Int j simul model 18 (2019) 4, 678-688
ISSN 1726-4529 Original scientific paper

AN INTELLIGENT OPTIMIZATION ALGORITHM FOR BLOCKING FLOW-SHOP SCHEDULING BASED ON DIFFERENTIAL EVOLUTION

Xu, L. Z.; Xie, Q. S.[#]; Yuan, Q. N. & Huang, H. S.
Key Laboratory of Advanced Manufacturing Technology, Guizhou University, Guiyang 550025, China
E-Mail: qsxie@gzu.edu.cn ([#] Corresponding author)

Abstract
Owing to its large scale, the blocking flow-shop scheduling problem (BFSP) cannot be solved effectively by traditional optimization methods. To solve the problem, this paper develops a novel intelligent optimization algorithm based on differential evolution (DE) for the BFSP with a single objective: minimizing the total flow time (TFT). On the one hand, a new heuristic method was introduced to balance the quality and diversity of the initial population. On the other hand, a new operator was adopted to update the acceleration, velocity and position of each particle. In this way, the population will not converge prematurely to local optimums, and the local and global search abilities are perfectly balanced. Simulation on standard test set proves that our algorithm outperformed most commonly used methods in solving the BFSP.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Blocking Flow-Shop Scheduling Problem (BFSP), Differential Evolution (DE), Intelligent Optimization Algorithm, Gravitational Search Algorithm (GSA)

1. INTRODUCTION

Production scheduling problem is a typical combinatorial optimization problem. The problem has been proved to be non-deterministic polynomial-time (NP) hard, because of its inherent complexity, strong constraint and multiple objectives. With the growing scale of the problem, the difficulty of problem-solving increases exponentially.

Early on, the production scheduling problems are generally small in scale. These small-scale problems can be solved well by traditional optimization methods, such as integer programming [1], dynamic programming [2] and branch-and-bound algorithm [3]. By these methods, the optimal solution is obtained by problem modelling and model analysis. With the increase of the scale of production scheduling problems, however, the traditional optimization methods can no longer find the optimal solution. This calls for the design of a new intelligent optimization algorithm for largescale production scheduling problems.

Intelligent optimization algorithm [4-8] is a kind of optimization method inspired by biological system or physical phenomenon in nature. Many classical optimization algorithms are intelligent optimization algorithms, including genetic algorithm (GA), simulated annealing (SA) algorithm, particle swarm optimization (PSO) [9], ant colony optimization (ACO) [10], differential evolution (DE) [11], and many hybrid algorithms [12-14]. An intelligent optimization algorithm does not necessarily output the optimal solution. But the solution basically falls in the acceptable time range.

There are some common defects of intelligent optimization algorithms, namely, parameter sensitivity, and proneness to local optimum trap. To overcome these defects, this paper probes deep into the search mechanism and operation principle of intelligent optimization algorithms, develops an intelligent optimization algorithm based on the DE, and applies the proposed algorithm to solve the blocking flow-shop scheduling problem (BFSP).

2. RELATED WORKS

The BFSP is a typical subproblem of production scheduling. In the BFSP, the jobs may be blocked on the production line due to the lack of buffer between machines. Many heuristic methods have been proposed to solve the BFSP. For example, Shao and Pi [15] minimized the total makespan of the BFSP, using the specific information of problem constraints. Based on directed search, Viagas et al. [16] created a constructive heuristic method to minimize the total flow time (TFT) of the BFSP. Ribas et al. [17] minimized the TFT of the BFSP with two heuristic methods. Han et al. [18] put forward a mixed heuristic method for the multi-objective BFSP.

Meta-heuristic methods are also widely adopted to solve the BFSP. For instance, Pan et al. [19] proposed a discrete artificial bee colony (ABC) algorithm to minimize the TFT of the BFSP. Liao et al. [20] designed a PSO-based meta-heuristic method to minimize the total delay of the BFSP. Dai et al. [21] developed a SA algorithm to solve multi-objective BFSP.

