Int j simul model 18 (2019) 3, 500-509
ISSN 1726-4529 Original scientific paper

A MULTI-OBJECTIVE HYBRID DIFFERENTIAL OPTIMIZATION ALGORITHM FOR FLOW-SHOP SCHEDULING PROBLEM

Pei, J. Y. & Shan, P.[#]
School of Business, Jiangnan University, Wuxi 214122, China
E-Mail: shanp@jiangnan.edu.cn ([#] Corresponding author)

Abstract
This paper puts forward a multi-objective hybrid difference optimization algorithm to solve multi-objective flow-shop scheduling problem (FSP). The hybrid algorithm inherits the merits of differential evolution vector operation, and makes dynamic adjustments to the search direction based on historical data. However, the basic differential evolution algorithm is prone to the local optimum trap, due to the low population diversity in the later stage of evolution. To solve the problem, a hybrid sampling strategy was introduced obtain the distribution information of solution sets and to design the mutation operator of differential evolution, thus improving the convergence of the hybrid algorithm. Finally, our algorithm was applied to solve FSPs through simulation. The simulation results show that our algorithm greatly outperformed the basic multi-objective evolutionary algorithm in convergence and distribution performance.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Flow-Shop Scheduling Problem (FSP), Multi-Objective Optimization, Hybrid Differential Evolution, Genetic Algorithms (GA)

1. INTRODUCTION

The job-shop scheduling problem (JSP) attempts to achieve a single objective or multiple objectives (e.g. maximum number of jobs and minimum cost), under certain constraints (e.g. completion time and processing sequence) through rational allocation of the limited resources of the job-shop. The flow-shop [1, 2] is a job-shop capable of automatic adjustment and adaptive management of jobs. It consists of linkage information control, raw material transport, and smart management. Therefore, the flow-shop scheduling problem (FSP) is a combinatorial optimization problem with high complexity and multiple constraints. It is difficult to obtain the optimal solution of the FSP.

In actual job-shops, the decision-maker usually selects one or more orders in the light of the resource constraint and the consumer demand, aiming to minimize the time consumption or enhance machine efficiency. Therefore, if some constraints are known, a complex scheduling problem can be greatly simplified. The mathematical models for real-world optimization problems must have various constraints. The general form of constrained optimization can be described as:

$$\min f_m(x), x \in R^n \tag{1}$$

s.t.

$$g_i(x) \le 0,\ \ i = 1, \dots, q$$
$$h_j(x) = 0,\ \ j = q+1, \dots, k$$
$$L_t \le x_t \le U_t,\ \ t = 1, \dots, n$$

where, L_t and U_t are the lower and upper bounds of variable x_t, it can form search space S^2. It needs to satisfy q inequality constraints (linear or non-linear) and k-q equality constraints.

This paper puts forward a multi-objective hybrid differential evolution algorithm to solve the FSP with multiple objectives. Firstly, the distribution of solution sets was obtained using

hybrid sampling strategy, which considers the changes of minimizing the completion time and the machine load. During differential evolution, several mutation operators were designed to select alternative individuals in varied ways, thereby guiding the search direction, speeding up convergence and improving distribution performance.

2. LITERATURE REVIEW

The FSP, as a classical JSP, has been extensively studied in the field of real production. It is also known as permutation FSP. The main constraint on the problem is that the operations of a job should be implemented on different machines, but following the same sequence. The existing algorithms for the FSP fall into two categories: optimization algorithms based purely on mathematics [3] and swarm intelligent optimization algorithms inspired by bionics [4, 5].

Intelligent optimization algorithms combine the merits of natural learning and artificial intelligence. Both single-objective optimization problems (e.g. function optimization and wave crest search) and multi-objective ones can be solved satisfactorily by these algorithms. In particular, intelligent optimization algorithms converge quickly to the near-optimal or optimal solution set of multi-objective JSPs. As a result, many scholars have applied intelligent optimization algorithms to solve the FSP.

The most popular intelligent optimization algorithms include the genetic algorithm (GA) [6-8], the simulated annealing (SA) algorithm [9], the tabu search (TS) [10], the neural network (NN) [11] and the particle swarm optimization (PSO) [12]. Each of them has its own strengths. However, a single intelligent optimization algorithm often shows different degrees of inadequacy in handling different types of problems. To overcome the defect, the design of hybrid algorithms has emerged as a new hotspot. The hybrid algorithm is generally created by fusing different algorithms into the basic framework, and thus features the advantages of multiple types of algorithms.

