Research Article

Differential Evolution Algorithm Combined with Uncertainty Handling Techniques for Stochastic Reentrant Job Shop Scheduling Problem

Rong Hu ^{1,2}, Xing Wu ^{1,3}, Bin Qian ², Jianlin Mao ², and Huaiping Jin ²

¹Faculty of Mechanical & Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China

²Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China

³Yunnan Vocational College of Mechanical and Electrical Technology, Kunming 650500, China

Correspondence should be addressed to Xing Wu; xwu@kust.edu.cn

Received 5 March 2021; Revised 11 November 2021; Accepted 18 December 2021; Published 31 January 2022

Academic Editor: Xue-bo Jin

Copyright © 2022 Rong Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper considers two kinds of stochastic reentrant job shop scheduling problems (SRJSSP), i.e., the SRJSSP with the maximum tardiness criterion and the SRJSSP with the makespan criterion. Owing to the NP-complete complexity of the considered RJSSPs, an effective differential evolutionary algorithm (DEA) combined with two uncertainty handling techniques, namely, DEA_UHT, is proposed to address these problems. Firstly, to reasonably control the computation cost, the optimal computing budget allocation technique (OCBAT) is applied for allocating limited computation budgets to assure reliable evaluation and identification for excellent solutions or individuals, and the hypothesis test technique (HTT) is added to execute a statistical comparison to reduce some unnecessary repeated evaluation. Secondly, a reentrant-largest-order-value rule is designed to convert the DEA's individual (i.e., a continuous vector) to the SRJSSP's solution (i.e., an operation permutation). Thirdly, a conventional active decoding scheme for the job shop scheduling problem is extended to decode the solution for obtaining the criterion value. Fourthly, an *Insert*-based exploitation strategy and an *Interchange*-based exploration strategy are devised to enhance DEA's exploitation ability and exploration ability, respectively. Finally, the test results and comparisons manifest the effectiveness and robustness of the proposed DEA_UHT.

1. Introduction

The job shop scheduling problem (JSSP) is one of the most well-known combinatorial optimization problems. As a variant of the JSSP, the reentrant JSSP (RJSSP) can be found in many real-life production and manufacturing environments, such as in semiconductor manufacturing enterprises [1, 2] and textile factories [3]. The main characteristic of the RJSSP is that one job may visit a set of machines more than once during the production process. Since the JSSP has NP-complete property [4] and it can be reduced to the RJSSP, the latter is also NP-complete. The studies on the RJSSP are still limited. Pan and Chen [5] presented two extended binary integer programming formulations for the RJSSP. Topaloglu and Kilincli [6] proposed a modified shifting bottleneck

heuristic for the RJSSP with makespan criterion. Chen et al. [7] presented a hybrid estimation of distribution algorithm for the RJSSP with sequence-dependent setup times to minimize makespan.

As we all know, uncertainties are very common in many real-world manufacturing scenarios. At present, some uncertain JSSPs have been studied. Most of them are the JSSP with stochastic processing times [8–22], and the rest are the JSSP with probabilistic machine availability [23], the JSSP with fuzzy processing times [24], and the JSSP with fuzzy machine availability [25]. In these studies, evolutionary algorithms are the main methodology, and makespan or tardiness/earliness is the main optimization criterion. In terms of the JSSP with stochastic processing times, how to evaluate the solution's expected objective value

(i.e., criterion value) reasonably is one of the key links in the design of the algorithm. The direct method is to sample the corresponding stochastic model multiple times to obtain multiple objective values, and then use the average of these sampling values as an approximate estimation of the expected objective value. Obviously, the more the sampling times, the more stable and reliable the estimated value is. This method is the most used method in existing research. However, such a method usually has a high computation cost. Thus, several researchers have adopted three types of techniques to control and reduce the computation cost. The first is the optimal computing budget allocation technique (OCBAT) [26]. Zhang et al. [13] and Yang et al. [16] utilized it to allocate limited computation budgets to assure more reliable evaluation for better solutions. The second is the problem conversion technique. Shen and Zhu [19] used it to convert the stochastic models of the JSPs to the deterministic ones, in which the confidence or robustness level is considered. Then, the objective value of each solution was calculated via these deterministic models. The third is the objective value prediction technique, which is mainly driven by the simulation metamodel. Ghasemi et al. [22] constructed a metamodel by using genetic programming (GP) to directly predict the solution's objective value. However, as for the uncertain RJSSPs, only Mohammadi [27] designed a simulated annealing algorithm and a genetic algorithm for the RJSSP with fuzzy processing times to minimize makespan, and the stochastic RJSSP has not been studied until now.

From the above literature review, it can be known that it is very necessary to carry out a study on the stochastic RJSSP. Hence, the RJSSP with stochastic processing time is considered here. Recently, many effective evolutionary algorithms have been successfully designed to deal with different complex optimization problems [28–37]. For example, Duan et al. [28] developed a hybrid estimation of distribution algorithm for the distributed permutation flow shop scheduling problem (FSSP) with flowline eligibility. Guo et al. [31] proposed a particle swarm optimization algorithm for the dynamic multi-objective vehicle routing problem. Sang et al. [33] presented three invasive weed optimization algorithms for the distributed assembly permutation FSSP. Guo et al. [36] devised a novel interactive preference-based multiobjective evolutionary algorithm for the optimization problem of the bolt supporting network. Zhang et al. [37] designed a matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation FSSP. Computational comparisons demonstrated that these evolutionary algorithms can obtain very satisfactory solutions with a reasonable computational time. Due to the NP-completeness of the stochastic RJSSP, an effective population-based evolutionary algorithm, i.e., differential evolution algorithm (DEA), is also selected to deal with this complex problem. DEA was firstly presented for addressing optimization problems in continuous domain [38, 39]. Because of its simple concept and quick convergence, DEA has gained many applications in a variety of fields. Nevertheless, owing to its continuous nature, the studies on DEA for the JSPs is still very limited. Thus, in this paper, an

enhanced DEA integrated with uncertainty handling techniques (DEA_UHT) is designed for solving the stochastic RJSSP under the maximum tardiness $T_{\max}$ and the stochastic RJSSP under the makespan $C_{\max}$.

The main features of DEA_UHT lie in two aspects: the reasonable uncertainty handling techniques and the effective DEA-based search. As for the uncertainty handling techniques, not only is the aforementioned OCBAT used to identify better solutions (i.e., individuals) with high confidence under limited computation cost but also the hypothesis test technique (HTT) is added to avoid the unnecessary repetitive evaluation for the poor solutions. That is, the solutions with high quality can be selected during the DEA's search process, which is helpful to guide the algorithm search faster to the truly promising regions in the stochastic solution space. As for the DEA-based search, an *Insert*-based exploitation strategy and an *Interchange*-based exploration strategy are devised to enhance DEA's exploitation ability and exploration ability, respectively. The test results and comparisons verify the effectiveness and robustness of the proposed DEA_UHT.

