International Journal of Industrial Engineering, 30(2), 373-386, 2023

ROBUST OPTIMIZATION OF STOCHASTIC HYBRID JOB-SHOP SCHEDULING WITH MULTIPROCESSOR TASK

Junyan Wang, Hua Qu*, Kun Fan, and Lang Zhou

School of Economics and Management
Beijing Forestry University
Beijing, China
*Corresponding author's e-mail: quhua@bjfu.edu.cn

Due to the large number of uncertainties in the production workshop, the actual performance of the scheduling scheme deviated significantly from the theoretical value. In order to enhance its anti-jamming capability, this paper developed the robust optimization of stochastic hybrid job-shop scheduling with multiprocessors tasks. Firstly, predictable uncertainties were abstracted into processing time variations and described by scenario analysis in the modeling process. Secondly, based on the analysis of the advantages and disadvantages of traditional robust optimization models, a new Expected C_{max} and the Worst scenario Model (ECWM) was proposed. The model improved the single-index robust optimization model and avoided the disadvantage that the Max Regret Model is computationally intensive. Finally, the effectiveness of ECWM is verified by simulation experiments. The results show that the scheduling obtained by ECWM has good average performance and anti-risk ability, which indicates that the model achieves a good balance in scheduling performance enthusiasm and risk resistance.

Keywords: Robust Optimization; Multiprocessor Task; Job-Shop Scheduling; Hybrid Scheduling.

(Received on April 19, 2022; Accepted on January 26, 2023)

1. INTRODUCTION

Workshop scheduling occupies an important position in the production management of manufacturing enterprises and plays an important role in improving the productivity of enterprises, reducing costs and enhancing market competitiveness. The Job-shop Scheduling Problem (JSP) is one of the most classic models in the field of workshop scheduling and is often solved by using genetic algorithm (Huang and Wang, 2016; Tan *et al.*, 2019), artificial neural networks (Lei *et al.*, 2022), and various swarm intelligence algorithms (Long *et al.*, 2022; Gu, 2021), etc. The JSP assumes that each process can only be completed on one processor. However, in the actual production workshop, there are often situations where certain working procedures require multiple processors (equipment or workers, etc.) to process simultaneously, which challenges the application of JSP. This JSP problem with multiprocessor task processing requirements is called the Hybrid Job-shop Scheduling with Multiprocessor Task (HJSMT), which can also be called Hybrid Multiprocessor Tasks Job-shop Scheduling. It can be seen as a hybrid scheduling problem that combines JSP and Multiprocessor task scheduling (MTS) (Fan *et al.*, 2018). Among them, the MTS requires multiple processors to process the workpiece processing task simultaneously, but the task is assigned only once and ends without the concept of working procedure. HJSMT scheduling is more commonly used in practical single-piece discrete manufacturing shops, parallel computing systems and multi-task assignments. However, there is little research on HJSMT by domestic and foreign scholars.

The generalized HJSMT is an NP-hard problem. Its model is characterized by high complexity and many constraints. For a long time, the research dedicated to HJSMT has only included: proposing an approximate scheduling method and an integrated method based on the taboo search to determine the scheduling scheme and studying only the insertion strategy of HJMST with workpiece insertion without implementing the design of the scheduling scheme (Gröflin *et al.*, 2008), etc. In recent years, in order to better solve the HJSMT problem, Fan *et al.* (2019) and Zhai *et al.* (2018) proposed an improved decentralized algorithm and particle swarm optimization algorithm, respectively, which achieved excellent results. In addition, in the network parallel computing system, the multi-step scheduling (Wang *et al.*, 2017) with multiprocessor task requirements is substantially consistent with the HJSMT problem in this paper, while the hybrid particle swarm optimization algorithm proposed in this reference provides a new idea for the in-depth study of the algorithm. In addition to the study of different algorithms for solving this problem, Fan *et al.* (2018) implement the whole simulation process of HJSMT using Plant Simulation software.

The scheduling problem in actual production faces a complex and changing dynamic environment, such as changes in material supply, sudden machine breakdowns (Soofi *et al*., 2021), changes in workers' working conditions or insertion of new jobs (Seyyedi *et al.*, 2021), etc. These uncertain factors will affect the implementation effect of the scheduling scheme and even make the scheme infeasible. Therefore, it is necessary to develop the study of the HJSMT scheduling problem under an uncertain environment, that is, the stochastic HJSMT problem.

If a scheduling scheme can also perform well when uncertain events occur, then this attribute is called "robustness". In order to get robust scheduling, the common method is to consider these uncertain factors in modeling. Mckay *et al*. (1989) divided the uncertainties in actual production into two categories: predictable uncertainty and unpredictable uncertainty. Because unpredictable uncertainties happen accidentally and unpredictably, they cannot be considered in the modeling of uncertainty problems.

In summary, this paper continues to study the stochastic HJSMT problem based on the deterministic model research. Considering the risk resistance capacity of the scheduling scheme and the predictable uncertainty factors, this paper uses the scenario analysis method to establish the stochastic HJSMT model and proposes a new robust optimization model, that is, the Expected Cmax and Bad-scenario Robust Model.

2. HYBRID JOB-SHOP SCHEDULING WITH MULTIPROCESSOR TASK

As mentioned above, the HJSMT problem is a more complex hybrid scheduling model that combines JSP and MTS. The problem is described as follows: there is a batch of workpieces to be processed by a certain number of processors, and each workpiece has a certain number of processing procedures. Each working procedure needs to occupy at least one processor at the same time. How to arrange the processing order of the processes on the processors to achieve the optimal index of an optimization.

(1) The workpiece should be processed according to the processing route, i.e., the first working procedure O_{i1}, then the working procedure O_{i2}, ..., and finally working procedure $O_{i|J_i|}$;
(2) The next working procedure can only start when the previous one is finished;
(3) Once the working procedure is started, it must be completed, and no interruption is allowed in the middle;
(4) A processor can only process one working procedure at a certain time, but it occupies 1 processor or more at the same time.
(5) There is no priority order between workpieces.

