 0.7, 0.8, 0.9\}$, $Mr \in \{0.1, 0.2, 0.3\}$, $F \in \{0.5, 0.6, 0.7, 0.8\}$. The performance of MSHEA_SDDE on the problem of three factories, 20 jobs and five machines was selected for the comparison test. The test results are shown in Table 1.

Table 1. Comparison of MSHEA_SDDE on 20_5 problems with different parameters.

Cr	*Mr*	*F*	*GD*	*IGD*	*HV*	*SP*	*Spread*
0.6	0.1	0.5	8.13E-02	7.03E-02	1.80E+05	3.89E+01	8.89E-01
		0.6	6.82E-02	5.66E-02	2.00E+05	2.83E+01	8.62E-01
		0.7	7.08E-02	6.29E-02	1.88E+05	3.15E+01	8.62E-01
		0.8	6.90E-02	6.28E-02	1.88E+05	3.19E+01	8.62E-01
	0.2	0.5	7.11E-02	6.35E-02	1.89E+05	4.65E+01	8.91E-01
		0.6	6.80E-02	6.15E-02	1.92E+05	3.70E+01	8.86E-01
		0.7	6.64E-02	5.77E-02	1.96E+05	3.60E+01	8.87E-01
		0.8	6.09E-02	5.12E-02	2.13E+05	4.22E+01	8.65E-01
	0.3	0.5	6.07E-02	5.31E-02	2.05E+05	4.09E+01	8.78E-01
		0.6	7.52E-02	6.29E-02	1.88E+05	3.97E+01	8.90E-01
		0.7	6.51E-02	5.84E-02	1.96E+05	3.41E+01	8.76E-01
		0.8	6.28E-02	5.26E-02	2.07E+05	3.34E+01	8.48E-01
0.7	0.1	0.5	7.81E-02	6.76E-02	1.86E+05	4.57E+01	8.79E-01
		0.6	7.70E-02	6.97E-02	1.80E+05	3.75E+01	9.02E-01
		0.7	7.31E-02	6.52E-02	1.88E+05	3.34E+01	8.86E-01
		0.8	5.98E-02	5.54E-02	1.99E+05	3.42E+01	8.63E-01
	0.2	0.5	7.35E-02	6.24E-02	1.86E+05	3.33E+01	8.74E-01
		0.6	6.39E-02	5.51E-02	2.01E+05	3.91E+01	8.77E-01
		0.7	6.98E-02	5.71E-02	2.00E+05	3.43E+01	8.64E-01
		0.8	6.76E-02	5.43E-02	2.06E+05	3.33E+01	8.54E-01
	0.3	0.5	6.19E-02	5.03E-02	2.12E+05	3.44E+01	8.76E-01
		0.6	6.12E-02	5.07E-02	2.13E+05	4.85E+01	8.99E-01
		0.7	6.19E-02	4.99E-02	2.12E+05	3.08E+01	8.62E-01
		0.8	5.90E-02	5.15E-02	2.13E+05	5.17E+01	8.68E-01
0.8	0.1	0.5	8.03E-02	7.24E-02	1.75E+05	4.74E+01	8.96E-01
		0.6	7.71E-02	7.15E-02	1.78E+05	5.55E+01	9.24E-01
		0.7	6.79E-02	6.04E-02	1.91E+05	5.35E+01	9.00E-01
		0.8	7.65E-02	6.53E-02	1.86E+05	4.88E+01	8.93E-01
	0.2	0.5	6.65E-02	6.24E-02	1.93E+05	3.48E+01	8.66E-01
		0.6	6.86E-02	6.23E-02	1.88E+05	3.21E+01	8.81E-01
		0.7	6.41E-02	5.74E-02	1.99E+05	5.84E+01	9.01E-01
		0.8	**4.89E-02**	**4.31E-02**	2.21E+05	2.96E+01	8.66E-01
	0.3	0.5	7.18E-02	6.50E-02	1.86E+05	3.97E+01	8.82E-01
		0.6	5.75E-02	5.09E-02	2.13E+05	2.88E+01	8.56E-01
		0.7	5.51E-02	4.78E-02	2.19E+05	3.78E+01	8.57E-01
		0.8	5.57E-02	4.43E-02	**2.25E+05**	4.18E+01	**8.31E-01**
0.9	0.1	0.5	7.56E-02	6.94E-02	1.78E+05	4.31E+01	9.20E-01
		0.6	7.42E-02	6.72E-02	1.81E+05	3.14E+01	9.03E-01
		0.7	6.69E-02	5.97E-02	1.95E+05	3.00E+01	8.79E-01
		0.8	6.66E-02	5.59E-02	2.00E+05	5.68E+01	8.84E-01
	0.2	0.5	5.66E-02	5.06E-02	2.10E+05	**2.54E+01**	8.76E-01
		0.6	6.67E-02	5.87E-02	1.96E+05	3.59E+01	8.97E-01
		0.7	6.39E-02	5.58E-02	2.02E+05	4.12E+01	8.71E-01
		0.8	5.31E-02	4.65E-02	2.21E+05	4.29E+01	8.64E-01
	0.3	0.5	5.95E-02	5.46E-02	2.05E+05	3.50E+01	8.92E-01
		0.6	6.65E-02	5.46E-02	2.07E+05	3.94E+01	8.64E-01
		0.7	5.24E-02	4.50E-02	2.23E+05	4.73E+01	8.81E-01
		0.8	5.32E-02	4.44E-02	2.22E+05	3.52E+01	8.64E-01