The remainder of the paper is organized as follows. Two kinds of stochastic RJSSPs are introduced in the next section. The basic DEA, OCBAT, and HTT are briefly described in Section 3. After presenting the solution representation, the extended decoding scheme, the revised OCBAT, and two special crossover strategies, DEA_UHT is proposed via combining DEA with OCBAT and HTT in Section 4. The test results and comparisons are given and analyzed in Section 5. Finally, concluding remarks and further study directions are summarized in Section 6.

2. Stochastic RJSSP

Consider the stochastic RJSSP that consists of n jobs and m machines. Each job should be processed on each machine L ($L > 1$) times. The processing time of each job on each machine is a random value under a certain distribution. Let $\pi = [\pi_1, \pi_2, \dots, \pi_{n \times m \times L}]$ denote the operation-based solution or sequence to be processed, π_j ($j \in \{1, \dots, n \times m \times L\}$) and $\pi_j \in \{1, \dots, n\}$ the j th job in π , $l(\pi_j)$ the repeat or reentrant times of job π_j in $[\pi_1, \pi_2, \dots, \pi_j]$, ξ the noise or uncertain factors, $p(\pi_j, k, l(\pi_j), \xi)$ the $l(\pi_j)$ th stochastic processing time of job π_j on machine k , $E(C(\pi_j, l(\pi_j), \xi))$ the $l(\pi_j)$ th expected completion time of job π_j , $d(\pi_j)$ the due date of job π_j , $E(L(\pi_j, \xi)) = E(C(\pi_j, L, \xi)) - d(\pi_j)$ the expected lateness of job π_j , $E(T(\pi_j, \xi)) = \max\{E(L(\pi_j, \xi)), 0\}$ the expected tardiness of job π_j . Here $p(\pi_j, k, l(\pi_j), \xi)$ is supposed to be subjected to a uniform distribution $U[(1 - \xi)E(P_{\pi_j, k, l(\pi_j)}), (1 + \xi)E(P_{\pi_j, k, l(\pi_j)})]$, where $P_{\pi_j, k, l(\pi_j)}$ is the expected processing time and ξ denotes noise magnitude.

Then, the expected maximum tardiness $E(T_{\max}(\pi, \xi))$ can be given as

$$E(T_{\max}(\pi, \xi)) = \max\{E(T(1, \xi)), \dots, E(T(n, \xi))\}. \quad (1)$$

The stochastic RJSSP under the maximum tardiness (SRJSSP- $T_{\max}$) can be formulated as

$$\min_{\pi \in \Pi} E(T_{\max}(\pi, \xi)), \quad (2)$$

where Π denotes the set of all possible permutations.

Similarly, the expected makespan $E(C_{\max}(\pi, \xi))$ can be given as

$$E(C_{\max}(\pi, \xi)) = \max\{E(C(\pi_1, L, \xi)), \dots, E(C(\pi_n, L, \xi))\}. \quad (3)$$

The stochastic RJSSP under the makespan (SRJSSP- $C_{\max}$) can be formulated as

$$\min_{\pi \in \Pi} E(C_{\max}(\pi, \xi)), \quad (4)$$

where Π denotes the set of all possible permutations.

Obviously, infinite sampling of the above problem model can make the objective values of equations (1) and (3) stable, but in fact it is practically impossible. In the actual calculation, the value of equations (1) or (3) is usually approximated by the average of finite sampling values (e.g., Monte Carlo simulation) [8–18]. Thus, $E(T_{\max}(\pi, \xi))$ can be estimated with

$$\overline{T_{\max}(\pi, \xi)} = \sum_{l=1}^{LL} T_{\max} \frac{(\pi, \xi)}{LL}. \quad (5)$$

and $E(C_{\max}(\pi, \xi))$ can be estimated with

$$\overline{C_{\max}(\pi, \xi)} = \sum_{l=1}^{LL} C_{\max} \frac{(\pi, \xi)}{LL}. \quad (6)$$

3. Introduction to DEA, OCBAT, and HTT

3.1. DEA. Differential evolution algorithm (DEA) [38, 39] is a kind of evolutionary algorithms for solving continuous optimization problems. The basic DEA aims at finding the global optimal solutions by using the differentiations among current individuals. Specifically, each individual's search behavior is dynamically controlled via changing the differentiation's direction and step length. At each generation, the *mutation* and *crossover* operators are applied to current individuals to generate a candidate population, and then the *selection* operator is utilized to determine a new population of the next generation. Nowadays, there are several variants/schemes of DEA.

3.2. OCBAT. OCBAT is a dynamic allocation technique of computing budget [26, 40]. Specifically, it can optimize the sampling times for each candidate solution so that the truly better solutions can be identified under a given computation cost. Denote S the total number of potential solutions, n_i^k (resp. n_b^k) the computation cost or sampling times allocated to solution i (resp. the current best solution) after the k th allocation, s_i^k (resp. s_b^k) the standard deviation among the sampling objective values of solution i (resp. the current best solution) after the k th allocation, $\bar{f}_i^k$ (resp. $\bar{f}_b^k$) the average sampling objective value of solution i (resp. the current best solution) after the k th allocation, $\delta_{b,i}^k = \bar{f}_b^k - \bar{f}_i^k$ the difference between $\bar{f}_b^k$ and $\bar{f}_i^k$ after the k th allocation, T^k the total

sampling times after the k th allocation, and T the given total sampling times.

According to the above definitions, the standard OCBAT's algorithm is provided as follows:

Step 0. Set $k = 0$, $n_i^k = n_0$ ($i = 1, 2, \dots, S$) and $T^k = \sum_{i=1}^S n_i^k$.

Step 1. If $\sum_{i=1}^S n_i^k \geq T$, the algorithm should be stopped.

Step 2. Execute n_i^k samples (i.e., simulation replications) for each solution i , and then set $T^{k+1} = T^k + \Delta$.

Step 3. Calculate each n_i^{k+1} via using $n_b^{k+1} = s_b^k$ ($\sum_{i=1, i \neq b}^S (n_i^k)^2 / (s_i^k)^2$)^{1/2} and $n_i^{k+1} / n_j^{k+1} = ((s_i^k / \delta_{b,i}^k) / (s_j^k / \delta_{b,j}^k))^2$ ($i \neq j \neq b$).

Step 4. Execute $\max\{n_i^{k+1} - n_i^k, 0\}$ samples for each solution i .

Step 5. Set $k = k + 1$, and return to Step 1.

Based on Reference [26], the recommended value ranges of n_0 and Δ are [5, 20] and [$S/5$, $S/10$], respectively.