The evaluation indices of blocking flow-shop scheduling are another research hotspot. The commonly used indices include the total makespan, the total delay, and the TFT. Among them, the total delay refers to the waiting time between jobs in a production task. Trabelsi et al. [22] suggested solving the mixed BFSP based on the specific structure of blocking constraints and objective functions. Tasgetiren et al. [23] proposed three hybrid iterative greedy algorithms to solve distributed BFSP. Li et al. [24] set up an improved fruit fly algorithm to minimize the total makespan of the BFSP.

3. ALGORITHM DESIGN

3.1 Description of the BFSP

The BFSP got its name from the fact that the jobs may be blocked on the production line, due to the lack of buffer between machines. In this paper, the DE is adopted to design an algorithm to solve the BFSP with a single objective: minimizing the TFT.

Let N be the number of jobs in the flow shop, and M be the number of machines in series. It is assumed that all the jobs have the same operation sequence, and no buffer exists between adjacent machines. A blocking occurs in the production line when job j is ready to be transferred from machine i to machine i+1, but the latter machine is occupied by another job. In this case, job j is blocked on machine i until machine i+1 is idle. Fig. 1 is a Gantt chart of a BFSP with four machines and three jobs.

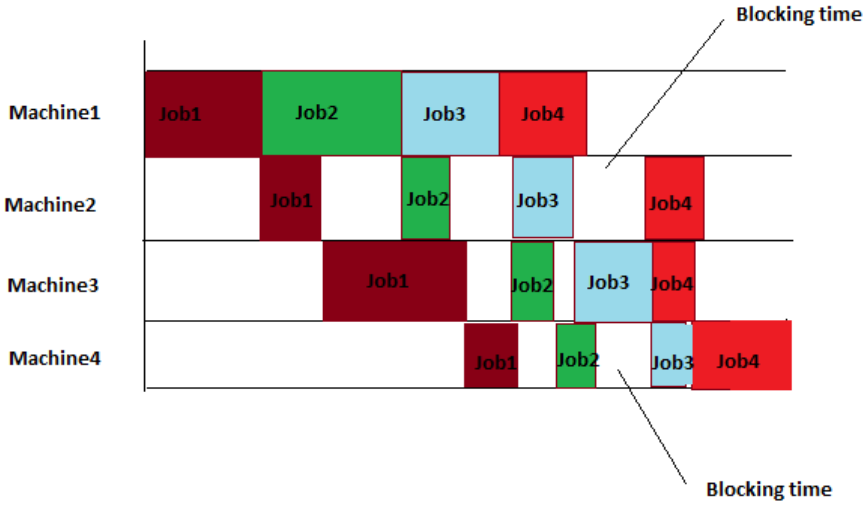

Figure 1: Gantt chart of a BFSP.

As shown in Fig. 1, when job 2 has been fully processed on machine 2, it cannot be transferred to machine 3 immediately. This is because machine 3 is currently occupied by job 1. Then, job 2 is blocked on machine 2, which delays the processing of job 3.

Mathematically, the BFSP with the objective of minimizing the TFT can be described as $F_{block}\sum T_i$, where $\sum T_i$ is the TFT. Let $\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(n)\}$ be the set of jobs, and p_{ij} be the processing time of job i on machine j. Then, the TFT of an operation sequence can be expressed as:

$$T_{\sigma(1),0} = 0 \tag{1}$$

$$T_{\sigma(1),k} = T_{\sigma(1),k-1} + p_{\sigma(1),k} k = 1,2,\ldots,m-1 \tag{2}$$

$$T_{\sigma(j),0} = T_{\sigma(j-1),1} \tag{3}$$

$$T_{\sigma(j),k} = \max\{T_{\sigma(j),k-1} + p_{\sigma(j),k}, T_{\sigma(j-1),k+1}\} \;\; j = 2,\ldots,n \;\; k = 1,2,\ldots,m-1 \tag{4}$$