As mentioned earlier, the scheduling problem in Wang *et al*. (2017) is substantially consistent with HJSMT. Since the authors will continue to use the HPSO algorithm (Wang *et al*., 2017) previously designed by the research team to solve the newly proposed robust optimization model in this paper, the mathematical description of the HJSMT problem also uses a model consistent with that of Wang *et al*. (2017), namely:

Objective function:

$$\min C_{max} = x_{df} \tag{1}$$

s.t.

$$x_{i,j'} - x_{ij} \geq p_{i,j'}, for\ all\ \{O_{ij}, O_{i,j'}\} \in A \tag{2}$$
$$x_{ij} - x_{i',j'} \geq p_{i',j'} \vee x_{i',j'} - x_{ij} \geq p_{i,j}, for\ all\ \{O_{ij}, O_{i',j'}\} \in B \tag{3}$$
$$x_{ij} - x_{ds} \geq 0 \vee x_{ds} - x_{ij} \geq p_{ij}, for\ all\ O_{ij} \in I \tag{4}$$

The meaning of the mathematical symbol is as follows: there are n workpieces $J = \{J_1, J_2, \cdots, J_n\}$ to be processed on m processor $M = \{M_1, M_2, \cdots, M_m\}$. $I = \{O_{ij} | i = 1,2, \cdots, n; j = 1,2, \cdots, |J_i|\}$ stands for the working procedure set. O_{ij} takes up all the processors in the processor set m_{ij} at the same time and the time required is P_{ij}. The "start virtual process" O_{ds} and the "end virtual process" O_{df} are defined, which represent the first and last working procedure of the entire process, respectively. x_{ij} stands for the start processing time of the working procedure $O_{ij} (\subseteq I \cup \{O_{ds}, O_{df}\})$. In addition, $A = \{\{O_{ij}, O_{i'j'}\} | O_{ij}, O_{i',j'} \in I; i = i'; j' > j\}$. $\{O_{ij}, O_{i',j'}\} \in A$ represents different pairs of working procedure from the same workpiece. $B = \{\{O_{ij}, O_{i'j'}\} | O_{ij}, O_{i'j'} \in I; fix_{i'j'} \cap fix_{ij} \neq \emptyset; i \neq i'\}$. $\{O_{ij}, O_{i',j'}\} \in B$ represents pairs of working

procedure from different workpieces and at least one processor in the set of processors required by working procedure $\{O_{ij}, O_{i',j'}\}$ is the same (Tavakkoli *et al*., 2005).

3. STOCHASTIC ROBUST OPTIMIZATION MODEL

In studying stochastic workshop scheduling problems, for the predictable uncertainty factors, t-modeling methods mainly include the probability distribution method (Tavakkoli *et al*., 2005), the fuzzy set method (Liu *et al*., 2011) and the scenario analysis method. Both the probability distribution method and the fuzzy set method only consider the large probability events while ignoring the smaller probability events when modeling stochastic problems, so the resulting solutions are statistically effective.

In order to solve the above problems, the scenario analysis method is mainly used in the research of robust optimization for stochastic problems at home and abroad. The basic idea of this method is to represent the stochastic environments in tree form according to the time stage, starting from the root node and continuously branching to the leaf node, so as to consider all possible scenarios (Zhang and Wang, 2009). In the field of workshop scheduling, the scenario method has been used in many studies (Talla and Leus, 2014; Wang *et al*., 2012) and achieved good results.

3.1 Stochastic HJSMT problem based on scenario description

For the stochastic HJSMT problem, this paper reflects the uncertain factors as the change of processing time P_{ij}, so it is not fixed, but has multiple possible values according to the occurrence of uncertain events. Each uncertain event is a scenario with a certain probability, which represents a possible realization of random variables.

In this paper, a possible value of processing time $p_{ij}, i = 1,2,\dots,n; j = 1,2,\dots,|J_i|$ for all working procedure of stochastic HJSMT is regarded as a scenario, and the probability of occurrence of the scenario is assumed to be known. The symbols used in this paper are as follows:

Λ: denotes the set of all scenarios; λ denotes a certain scenario, $\lambda \in \Lambda$; Λ_β: denotes a bad scenario set. $Pr(\lambda)$: denotes the probability of occurrence of the scenario λ; s: denotes a solution to the problem, i.e., scheduling; s_λ^*: optimal scheduling under scenario λ; $C(s,\lambda)$: maximum completion time of solution s under scenario λ, i.e., C_{max}; $C^*(s,\lambda)$:the optimal performance in all scenarios. p_{ij}^λ: Processing time of working procedure O_{ij} under scenario λ. Obviously, there is a relationship between λ and p_{ij}^λ: $\lambda = \{p_{ij}^\lambda | i = 1,2,\dots,n; j = 1,2,\dots,|J_i|\}$, i.e., the scenario λ represents one possible realization of the processing time of all working procedure.

3.2 Traditional robust optimization model

In this subsection, the traditional robust optimization model is briefly introduced and analyzed, and a new robust optimization model is proposed based on it.

3.2.1 Expected C_{max} Model

The starting point of the Expected C_{max} Model (ECM) is very simple. Its goal is to minimize the weight C_{max} of all scenarios in the scenario set Λ, and the weight is generally the probability of the scenario. ECM can be regarded as the pursuit of excellent performance in the statistical sense.

$$\min_s EC(s) = \sum_{\lambda \in \Lambda} Pr(\lambda) \times C(s,\lambda) . \tag{5}$$

3.2.2 Expected C_{max} and its Variance Model

The Expected C_{max} and its Variance Model (ECVM) adds the variance of the scheduling performance as one of the optimization metrics based on the ECM, which is intended to control the fluctuation of the scheduling performance so as to be robust. Where $0 \le \alpha_V \le 1$.

$$\min_s ECV(s) = (1-\alpha_v) + \alpha_v \sum Pr(\lambda) \times [C(s,\lambda) - EC(s)]^2 . \tag{6}$$

3.2.3 Worst C_{max} Model

The optimization goal of the Worst C_{max} Model (WCM) is to minimize the worst performance $\max_{\lambda\in\Lambda}\{C(s,\lambda)\}$.

$$\min_{s} WC(s) = \max_{\lambda\in\Lambda}\{C(s,\lambda)\}. \tag{7}$$

WCM only considers the worst scenario and does not pursue good performance in other scenarios, so its decision-making is highly conservative (Wang *et al*., 2012).

3.2.4 Max Regret Model

The Max Regret Model (MRM) takes the relative difference between the performance $C(s,\lambda)$ of the scenario λ under scheduling s and the optimal performance $C(s_\lambda^*,\lambda)$ of the scenario λ as the optimization index. Where is the optimal scheduling s_λ^* in the scenario λ.

$$\min_{s} MR(s) = \max_{\lambda\in\Lambda}\{C(s,\lambda) - C^*(s,\lambda)\}. \tag{8}$$

MRM is an important model in the field of decision-making. The idea is that when something happens, the value of the loss is formed because the decision maker did not choose the strategy with the greatest gain.

3.2.5 Expected C_{max} and Bad-scenario Robust Model

Considering the advantages and disadvantages of the previous four types of robust optimization models, Wang *et al*. (2012) made an improvement on ECM and MRM, that is, all scenario sets whose performance exceeds the expected performance to a certain extent ($\beta EC(s)$) are defined as bad scenario sets, and the "exceeding" part is taken as the optimization index, which is Expected C_{max} and Bad-scenario Robust Model (ECBM) (Wang *et al*., 2012).

$$\min_{s} ECBR(s) = (1-\alpha_B)EC(s) + \alpha_B \sum_{\lambda\in\Lambda_\beta} Pr(\lambda) \times [C(s,\lambda) - \beta EC(s)], \tag{9}$$

where $0 \le \alpha_B \le 1$. This model not only includes the pursuit of expected scheduling performance (EC(s)) but also suppresses the appearance of bad scenarios and has good robustness.