3.3. HTT. HTT is a statistical inference method that can be used to judge whether the difference between two solutions i and j is caused by sampling error or essential difference [41]. Let the null hypothesis H_0 be " $f_i = f_j$," and then the alternative hypothesis H_1 is " $f_i \neq f_j$." Based on the statistical theory, the critical region of H_0 is defined as follows:

$$|\bar{f}_i - \bar{f}_j| \geq T_{\alpha/2} = t_{\alpha/2}(n_i + n_j - 2)$$

$$\times \sqrt{\left[\frac{(n_i - 1)s_i^2 + (n_j - 1)s_j^2}{(n_i + n_j - 2)} \right]} \times \sqrt{\left[\frac{(n_i + n_j)}{(n_i n_j)} \right]}, \quad (7)$$

where t is the t distribution and α is the evidence level. When $|\bar{f}_i - \bar{f}_j| > T_{\alpha/2}$, it can be judged that solution j is better than solutions i .

4. DEA_UHT for Stochastic RJSSP

4.1. Encoding Scheme. Since DEA's individual is a real vector, a conventional DEA cannot be directly used to deal with the discrete optimization problem. Thus, we utilize a reentrant-largest-order-value (RLOV) rule derived from the existing largest-order-value (LOV) rule [42] to transform DEA's i th individual $X_i = [x_{i,1}, x_{i,2}, \dots, x_{i,n \times m \times L}]$ to the job sequence $\pi_i = [\pi_{i,1}, \pi_{i,2}, \dots, \pi_{i,n \times m \times L}]$. Based on the RLOV rule, the elements in X_i are firstly ranked by descending order to obtain a temporary discrete sequence $\varphi_i = [\varphi_{i,1}, \varphi_{i,2}, \dots, \varphi_{i,n \times m \times L}]$. Then, the operation-based solution π_i can be generated via

$$\pi_{i,k} = \left\lceil \frac{\varphi_{i,k} - 1}{m \times L} \right\rceil + 1, \quad (8)$$

where $\lceil x \rceil$ is an operator to get the integral part of (e.g., $\lceil 2.78 \rceil = 2$). Table 1 illustrates the encoding of vector X_i in DEA for a small RJSSP ($L = 3, n = 2, m = 1$).

TABLE 1: Solution's encoding scheme.

Dimension k	1	2	3	4	5	6
$x_{i,k}$	2.16	0.33	1.51	3.67	0.86	2.79
$\varphi_{i,k}$	3	6	4	1	5	2
$\pi_{i,k}$	1	2	2	1	2	1

4.2. Decoding Scheme. Based on the characteristics of RJSSP, an optimal solution must be an active schedule under regular criteria (e.g., the maximum tardiness $T_{\max}$ and the makespan $C_{\max}$) [43]. This means that the solution's quality can be improved via transforming the semi-active schedule into the active one. Therefore, when decoding a solution to a feasible schedule, each operation should be filled in the first idle time interval in the current Gantt chart instead of the last idle time interval. This active decoding scheme is utilized in our proposed algorithm.

Let $M_{i,j,l}$ be the machine number of the j th operation that processes the l th reentrant job I , $T_{i,j,l}$ the processing time of the l th reentrant job j on machine i , and $S(\pi_{i,j})$ the possible earliest starting time of job $\pi_{i,j}$. Notice that the value of $S(\pi_{i,j})$ should meet the following inequality:

$$\max\{S(\pi_{i,j}), t_1\} + T_{M(\pi_{i,j}), \pi_{i,j}, l}(\pi_{i,j}) \leq t_2, \quad (9)$$

where $M(\pi_{i,j}) = M_{\pi_{i,j}, h(\pi_{i,j}), l(\pi_{i,j})}$, and $[t_1, t_2]$ represents any idle time interval on the machine.

Based on the above definitions, the decoding scheme that generates an active schedule for a RJSSP's solution π_i is provided as follows:

Step 1. Set $k(w) = 1$ and $l(w) = 1$, $w = 1, \dots, n$ and $j = 1$.

Step 2. If $k(\pi_{i,j}) > 1$ and $(k(\pi_{i,j}) \bmod m = 1)$, then set $l(\pi_{i,j}) = l(\pi_{i,j}) + 1$.

Step 3. Set $h(\pi_{i,j}) = k(\pi_{i,j}) - (l(\pi_{i,j}) - 1) \times m$.

Step 4. Calculate $S(\pi_{i,j})$, and set $k(\pi_{i,j}) = k(\pi_{i,j}) + 1$.

Step 5. Set $j = j + 1$. If $j \leq n \times m \times L$, then return to Step 2.

4.3. Application of Roulette Wheel Rule in OCBAT. For stochastic RJSSPs, the difference between $\delta_{b,i}^k$ and $\delta_{b,j}^k$ ($i \neq j$) is usually much larger than that between s_i and s_j . This can cause $\sum_{i=1}^S \max\{n_i^{k+1} - n_i^k, 0\} \gg \Delta$. Based on Step 4 of the standard OCBAT (see Section 3.2), the repeated times of Steps 1–5 of this standard algorithm may be much less than $(T - n_0)/\Delta$. That is, the number of times that OCBAT can be used to dynamically allocate computing budget is very limited. As a result, the reliable evaluation ability of OCBAT cannot be well utilized.

Under this situation, the so-called “most starving” rule [40] is usually used to handle this obstacle. However, this rule sets Δ to 1 and needs to execute the OCBAT's Steps 1–5 T times. This may increase the running time of the OCBAT's algorithm. Since the OCBAT decides the budget fractions instead of the number, here we utilize the “roulette wheel” rule to allocate Δ . That is, Step 4 of the standard OCBAT should be changed to the following two substeps:

Step 4.1. Calculate the allocation probability P_i^{k+1} ($P_i^{k+1} = n_i^{k+1} / \sum_{i=1}^S n_i^{k+1}$) for each solution i at $(k+1)$ th allocation.

Step 4.2. Apply the “roulette wheel” rule to allocate Δ to each solution i according to P_i^{k+1} , and update each n_i^{k+1} .

4.4. Special Exploitation and Exploration Strategies. Based on References [44, 45], the solution space of the RJSSP can be regarded as a “big valley,” where the local optima are often close to each other, and close to the global optima at the bottom of the “big valley,” and either the *Insert*-based operator or the *Interchange*-based operator can execute a more compact search. Meanwhile, DEA has been proved to be an effective algorithm to find the promising solutions or regions in the whole valley [42, 44]. As for the stochastic RJSSP, the shape and the bottom regions of the big valley are always changing. Thus, to improve our DEA_UHT's search ability, we devise an *Insert*-based exploitation strategy and an *Interchange*-based exploration strategy to replace DEA's original *mutation* and *crossover*, which cannot obtain enough promising solutions. The former strategy selects some reliable good individuals identified by OCBAT and HTT to replace some poor individuals, and executes an *Insert*-based operator on these selected individuals. The latter strategy performs an *Interchange*-based operator guided by DEA's basic scheme on all individuals except those replaced poor individuals. The detailed procedures of these strategies are provided in Step 5 in the next section.