$$T_{\sigma(j),m} = T_{\sigma(j),m-1} + p_{\sigma(j),m} \;\; j = 1,\ldots,n \tag{5}$$

$$TFT(\sigma) = \sum_{j=1}^{n} T_{\sigma(j),m} \tag{6}$$

where, $T_{\sigma(j),0}, j = 1,\ldots,n$ is the start time of job j on the first machine (the start time of the first job on the first machine is defined as zero); $T_{\sigma(j),k}\; k = 1,\ldots,m$ is the completion time of job j on machine k. The objective of minimizing the TFT is to find an operation sequence σ^* that satisfies the inequality $TFT(\sigma^*) \leq TFT(\sigma)\; \forall \sigma \in S$, where S is the set of possible operation sequences.

3.2 DE-based intelligent optimization algorithm

Considering the above BFSP, the author set up a new DE-based intelligent optimization algorithm to minimize the TFT. During algorithm construction, the gravitational search algorithm (GSA) [25, 26] was improved to solve discrete problems: In the encoding process, the job sequence was directly represented by the sequence of integers; a new initialization method was introduced to enhance the quality of initial solution and maintain the diversity of the initial solution set; the acceleration, velocity and position of each particle are updated by different processing methods. In the improved GSA, a particle in the t^{th} generation can be expressed as $X^t = [x_1^t, x_2^t, \ldots, x_n^t]$, where x_i^t is the number of the job at position i.

(1) Population initialization

In the improved GSA, a total of num initial solutions are generated at the beginning. Each initial solution represents an operation sequence of a job. The fitness of each particle is computed by the TFT method, Eq. (6). The initial population is generated by a new method called variable neighbours heuristic (*VNH(n)*). Specifically, the first job of an initial solution is selected from the x particles in the population. Each initial solution is generated by:

$$td(i,k) = \mu * \left(\sum_{j=1}^{m} (T_{k+1,j}(\sigma * i) - T_{k,j}(\sigma) - p_{i,j}) \right) + (1-\mu) * (T_i - T_{[k-1]}) \tag{7}$$

The generation process in Eq. (7) can be divided into two steps:

Step 1. A sequence of consecutive integers is generated, depending on the number of jobs. Job $t\; (t = 1,\ldots,n)$ is selected as the first job of the new sequence σ. Let $k = 1$.

Step 2. If $k < n$, the td value is calculated by Eq. (7) for each job i that has not been determined. The job with the smallest td value is chosen as the next job. If two jobs have the same td value, the one that minimizes the TFT value of the sequence is selected as the next job.

The initial job sequence can be obtained through the above steps. The detailed process of *VNH(n)* is defined as Algorithm 1:

Algorithm 1

Input: The initial solution τ, which is a sequential integer sequence from 1 to n.
For ($i = 1$ to n) do
τ_f is taken as the first job of τ
A sequence is generated by $VNH\ \tau 1 = VNH(\tau_f)$
For ($j = n - \sigma + 1$ to n) do
Job $\tau_j$is selected from $\tau 1$
$\tau_j$is inserted into all possible positions of $\tau 1$ and $TFT$is evaluated
End for
The sequence $\tau 2$ with the lowest TFT is obtained
If ($TFT\ (\tau 2) < TFT\ (\tau 1)$) $\tau^{\mathrm{i}} = \tau 2$;
else $\tau^{\mathrm{i}} = \tau 1$
End for
Output: a set of sequence $\{\tau^1, \tau^2, \dots, \tau^{\mathrm{n}}\}$ and corresponding TFT.

In the improved GSA, n sequences are produced by the *VNH*(n). Then, the population size num is determined through the following method:$if\,(n \le 20), num = 20; else num = 50$. After that, $TFT(\tau^i)(i = 1, \dots, n)$ are ranked in ascending order according to the sequences generated by *VNH*(n), and the top num particles are assigned to the initial population. The heuristic method ensures the quality and diversity of the initial population.