Among the above five types of traditional robust optimization models, ECM is only the pursuit of performance in a statistical sense; ECVM uses variance as the optimization objective to suppress performance fluctuations, but the variance will inhibit the appearance of good scenarios; WCM only optimizes the performance of the worst scenarios; MRM takes into account both performance enthusiasm and risk resistance, but the amount of calculation is huge; ECBM improves ECM well, which is a statistical pursuit of average scheduling performance and risk resistance. In order to effectively optimize the performance of bad scenarios while optimizing the average performance of HJSMT scheduling, this paper proposes a new expectation-worst scenario robust optimization model to achieve this goal.

3.3 Robust optimization model construction and analysis of HJSMT for Expected C_{max} and the Worst scenario Model

3.3.1 Expected C_{max} and the Worst scenario Model

As mentioned above, ECBM pursues scheduling robustness in a statistical sense. However, this paper hopes to get such scheduling: it performs well enough in general scenarios, and it does not perform badly in some bad scenarios. For this reason, we learn from the practice of WCM, take $\max_{\lambda\in\Lambda}\{C(s,\lambda)\}$ as one of the optimization goals, and add the optimization of the average performance of HJSMT scheduling on this basis so that the Expected C_{max} and the Worst scenario Model (ECWM) is obtained:

$$\min_{s} ECW(s) = (1-\alpha_w)EC(s) + \alpha_w \max_{\lambda\in\Lambda}\{C(s,\lambda)\}, \tag{10}$$

where $0 \le \alpha_W \le 1$. Here, $\lambda_w = \arg\max_{\lambda\in\Lambda}\{C(s,\lambda)\}$ is referred to as the worst scenario of scheduling s.

Since the weighting of the first part of $EC(s)$ in the above equation (7) is used to improve the average performance of HJSMT scheduling, and the weighting of the second part of $WC(s)$ is used to improve the performance of the worst scenario. ECWM is essentially a robust optimization model that combines ECM and WCM.

3.3.2 ECWM Robust Scheduling Model Analysis

As a combined model, the performance of ECWM must be affected by two purely biased uncertain models. Discussion on the properties of ECWM: different values of α_W can achieve the balance of average performance and robustness in different degrees, among which property 1 and property 2 are obvious.

Property 1: if $\alpha_W = 0$, then ECWM degenerates to ECM： $ECW(s) = EC(s)$. So ECWM has the characteristics of pursuing average performance optimization.

Property 2: if $\alpha_W = 1$, then ECWM degenerates to WCM： $ECW(s) = WC(s)$; For the optimal solution s^* of WCM, when ECWM takes $\alpha_W = 1$, its optimal solution is also s^*, which shows that ECWM has the same optimal choice as WCM in terms of anti-risk ability.

According to property 1 and property 2, ECWM has both ECM and WCM characteristics, which can balance the pursuit of anti-risk ability and average performance.

Property 3: When $\alpha_W \in [0,1]$, for ECWM, the following properties can be obtained:

$$EC^{ECM} \le EC^{ECWM} \le EC^{WCM} \text{ and } WC^{WCM} \le WC^{ECWM} \le WC^{ECM}.$$

For property 3, the following process is given to prove: the equation (7) can be transformed into:

$$EC(s)\text{-}\ \alpha_w \left(EC(s) - \max_{\lambda \in \Lambda} \{C(s,\lambda)\}\right). \tag{11}$$

Since $WC(s) = \max_{\lambda \in \Lambda} \{C(s,\lambda)\}$ is the worst-case scenario performance, there must be $EC(s) \le \max_{\lambda \in \Lambda} \{C(s,\lambda)\}$,so when the value of ECW(s) is the smallest, it satisfies $\alpha_w = 0$, and degenerates to the ECM model at this time, so $EC^{ECM} \le EC^{ECWM}$.

Let s^* be the optimal solution of ECWM, then for any schedule s of ECWM, $EC^S \ge EC^*$, and when $\alpha_w = 1$, ECWM(s) degenerates to WC(s). At this time, for the optimal solution s^* of ECWM, which represents the optimal solution for the worst scenario under ECWM, so for any scenario λ, the corresponding scheduling s has $EC^s \le EC^{WCM}$. In summary, $EC^{\text{ECM}} \le EC^{ECWM} \le EC^{WCM}$.

Similarly, let s^* be the optimal solution of WCM, and it has been proved that WCM is the realization of a specific scenario of ECWM, and s^* is the worst scenario performance of the optimal solution. Therefore, for any scheduling s of ECWM, it must satisfy $WC^S \ge WC^*$. When $\alpha_w = 0$, ECWM degenerates to ECM. At this time, ECWM's WC^* performance is the worst case and does not have robustness and only pursues average performance, and WC^{ECM} satisfies $WC^{ECM} \ge WC^S$. In summary: $WC^{WCM} \le WC^{ECWM} \le WC^{ECM}$.

Property 3 reveals the inherent logic of ECWM in balancing performance and robustness. In theory, its performance is not worse than ECM, and its robustness is not worse than WCM. Therefore, ECWM can achieve the goal of avoiding significant deterioration in bad scenarios while ensuring average performance.

4. SOLVING THE HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM FOR HJSMT

4.1 Classical particle swarm optimization algorithm

PSO algorithm is a kind of bionic algorithm which was first proposed by Kennedy and Eberhart (1995). PSO is essentially an iterative algorithm, which has the advantages of easy to use, high accuracy, and fast convergence speed.

The update equation of particle position and velocity in the PSO algorithm is as follows:

$$v_{id}^{(k+1)} = wv_{id}^{(k)} + c_1 rand_1^{(k)}(Pbest_{id}^{(k)} - x_{id}^{(k)}) + c_2 rand_2^{(k)}(Gbest_d^{(k)} - x_{id}^{(k)}) \tag{12}$$

$$x_{id}^{(k+1)} = x_{id}^{(k)} + v_{id}^{(k)}\ , \tag{13}$$

where $v_{id}^{(k)}$ and $x_{id}^{(k)}$ denote the velocity and position of the d-th dimension of the *i*-th particle in the *k*-th iteration, respectively. The optimal solution that can be found by an individual particle is called the "individual extreme value", denoted by *pbest*; the optimal value that can be found by the whole particle population is called "global optimum", denoted by *gbest*.