Many scholars have introduced evolutionary algorithms to solve the FSP, because the problem has multiple constraints and objectives [13-15]. For example, Gao et al. [16] proposed two constructive heuristic methods embedded with local search and modified iteration operators, and combined them into a hybrid heuristic algorithm that can minimize the makespan under constraints. Focusing on the FSP with grouped jobs, Sun et al. [17] relied on fitness calculation and local search to maintain the population diversity and convergence, respectively. Drawing on the structure of food chain and the principle of survival of the fittest, Nabipoor Afruzi et al. [18] improved the non-dominated sorting genetic algorithm II (NSGA-II), and enhanced the convergence performance through radical competition mode.

The differential evolution [19] has now been extended to multi-objective optimization problems. This heuristic global search algorithm executes the mutation operation before crossover. The floating-point encoding blesses the algorithm with high global search efficiency in continuous space. For the maximal makespan, Lwin and Qu [20] designed a guidance agent based on probabilistic model for adaptive differential evolution with neighbourhood search, which guides the child chromosome selection of differential evolution algorithm. Shao and Pi [21] constructed a distribution model with an adaptive scaling factor, counted the dominant individuals in population evolution, and changed the selection mode of self-suitability, thereby avoiding the local optimum trap. Based on probability model, Soltani and Karimi [22] used the distribution estimation algorithm to keep the differential evolution algorithm away from the local optimum trap, and solve the mixed zero-idle permutation FSP.

3. MULTI-OBJECTIVE HYBRID DIFFERENTIAL OPTIMIZATION BASED ON HYBRID SAMPLING

3.1 Basic differential evolution algorithm

Differential evolution is a population-based evolutionary algorithm. In the algorithm, the nondeterministic polynomial (NP) real parameter vectors are taken as the population of each generation. Each vector, also known as a chromosome, forms a candidate solution to the optimization problem aiming to minimize or maximize $f(\vec{x})$, where $\vec{X} = [x_1, x_2, \dots, x_D]^T$ is a vector of *D*-dimensional decision variables. For each decision variable, the lower and upper bounds are denoted as $x_{j,min}$ and $x_{j,max}$, respectively.

The initial aim of the basic differential evolution algorithm is to set control parameters and initialize the population. The algorithm parameters include population size (*NP*), scale factor (*F*) and crossover probability (*Cr*). The quality of the parameter set directly bears on the convergence speed to the global near-optimal or optimal solution set.

Let $G = 0, 1, \dots, G_{max}$ be the generations. Then, the i^{th} vector of the population can be described as:

$$\overrightarrow{X_{i,G}} = [x_{1,i,G}, x_{2,i,G}, \dots, x_{D,i,G}]^T \tag{2}$$

The initial population of vectors ($G = 0$) is randomly generated to cover as much space in the search space as possible. Hence, the j^{th} component of the i^{th} vector can be initialized as:

$$x_{j,i,0} = x_{j,min} + (x_{j,max} - x_{j,min}) \times rand(0,1) \tag{3}$$

where, *rand* (0, 1) is a random function, which returns a real number evenly distributed in [0, 1].

During the evolution process, mutation can be regarded as disturbance to individuals. In the basic differential evolution algorithm, the current parent vector is called the target vector. The mutation vector $V_{i,G} = [v_{1,i,G}, v_{2,i,G}, \dots, v_{D,i,G}]^T$ constructed by mutation operator is called the composite vector, which corresponds to the i^{th} target vector $X_{i,G}$. The details on the mutation operation are as follows:

$$V_{i,G} = X_{best,G} + F \times (X_{r_1^j,G} - X_{r_2^j,G}) \tag{4}$$

$$V_{i,G} = X_{best,G} + F \times \left(X_{r_1^j,G} - X_{r_2^j,G}\right) + F \times \left(X_{r_3^j,G} - X_{r_4^j,G}\right) \tag{5}$$

$$V_{i,G} = X_{r_1^j,G} + F \times \left(X_{r_2^j,G} - X_{r_3^j,G}\right) \tag{6}$$