4.5. DEA_UHT. In DEA_UHT, DEA-based search is devised according to the DE/rand-to-best/1/exp strategy to perform global search or exploration, where the base vector is the current best individual. Therefore, the information of the best individual can be shared by other individuals. By incorporating OCBAT, HTT, and two special exploitation and exploration strategies into DEA, a hybrid approach, namely, DEA_UHT, is proposed for the stochastic RJSSP.

Denote G a generation, $\text{Pop}(t)$ a population with size PS at generation t , $W_Pop(t)$ an archive set to save the individuals which need to adopt the *special Insert*-based *crossover*, $X_i(t)$ the i th individual with dimension N ($N = n$) in $\text{Pop}(t)$, $\text{Tmp_Pop}(t)$ a temporary population with size PS in generation t , $\text{Tmp_}X_i(t)$ the i th individual in $\text{Tmp_Pop}(t)$, $x_{i,k}(t)$ the k th variable/component of individual $X_i(t)$, tmp_k the k th variable of tmp , tmp01_k the k th variable of tmp01 , CR the crossover probability, F the scaling factor, $O_d(x_{i,k}(t)) = \varphi_{i,k}(t)$ the ordinal number of $x_{i,k}(t)$ in $X_i(t)$ in descending order (see Section 4.1), $RO_d(x_{i,k}(t)) = \pi_{i,k}(t)$ the $x_{i,k}(t)$'s corresponding job number in the job permutation $\pi_i(t)$ (see Section 4.1), and $\text{random}(0, 1)$ the random value in the interval $[0, 1]$. Based on these definitions, the framework of DEA_UHT is given as follows:

Step 1. Input N , $PS \geq 3$, $CR \in [0, 1]$, and let bounds be lower($x_{i,k}$) = 0 and upper($x_{i,k}$) = 4, $k = 1, \dots, N$.

Step 2. Population initialization

Step 2.1. Generate $x_{i,k}(0) = \text{lower}(x_{i,k}) + \text{random}(0, 1) * (\text{upper}(x_{i,k}) - \text{lower}(x_{i,k}))$, $k = 1, \dots, N$ for $i = 1, \dots, PS$

Step 2.2. Use the algorithm of OCBAT in Sections 3.2 and 4.3 to allocate T sampling budgets (i.e., evaluation times) for all individuals while estimating the objective values of all individuals and set $W_Pop(0) = \text{null set}$.

Step 3. Set $t = 1$ and select an individual $X_{\text{best}}(0)$ from $Pop(0)$ as bestit with the minimum objective value.

Step 4. Evolution phase (between Steps 4 and 9). Set $i = 1$.

Step 5. Mutation and crossover phase.

Step 5.1. If $X_i(t-1)$ belongs to $W_Pop(t-1)$, then execute the *Insert*-based exploitation (see Section 4.4).

Step 5.1.1. Randomly select one individual from the best five percent of $Pop(t-1)$ and set it as tmp .

Step 5.1.2. Randomly generate u and v ($u < v$), and randomly set r_i to 1 or 2.

Step 5.1.3. If $r_i = 1$, then insert tmp_u after tmp_v .

Step 5.1.4. If $r_i = 2$, then insert tmp_v before tmp_u .

Step 5.1.5. Set $\text{tmp}_X(t) = \text{tmp}$ and go to Step 6.

Step 5.2. Perform the *Interchange*-based exploration (see Section 4.4). Set the trial vector $\text{tmp} = X_i(t-1)$, $\text{tmp01} = X_i(t-1)$ and $L' = 0$. Randomly select $r1, r2 \in (1, \dots, PS)$, where $r1 \neq r2 \neq i$, and randomly select $k \in (1, \dots, N)$.

Step 5.2.1. Set $\text{tmp}_k = \text{tmp}_k + F * (\text{bestit}_k - \text{tmp}_k) + F * (x_{r1,k}(t-1) - x_{r2,k}(t-1))$, and $v1 = O_d(\text{tmp}_k)$.

Step 5.2.2. If $(RO_d(\text{tmp}_k) < RO_d(\text{tmp01}_k))$, then find $u1$ satisfying $O_d(\text{tmp01}_{u1}) = v1$ and interchange tmp01_k with tmp01_{u1} ; else randomly generate $v1$ ($(RO_d(\text{tmp01}_k) < RO_d(\text{tmp01}_{v1}))$) and interchange tmp01_k with tmp01_{v1} .

Step 5.2.3. Set $\text{tmp} = \text{tmp01}$.

Step 5.2.4. Set $k = (k+1) \bmod N$ and $L' = L' + 1$.

Step 5.2.5. If $\text{random}(0, 1) < CR$ and $(L' < 2)$, then return to Step 5.2.1.

Step 5.2.6. Set $\text{Tmp}_X(t) = \text{tmp}$.

Step 6. Set $i = i+1$. If $i \leq PS$, then return to Step 5.

Step 7. Use the algorithm of OCBAT in Sections 3.2 and 4.3 to estimate the objective values of all individuals of $\text{Tmp_pop}(t)$.

Step 8. Selection phase. Perform HHT in Section 3.3 on $\text{Tmp}_X(t)$ with $X_i(t-1)$, $i = 1, \dots, PS$.

Step 8.1. If the null hypothesis holds and $\bar{J}(X_i(t-1)) - \bar{J}(\text{Tmp}_X(t)) > \tau$, then set a Boolean variable $b_replace = \text{true}$; else set $b_replace = \text{false}$.

Step 8.2. If $b_replace = \text{true}$ then set $X_i(t) = \text{Tmp}_X(t)$; else set $X_i(t) = X_i(t-1)$.

Step 8.3. If $b_replace = \text{true}$ and $\bar{J}(\text{Tmp}_X(t)) < \bar{J}(\text{bestit})$, then set $\text{bestit} = \text{Tmp}_X(t)$.

Step 8.4. If $b_replace = \text{false}$ and $X_i(t-1)$ is in the worst 20 percent of $Pop(t-1)$, then add $X_i(t-1)$ to $W_Pop(t)$.

Step 9. Set $t = t+1$. If $t \leq t_max$ (the maximum generation), then return to Step 4.

Step 10. Output π_{output} ($\pi_{\text{output}} = \text{bestit}$) and its objective value.

From the above procedure, it can be known that DEA_UHT not only utilizes OCBAT to ensure reliable evaluation for good solutions but also adopts HTT to reduce repetitive search. Furthermore, the DEA-based search combined with two special strategies is devised to effectively execute exploration and exploitation in solution space. Meanwhile, “ $(L' < 2)$ ” in Step 5.2.5 is used to control the search range of the *Interchange*-based exploration. In addition, it is worth noting that although Step 5.1 (resp. Step 5.2) directly performs the *Insert* (resp. *Interchange*) operation on the individual, it is equivalent to performing the same *Insert* (resp. *Interchange*) operation on the corresponding permutation of this individual (see the RLOV rule in Section 4.1).