From the experimental results in Table 1, it can be seen that the algorithm performs better when the crossover rate is 0.8 and the scale ratio F is 0.8. When the mutation rate is 0.2, the combination achieves two first, and when the mutation rate is 0.3, the combination achieves two first and one second.

Therefore, the parameter combinations of crossover rate $Cr = 0.8$, mutation rate $Mr = 0.3$ and SDDE scale ratio $F = 0.8$ were used in the subsequent experiments.

The algorithm parameter settings are shown in Table 2. All algorithms set the same population size and use the same maximum number of iterations, and the crossover rate and mutation rate are set to a fixed value. Among them, compared with MSHEA-SDDE, HMOEA-DE does not consider using the DE strategy at different times at different stages.

Table 2. Parameter settings.

	MSHEA-SDDE	HMOEA-DE	MOHEA	NSGA-II	SPEA2	MOEA/D
population size	100	100	100	100	100	100
elite population size	50	50	50	\	50	\
subpopulation size	50	50	50	\	\	\
crossover rate	0.8	0.8	0.8	0.8	0.8	0.8
mutation rate	0.3	0.3	0.3	0.3	0.3	0.3
scale ratio *F*	0.8	0.8	\	\	\	\
max generation	600	600	600	600	600	600

5.2. *Effectiveness of strategies*

Two key strategies SDDE_1 and SDDE_2 in MSHEA-SDDE were introduced in the previous section. It is worth mentioning that in the early stage of the evolutional process, although there are individuals with various fitness values in the population, most of the individuals with poor performance may be caused by too few iterations. Therefore, the SDDE_1 strategy adopted from the beginning of the iteration may affect the diversity of individuals in the population and reduce the distribution performance. Similarly, if the SDDE_2 strategy is applied in the early or middle iteration of the MSHEA-SDDE, the elite individuals with good PDDR-FF values in the central region of the population may evolve toward the local region solution earlier. As a result, the excellent individuals in the central region are destroyed, which affects the convergence performance of the entire algorithm.

According to the results in the parameter settings section, in order to prove these ideas and verify the appropriate use timing of SDDE_1 and SDDE_2 in the MSHEA-SDDE, this part of the experiment involved 12 groups in the MSHEA-SDDE with different timing combinations of the two strategies in 12 fuzzy distributed flow factories of different scales. The test on the scheduling problem used the five indicators of *GD*, *IGD*, *HV*, *SP* and *Spread* to verify the convergence and distribution performance of the MSHEA-SDDE. Due to the large amount of experimental data and it being difficult to fully display, we recorded the statistics of the top three rankings of different combinations on 12 problems in the table, and each indicator had 36 results on 12 problems. As well as the statistics of the number of times that different combinations ranked first on 12 problems, each indicator had 12 results on 12 problems. The specific statistical results are shown in Tables 3 and 4.

As shown in Tables 3 and 4, the usage timings of SDDE_1 and SDDE_2 are separated by "/", where "x%E/y%E" means that SDDE_1 was used when MSHEA-SDDE ran to x%, and the MSHEA-SDDE used SDDE_2 when running to y%. For example, "15%E/66%E" means that SDDE_1 was added after the MSHEA-SDDE algorithm iterated to 15%, and SDDE_1 was adjusted to SDDE_2 after the MSHEA-SDDE algorithm iterated to 66%. From the statistical results in Table 3, the combination result of "15%E/90%E" ranks the top three times the most for the five indicators, and the performance of the three indicators *GD*, *IGD* and *HV* is more obvious. In Table 4, the combination of "15%E/90%E" shows less obvious advantages, but still outperformed other combinations in terms of *GD*, *IGD*, *HV* and *SP*, which is basically consistent with our previous study idea. Therefore, adding the SDDE_2 strategy at an appropriate time in the late iteration of the MSHEA-SDDE can not only ensure the convergence performance of the MSHEA-SDDE, but it can also improve the distribution performance. In the subsequent experiments, the timing of "15%E/90%E" was used in MSHEA-SDDE for comparative experiments.

Table 3. Statistics of SDDE combination use timing (comparison times of the top three).