$$V_{i,G} = X_{r_1^j,G} + F \times \left(X_{r_2^j,G} - X_{r_3^j,G}\right) + F \times \left(X_{r_4^j,G} - X_{r_5^j,G}\right) \tag{7}$$

$$V_{i,G} = X_{i,G} + K \times \left(X_{best,G} - X_{i,G}\right) + F \times \left(X_{r_1^j,G} - X_{r_2^j,G}\right) \tag{8}$$

$$V_{i,G} = X_{i,G} + K \times \left(X_{best,G} - X_{i,G}\right) + F \times \left(X_{r_1^j,G} - X_{r_2^j,G} + X_{r_3^j,G} - X_{r_4^j,G}\right) \tag{9}$$

$$U_{i,G} = X_{i,G} + K \times \left(X_{r_1^j,G} - X_{i,G}\right) + F \times \left(X_{r_2^j,G} - X_{r_3^j,G}\right) \tag{10}$$

where, r_n^i is a random number within [1, NP]; *F* is the scale factor that positively controls the scale of each vector; $X_{best,G}$ is the individual vector with the best fitness in the G^{th} generation; *K* is a random number within [0, 1].

After mutation, the target vectors X_i are crossed over in pairs to produce test vectors $U_{i,G} = \{u_{i,G}^1, u_{i,G}^2, \dots, u_{i,G}^D\}$. The binomial uniform crossover can be defined as:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if } \left(rand_j[0,1] \le CR\right) \text{ or } (j = j_{rand}) \\ x_{i,G}^j, & \text{otherwise} \end{cases} \tag{11}$$

where, *CR* is the user-defined crossover rate within [0, 1]; j_{rand} is a random integer within $[1, D]$.

For a newly generated test vector, any parameter falling outside its upper and lower bounds are reinitialized randomly within the preset range in a uniform manner. The objective function values of all test vectors are computed for the selection operation. During the selection, the objective function value of each test vector $f(U_{i,G})$ is compared with that of the corresponding target vector $f(X_{i,G})$ in the current population. The vector with the larger value will be retained in the next generation. If the two vectors have equal objective function values, then the test vector will be retained. The selection operation can be described as:

$$X_{i+1,G} = \begin{cases} U_{i,G}, & \text{if } \left(f(U_{i,G} \le f(X_{i,G})\right) \\ X_{i,G}, & \text{otherwise} \end{cases} \tag{12}$$

These above steps are repeated generation after generation until satisfying the termination condition.

3.2 Multi-objective hybrid difference optimization algorithm

Differential evolution can improve convergence performance by guiding search direction and exploiting the solution space through individual differences. Compared with the basic differential evolution algorithm, the hybrid difference optimization algorithm both enhances the convergence and allocation performance of the FSP, but also maintains the diversity of solutions.

To improve the FSP performance, the author proposed an algorithm based on differential evolution and hybrid differential optimization. The framework of the proposed algorithm can be set up in the following steps.

Stage 1: Following the sampling strategy of VEGA, the individuals with better target values in $P(t)$ were selected to form multiple small populations.

Stage 2: The small populations and the elite population $A(t)$ from Stage 1 were combined into a mating pool. In the mating pool, the individuals who perform well on different targets were stored in *Pop*-1 and *Pop*-2, respectively. The elite individuals with good PDDR-FF values were stored in the elite population. Overall, one third of the individuals perform well on one objective, one third on another objective and the remaining one third on multiple objectives.

Stage 3: New individuals were generated, and subjected to mutation and crossover, forming a new population $P(t+1)$.

Stage 4: $A(t)$ and $P(t)$ were combined into a temporary elite population $A'(t)$, the PDDR-FF values of all individuals in $A'(t)$ were calculated, and all these individuals were ranked in ascending order of fitness. Then, $|A(t)|$ individuals were replicated to form $A'(t+1)$.

Stage 5: The solution set $A''(t+1)$ obtained after mutation was fused with the temporary elite population $A'(t+1)$. According to the PDDR-FF value, the excellent individuals in the fused population were selected to generate the final population $A(t+1)$.

The evolution process of our algorithm is shown in Fig. 1.