4.2 Hybrid particle swarm optimization algorithm

PSO algorithm is widely used in scheduling problems, and there are many improved algorithms (Jia *et al*., 2012; Lv *et al*., 2021). This is because the traditional PSO algorithm itself is defined in the continuous domain optimization algorithm, but the definition of the scheduling problem is often discrete. At the same time, although the PSO algorithm performs well in the global search, it will miss some high-quality solutions in the local search process, so it needs to be adjusted and improved in dealing with the scheduling problem.

As a hybrid scheduling model of JSP and MTS, HJSMT is more complex than JSP. In HJSMT, working procedure need to be processed by multiple processors at the same time. Because it does not meet the constraint of the JSP model that "one working procedure can only be processed on one machine", many classical algorithms suitable for JSP cannot solve HJSMT problems. To this end, we implement the HPSO algorithm to solve the problem of this paper based on the previous research results of our research team (Wang *et al*., 2017). The algorithm adds a simulated annealing algorithm (SA) to GPSO (Gao *et al.*, 2015), which improves the local search ability of the algorithm, and can effectively solve the discrete combinatorial optimization problem of HJSMT.

HPSO adopts a new position update method and a local search method based on simulated annealing. At the same time, in order to ensure the performance of the algorithm, a memory of size *b* is introduced to record the best *b* particles found by the particle swarm. Swarms tend to be large, and local searches for all particles result in huge calculations and reduce the efficiency of the algorithm, so only particles in the memory are searched locally. Here, the fitness value of a particle is defined as the maximum completion time of the scheduling scheme that the particle represents. The HPSO flow is shown in Figure 1.

4.3 Particle encoding strategy

This paper adopts the most widely used process-based coding method: using $|J_i|$ represents the number of working procedures ($i = 1,2,3 \ldots, n$) of workpiece *i* and represents a particle as a vector of length $\sum_{i=1}^{n} |J_i|$, which is used to represent a scheduling scheme. Each element of the vector represents the corresponding task, and element i occurs $|J_i|$ times and the order of occurrence represents the corresponding step. For example, when there are 3 workpieces, each with 3 working procedures, a vector (particle) is (1,2,3,3,1,1,3,2,2), and its corresponding processing order is (O_{11}, O_{21}, O_{31}, O_{32}, O_{12}, O_{13}, O_{33}, O_{22}, O_{23}) (O_{ij} represents the *j*-th working procedure of the *i*-th workpiece).

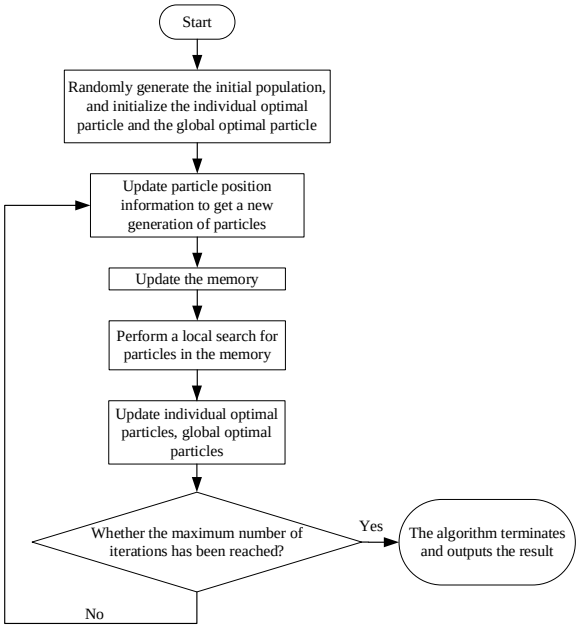

Figure 1. HPSO algorithm flow

4.4 Particle update method

In HPSO, the update of particle position can be regarded as the information exchange between the current particle and the individual optimal particle and the global optimal particle. Inspired by the genetic algorithm, the crossover operation is used here to achieve the information exchange between particles. In each iteration, the particles are updated in the following way.

Step 1: Cross the current particle with the individual optimal particle to get the new particle P1.
Step 2: Cross the current particle with the global optimal particle to get the new particle P2.
Step 3: Compare the fitness values of P1 and P2, and replace the current particle with the new particle with the smaller fitness value.

It is worth noting that in step 3, even if the fitness value of the current particle is smaller than the fitness value of the new particle, the replacement operation is still performed. By means of the particle update described above, the particles move from one position to another and search for the optimal solution in the solution space. Using A1 and A2 to represent the particles to be crossed, the new particle is obtained as B after crossover, and the crossover operation (Wang *et al*., 2017) is defined as follows.

Step 1: Divide the task set $S = \{J_i | i = 1,2,\ldots,n\}$ into two subsets S1 and S2 at random.
Step 2: The child particle B1 inherits the elements belonging to S1 in A1 and the elements belonging to S2 in A2.
Step 3: The child particle B2 inherits the elements belonging to S2 in A1 and the elements belonging to S1 in A2.
Step 4: Compare the fitness values of child particles B1 and B2, and output the child particle with smaller fitness value, i.e., new particle B.

Figure 2 shows an example of crossover operation, when S1={J2, J3} and S2={J1}, the new particle B1 retains elements 2 and 3 in A1 and element 1 in A2, and B2 retains element 1 in A1 and elements 2 and 3 in A2. By this crossover operation, the child particles inherit some information from the parent particles and "mutate" on this basis to search for the optimal solution of the problem on a larger scale.

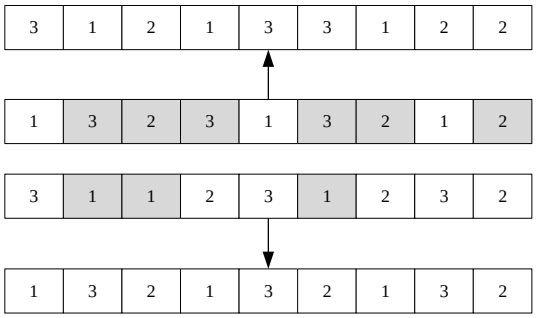

Figure 2. Crossover operation to obtain child particles

4.5 Particle local search method

Although the traditional particle swarm optimization algorithm has good global search capability, the local search capability is weak, and it is easy to miss high-quality solutions. For this reason, HPSO adds a local search algorithm based on simulated annealing on the basis of PSO. The simulated Annealing algorithm (SA) is a local search algorithm simulating the solid annealing principle. Unlike the traditional local search algorithm (hill climbing algorithm), SA can accept inferior solutions with a certain probability and has the ability to jump out of the local optimum, which has good effect on multi-peak optimization problems with good results. In each iteration, the b particles in the memory will be used as the initial solution of SA for local search.