SDDE(1st,2nd,3rd)	*GD*	*IGD*	*HV*	*SP*	*Spread*
15%E/66%E	3	2	3	1	2
15%E/75%E	4	4	3	5	3
15%E/83%E	1	2	1	2	**6**
15%E/90%E	**9**	**9**	**10**	**6**	2
25%E/66%E	4	3	3	5	3
25%E/75%E	0	1	0	1	2
25%E/83%E	2	1	1	1	1
25%E/90%E	1	2	2	3	1
33%E/66%E	3	3	3	3	5
33%E/75%E	3	2	2	4	3
33%E/83%E	2	3	2	1	4
33%E/90%E	4	4	6	4	4

Table 4. Statistics of SDDE combination use timing (comparison times of the first one).

SDDE(1st)	*GD*	*IGD*	*HV*	*SP*	*Spread*
15%E/66%E	1	0	1	1	0
15%E/75%E	1	1	1	1	1
15%E/83%E	0	0	0	1	0
15%E/90%E	**3**	**4**	3	**5**	1
25%E/66%E	1	3	3	1	2
25%E/75%E	0	0	0	0	2
25%E/83%E	1	0	0	0	0
25%E/90%E	0	0	0	2	0
33%E/66%E	2	1	1	1	1
33%E/75%E	0	0	0	0	1
33%E/83%E	1	1	1	0	2
33%E/90%E	2	2	2	0	2

Other components of the proposed strategy are the exchange sequence strategy and the hybrid sampling strategy, whose effectiveness has been verified in papers [5, 44] and will no longer be verified here.

5.3. Results and discussion

This section analyzes algorithm performance through various indexes. Tables 5 to 10 show the comparative test results of all algorithms on the six evaluation indicators. The significance analysis results obtained by the Wilcoxon rank sum test are shown in the H.T. column of the tables.

Table 5 shows the results of the *GD* indicator, which reflects the convergence performance of the algorithm. From the experimental results, MSHEA-SDDE performed best on 10 test questions and second on two test questions. In the Wilcoxon rank sum test results, except for HMOEA-DE, MSHEA-SDDE had a significant difference with all other comparative algorithms. It can be proved that MSHEA-SDDE has better convergence performance, and that it is also better than other comparison algorithms.

The *IGD* indicator can not only be used to evaluate the convergence performance of the solution, but it can also be used to judge the difference between the distribution of the solution and the distribution of the solution on the PF. The smaller the value of *IGD*, the better the convergence performance and distribution performance of the algorithm. From the *IGD* indicator results shown in Table 6 and the Wilcoxon rank sum test results, MSHEA-SDDE also performed best again on 10 questions, and performed second best on two questions. The experimental results prove that MSHEA-SDDE not only has good convergence performance, but it also has good distribution performance.

Table 5. Average results and significance analysis for *GD* indicator.

GD	MSHEA-SDDE	HMOEA-DE		MOHEA		NSGA-II		SPEA2		MOEA/D	
	Mean	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.
20_5	**3.21E-02**	3.29E-02	*	1.06E-01	-	1.07E-01	-	1.06E-01	-	1.05E-01	-
20_10	**1.81E-02**	2.26E-02	-	2.81E-02	-	2.90E-02	-	2.85E-02	-	3.17E-02	-
20_20	**2.03E-02**	2.20E-02	*	3.09E-02	-	2.95E-02	-	3.10E-02	-	3.85E-02	-
50_5	**3.42E-02**	3.64E-02	*	1.97E-01	-	1.92E-01	-	1.91E-01	-	1.87E-01	-
50_10	3.59E-02	**3.21E-02**	*	5.30E-02	-	5.22E-02	-	4.63E-02	-	5.08E-02	-
50_20	**3.41E-02**	3.70E-02	*	1.05E-01	-	1.02E-01	-	9.99E-02	-	1.04E-01	-
100_5	**4.65E-02**	5.27E-02	*	1.40E-01	-	1.38E-01	-	1.36E-01	-	1.38E-01	-
100_10	**3.20E-02**	3.68E-02	*	1.06E-01	-	1.02E-01	-	1.04E-01	-	1.12E-01	-
100_20	5.14E-02	**5.03E-02**	*	5.41E-02	*	5.47E-02	*	5.15E-02	*	5.83E-02	-
200_10	**3.36E-02**	3.50E-02	*	6.09E-02	-	5.99E-02	-	5.91E-02	-	6.34E-02	-
200_20	**2.05E-02**	2.34E-02	*	9.38E-02	-	9.37E-02	-	9.04E-02	-	9.50E-02	-
500_20	**7.87E-03**	8.77E-03	*	8.82E-03	-	8.16E-03	*	8.02E-03	*	9.91E-03	-

Table 6. Average results and significance analysis for *IGD* indicator.