In our algorithm, a chromosome $\vec{X} = [x_1, x_2, \dots, x_n]^T$ is still represented as a *D*-dimensional real parameter vector. But the dimension *D* must satisfy the constraint $D = 2d$, where *d* is the total number of operations required in the FSP. Since there are two coding stages, the chromosomes are divided into two separate parts. The first part $\overrightarrow{X^1} = [x_1, x_2, \dots, x_n]^T$ is a random real number vector, where x_n is a random number within $[0, 1]$. The second part $\overrightarrow{X^2} = [x'_1, x'_2, \dots, x'_n]^T$ is the machine allocation for each operation. The former is only applied to fitness functions and the calculation of individual fitness, while the latter performs various genetic operations.

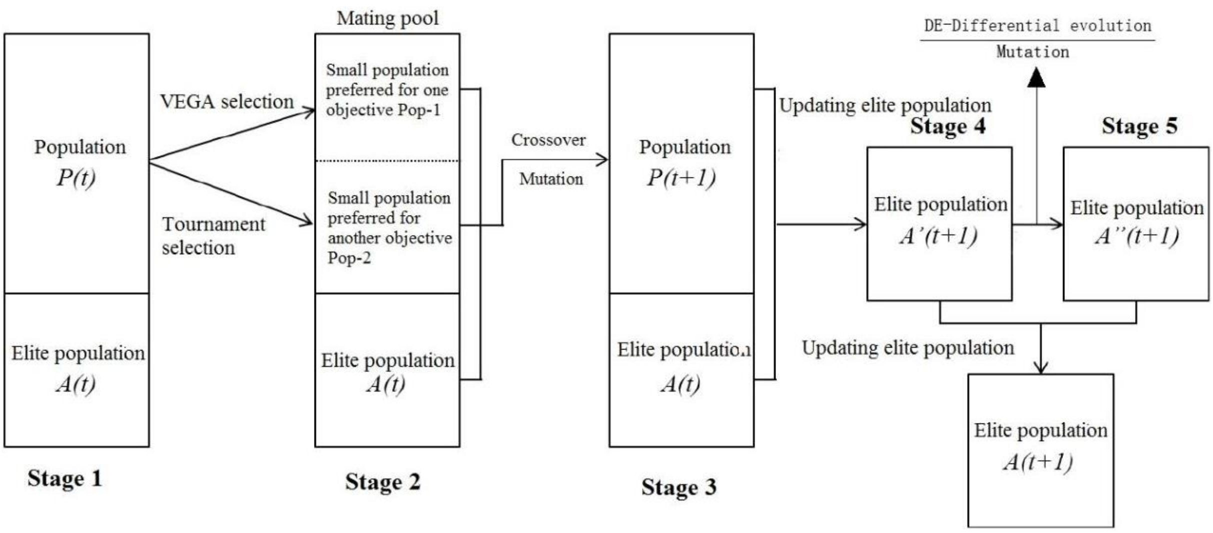

Figure 1: The evolution process of our algorithm.

Fig. 2 provides the details on the individual initialization. As shown in the figure, a random set of real vectors $\overrightarrow{X^1} = (0.1,\ 0.3,\ 0.5,\ 0.8)$ is generated, $\overrightarrow{X^1}$ is decoded as $\overrightarrow{X^2} = (3, 2, 4, 1)$, and the job processing sequence is obtained as $J_2,\ J_3,\ J_4,\ J_1$.

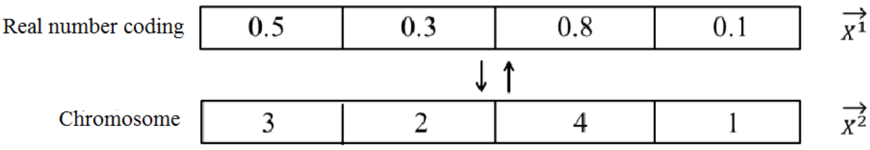

Figure 2: Detailed process of individual initialization.

4. IMPROVED MULTI-OBJECTIVE HYBRID DIFFERENTIAL OPTIMIZATION ALGORITHM FOR THE FSP

In the framework of hybrid differential evolution algorithm, the positions of fractional individuals were removed by hybrid sampling strategy. A full search-based differential operator is proposed by synthesizing vector evolution. This strategy combines global search-based vector evolution with new selection strategy.