A neighborhood particle is a new particle obtained by giving a small change to the current particle, and the local search process can be regarded as the process of generating neighborhood particles. In order to generate a neighborhood solution, a reasonable neighborhood structure needs to be set, and the advantages and disadvantages of the neighborhood structure are directly related to the SA search performance. For the encoding method in 4.3, the operations commonly used to generate neighborhood solutions are inverse order, insertion and interchange. Yang *et al. (*2012) points out that the interchange operation is more conducive to the large range search of the algorithm, so this paper uses the interchange operation to generate

the neighborhood solution, i.e., given a particle, two different elements are randomly selected, and their positions are exchanged to obtain the neighborhood particles, as shown in Figure 3.

When SA starts its search, the initial temperature T_0 should be set high enough to be able to search for all particles. In the initialization stage of the particle swarm optimization algorithm, the fitness values of the optimal particle and the worst particle are recorded: p_b and p_w, respectively, and the probability of accepting the worst particle is denoted as $p_r \in (0,1)$, so that there is $p_r = \exp[-(p_w - p_b)/T_0]$, from which the initial temperature can be obtained as $T_0 = -(p_w - p_b)/(ln\ pr)$. SA has a sampling step of *L* at each temperature, yielding a neighborhood particle each time through an interchange operation, and if the particle is better than the current particle, the current particle is replaced with that particle; If the particle is inferior to the current particle, the neighborhood particle is accepted with probability $p_r = \exp[-(p_w - p_b)/T_k]$. After *L* searches, SA performs a cooling operation, which uses the index cooling method: $T_k = \lambda T_{k-1}$.

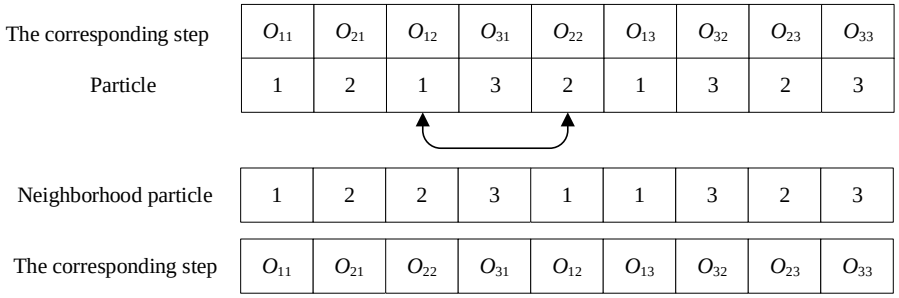

Figure 3. Interchange operation to obtain neighborhood particles

4.6 HPSO algorithm steps

The main steps of HPSO are as follows.

Step 1:	Given the population size popsize, the maximum number of iterations MaxIterion, set the current iteration *iter* = 1.
Step 2:	Randomly generate the initial particle population $P_t^{(0)}$, where t = 1, 2,…, popsize, and determine the fitness value $F(P_t^{(0)})$ for each particle.
Step 3:	Initialize the individual optimal particle $Pbest_t^{(0)} = P_t^{(0)}$, the global optimal particle $Gbest^{(0)} = \{Pbest_t^{(0)} \mid F(Pbest_{t'}^{(0)}) = \max_t\{F(Pbest_t^{(0)})\}\}$, and select b optimal $Pbest_t^{(0)}$ into the memory.
Step 4:	Update all current particle $P_t^{(iter-1)}$ with their individual optimal particles $Pbest_t^{(iter-1)}$ and global optimal particles $Gbest_t^{(iter-1)}$ to obtain the new particle $P_t^{(iter)}$.
Step 5:	Checks the fitness value of each new particle $F(P_t^{(iter)})$ and performs a replacement operation if the fitness value of the new particle is less than the fitness value of the worst particle in the memory.
Step 6:	Perform a local search for particles in the memory.
Step 7:	Update the individual optimal particle; if $F(Pbest_t^{(iter-1)}) \geq F(P_t^{(iter)})$, then $Pbest_t^{(iter)} = P_t^{(iter-1)}$, otherwise $Pbest_t^{(iter)} = Pbest_t^{(iter-1)}$; then update the global optimal particle.
Step 8:	If *iter* ≥ *MaxIterion*, the algorithm stops, and the output result is $Gbest^{(iter)}$; Otherwise, *iter* = *iter* + 1 and go back to step 4.

5. EXPERIMENT SIMULATION AND RESULT ANALYSIS

The difference between the solution of the stochastic HJSMT problem and the deterministic problem is that the objective function of the algorithm is different: the objective function of the deterministic HJSMT problem is the maximum completion time C_{max}; The objective function of the stochastic HJSMT problem is the various robust optimization models introduced in the previous section. Therefore, as long as the algorithm is determined, the robust optimization model introduced above can be used to solve any uncertain scheduling problem.

5.1 Design of stochastic HJSMT problem examples and selection of performance index

In order to test the performance of HPSO on the HJSMT problem, this paper adopts the method of GröflinandLinkert (2008). First, a batch of deterministic HJSMT problem examples P1~P8 with different scales are randomly generated (see Table 1).

The stochastic HJSMT problem examples are derived from the randomization of deterministic HJSMT problems P1 ~ P8. According to the practice of related Wang *et al.* (2012), stochastic HJSMT problem examples are generated as follows: The number of scenarios for each problem $|\Lambda|$ is equal to 50; assuming that the probability of all scenarios is the same, that is, $Pr(\lambda) = \frac{1}{50}$, $\lambda \in \Lambda$; random processing time is randomly selected within $[1,99]$, that is:

$$p_{ij}^{\lambda} \in U[1,99], i = 1,2,\ldots,n,\ j = 1,2,\ldots,|J_i|,\ \lambda \in \Lambda \tag{14}$$

In the selection of performance index, it is hoped that the selected performance index can reflect the enthusiasm and risk resistance of the robust optimization model, mainly including the following three aspects: (1) The average performance of scheduling in different scenarios; (2) The fluctuation of scheduling in different scenarios; (3) Scheduling performance in bad scenarios. Therefore, the expected performance EC, variance VC and the performance WC in the worst scenario are selected as performance indexes.

Table 1. Deterministic HJSMT problems P1~P8

Problem		P1	P2	P3	P4	P5	P6	P7	P8
Scale	n	5	5	5	10	10	10	20	20
	q	5	10	15	5	10	15	5	10

Note: n denotes the number of workpieces, and q denotes the number of working procedures.

In the simulation experiment, we first analyze the performance changes of ECWM under different weights α_W to find the best weight α_W^*, then use ECWM and traditional robust optimization model to solve the stochastic HJSMT problem, and the experimental results are compared and analyzed.

5.2 Variation of ECWM performance with different weights

In ECWM, the weight α_W plays a role in regulating the equilibrium relationship between scheduling enthusiasm and robustness. The scheduling obtained under different weights has different emphasis. In order to obtain the performance difference of ECWM under different weight α_W, the experiment uses ECWM under different weight to solve the random P1 problem. The values of the weight α_W are 0, 0.1, 0.2, …, 1. The algorithm runs 10 times under each weight, and the best fitness value is the experimental result. This is the maximum potential of the ECWM model, considering different weight values. The experimental results are summarized in Table 2.