IGD	MSHEA-SDDE	HMOEA-DE		MOHEA		NSGA-II		SPEA2		MOEA/D	
	Mean	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.
20_5	**4.07E-02**	4.30E-02	*	1.14E-01	-	1.14E-01	-	1.15E-01	-	1.19E-01	-
20_10	**2.25E-02**	2.73E-02	-	2.94E-02	-	2.93E-02	-	2.88E-02	-	3.68E-02	-
20_20	**2.04E-02**	2.21E-02	*	2.85E-02	-	2.79E-02	-	2.83E-02	-	3.88E-02	-
50_5	**4.07E-02**	4.21E-02	*	2.10E-01	-	2.06E-01	-	2.02E-01	-	2.07E-01	-
50_10	3.66E-02	**3.48E-02**	*	5.28E-02	-	5.12E-02	-	4.56E-02	-	5.49E-02	-
50_20	**3.30E-02**	3.63E-02	*	1.03E-01	-	9.86E-02	-	9.83E-02	-	1.04E-01	-
100_5	**4.44E-02**	5.11E-02	*	1.37E-01	-	1.34E-01	-	1.32E-01	-	1.37E-01	-
100_10	**2.80E-02**	3.03E-02	*	1.04E-01	-	1.01E-01	-	1.01E-01	-	1.08E-01	-
100_20	4.22E-02	**4.21E-02**	*	4.62E-02	*	4.55E-02	*	4.22E-02	*	4.87E-02	-
200_10	**2.88E-02**	3.13E-02	*	5.55E-02	-	5.38E-02	-	5.20E-02	-	5.58E-02	-
200_20	**1.85E-02**	2.07E-02	*	8.94E-02	-	8.95E-02	-	8.80E-02	-	9.10E-02	-
500_20	**7.35E-03**	7.42E-03	*	7.96E-03	*	7.89E-03	*	7.84E-03	*	9.75E-03	-

Table 7 shows the results for the *HV* indicator. The *HV* indicator can be used to evaluate the quality of the algorithm according to the size of the solution space. The larger the indicator value, the larger the dominance area of the solution obtained by the algorithm. From the experimental results, MSHEA-SDDE performed best on 10 questions. It can be seen that MSHEA-SDDE had better convergence and distribution performance than the other comparison algorithms.

Table 7. Average results and significance analysis for *HV* indicator.

HV	MSHEA-SDDE	HMOEA-DE		MOHEA		NSGA-II		SPEA2		MOEA/D	
	Mean	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.
20_5	**2.70E+05**	2.65E+05	*	1.28E+05	-	1.27E+05	-	1.27E+05	-	1.17E+05	-
20_10	**1.58E+05**	1.44E+05	-	1.37E+05	-	1.39E+05	-	1.36E+05	-	1.15E+05	-
20_20	**3.88E+05**	3.68E+05	*	3.07E+05	-	3.19E+05	-	3.16E+05	-	2.67E+05	-
50_5	**4.83E+06**	4.70E+06	*	1.50E+06	-	1.56E+06	-	1.62E+06	-	1.49E+06	-
50_10	1.13E+06	**1.18E+06**	*	9.06E+05	-	9.28E+05	-	1.02E+06	-	8.67E+05	-
50_20	**4.03E+06**	3.87E+06	*	1.52E+06	-	1.60E+06	-	1.68E+06	-	1.44E+06	-
100_5	**1.01E+07**	9.61E+06	*	3.55E+06	-	3.75E+06	-	3.83E+06	-	3.48E+06	-
100_10	**1.04E+07**	1.01E+07	*	3.64E+06	-	3.88E+06	-	3.86E+06	-	3.44E+06	-
100_20	4.02E+06	3.97E+06	*	3.71E+06	*	3.85E+06	*	**4.07E+06**	+	**3.42E+06**	-
200_10	**2.22E+07**	2.16E+07	*	1.38E+07	-	1.41E+07	-	1.45E+07	-	1.29E+07	-
200_20	**4.23E+07**	4.18E+07	*	9.98E+06	-	1.00E+07	-	1.06E+07	-	9.33E+06	-
500_20	**6.63E+07**	6.52E+07	*	6.42E+07	-	6.50E+07	*	6.62E+07	*	5.99E+07	-

Table 8. Average results and significance analysis for *SP* indicator.