Three different individuals s_1, s_2 and s_3 were selected randomly, according to the PDDR-FF values, provided that $eval(s_1) < eval(s_2) < eval(s_3)$. Each individual was selected evolutionarily by the following approach:

$$v_i = s_3 + K(s_1 - s_2) \tag{13}$$

The nature of PDDR-FF evaluation function determines that the development of the best individual depends on the worse individual. For example, if s_1 has been selected repeatedly for genetic manipulation, the selection process can be illustrated as:

$$\begin{cases} v_1^1 = s_1 + K(s_2 - s_3) & eval(s_2) < eval(s_3) < eval(s_1) \\ v_1^2 = s_1 + K(s_4 - s_5) & eval(s_4) < eval(s_5) < eval(s_1) \\ & \dots \\ v_1^n = s_1 + K(s_m - s_n) & eval(s_m) < eval(s_n) < eval(s_1) \end{cases} \tag{14}$$

where, v_1^1, v_1^2 and v_1^n are new individuals generated by genetic manipulation. If a good individual s_k is not selected, s_k is not necessarily a bad individual. If $A(t+1)$ is generated, s_k will be retained by PDDR-FF selection strategy, thanks to its good fitness.

The selected individuals were mutated by the following strategies:

Mutation strategy 1: If $eval(s_3) \leq 1$, then the current individual s_3 is a non-dominant solution, and individuals s_1 and s_2 are also smaller than or equal to 1. In this case, the mutation was performed by searching from the non-dominant solution s_3, and from the edge to the center along the Pareto boundary. This search pattern will improve the distribution performance. The search direction of this mutation strategy is shown in Fig. 3.

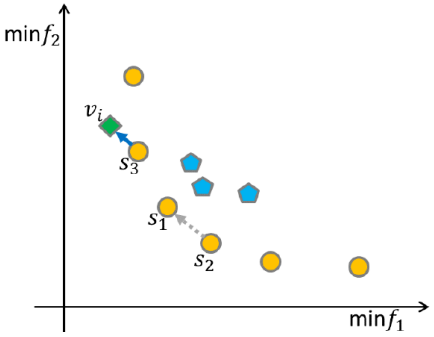

Figure 3: The search direction of mutation strategy 1.

Mutation strategy 2: If $eval(s_3) > 1$, then the current individual s_3 is the dominant solution. If $eval(s_1) > 1$, then $eval(s_2)$ is also greater than 1, and all three individuals are dominant solutions. In this case, the mutation was performed by searching from the dominant solution s_3, and from the edge to the center along the Pareto boundary. This search pattern will improve the distribution performance. The search direction of this mutation strategy is shown in Fig. 4.

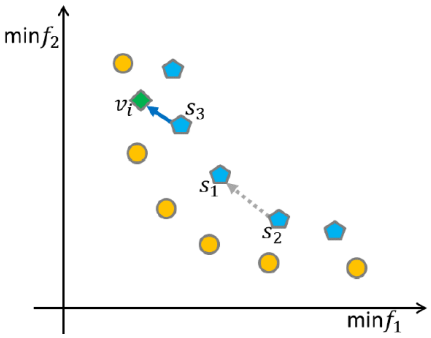

Figure 4: The search direction of mutation strategy 2: $eval(s_1) > 1$.

If $eval(s_1) \leq 1$ and $eval(s_2) > 1$, then both s_2 and s_3 are dominant solutions, while s_1 is non-dominant. In this case, the mutation was performed by searching from the dominant solution s_3, and from the dominant area to the non-dominant area along the Pareto boundary. This search pattern will improve the distribution performance. The search direction of this mutation strategy is shown in Fig. 5.

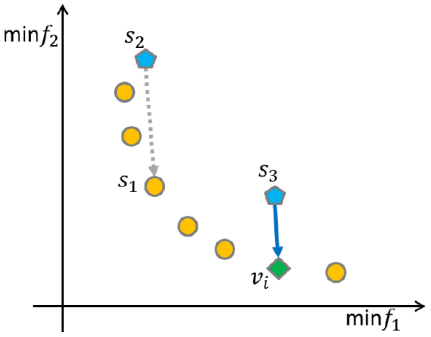

Figure 5: The search direction of mutation strategy 2: $eval(s_1) \leq 1$.