Table 2. ECWM performance with different weights

α_W	EC	VC	WC
0	554	3230	695
0.1	554	3230	695
0.2	556	3040	695
0.3	560	2746	665
0.4	566	2703	669
0.5	570	2966	657
0.6	570	2966	657
0.7	570	2966	657
0.8	570	2966	657
0.9	570	2966	657
1	570	2966	657

It can be seen from the results in Table 2 that when $\alpha_W = 0$, ECWM is ECM, and the enthusiasm of scheduling (the average value of scheduling in different scenarios) is the best, but the performance fluctuates greatly, and the robustness is also the worst. When $\alpha_W = 0.1, 0.2$, the result obtained is not much different from that when $\alpha_W = 0$. This is because the

weight α_W is too small, and the ECWM's robustness $\alpha_W \max_{\lambda \in \Lambda}\{C(s,\lambda)\}$ is not highlighted. With the gradual increase of α_W, the enthusiasm of scheduling is decreasing, while the robustness is getting better and better, and the performance fluctuation is also weakening. As can be seen in Figure 4, each performance indicator of scheduling reaches a steady state when $\alpha_W \geq 0.5$, which is the least aggressive and the best in terms of robustness and performance fluctuations at the same time. This is because too large weight α_W will cause $(1-\alpha_W)EC(s)$ to be too small, and ECWM's pursuit of enthusiasm is not being exploited. It is worth pointing out that when $\alpha_W = 0.3$, EC only increased by 6 units, but WC decreased by 30 units. At this time, the enthusiasm and robustness of scheduling reached a very good balance, so in the following experiments, the value of α_W is set to 0.3.

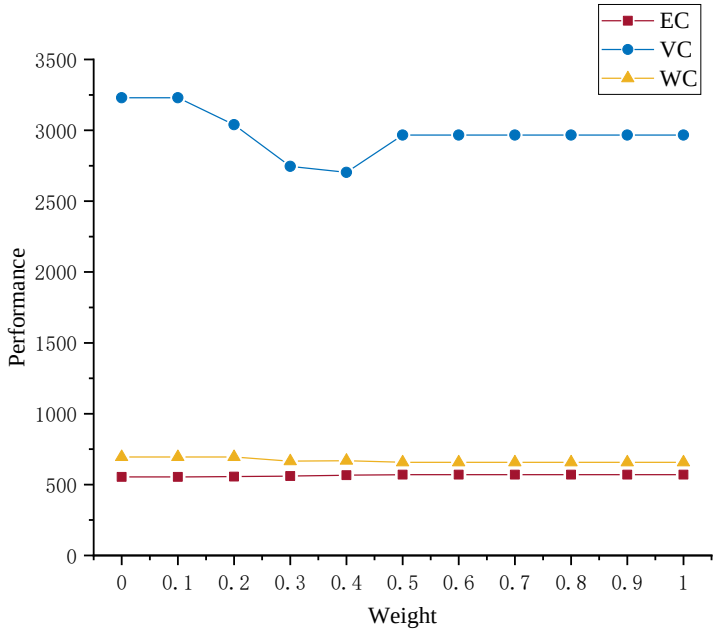

Figure 4. Performance variation of ECWM with different weights

5.3 Comparison of ECWM and traditional robust optimization models

The following experiments use ECWM and traditional robust optimization models to solve more complex stochastic HJSMT problems and analyze the advantages and disadvantages of these models by comparing the experimental results. This experiment still uses the HPSO algorithm, and its parameter setting and running environment are consistent with Section 3.2. According to the experimental results in the previous section, the weight of ECWM is set to 0.3, i.e., $\alpha_W = 0.3$. In addition, according to Wang *et al*. (2012), the weights and parameters of the traditional robust optimization model are set as follows: $\alpha_V = 0.5$, $\beta = 1.05$, $\alpha_B = 0.5$. Each model is used to solve each problem 10 times, and the best one is taken as the experimental result. The experimental results are summarized in Table 3.

Table 3. Comparison of ECWM and traditional robust optimization models

Example	Performance	ECM	ECVM	WCM	MRM	ECBM	ECWM
P1	EC	554*	594#	570	565	576	560^
	VC	3230#	1939*	2966	2780	2588^	2634
	WC	695	696#	657*	700	669	665^
P2	EC	790*	845#	802	796	834	794^
	VC	5899	2936*	4082^	6094#	5926	5806
	WC	956#	952	896*	946	923	900^
P3	EC	1060*	1163	1092#	1061^	1168	1069
	VC	9596#	3201*	6173	7781	4817^	9180
	WC	1278#	1275	1250*	1274	1275	1267^
P4	EC	681*	806#	725	701	694^	697

Example	Performance	ECM	ECVM	WCM	MRM	ECBM	ECWM
	VC	3512#	1781*	2738^	3339	3008	2813
	WC	874	899#	802*	849	866	806^
	EC	1122	1274#	1181	1146^	1167	1153
P5	VC	4762#	1091*	3124^	3421	3185	3601
	WC	1282	1336#	1241*	1269	1291	1239^
	EC	1416*	1567#	1510	1477	1497	1441^
P6	VC	5438	1110*	3482^	5148	5107	5918#
	WC	1665#	1646	1573*	1648	1593	1584^
	EC	1208*	1411#	1345	1234^	1248	1255
P7	VC	7665	2303*	6130	9293#	4797^	6938
	WC	1599#	1523	1495*	1506	1547	1524^
	EC	1613*	1876	1721	1630^	1676	1662
P8	VC	7416	1536*	8515#	7223	5401^	6806
	WC	1911	1962#	1766*	1821	1779^	1801

Note: * indicates the best value of this performance, ^ indicates the second-best value of this performance, and # indicates the worst value of this performance.

Among the three indexes, the expectation EC reflects the enthusiasm of scheduling to pursue excellent performance, the variance VC reflects the volatility of scheduling performance in different scenarios, and the worst scenario performance WC reflects the anti-risk ability of scheduling. As can be seen in Figures 5, 6 and 7, ECM, ECVM and WCM achieve the best values of their respective optimization metrics EC, VC and WC, respectively, in most cases, but are also the models with the worst values. This is because ECM and WCM belong to a single index optimization model, and it is difficult to consider the performance beyond their respective optimization index. ECVM takes variance VC as one of the optimization indexes, which can effectively reduce the fluctuation of scheduling in various scenarios, but it is at the expense of good performance enthusiasm and anti-risk ability. The other three models, MRM, ECBM and ECWM, all perform between the best value and the worst value in the performance index EC and WC. This result is consistent with the essence of the three models, that is, a balance and compromise between the single index optimization model ECM and WCM in different ways. Among the six robust optimization models, ECWM gets the most suboptimal value, which shows that the model achieves a good balance in terms of performance enthusiasm and risk resistance; The result of ECBM is a little lower than that of ECWM, and it also achieves a good balance in performance enthusiasm and risk resistance; The performance of MRM is slightly worse than that of ECBM, which is consistent with the conclusion of Wang *et al*. (2012). Because MRM uses the near-optimal solution of each scenario rather than the optimal solution (because HJSMT is an NP-hard problem, the optimal solution is difficult to obtain), it is difficult to accurately obtain the regret value $C(s,\lambda) - C^*(s,\lambda)$ of scheduling s, so the effect is not fully developed.