5. Simulation Results and Comparisons

5.1. Experimental Setup. To verify DEA_UHT's performance, ten instances of the deterministic RJSSP are created via the following steps:

Step 1. Select one out of ten JSSP's classic benchmarks [46] in turn as *PP*

(i.e., FT06, FT10, FT20, LA01, LA06, LA11, LA16, LA21, LA26, and LA31 [46]).

Step 2. Set the reentrant times $L = 2$.

Step 3. Set each $M_{i,j,1}$ and each $T_{i,j,1}$ to the corresponding data in *PP*.

Step 4. Set $M_{i,j,2} = M_{i+1,j,1}$ for $i = 1, \dots, n-1$ and $j = 1, \dots, m$.

Step 5. Set $M_{n,j,2} = M_{1,j,1}$ for $j = 1, \dots, m$.

Step 6. Set $T_{i,j,2} = T_{i,j+1,1}$ for $j = 1, \dots, n-1$ and $i = 1, \dots, m$.

Step 7. Set $T_{i,n,2} = T_{i,1,1}$ for $i = 1, \dots, m$.

Using the above steps, ten RJSSP's instances can be created. These instances are named FT06_R2, FT10_R2, FT20_R2, LA01_R2, LA06_R2, LA11_R2, LA16_R2, LA21_R2, LA26_R2, and LA31_R2, respectively. Each of these names is formed by adding “_R2” after the name of the corresponding JSSP's benchmark. Obviously, these RJSSP's instances are obtained by simply extending the classical JSSP's instances. Since each $M_{i,j,1}$ and $T_{i,j,1}$ are randomly generated between $[1, n]$ and $[1, 100]$, respectively, the internal correlation within each RJSSP's instance can be ignored. Using the above steps to obtain the RJSSP's instances is just for the convenience of other researchers to generate the same instances.

TABLE 2: Comparisons of PDEA, DEA_UHT_O, and DEA_UHT.

Problem	n, m, L	ξ	PDEA		DEA_UHT_O		DEA_UHT	
			BRE	ARE	BRE	ARE	BRE	ARE
S1_FT06_R2_ $T_{\max}$	6, 6, 2	0.3	17.00	22.53	16.00	21.37	11.00	15.13
S1_LA01_R2_ $T_{\max}$	10, 5, 2	0.3	409.00	479.07	429.00	469.87	282.00	369.27
S1_LA06_R2_ $T_{\max}$	15, 5, 2	0.3	848.00	924.57	777.00	902.17	614.00	760.17
S1_LA11_R2_ $T_{\max}$	20, 5, 2	0.3	1299.00	1374.03	1244.00	1341.90	1071.00	1179.30
S1_FT10_R2_ $T_{\max}$	10, 10, 2	0.3	579.00	721.17	537.00	694.57	199.00	363.57
S1_FT20_R2_ $T_{\max}$	20, 5, 2	0.3	1429.00	1518.73	1371.00	1489.83	955.00	1090.87
S1_LA16_R2_ $T_{\max}$	10, 10, 2	0.3	558.00	621.13	485.00	585.07	221.00	315.97
S1_LA21_R2_ $T_{\max}$	15, 10, 2	0.3	964.00	1128.67	1011.00	1075.13	407.00	538.00
S1_LA26_R2_ $T_{\max}$	20, 10, 2	0.3	1436.00	1604.73	1449.00	1559.93	847.00	949.57
S1_LA31_R2_ $T_{\max}$	30, 10, 2	0.3	2469.00	2611.97	2300.00	2546.90	1573.00	1848.00
S1_FT06_R2_ $T_{\max}$	6, 6, 2	0.6	19.00	25.80	17.00	22.47	10.00	16.37
S1_LA01_R2_ $T_{\max}$	10, 5, 2	0.6	464.00	539.60	414.00	487.43	307.00	379.43
S1_LA06_R2_ $T_{\max}$	15, 5, 2	0.6	872.00	947.50	810.00	915.33	626.00	780.53
S1_LA11_R2_ $T_{\max}$	20, 5, 2	0.6	1304.00	1418.83	1289.00	1366.57	1170.00	1252.27
S1_FT10_R2_ $T_{\max}$	10, 10, 2	0.6	625.00	765.93	534.00	698.93	176.00	375.43
S1_FT20_R2_ $T_{\max}$	20, 5, 2	0.6	1477.00	1586.27	1403.00	1517.70	1043.00	1199.67
S1_LA16_R2_ $T_{\max}$	10, 10, 2	0.6	503.00	675.83	499.00	617.30	244.00	370.60
S1_LA21_R2_ $T_{\max}$	15, 10, 2	0.6	1035.00	1172.10	1002.00	1100.77	557.00	685.80
S1_LA26_R2_ $T_{\max}$	20, 10, 2	0.6	1465.00	1663.93	1407.00	1559.27	928.00	1033.50
S1_LA31_R2_ $T_{\max}$	30, 10, 2	0.6	2560.00	2689.83	2405.00	2579.50	1761.00	1993.90
Average			1016.60	1124.61	969.95	1077.60	650.10	775.87

For any created instance, each job's due date can be obtained via the following steps:

Step 1. Randomly created a permutation $\pi = [\pi_1, \pi_2, \dots, \pi_{n \times m \times L}]$ for each RJSSP's instance z .

Step 2. Calculate each $C_z(\pi_j, L)$ ($\pi_j \in \pi$) via the decoding scheme in Section 4.2.

Step 3. Specify each job's due date via

$$d_z(\pi_j) = C_z(\pi_j, L) + \text{random} \left[\frac{-C_z(\pi_j, L)}{10}, \frac{C_z(\pi_j, L)}{10} \right], \quad (10)$$

where $\text{random}[-C_z(\pi_j, L)/10, C_z(\pi_j, L)/10]$ is a random value between $[-C_z(\pi_j, L)/10, C_z(\pi_j, L)/10]$. Because the ability to address tight due dates affects an enterprise's competence and the problem under tight due dates is more difficult to solve than under loose due dates, (10) is designed to obtain a tight due date.