(2) Calculation of particle acceleration

In improved GSA, the acceleration, velocity and position of each particle is updated by a novel mechanism. First, the acceleration update formula is refined as:

$$a_i(t) = \sum_{j \in kbest, j \neq i} \frac{HR_{i,j}(t) * rand * W(t) * D(j)}{ra + \frac{HR_{i,j}(t)}{n}} \otimes (x_j(t) \ominus x_i(t)) \tag{8}$$

where, $rand$ is a random variable in [0, 1]; $W(t)$ is the gravitational coefficient; $D(j)$ is the quality after normalizing the TFT of sequence j; $HR_{i,j}(t)$ is the Hamming distance between sequences i and j at the t^{th} iteration; $\otimes$ and $\ominus$ are the acceleration update operators.

The particle acceleration is defined as Algorithm 2.

Algorithm 2

Input: A set of sequences $\{\tau_j\}(j = 1, \dots, kbest)$
For $i = 1$ to n do
An integer is randomly selected form $[1, \dots, kbest] \to k$
$a[i] = \tau_k[i]$
End for
If there are multiple identical elements in a do
One element is randomly kept and other elements are set to be zero
End if
Output: acceleration a is obtained.

(3) Calculation of particle velocity

The velocity update formula is redefined as:

$$v_i(t+1) = v_i(t) \odot a_i(t) \tag{9}$$

where, $\odot$ is the velocity update operator. The velocity update of a particle is implemented in two steps.

Step 1. It is assumed that vc is a constant ($0 < vc < 1$). For $j = 1, \dots, n$, if $rand < vc$, then $v_i(t+1)[j] = v_i(t)[j]$; otherwise $v_i(t+1)[j] = a_i(t)[j]$.

Step 2. If there are multiple identical elements in the $v_i(t+1)$ sequence, then the processing method is the same as in the summation operation.

(4) Calculation of particle position

The position update formula is redefined as:

$$x_i(t+1) = x_i(t) \oplus v_i(t+1) \tag{10}$$

where, $\oplus$ is the position update operator. The position update of a particle includes three operations, which are summed up as Algorithm 3.

Algorithm 3

Input: the best sequence τ^* and the original sequence τ are selected.
For $i = 1$ to n do
If $\tau(i) \neq \tau^*(i)$ do
For $j = 1$ to n do
Find $\tau(j) = \tau^*(i), key = j, keyvalue = \tau(j)$
End for
If ($key > i$) (forward insertion) do
A sequence $\tau 1$ is generated by inserting $keyvalue$ into the front of $\tau(k)$
TFT of new sequence $\tau 1$ is evaluated
If ($TFT\ (\tau 1) < TFT\ (\tau)$) do

$$\tau = \tau 1; TFT(\tau) = TFT(\tau 1)$$

End if
End if
End if
End for
Output: the best sequence with the lowest TFT.

If the current particle to be updated is the same as the optimal particle in the population, then the two sequences must be identical. In this case, the particle position needs to be updated by variable neighbourhood operators.

To avoid premature convergence, the improved GSA must go through, an adaptive perturbation: the current population is redistributed if its particles are concentrated to a certain extent.

In this paper, the population diversity is measured by a new method:

$$dif = \frac{\sum_{i=1}^{num} \sum_{j=1}^{j=kbest} HR(x_i(t), x_j(t))}{kbest * num * n} \tag{11}$$

where, $HR(x_i(t), x_j(t)$ is the Hamming distance between sequences i and j at the t^{th} iteration. Since $HR(x_i(t), x_j(t)$ falls within $[0, n]$, the population diversity coefficient div must belong to $[0, 1]$. In the light of Eq. (1), the current population is considered to be highly diverse if dif exceeds a certain threshold, and weakly diverse or even uniform if dif approaches 0.

According to the dif value, the population is subjected to adaptive perturbation at the probability $ap = e^{-K*div}$, where K is a control parameter. If $rand \leq ap$, the local search strategy of path relinking [27] will be introduced to disperse the current sequence and avoid the local optimum trap. The optimal sequence of the current population is selected as the target sequence, and the sequence to be promoted is taken as the original sequence. When the original sequence is converted to the target sequence, a path is established by inserting the jobs in the forward or backward direction. If $rand > sa$, the population will not fall into the local optimum trap.