If $eval(s_1) \le 1$ and $eval(s_2) \le 1$, then s_3 is a dominant solution, while both s_1 and s_2 are non-dominant. In this case, the mutation was performed by searching from the dominant solution s_3, and from the edge to the center along the Pareto boundary. This search pattern will improve the distribution performance. The search direction of this mutation strategy is shown in Fig. 6.

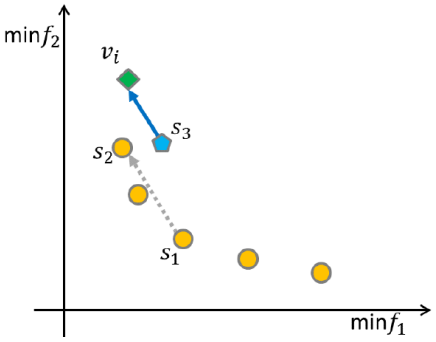

Figure 6: The search direction of mutation strategy 2: $eval(s_1) \le 1$ and $eval(s_2) \le 1$.

5. SIMULATION AND RESULTS ANALYSIS

Our algorithm was simulated with the set of special test problems of the GA. The mathematical expressions of the standard test problems are given in Table I.

Firstly, both our algorithm and the basic differential evolution algorithm were simulated on the three benchmark problems of P1, P2 and P3. Each problem was simulated 50 times. To minimize the simulation error, the initial populations of the two algorithms were the same in each simulation. In addition, the two algorithms use the same encoding method and genetic operator. The population size, maximum number of iterations, elite population size and crossover probability were set to 100, 500, 50 and 0.85, respectively. The two parameters of differential evolution algorithm were configured as $K = 0.5$ and $F = 0.5$. The simulation results are shown in Table II.

The simulation results show that the chromosome length increased with the problem scale. Our algorithm outperformed the basic differential evolution algorithm in convergence and distribution.

Our algorithm was further verified through a case study on a 6 job, 6 machine FSP. The details on the FSP are provided in Table III.

Table I: The standard test problems.

Problem	Number of variables	Interval	Objective functions
P1	20	[0.1]	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + \sum_{i=1}^{n} x_i/(n+1)$
P2	20	[0.1]	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + \sum_{i=1}^{n} x_i/(n+1)$
P3	20	[0.1]	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)} - \frac{x_1}{g(x)} \sin 10\pi x_i]$ $g(x) = 1 + \sum_{i=1}^{n} x_i/(n+1)$

Table II: The simulation results of the two algorithms.

Problem	Mean value		Standard deviation	
	Our algorithm	Basic differential evolution algorithm	Our algorithm	Basic differential evolution algorithm
P1	2.03E-01	1.32E-02	7.62E-02	2.75E-03
P2	1.98E-01	1.33E-02	5.76E-02	2.38E-03
P3	1.12E-01	1.49E-02	6.17E-02	1.77E-03

Table III: The 6 job, 6 machine FSP.

Jobs	Operations	Machines					
		M_1	M_2	M_3	M_4	M_5	M_6
W_1	O_{11}	45	40	48	32	36	38
	O_{12}	16	--	--	--	--	--
	O_{13}	--	32	36	22	19	81
	O_{14}	4	--	5	3	--	2
	O_{15}	8	10	--	6	--	6
W_2	O_{21}	26	22	--	20	--	22
	O_{22}	--	38	35	33	28	--
	O_{23}	13	11	--	--	17	9
	O_{24}	--	9	7	6	6	5
	O_{25}	29	--	27	--	25	23
W_3	O_{31}	23	25	--	--	22	19
	O_{32}	30	--	--	28	26	--
	O_{33}	9	--	7	--	5	--
	O_{34}	20	--	18	--	--	18
	O_{35}	8	--	7	--	7	5
W_4	O_{41}	8	7	--	6	4	--
	O_{42}	--	--	5	4	--	3
	O_{43}	--	14	12	11	--	9
	O_{44}	22	17	--	15	13	10
	O_{45}	18	16	15	12	--	11
W_5	O_{51}	27	26	--	--	22	--
	O_{52}	9	--	--	--	--	6
	O_{53}	13	--	11	9	8	5
	O_{54}	11	9	--	--	9	8
	O_{55}	14	12	10	--	10	7
W_6	O_{61}	10	9	9	8	7	8
	O_{62}	26	24	22	21	--	20
	O_{63}	16	--	--	14	12	13
	O_{64}	9	7	--	--	8	6
	O_{65}	12	10	9	--	6	--

The parameters of our algorithm were initialized as: population size = 500, maximum number of iterations = 200, elite population size = 100, and crossover probability = 0.85. The scheduling objectives are to minimize the makespan and machine load.