SP	MSHEA-SDDE	HMOEA-DE		MOHEA		NSGA-II		SPEA2		MOEA/D	
	Mean	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.
20_5	4.34E+01	**3.30E+01**	*	5.27E+01	*	5.26E+01	-	5.20E+01	-	4.28E+01	*
20_10	**4.05E+01**	4.54E+01	*	5.64E+01	*	7.19E+01	-	6.91E+01	*	6.97E+01	-
20_20	1.18E+02	**8.37E+01**	*	9.01E+01	*	1.03E+02	*	1.33E+02	*	1.20E+02	*
50_5	2.09E+02	**1.41E+02**	*	3.71E+02	-	2.67E+02	*	3.07E+02	-	4.31E+02	-
50_10	**1.91E+02**	2.56E+02	*	2.50E+02	*	3.53E+02	-	2.59E+02	*	3.24E+02	*
50_20	3.06E+02	**2.94E+02**	*	3.62E+02	*	4.36E+02	*	3.68E+02	*	3.64E+02	*
100_5	8.90E+02	**5.88E+02**	+	7.80E+02	*	7.35E+02	*	8.98E+02	*	8.82E+02	*
100_10	7.11E+02	**6.03E+02**	*	8.12E+02	*	9.37E+02	*	9.27E+02	*	1.16E+03	*
100_20	1.01E+03	9.16E+02	*	1.08E+03	*	9.15E+02	*	**7.68E+02**	*	1.20E+03	*
200_10	1.99E+03	2.29E+03	*	**1.86E+03**	*	2.28E+03	*	2.55E+03	*	2.29E+03	*
200_20	2.22E+03	2.44E+03	*	1.86E+03	*	2.33E+03	*	**1.80E+03**	*	2.27E+03	*
500_20	1.00E+04	6.44E+03	+	8.54E+03	*	8.97E+03	*	7.21E+03	*	**5.29E+03**	+

Table 9. Average results and significance analysis for *Spread* indicator.

Spread	MSHEA-SDDE	HMOEA-DE		MOHEA		NSGA-II		SPEA2		MOEA/D	
	Mean	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.	Mean	H.T.
20_5	**8.53E-01**	8.59E-01	*	9.47E-01	-	9.44E-01	-	9.51E-01	-	9.52E-01	-
20_10	**8.19E-01**	8.82E-01	*	8.79E-01	*	8.69E-01	*	9.18E-01	-	9.00E-01	-
20_20	8.62E-01	**8.55E-01**	*	9.03E-01	*	8.74E-01	*	9.03E-01	*	9.06E-01	*
50_5	**8.71E-01**	9.20E-01	*	9.58E-01	-	9.49E-01	-	9.57E-01	-	9.75E-01	-
50_10	**8.45E-01**	9.02E-01	*	8.79E-01	*	9.30E-01	-	8.78E-01	*	8.79E-01	*
50_20	**8.92E-01**	9.38E-01	*	9.42E-01	*	9.62E-01	-	9.38E-01	*	9.58E-01	*
100_5	**8.61E-01**	8.72E-01	*	9.43E-01	-	9.49E-01	-	9.53E-01	-	9.65E-01	-
100_10	**8.10E-01**	8.53E-01	*	9.35E-01	-	9.51E-01	-	9.52E-01	-	9.62E-01	-
100_20	9.10E-01	9.06E-01	*	9.07E-01	*	9.22E-01	*	**9.05E-01**	*	9.05E-01	*
200_10	**8.67E-01**	8.78E-01	*	9.27E-01	-	9.31E-01	-	9.35E-01	-	9.59E-01	-
200_20	8.36E-01	**7.82E-01**	*	9.49E-01	-	9.71E-01	-	9.48E-01	-	9.59E-01	-
500_20	8.02E-01	7.90E-01	*	8.17E-01	*	8.19E-01	*	**7.53E-01**	*	8.53E-01	*

In terms of the *SP* and *Spread* indicators in Tables 8 and 9, MSHEA-SDDE performed well on the *Spread* indicator and had the best performance on eight questoins, but had average performance on the *SP* indicator. It is because the DFFSP is based on the actual problem with a variety of constraints and the number of final solutions obtained by the algorithm is small. As a result, the continuity performance of the individuals in the final solution set is not very prominent, which makes the performance of the *SP* indicator unsatisfactory and affects the *SP* indicator. The *Spread* indicator performed better, which indicates that the final solution set obtained by the algorithm is more widely distributed. In general, MSHEA-SDDE has better distribution performance than the other comparison algorithms.

In addition, the Wilcoxon rank sum test results of MSHEA-SDDE and MOHEA-DE were mostly insignificant. From the experimental results of the *C* indicator in Table 10, MSHEA-SDDE outperformed HMOEA-DE on nine questions, and almost outperformed other comparative algorithms on all problems. It indicates that the solutions obtained by MSHEA-SDDE are better than HMOEA-DE, and that the MSHEA-SDDE has the best convergence performance.

Table 10. Average results and significance analysis for *C* indicator.