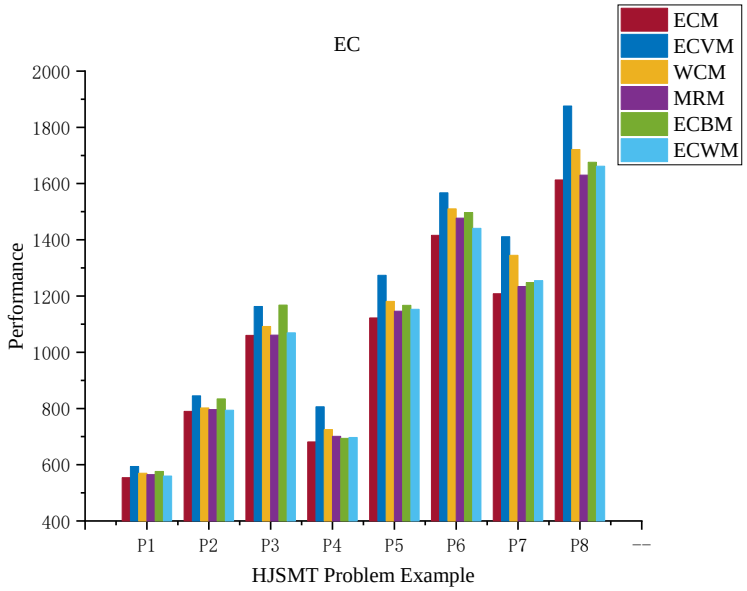

Figure 5. Comparison of EC metrics between ECWM and traditional robust optimization model

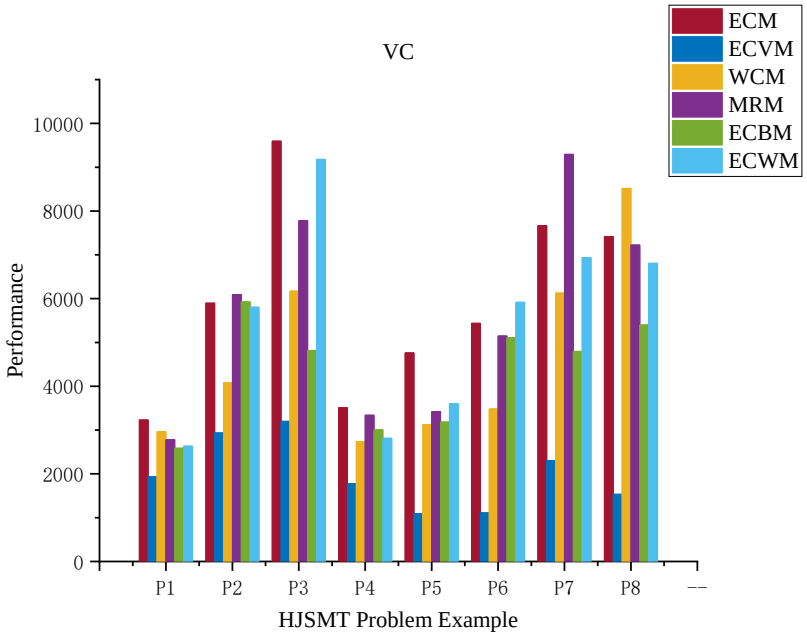

Figure 6. Comparison of VC metrics between ECWM and traditional robust optimization model

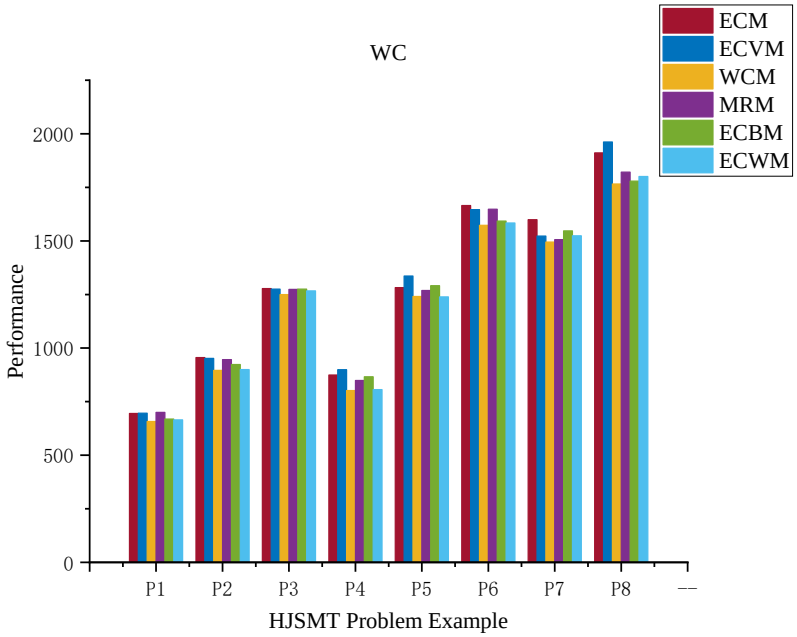

Figure 7. Comparison of WC metrics between ECWM and traditional robust optimization model

Although WCM is the most robust model, which has the highest risk resistance (the optimal WC value) in all HJSMT problems, WCM is too conservative, and its enthusiasm for excellent performance is low. ECVM gets the worst performance, which shows that ECVM performs the worst in terms of good performance enthusiasm and risk resistance. Because ECVM takes variance VC as the optimization index, it not only suppresses the performance of bad scenarios but also suppresses the performance of good scenarios, which worsens the performance of ECVM in performance indexes EC and WC. However, ECVM is the best in terms of performance fluctuation of scheduling in various scenarios.

6. CONCLUSIONS

In conclusion, the ECWM model proposed in this paper is not the best performance in a single performance, but considering the robustness, performance enthusiasm and risk resistance of scheduling results, ECWM performs better than the other five models.