Next, the instances of the stochastic RJSSP can be obtained by changing each original processing time $p(\pi_j, k, l(\pi_j))$ of the above RJSSP's instances to a uniform distribution $U[[(1 - \xi)E(P_{\pi_j, k, l(\pi_j)}), (1 + \xi)E(P_{\pi_j, k, l(\pi_j)})]]$, where $E(P_{\pi_j, k, l(\pi_j)})$ is the expected processing time that equals to the original $p(\pi_j, k, l(\pi_j))$ and ξ denotes noise magnitude. To test the performance of the algorithm more comprehensively, two kinds of stochastic RJSSPs are considered. The first kind is the stochastic RJSSP under $T_{\max}$. Ten instances with $\xi = 0.3$ and them with $\xi = 0.6$ are generated. These instances are denoted by S1_Iname_R2_ $T_{\max}$, where Iname indicates that S1_Iname_R2_ $T_{\max}$ is derived from the deterministic instance Iname. The second kind is the stochastic RJSSP under $C_{\max}$,

whose ten instances with $\xi = 0.6$ are generated. Similarly, these instances are denoted by S1_Iname_R2_ $C_{\max}$. Moreover, since it is a common scenario that only a part of the machines are unstable, a variant of the first kind of problem is also considered. This variant, namely, S2_Iname_R2_ $T_{\max}$, only sets the processing time of each job on machine one and machine three to a uniform distribution $U[[(1 - \xi)E(P_{\pi_j, k, l(\pi_j)}), (1 + \xi)E(P_{\pi_j, k, l(\pi_j)})]]$ ($k = 1, 3$ and $\xi = 0.6$).

The proposed DEA_UHT and the other compared algorithms were implemented in Delphi 2010 and the experiments were executed on an Intel 3.0 GHz PC with 8 GB memory. The main parameters of DEA_UHT are set as follows: the population size $PS = 100$, $F = 0.7$, $CR = 0.1$, $T = 1000$, $n_0 = 5$, and $\Delta = 10$. The maximum generation of each compared algorithm is set to 300. Each compared algorithm is executed thirty times independently on each instance.

To show the effectiveness of DEA_UHT, two variants of it are compared. The abbreviations of these variants are given as follows:

- (i) PDEA: DEA_UHT without OCBAT, HTT, the *Insert*-based exploitation strategy and the *Inter-change*-based exploration strategy. The details can be described as follows:
 - (1) *DEA_UHT without OCBAT*. At Steps 2.2 and 7 of DEA_UHT, the algorithm of OCBAT is replaced with twenty evaluation times for each solution's performance estimation.
 - (2) *DEA_UHT without HTT*. At Step 8 of DEA_UHT, the HTT-based selection is replaced with the normal DE's *selection*.
 - (3) DEA_UHT without the *Insert*-based exploitation strategy: Step 5.1 is deleted.

TABLE 3: Comparisons of PDEA, DEA_UHT_O, and DEA_UHT on two other kinds of stochastic RJSSP.

Problem	n, m, L	ξ	PDEA		DEA_UHT_O		DEA_UHT	
			BRE	ARE	BRE	ARE	BRE	ARE
S2_FT06_R2_ $T_{\max}$	6, 6, 2	0.6	19.00	23.77	<i>17.00</i>	<i>21.60</i>	10.00	15.13
S2_LA01_R2_ $T_{\max}$	10, 5, 2	0.6	<i>414.00</i>	485.80	423.00	470.63	282.00	371.93
S2_LA06_R2_ $T_{\max}$	15, 5, 2	0.6	862.00	937.60	<i>811.00</i>	<i>907.67</i>	650.00	781.43
S2_LA11_R2_ $T_{\max}$	20, 5, 2	0.6	1319.00	1387.10	<i>1299.00</i>	<i>1339.53</i>	1136.00	1243.43
S2_FT10_R2_ $T_{\max}$	10, 10, 2	0.6	525.00	718.60	<i>512.00</i>	<i>681.37</i>	210.00	316.20
S2_FT20_R2_ $T_{\max}$	20, 5, 2	0.6	<i>1381.00</i>	1528.30	1441.00	<i>1512.57</i>	1022.00	1192.00
S2_LA16_R2_ $T_{\max}$	10, 10, 2	0.6	521.00	629.07	<i>475.00</i>	<i>596.47</i>	262.00	362.90
S2_LA21_R2_ $T_{\max}$	15, 10, 2	0.6	989.00	1133.57	<i>963.00</i>	<i>1101.43</i>	411.00	569.77
S2_LA26_R2_ $T_{\max}$	20, 10, 2	0.6	1421.00	1589.57	<i>1418.00</i>	<i>1532.90</i>	855.00	981.10
S2_LA31_R2_ $T_{\max}$	30, 10, 2	0.6	2371.00	2588.87	<i>2288.00</i>	<i>2542.00</i>	1758.00	1910.33
S1_FT06_R2_ $C_{\max}$	6, 6, 2	0.6	108.00	115.87	<i>107.00</i>	<i>112.50</i>	102.00	108.70
S1_LA01_R2_ $C_{\max}$	10, 5, 2	0.6	<i>1407.00</i>	1499.43	<i>1406.00</i>	<i>1452.53</i>	1332.00	1375.33
S1_LA06_R2_ $C_{\max}$	15, 5, 2	0.6	1913.00	2031.70	<i>1908.00</i>	<i>1959.97</i>	1852.00	1877.40
S1_LA11_R2_ $C_{\max}$	20, 5, 2	0.6	2555.00	2652.23	<i>2525.00</i>	<i>2610.67</i>	2444.00	2476.73
S1_FT10_R2_ $C_{\max}$	10, 10, 2	0.6	<i>2186.00</i>	2355.87	2196.00	<i>2299.10</i>	1961.00	2045.67
S1_FT20_R2_ $C_{\max}$	20, 5, 2	0.6	<i>2678.00</i>	2829.07	<i>2692.00</i>	<i>2782.90</i>	2490.00	2572.93
S1_LA16_R2_ $C_{\max}$	10, 10, 2	0.6	<i>2136.00</i>	2271.33	2142.00	<i>2241.23</i>	1864.00	2004.90
S1_LA21_R2_ $C_{\max}$	15, 10, 2	0.6	2871.00	2963.77	<i>2769.00</i>	<i>2863.10</i>	2284.00	2485.43
S1_LA26_R2_ $C_{\max}$	20, 10, 2	0.6	3440.00	3592.53	<i>3380.00</i>	<i>3485.20</i>	2843.00	2982.27
S1_LA31_R2_ $C_{\max}$	30, 10, 2	0.6	<i>4408.00</i>	4694.07	<i>4465.00</i>	<i>4585.60</i>	3871.00	4071.83
Average			1676.20	1801.41	<i>1661.85</i>	<i>1754.95</i>	1381.95	1487.27

FIGURE 1: Violin plots for comparison results in Table 1. (a) PDEA vs DEA_UHT. (b) DEA_UHT_O vs DEA_UHT.

- (4) DEA_UHT without the *Interchange*-based exploitation strategy: Steps 5.2.2 and 5.2.3 are deleted, and “($L' < 2$)” in Step 5.2.5 is replaced with “($L' < N$).”
- (2) DEA_UHT_O. DEA_UHT without HTT, the *Insert*-based exploitation strategy and the *Interchange*-based exploration strategy.