C	A: MSHEA-SDDE B: HMOEA-DE D: MOHEA E: NSGA-II F: SPEA2 G: MOEA/D									
	C(A, B), *C*(B, A)		*C*(A, D), *C*(D, A)		*C*(A, E), *C*(E, A)		*C*(A, F), *C*(F, A)		*C*(A, G), *C*(G, A)	
20_5	**2.67E-01**	1.00E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
20_10	**2.33E-01**	1.33E-01	**3.67E-01**	3.33E-02	**3.33E-01**	0.00E+00	**3.33E-01**	3.33E-02	**5.67E-01**	3.33E-02
20_20	**3.67E-01**	1.00E-01	**5.00E-01**	0.00E+00	**4.67E-01**	3.33E-02	**5.33E-01**	3.33E-02	**7.33E-01**	0.00E+00
50_5	**2.33E-01**	1.67E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
50_10	2.00E-01	**2.67E-01**	**5.33E-01**	6.67E-02	**4.67E-01**	1.00E-01	**2.67E-01**	6.67E-02	**4.33E-01**	0.00E+00
50_20	**3.67E-01**	2.33E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**9.67E-01**	0.00E+00
100_5	**3.00E-01**	1.33E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
100_10	**3.33E-01**	2.00E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
100_20	2.00E-01	**2.33E-01**	**1.67E-01**	6.67E-02	**1.67E-01**	6.67E-02	1.33E-01	1.33E-01	**2.00E-01**	6.67E-02
200_10	2.67E-01	**3.00E-01**	**8.67E-01**	0.00E+00	**9.00E-01**	0.00E+00	**9.00E-01**	0.00E+00	**1.00E+00**	0.00E+00
200_20	**2.00E-01**	1.67E-01	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00	**1.00E+00**	0.00E+00
500_20	**3.33E-02**	0.00E+00	**1.33E-01**	0.00E+00	**1.33E-01**	6.67E-02	**1.00E-01**	3.33E-02	**1.67E-01**	0.00E+00

From the experimental results of the *C* indicator, it can be further seen that the solution obtained by MSHEA-SDDE is closer to the real PF. Therefore, the model of the MoDFFSP proposed in this paper is effective, and the proposed MSHEA-SDDE has better convergence and distribution performance than other comparative algorithms.

6. Conclusions

In this paper, MSHEA-SDDE has been proposed to solve the MoDFFSP. This algorithm consists of a global search strategy based on a hybrid selection strategy and a local search strategy based on multi-stage sequence difference DE. In the hybrid sampling strategy, the population is divided into multiple sub-populations according to different objectives so that the population evolves to multiple regions of the PF. The multi-stage sequence difference DE strategy considers the different requirements for convergence and distribution performance of the MSHEA-SDDE in different iteration periods and uses the corresponding sequence difference strategy in different periods to further improve the convergence and distribution performance of MSHEA-SDDE. In the first iteration stage of the MSHEA-SDDE, the global search strategy of hybrid selection is used to make the population converge to the PF quickly. In the second stage of the MSHEA-SDDE iteration, a local search strategy based on sequence difference is used to improve the convergence performance by improving the performance of individuals with poor fitness values in the population. In the third iteration stage of the MSHEA-SDDE, the evolutional direction of the sequence differential strategy is adjusted to guide the population to search multiple areas of the PF, which is used to improve the distribution performance. An encoding and decoding approach with genetic operators relevant to the problem is included. The experimental results show that MSHEA-SDDE has better convergence and distributional performance than other multiobjective optimization algorithms.

The proposed MSHEA-SDDE can produce good scheduling solutions, but it also has the shortcoming that the CPU running time is not the shortest. The main reason is the computational time overhead of differential evolution search, but the experimental analysis shows that it can significantly improve the convergence and distribution performance of the MSHEA-SDDE. Therefore, how to reduce the computation time of the MSHEA-SDDE needs to be further investigated. With the development of industrialization, there will be a variety of variations of shop scheduling. These scheduling problems are more complex and require intelligent algorithms for optimization. In the future work, it is appropriate to consider some strategies such as the ABC algorithm, ant colony algorithm, simulated annealing, tabu search and other optimization algorithms, which can be improved and used as a local search strategy to improve intelligent algorithms, or the use of techniques such as neural networks and deep learning, which may also produce good results.

Acknowledgments

This research work was supported by the National Natural Science Foundation of China (U1904167), Innovative Funds Plan of Henan University of Technology (2020ZKCJ02), Innovative Research Team (in Science and Technology) in University of Henan Province (21IRTSTHN018), Zhengzhou Science and Technology Collaborative Innovation Project (21ZZXTCX19) and the Grant-in-Aid for Scientific Research (C) of the Japan Society of Promotion of Science (19K12148).

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. L. Wang, W. Shen, *Process Planning and Scheduling for Distributed Manufacturing*, Springer Science & Business Media, 2007. https://doi.org/10.1007/978-1-84628-752-7
2. X. Chen, V. Chau, P. Xie, M. Sterna, J. Błażewicz, Complexity of late work minimization in flow shop systems and a particle swarm optimization algorithm for learning effect, *Comput. Ind. Eng.*, **111** (2017), 176–182. https://doi.org/10.1016/j.cie.2017.07.016
3. X. Yu, M. Gen, *Introduction to Evolutionary Algorithms*, Springer Science & Business Media, 2010. https://doi.org/10.1007/978-1-84996-129-5
4. Y. Abdi, M. Feizi-Derakhshi, Hybrid multi-objective evolutionary algorithm based on search manager framework for big data optimization problems, *Appl. Soft Comput.*, **87** (2020), 105991. https://doi.org/10.1016/j.asoc.2019.105991
5. W. Zhang, D. Yang, G. Zhang, M. Gen, Hybrid multiobjective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search for VRPTW, *Expert Syst. Appl.*, **145** (2020), 113151. https://doi.org/10.1016/j.eswa.2019.113151
6. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6** (2002), 182–197. https://doi.org/10.1109/4235.996017