Scheduling cycle and scheduling factor are two important influencing factors of the FSP. If the cycle is too short, the scheduling plan needs to be modified repeatedly; if the cycle is too long, the machine utilization may not be satisfactory. Hence, the rescheduling cycle was set to 30, 60, 90 and 120, respectively. The results of the makespan are shown in Table IV below.

Table IV: Results (makespans) of different scheduling cycles.

Times	Cycle			
	30	60	90	120
1	29,832	29,745	32,689	35,790
2	29,783	29,712	32,517	35,562
3	29,912	29,985	33,598	36,127
4	30,418	30,426	34,325	37,650
5	29,863	29,865	33,690	36,851

The data in Table IV show that the makespans of the five operations in four different cycles basically exhibited a U-shaped distribution. This means our algorithm achieved the minimize makespan and machine load, and ensured the scheduling efficiency and stability of the FSP.

6. CONCLUSIONS

For the multi-objective FSP, this paper proposes a hybrid multi-objective differential evolution algorithm based on individual distribution. Considering the dominant and non-dominant relationships among individuals, the individuals with better PDDR-FF values were selected to the set of elite solutions. During the selection, a two-part mating pool was generated according to fitness function and target value, which are respectively close to the centre and edge of Pareto boundary. Based on the distance to the Pareto boundary, alternative individuals were selected in multiple ways to guide the search direction and improve the convergence and distribution. The simulation results show that our algorithm outperformed the basic multi-objective evolutionary algorithm in convergence and distribution performance.

ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China under Grant No. 71871106, Jiangsu Province Social Science Foundation under Grant No. 18GLD014, Jiangsu Province University Philosophy and Social Science Foundation under Grant No. 2018SJA0815, Chinese University Research Foundation under Grant No. 2019JDZD06 and Chinese University Research Foundation for Young Scholars under Grant No. JUSRP11882.

REFERENCES

[1] Arunachalam, A. P. S.; Idapalapati, S.; Subbiah, S. (2015). Multi-criteria decision making techniques for compliant polishing tool selection, *The International Journal of Advanced Manufacturing Technology*, Vol. 79, No. 1-4, 519-530, doi:10.1007/s00170-015-6822-y

[2] Yang, Z.; Liu, C.-G. (2018). A multi-objective genetic algorithm for a fuzzy parallel blocking flow shop scheduling problem, *Academic Journal of Manufacturing Engineering*, Vol. 16, No. 2, 3-11

[3] Golmakani, H. R.; Birjandi, A. R. (2013). A two-phase algorithm for multiple-route job shop scheduling problem subject to makespan, *The International Journal of Advanced Manufacturing Technology*, Vol. 67, No. 1-4, 203-216, doi:10.1007/s00170-013-4767-6

[4] Chen, W.; Hao, Y. F. (2018). Genetic algorithm-based design and simulation of manufacturing flow shop scheduling, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 702-711, doi:10.2507/IJSIMM17(4)CO17

[5] Huang, C. J.; Zhou, X. H.; Hou, D. S. (2018). Online no-wait scheduling of leather workshop supply chain based on particle swarm optimization, *Journal Européen des Systèmes Automatisés*, Vol. 51, No. 1-3, 153-167, doi:10.3166/JESA.51.153-167