5.2. *Simulation Results and Comparisons.* The statistical results are summarized in Tables 2 and 3, where ξ is the noise

magnitude, *BRE* is the best expected maximum tardiness or makespan, and *ARE* is the average expected maximum tardiness or makespan. To make the statistical results clearer, in Tables 2 and 3, the best and the second-best values in each row are represented by using the bold and the italic fonts, respectively. For the final output solution π_{output} obtained by each compared algorithm at each run, its expected objective value $E(T_{\max}(\pi_{\text{output}}, \xi))$ or $E(C_{\max}(\pi_{\text{output}}, \xi))$ is calculated with the expected processing time (i.e., the processing time without noise). Obviously, *BRE* and *ARE* can reflect the performance of these compared algorithms in a statistical

FIGURE 2: Violin plots for comparison results in Table 2. (a) PDEA vs DEA_UHT. (b) DEA_UHT_O vs DEA_UHT.

FIGURE 3: Computing budget allocation of OCBAT in DEA_UHT for $S1_LA11_R2_{T_{\max}}$ (generation $t = 20$).

sense. Moreover, to further show the statistical distributions of the results in Tables 1 and 2, the violin plots for the algorithm pairs “PDEA vs. DEA_UHT” and “DEA_UHT_O vs. DEA_UHT” are given in Figures 1 and 2.

From Tables 2 and 3, it is clear that both the *BRE* values and the *ARE* values obtained by DEA_UHT are better than those obtained by PDEA, which validates the effectiveness via incorporating OCBAT, HTT, the *Insert*-based exploitation strategy, and the *Interchange*-based exploration strategy into PDEA. This also means that DEA_UHT can achieve a better balance between exploration and exploitation than PDEA and DEA_UHT_O. Meanwhile, most of the *BRE* values and all of the *ARE* values obtained by DEA_UHT_O are better than those obtained by PDEA,

from which it can be concluded that OCBAT is necessary to improve the optimization performance. It is worth noting that the above conclusions can still be drawn when considering the instances in Tables 1 and 2, where $\xi = 0.1$ or 0.3 . In addition, from Figures 1 and 2, it can be seen that DEA_UHT has better statistical distributions and better median lines compared with PDEA and DEA_UHT_O. These violin plots confirm the competitiveness of DEA_UHT.

To show the role of OCBAT, the computing budget allocation of OCBAT in DEA_UHT for the middle-size instance $S1_LA11_R2$ is provided in Figure 3. In addition, to illustrate the effectiveness of DEA_UHT, the convergence curves of the compared algorithms on the same instance are

FIGURE 4: Convergence curves of the compared algorithms on S1_LA11_R2_ $T_{\max}$ (noise magnitude = 0.6).

given in Figure 4, where the curves are formed by connecting the $E(T_{\max}(\pi, \xi))$'s value of the best individual *bestit* at each generation of the corresponding algorithm. $E(T_{\max}(\pi, \xi))$ here is calculated by setting $\xi = 0$ (i.e., $E(T_{\max}(\text{bestit}, \xi)) = T_{\max}(\text{bestit})$). The computing budget allocation and the convergence curves for other instances are similar as those in Figures 3 and 4.

From Figure 3, it is shown that OCBAT can intelligently assign more evaluation times to better individuals. That is to say, OCBAT can enhance the algorithm's ability to find more reliable high-quality solutions or regions at each generation under the same computing budget. From Figure 4, it can be seen that the convergence speed of DEA_UHT is obviously faster than those of the other compared algorithms. This manifests that DEA_UHT can guide the whole population to move to the bottom of the expected "big valley" (i.e., the bottom of the "big valley" without noise) in solution space faster, and thus has better search performance. Moreover, the $E(T_{\max}(\pi, \xi))$'s values of DEA_UHT_O are less than those of PEDA in most cases, which verifies that OCBAT helps to guide the search to really promising regions in noisy environment.

6. Conclusion and Future Research

This paper proposed a novel DEA_UHT by combining DEA with OCBAT and HTT to address the stochastic RJSSP. This kind of problem has many real-world applications in modern enterprises. To the best of the authors' knowledge, this is the first time that DEA has been applied to such an important problem. With the help of OCBAT to intelligently allocate limited sampling budgets to perform reliable identification on excellent or good individuals, and with the help of HTT to reserve good individuals with high confidence, as well as with the *Insert*-based and *Interchange*-based strategies to effectively execute enhanced exploitation and

exploration, DEA_UHT is of good performance for different kinds of stochastic RJSSP. In the future, our work is to extend DEA_UHT to other kinds of stochastic scheduling problems.

Data Availability

Data were curated by the authors and are available upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (61963022 and 62173169) and the Basic Research Key Project of Yunnan Province (202101AS070097).

References

- [1] E. Levner, V. Kats, D. Alcaide López de Pablo, and T. C. E. Cheng, "Complexity of cyclic scheduling problems: a state-of-the-art survey," *Computers & Industrial Engineering*, vol. 59, no. 2, pp. 352–361, 2010.
- [2] A. K. Gupta and A. I. Sivakumar, "Job shop scheduling techniques in semiconductor manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 27, no. 11, pp. 1163–1169, 2006.
- [3] S. Topaloglu and G. Kilincli, "A new bottleneck-based heuristic for reentrant job shops: a case study in a textile factory," in *Operations Research Proceedings 2008*, Springer, Berlin, Germany, pp. 159–164, 2009.
- [4] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