7. E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, *TIK Rep.*, **103** (2001). https://doi.org/10.3929/ETHZ-A-004284029

8. Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Comput.*, **11** (2007), 712–731. https://doi.org/10.1109/TEVC.2007.892759

9. W. Zhang, J. Lu, H. Zhang, C. Wang, M. Gen, Fast multi-objective hybrid evolutionary algorithm for flow shop scheduling problem, in *Proceedings of the Tenth International Conference on Management Science and Engineering Management*, Springer, Singapore, (2017), 383–392. https://doi.org/10.1007/978-981-10-1837-4_33

10. W. Zhang, Y. Wang, Y. Yang, M. Gen, Hybrid multiobjective evolutionary algorithm based on differential evolution for flow shop scheduling problems, *Comput. Ind. Eng.*, **130** (2019), 661–670. https://doi.org/10.1016/j.cie.2019.03.019

11. H. A. E. Khalifa, F. Smarandache, S. S. Alodhaibi, A fuzzy approach for minimizing machine rental cost for a specially-structured three-stages flow-shop scheduling problem in a fuzzy environment, in *Handbook of Research on Advances and Applications of Fuzzy Sets and Logic*, IGI Global, (2022), 105–119. https://doi.org/10.4018/978-1-7998-7979-4.ch005

12. B. Goyal, S. Kaur, Specially structured flow shop scheduling models with processing times as trapezoidal fuzzy numbers to optimize waiting time of jobs, in *Soft Computing for Problem Solving*, Springer, Singapore, (2021), 27–42. https://doi.org/10.1007/978-981-16-2712-5_3

13. V. Vinoba, N. Selvamalar, Improved makespan of the branch and bound solution for a fuzzy flow-shop scheduling problem using the maximization operator, *Malaya J. Mat.*, **7** (2019), 91–95. https://doi.org/10.26637/MJM0701/0018

14. M. Li, B. Su, D. Lei, A novel imperialist competitive algorithm for fuzzy distributed assembly flow shop scheduling, *J. Intell. Fuzzy Syst.*, **40** (2021), 4545–4561. https://doi.org/10.3233/JIFS-201391

15. B. Xi, D. Lei, Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time, *Complex Syst. Model. Simul.*, **2** (2022), 113–129. https://doi.org/10.23919/CSMS.2022.0002

16. J. Li, J. Li, L. Zhang, H. Sang, Y. Han, Q. Chen, Solving type-2 fuzzy distributed hybrid flowshop scheduling using an improved brain storm optimization algorithm, *Int. J. Fuzzy Syst.*, **23** (2021), 1194–1212. https://doi.org/10.1007/S40815-021-01050-9

17. J. Cai, D. Lei, A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time, *Complex Intell. Syst.*, **7** (2021), 2235–2253. https://doi.org/10.1007/S40747-021-00400-2

18. L. Wang, D. Li, Fuzzy distributed hybrid flow shop scheduling problem with heterogeneous factory and unrelated parallel machine: A shuffled frog leaping algorithm with collaboration of multiple search strategies, *IEEE Access*, **8** (2020), 214209–214223. https://doi.org/10.1109/ACCESS.2020.3041369

19. J. Cai, R. Zhou, D. Lei, Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: collaborative variable search, *J. Intell. Fuzzy Syst.*, **38** (2020), 3189–3199. https://doi.org/10.3233/JIFS-191175

20. K. Ying, S. Lin, C. Cheng, C. He, Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems, *Comput. Ind. Eng.*, **110** (2017), 413–423. https://doi.org/10.1016/j.cie.2017.06.025

21. M. E. Baysal, A. Sarucan, K. Büyüközkan, O. Engin, Distributed fuzzy permutation flow shop scheduling problem: a bee colony algorithm, in *International Conference on Intelligent and Fuzzy Systems*, Springer, Cham, (2020), 1440–1446. https://doi.org/10.1007/978-3-030-51156-2_167

22. J. Lin, Z. Wang, X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, *Swarm Evol. Comput.*, **36** (2017), 124–135. https://doi.org/10.1016/j.swevo.2017.04.007

23. M. E. Baysal, A. Sarucan, K. Büyüközkan, O. Engin, Artificial bee colony algorithm for solving multi-objective distributed fuzzy permutation flow shop problem, *J. Intell. Fuzzy Syst.*, **42** (2022), 439–449. https://doi.org/10.3233/JIFS-219202

24. K. Wang, Y. Huang, H. Qin, A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown, *J. Oper. Res. Soc.*, **67** (2016), 68–82. https://doi.org/10.1057/jors.2015.50