The complete pseudo code for the DE-based intelligent optimization algorithm is given as Algorithm 4, in which the TFT after insertion and exchange is obtained by a fast TFT calculation method.

Algorithm 4

Input: According to *VNH(n)*, initialize the population of *num* agents

Termination condition *Ter_con* is set

While (Not *Ter_con*) do

it = current iterations

$$t0 = \frac{it}{it + vt * n}$$

The best value, the worst value, and the corresponding sequence are calculated

The gravitational coefficient $G(it) = (1 - t0) + G0$ is calculated

The $kbest = round(\frac{2+(1-t0)*(50-2)*num}{50})$ is calculated

The Hamming distance between the best sequence and each sequence is calculated, and the result is given to $bestR$

The acceleration and the velocity of the population is calculated

For $i = 1$ to num do

If $bestR(i) = 0$ do

Variable neighborhood operators are implemented, new sequence $\tau(i)^*$is obtained

If $TFT(\tau(\mathrm{i})^*) < TFT(\tau(i))$ do

$\tau(\mathrm{i}) = \tau(\mathrm{i})^*; TFT(\tau(\mathrm{i})) = TFT(\tau(\mathrm{i})^*)$

Else $if\ rand < exp\{\frac{-(TFT(\tau(i))-TFT(\tau(i)^*))}{T}\}$

$\tau(\mathrm{i}) = \tau(\mathrm{i})^*; TFT(\tau(\mathrm{i})) = TFT(\tau(\mathrm{i})^*)$

End if

Else if $rand < ap$

The path relinking is conducted to disperse the current population sequence, generating a new sequence

Else

The population is not in danger of falling into local optimum, a new sequence is generated

End if

End if

End for

End while

Output: τ_{best} and TFT (τ_{best})

4. SIMULATION AND RESULTS ANALYSIS

The proposed algorithm was verified through discrete event simulation on the MATLAB.

4.1 Parameter settings

The optimal combination of *VNH*(n) parameters, namely, μ and δ, was determined through complete factorization. The performance of each parameter combination was tested on a standard test set: $n \in \{10, 50, 100, 200\}$ and $m \in \{5, 10, 20\}$. There are 12 instances in the test set, each of which has 5 different sub-instances. The test set was used in all subsequent experiments on parameter setting.

The parameter combination of the *VNH*(n) was set to:

$\delta \in \{5, 10, 15, 20\}$,

$\mu \in \{0.5, 0.55, 0.65, 0.7, 0.75, 0.8, 0.85\}$.

Therefore, 28 sets of experiments were carried out, and the performance of each combination was measured by relative transient percentage (RTP):

$$RTP = \sum_{i=1}^{28} \frac{TFT(i) - TFT(best)}{TFT(best)} * 100 \tag{12}$$

where, $TFT(i)$ is the total flow time of instance i; $TFT(best)$ is the smallest total flow time among all parameter combinations. Let ARTP be the average RTP obtained for each problem instance for each parameter combination. The results of ARTP (Fig. 2) show that the optimal parameter combination of $VNH(n)$ is $\mu = 0.75$ and $\delta = 20$.