- [5] J. C.-H. Pan and J.-S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," *Computers & Operations Research*, vol. 32, no. 5, pp. 1197–1212, 2005.
- [6] S. Topaloglu and G. Kilincli, "A modified shifting bottleneck heuristic for the reentrant job shop scheduling problem with makespan minimization," *International Journal of Advanced Manufacturing Technology*, vol. 44, no. 7, pp. 781–794, 2009.
- [7] S.-F. Chen, B. Qian, B. Liu, R. Hu, and C.-S. Zhang, "Bayesian statistical inference-based estimation of distribution algorithm for the re-entrant job-shop scheduling problem with sequence-dependent setup times," *Intelligent Computing Methodologies*, vol. 8589, pp. 686–696, 2014.
- [8] D. Golenko-Ginzburg, S. Kesler, and Z. Landsman, "Industrial job-shop scheduling with random operations and different priorities," *International Journal of Production Economics*, vol. 40, no. 2, pp. 185–195, 1995.
- [9] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *Journal of Mathematical Analysis and Applications*, vol. 355, no. 1, pp. 63–81, 2009.
- [10] J. Gu, M. Gu, C. Cao, and X. Gu, "A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem," *Computers & Operations Research*, vol. 37, no. 5, pp. 927–937, 2010.
- [11] D. Lei, "Simplified multi-objective genetic algorithms for stochastic job shop scheduling," *Applied Soft Computing*, vol. 11, no. 8, pp. 4991–4996, 2011.
- [12] S.-C. Horng, S.-S. Lin, and F.-Y. Yang, "Evolutionary algorithm for stochastic job shop scheduling with random processing time," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3603–3610, 2012.
- [13] R. Zhang, S. Song, and C. Wu, "A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem," *Knowledge-Based Systems*, vol. 27, pp. 393–406, 2012.
- [14] M. van den Akker, K. van Blokland, and H. Hoogeveen, "Finding robust solutions for the stochastic job shop scheduling problem by including simulation in local search," in *Experimental Algorithms*, vol. 7933, pp. 402–413, Springer, Berlin, Germany, 2013.
- [15] X. Hao, L. Lin, M. Gen, and K. Ohno, "Effective estimation of distribution algorithm for stochastic job shop scheduling problem," *Procedia Computer Science*, vol. 20, pp. 102–107, 2013.
- [16] H.-A. Yang, Y. Lv, C. Xia, S. Sun, and H. Wang, "Optimal computing budget allocation for ordinal optimization in solving stochastic job shop scheduling problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 619254, 10 pages, 2014.
- [17] P. Sharma and A. Jain, "Performance analysis of dispatching rules in a stochastic dynamic job shop manufacturing system with sequence-dependent setup times: simulation approach," *CIRP Journal of Manufacturing Science and Technology*, vol. 10, pp. 110–119, 2015.
- [18] S. Kemmoe-Tchomte, D. Lamy, and N. Tchernev, "A meta-heuristic based on simulation for stochastic Job-shop optimization," in: IEEE, in *Proceedings of the 2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pp. 108–116, October 2015.
- [19] J. Shen and Y. Zhu, "Chance-constrained model for uncertain job shop scheduling problem," *Soft Computing*, vol. 20, no. 6, pp. 2383–2391, 2016.
- [20] X. Hao, M. Gen, L. Lin, and G. A. Suer, "Effective multi-objective EDA for bi-criteria stochastic job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 28, no. 3, pp. 833–845, 2017.
- [21] S. Shoval and M. Efatmaneshnik, "A probabilistic approach to the stochastic job-shop scheduling problem," *Procedia Manufacturing*, vol. 21, pp. 533–540, 2018.
- [22] A. Ghasemi, A. Ashoori, and C. Heavey, "Evolutionary learning based simulation optimization for stochastic job shop scheduling problems," *Applied Soft Computing*, vol. 106, pp. 1–18, Article ID 107309, 2021.
- [23] Y.-K. Lin, P.-C. Chang, L. C.-L. Yeng, and S.-F. Huang, "Bi-objective optimization for a multistate job-shop production network using NSGA-II and TOPSIS," *Journal of Manufacturing Systems*, vol. 52, pp. 43–54, 2019.
- [24] D. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 37, no. 1–2, pp. 157–165, 2008.
- [25] S. M. K. Hasan, R. Sarker, and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns," *International Journal of Production Research*, vol. 49, no. 16, pp. 4999–5015, 2011.
- [26] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems*, vol. 10, no. 3, pp. 251–270, 2000.
- [27] G. Mohammadi, "Scheduling parallel machines with sequence dependent set-up times in job shop industries using GA and SA," *International Journal of Systems Science: Operations & Logistics*, vol. 5, no. 3, pp. 282–294, 2018.
- [28] W. Z. Duan, Z. Y. Li, M. C. Ji, Y. X. Yang, S. Y. Wang, and B. Liu, "A hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling with flowline eligibility," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 2581–2587, Vancouver, Canada, July 2016.
- [29] W. Z. Duan, Z. Y. Li, Y. X. Yang, B. Liu, and K. Y. Wang, "EDA based probabilistic memetic algorithm for distributed blocking permutation flowshop scheduling with sequence dependent setup time," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 992–999, Donostia, Spain, June 2017.
- [30] B. Qian, Z.-c. Li, and R. Hu, "A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem," *Applied Soft Computing*, vol. 61, pp. 921–934, 2017.
- [31] Y.-N. Guo, J. Cheng, S. Luo, D. Gong, and Y. Xue, "Robust dynamic multi-objective vehicle routing optimization method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1891–1903, 2018.
- [32] Y. Guo, H. Yang, M. Chen, J. Cheng, and D. Gong, "Ensemble prediction-based dynamic robust multi-objective optimization methods," *Swarm and evolutionary computation*, vol. 48, pp. 156–171, 2019.
- [33] H.-Y. Sang, Q.-K. Pan, J.-Q. Li et al., "Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion," *Swarm and evolutionary computation*, vol. 44, pp. 64–73, 2019.
- [34] Z. C. Li, B. Qian, R. Hu, L. L. Chang, and J. B. Yang, "An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups," *Knowledge-Based Systems*, vol. 173, pp. 83–112, 2019.

- [35] Y. Han, J. Li, H. Sang, Y. Liu, K. Gao, and Q. Pan, "Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time," *Applied Soft Computing*, vol. 93, Article ID 106343, 2020.
- [36] Y.-N. Guo, X. Zhang, D.-W. Gong, Z. Zhang, and J.-J. Yang, "Novel interactive preference-based multiobjective evolutionary optimization for bolt supporting networks," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 750–764, 2020.
- [37] Z. Q. Zhang, B. Qian, R. Hu, H. P. Jin, L. Wang, and J. B. Yang, "A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 60, pp. 1–22, Article ID 100785, 2021.
- [38] R. Storn and K. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [39] V. Feoktistov, *Differential Evolution: In Search of Solutions*, Springer, Berlin, Germany, 2006.
- [40] C.-H. Chen, S. E. Chick, L. H. Lee, N. A. Pujowidianto, and A. Pujowidianto, "Ranking and Selection: Efficient Simulation Budget Allocation," *Handbook Of Simulation Optimization*, Springer, Berlin, Germany, pp. 45–80, 2015.
- [41] X. Liu, *Introduction to Statistics*, Tsinghua university Press, Beijing, China, 1998.
- [42] B. Qian, L. Wang, R. Hu, W. L. Wang, D. X. Huang, and X. Wang, "A hybrid differential evolution for permutation flow-shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 757–777, 2008.
- [43] L. Wang, *Shop Scheduling with Genetic Algorithms*, Tsinghua University & Springer Press, Beijing, China, 2003.
- [44] B. Qian, L. Wang, D. X. Huang, and X. Wang, "Scheduling multi-objective job shops using a memetic algorithm based on differential evolution," *International Journal of Advanced Manufacturing Technology*, vol. 35, no. 9-10, pp. 1014–1027, 2008.
- [45] T. Schiavinotto and T. Stützle, "A review of metrics on permutations for search landscape analysis," *Computers & Operations Research*, vol. 34, no. 10, pp. 3143–3153, 2007.
- [46] S. Lawrence, *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques*, Graduate school of industrial administration, Carnegie Mellon University, Pittsburgh, PA, USA, 1984.