Since there is usually a large deviation between the theoretical and actual performance of the scheduling scheme obtained from the deterministic model when implemented in an uncertain environment, this requires that the scheduling needs to be risk-resistant, i.e., it can maintain good performance even when encountering uncertain events. To this end, this paper takes the initiative to consider uncertainties in real production workshop and uses scenario analysis to describe the processing time in stochastic HJSMT problems and proposes a new robust optimization model for stochastic HJSMT, namely the Expected C_{max} and the Worst scenario Model (ECWM). In the simulation experiment, ECWM is used to solve the stochastic HJSMT problem, and the solution results are compared with traditional robust optimization models ECM, ECVM, WCM, MRM and ECBM. The results show that the scheduling obtained by ECWM has not only good average performance but also good anti-risk ability, which indicates that the model has reached a good balance between scheduling performance enthusiasm and risk resistance.

Although the performance of ECWM proposed in this paper is excellent, it needs a lot of computation and takes a long time to solve. This disadvantage is also the disadvantage of traditional robust optimization algorithms such as ECM, ECVM, WCM, MRM and ECBRM. From the perspective of reducing the amount of computation, future research can develop a robust optimization model with less computation and faster solution speed. In addition, sensitivity analysis can also be used to find out the indexes that may affect the robustness of HJSMT, and the objective function that meets the requirements can be constructed and optimized.

ACKNOWLEDGEMENT

The helpful comments and suggestions of the anonymous referees will be much appreciated by the authors. This research is supported by the Ministry of Education Humanities and Social Sciences Foundation of China (No.21YJA630012).

REFERENCES

Fan, K., Wang, M., Zhai, Y., and Li, X. (2019). Scatter search algorithm for the multiprocessor task job-shop scheduling problem. *Computers & Industrial Engineering*, 127: 677–686.

Fan, K., Zhai, Y., Li, X., and Wang, M. (2018). Review and classification of hybrid shop scheduling. *Production Engineering*, 12(5): 597–609.

Fan, K., Zhai, Y., Li, X., and Zhou, L. (2018). Simulation research on hybrid Job-Shop scheduling with multiprocessor task. *Manufacturing Automation*, 40(12): 94-99（in Chinese).

Gao, L., Li, X., Wen, X., Lu, C., and Wen, F. (2015). A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Computers & Industrial Engineering*, 88(C): 417–429.

Gröflin, H., linkert, A., andDinh, NP. (2008). Feasible job insertions in the multi-processor-task job shop. *European Journal of Operational Research*, 185: 1308–1318.

Gu, X. (2021). Application Research for Multiobjective Low-Carbon Flexible Job-Shop Scheduling Problem Based on Hybrid Artificial Bee Colony Algorithm. *IEEE Access*, 9: 135899-135914.

Jia, Z., Zhu, J., and Chen, H. (2012). Statistical Modeling-Based Image Matching Algorithm for Solder Joints of Electronic Components. *Journal of South China University of Technology (Natural Science Edition)*, 40(1): 69–76 (in Chinese).

Kennedy, J. and Eberhart, R. (1995), Particle swarm optimization. Proc of the IEEE Int Conf on Neural Networks. *Piscataway: IEEE Service Center*, 4(8): 1942–1948.

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., and Tang, L. (2022). A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. *Expert Systems with Applications*, 205: 117796.

Li, J., Huang, Y., and Wang, J. (2016).Branch Population Genetic Algorithm for Extension Dual Resource Constrained Job Shop Scheduling Problem. *The Journal of Northwestern Polytechnical University*, 34(4): 635–641 (in Chinese).

Liu, A., Yang, Y., Xing, Q., Lu, H., and Zhang, Y. (2011). Multi-population genetic algorithm in multiobjective fuzzy and flexible Job Shop scheduling. *Computer Integrated Manufacturing Systems*, 17(9): 1954-1961 (in Chinese).

Long, X., Zhang, J., Zhou, K., and Jin, T. (2022). Dynamic Self-Learning Artificial Bee Colony Optimization Algorithm for Flexible Job-Shop Scheduling Problem with Job Insertion. *Processes*, 10(3):571.

Lv, Y., Fan, K., Qu, H., and Zhou, L. (2022). Research on Multi-objective Particle Swarm Algorithm for Solving Hybrid Job-shop Scheduling. *Journal of Chinese Computer Systems*, 43(01): 218-224 (in Chinese).

Mckay, K.N., Buzacott, J.A., and Safayeni, F.R. (1989). The scheduler's knowledge of uncertainty: *The missing link. Knowledge Based Production Management System*, 171-189.

Seyyedi, M. H., Fakoor Saghih, A. M., and Naji Azimi, Z. (2021). A Fuzzy Mathematical Model for Multi-objective Flexible Job-shop Scheduling Problem with New Job Insertion and Earliness/Tardiness Penalty. *International Journal of Industrial Engineering: Theory, Applications, and Practice*, 28(3).

Soofi, P., Yazdani, m., Amiri, M., and Adibi, M. (2021). Robust Fuzzy-Stochastic Programming Model and Meta-Heuristic Algorithms for Dual-Resource Constrained Flexible Job-Shop Scheduling Problem Under Machine Breakdown. *IEEE Access*, 9: 155740-155762.

Talla Nobibon, F. and Leus, R. (2014). Complexity Results and Exact Algorithms for Robust Knapsack Problems. *Journal of Optimization Theory and Applications*, 161(2): 533–552.

Tan, C., Neoh, S., Lim, C., Hanoun, S., Wong, W., Loo, C., Zhang, L., and Nahavandi, S. (2019). Application of an evolutionary algorithm-based ensemble model to job-shop scheduling. *Journal of Intelligent Manufacturing*, 30(2): 879-890.

Tavakkoli-Moghaddam, R., Jolai, F., Vaziri, F., Ahmed, P.K., and Azaron, A. (2005). A hybrid method for solving stochastic job shop scheduling problems. *Applied Mathematics and Computation*, 170(1): 185–206.

Wang, B., Yang, X.F., and Li, Q.Y. (2012). Bad-scenario set based risk-resisting robust scheduling model. *Zidonghua Xuebao/Acta Automatica Sinica*, 38(2): 270–278 (in Chinese).

Wang, M., Fan, K., Zhai, Y., and Li, X. (2017). Study of multiprocessor task scheduling problem in network parallel computing system. *Computer Engineering and Applications*, 53(10): 264–270 (in Chinese).

Yang, Z., Wang, T., Duan, Z., and Zhang, J. (2012). Reducing forecast errors by logarithmic transformations for complex time series. *2012 2nd International Conference on Consumer Electronics，Communications and Networks (CECNet).IEEE*, 2761-2764.

Zhai, Y., Fan, K., Wang, M., and Li, X. (2018). Improved Particle Swarm Optimization Algorithm for Solving Hybrid Job-shop Scheduling with Multiprocessor Task. *Journal of Chinese Computer Systems*, 39(09): 2107–2113 (in Chinese).

Zhang, R. and Wang, R. (2009). Scenario-based stochastic capacity planning model and decision risk analysis. *System Engineering Theory and Practice*, 29(01): 55–63 (in Chinese).