25. Z. Shao, W. Shao, D. Pi, Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem, *Swarm Evol. Comput.*, **59** (2020), 100747. https://doi.org/10.1016/j.swevo.2020.100747

26. J. Wang, L. Wang, A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling, *Comput. Ind. Eng.*, **168** (2022), 108126. https://doi.org/10.1016/j.cie.2022.108126

27. C. Lu, L. Gao, J. Yi, X. Li, Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China, *IEEE Trans. Ind. Inf.*, **17** (2020), 6687–6696. https://doi.org/10.1109/TII.2020.3043734

28. F. Zhao, H. Zhang, L. Wang, A pareto-based discrete jaya algorithm for multiobjective carbon-efficient distributed blocking flow shop scheduling problem, *IEEE Trans. Ind. Inf.*, **2022** (2022), 1–10. https://doi.org/10.1109/TII.2022.3220860

29. F. Zhao, S. Di, L. Wang, A hyperheuristic with q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem, *IEEE Trans. Cybern.*, **2022** (2022), 1–14. https://doi.org/10.1109/TCYB.2022.3192112

30. Z. Pan, D. Lei, L. Wang, A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling, *IEEE Trans. Cybern.*, **52** (2022), 5051–5063. https://doi.org/10.1109/TCYB.2020.3026571

31. F. Zhao, T. Jiang, L. Wang, A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time, *IEEE Trans. Ind. Inf.*, **2022** (2022), 1–12. https://doi.org/10.1109/TII.2022.3218645

32. C. Lu, Q. Liu, B. Zhang, L. Yin, A pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop, *Expert Syst. Appl.*, **204** (2022), 117555. https://doi.org/10.1016/j.eswa.2022.117555

33. J. Zheng, L. Wang, J. Wang, A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop, *Knowledge-Based Syst.*, **194** (2020), 105536. https://doi.org/10.1016/j.knosys.2020.105536

34. W. Zhang, W. Hou, D. Yang, Z. Xing, M. Gen, Multiobjective particle swarm optimization with directional search for distributed permutation flow shop scheduling problem, in *International Conference on Bio-Inspired Computing: Theories and Applications*, Springer, Singapore, (2019), 164–176. https://doi.org/10.1007/978-981-15-3425-6_14

35. S. Chanas, A. Kasperski, On two single machine scheduling problems with fuzzy processing times and fuzzy due dates, *Eur. J. Oper. Res.*, **147** (2003), 281–296. https://doi.org/10.1016/S0377-2217(02)00561-1

36. M. Sakawa, R. Kubota, Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms, *Eur. J. Oper. Res.*, **120** (2000), 393–407. https://doi.org/10.1016/S0377-2217(99)00094-6

37. B. Liu, Y. Liu, Expected value of fuzzy variable and fuzzy expected value models, *IEEE Trans. Fuzzy Syst.*, **10** (2002), 445–450. https://doi.org/10.1109/TFUZZ.2002.800692

38. J. J. Palacios, I. G. Rodriguez, C. R. Vela, J. Puente, Robust multiobjective optimisation for fuzzy job shop problems, *Appl. Soft Comput.*, **56** (2017), 604–616. https://doi.org/10.1016/j.asoc.2016.07.004

39. J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in *Proceedings of the First International Conference of Genetic Algorithms and Their Application*, Psychology Press, (1985), 93–100. https://doi.org/10.4324/9781315799674

40. W. Zhang, M. Gen, J. Jo, Hybrid sampling strategy-based multiobjective evolutionary algorithm for process planning and scheduling problem, *J. Intell. Manuf.*, **25** (2014), 881–897. https://doi.org/10.1007/s10845-013-0814-2

41. W. Li, J. Li, K. Gao, Y. Han, B. Niu, Z. Liu, et al., Solving robotic distributed flowshop problem using an improved iterated greedy algorithm, *Int. J. Adv. Rob. Syst.*, **16** (2019), 1729881419879819. https://doi.org/10.1177/1729881419879819

42. W. Xu, Y. Wang, D. Yan, Z. Ji, Flower pollination algorithm for multi-objective fuzzy flexible job shop scheduling, *J. Syst. Simul.*, **30** (2018), 4403. https://doi.org/10.16182/j.issn1004731x.joss.201811042

43. D. Lei, X. Guo, An effective neighborhood search for scheduling in dual-resource constrained interval job shop with environmental objective, *Int. J. Prod. Econ.*, **159** (2015), 296–303. https://doi.org/10.1016/j.ijpe.2014.07.026

44. W. Zhang, C. Li, M. Gen, W. Yang, Z. Zhang, G. Zhang, Multiobjective particle swarm optimization with direction search and differential evolution for distributed flow-shop scheduling problem, *Math. Biosci. Eng.*, **19** (2022), 8833–8865. https://doi.org/10.3934/mbe.2022410