[6] Jemilda, G.; Baulkani, S. (2018). Moving object detection and tracking using genetic algorithm enabled extreme learning machine, *International Journal of Computers Communications and Control*, Vol. 13, No. 2, 162-174
[7] Somashekhara, S. C. H.; Setty, A. K. Y.; Sridharmurthy, S. M.; Adiga, P.; Mahabaleshwar, U. S.; Lorenzini, G. (2019). Makespan reduction using dynamic job sequencing combined with buffer optimization applying genetic algorithm in a manufacturing system, *Mathematical Modelling of Engineering Problems*, Vol. 6, No. 1, 29-37, doi:10.18280/mmep.060104
[8] Gopi, A. P.; Lakshman Narayana, V.; Ashok Kumar, N. (2018). Dynamic load balancing for client server assignment in distributed system using genetic algorithm, *Ingénierie des Systèmes d'Information*, Vol. 23, No. 6, 87-98, doi:10.3166/ISI.23.6.87-98
[9] Sun, G.; Bin, S. (2018). A new opinion leaders detecting algorithm in multi-relationship online social networks, *Multimedia Tools and Applications*, Vol. 77, No. 4, 4295-4307, doi:10.1007/s11042-017-4766-y
[10] Gao, J.; Chen, R.; Deng, W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem, *International Journal of Production Research*, Vol. 51, No. 3, 641-651, doi:10.1080/00207543.2011.644819
[11] Chandrasekaran, M.; Devarasiddappa, D. (2014). Artificial neural network modeling for surface roughness prediction in cylindrical grinding of Al-SiC_p metal matrix composites and ANOVA analysis, *Advances in Production Engineering & Management*, Vol. 9, No. 2, 59-70, doi:10.14743/apem2014.2.176
[12] Jiang, C.-H.; Zhang, C.; Zhang, Y.-H.; Xu, H. (2017). An improved particle swarm optimization algorithm for parameter optimization of proportional–integral–derivative controller, *Traitement du Signal*, Vol. 34, No. 1-2, 93-110, doi:10.3166/TS.34.93-110
[13] Fernández-Navarro, F.; Hervás-Martínez, C.; García-Alonso, C.; Torres-Jimenez, M. (2011). Determination of relative agrarian technical efficiency by a dynamic over-sampling procedure guided by minimum sensitivity, *Expert Systems with Applications*, Vol. 38, No. 10, 12483-12490, doi:10.1016/j.eswa.2011.04.031
[14] Haider, A.; Mirza, J. (2015). An implementation of lean scheduling in a job shop environment, *Advances in Production Engineering & Management*, Vol. 10, No. 1, 5-17, doi:10.14743/apem2015.1.188
[15] Chaturvedi, J. (2014). Adaptive quantum inspired genetic algorithm for combinatorial optimization problems, *International Journal of Computer Applications*, Vol. 107, No. 4, 34-42, doi:10.5120/18743-9996
[16] Gao, K.; Pan, Q.; Suganthan, P. N.; Li, J. (2013). Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization, *The International Journal of Advanced Manufacturing Technology*, Vol. 66, No. 9-12, 1563-1572, doi:10.1007/s00170-012-4440-5
[17] Sun, Y.; Zhang, C.; Gao, L.; Wang, X.-J. (2011). Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects, *The International Journal of Advanced Manufacturing Technology*, Vol. 55, No. 5-8, 723-739, doi:10.1007/s00170-010-3094-4
[18] Nabipoor Afruzi, E.; Roghanian, E.; Najafi, A. A.; Mazinani, M. (2013). A multi-mode resource-constrained discrete time–cost tradeoff problem solving using an adjusted fuzzy dominance genetic algorithm, *Scientia Iranica*, Vol. 20, No. 3, 931-944, doi:10.1016/j.scient.2012.12.024
[19] Das, S.; Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, 4-31, doi:10.1109/TEVC.2010.2059031
[20] Lwin, K.; Qu, R. (2013). A hybrid algorithm for constrained portfolio selection problems, *Applied Intelligence*, Vol. 39, No. 2, 251-266, doi:10.1007/s10489-012-0411-7
[21] Shao, W.; Pi, D. (2016). A self-guided differential evolution with neighborhood search for permutation flow shop scheduling, *Expert Systems with Applications*, Vol. 51, 161-176, doi:10.1016/j.eswa.2015.12.001
[22] Soltani, S. A.; Karimi, B. (2015). Cyclic hybrid flow shop scheduling problem with limited buffers and machine eligibility constraints, *The International Journal of Advanced Manufacturing Technology*, Vol. 76, No. 9-12, 1739-1755, doi:10.1007/s00170-014-6343-0