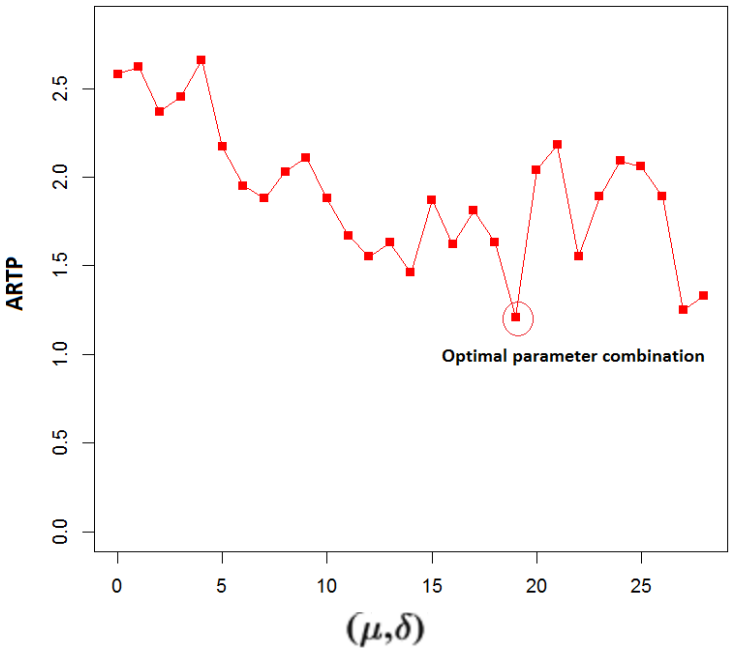

Figure 2: ARTP values for parameters μ and δ.

For the variable neighbourhood operator, the optimal combination of control parameters, namely, the block size in the beginning (bsb), the block size in the end (bse), the step coefficient (sc), the number of cycles (t), and the temperature control parameter (τP), was searched for through orthogonal experiment. According to the factor level of the parameters (Table I), the orthogonal matrix selected is $L_{16}(4^5)$.

Table I. Parameter factor levels of variable neighbourhood operator.

Parameter factor level	**1**	**2**	**3**	**4**
bsb	1	2	3	4
bse	10	15	20	25
sc	0.1	0.2	0.3	0.4
t	1	2	3	4
τP	0.2	0.4	0.6	0.8

In the orthogonal experiment, each parameter combination was tested five times on each instance. The ARTP of variable neighbourhood operator for each parameter combination is listed in Table II. The performance ranking of parameter combinations is shown in Table III.

As shown in Table III, t has a greater impact than any other parameter on the operator performance. This is attributable to its important influence on the operator's balance between runtime and search quality. Hence, the optimal combination of parameters is $bsb = 4, bse = 20,\ sc = 0.1$ and $\tau P = 0.4$.

Table II: Orthogonal test table of variable neighbourhood operator.

Number	Parameter					ARTP
	bsb	bse	sc	t	τP	
1	1	10	0.1	1	0.2	0.4752
2	1	15	0.2	2	0.4	0.6331
3	1	20	0.3	3	0.6	0.7902
4	1	25	0.4	4	0.8	0.8356
5	2	10	0.2	3	0.2	0.7129
6	2	15	0.1	4	0.4	0.8782
7	2	20	0.4	1	0.6	0.5312
8	2	25	0.3	2	0.8	0.6352
9	3	10	0.3	4	0.2	0.9162
10	3	15	0.4	3	0.4	0.7109
11	3	20	0.1	2	0.6	0.5983
12	3	25	0.2	1	0.8	0.6246
13	4	10	0.4	2	0.2	0.5183
14	4	15	0.3	1	0.4	0.5252
15	4	20	0.2	4	0.6	0.8733
16	4	25	0.1	3	0.8	0.5755

Table III: Parameter ranking of variable neighbourhood operator.

	Parameter				
	bsb	bse	sc	t	τP
1	0.676	0.662	0.647	0.552	0.672
2	0.682	0.681	0.712	0.596	0.656
3	0.711	0.687	0.717	0.682	0.696
4	0.632	0.672	0.651	0.869	0.667
Std.	0.036	0.019	0.043	0.147	0.017
Rank	3	4	2	1	5

Similarly, the five parameters of the DE-based intelligent optimization algorithm were adjusted through orthogonal test. Based on the results of the orthogonal test, the optimal parameter combination of the DE-based intelligent optimization algorithm is obtained as $vt = 2,\ GO = 1.8,\ ra = 1,\ vc = 0.2$ and $K = 2$.

4.2 Discrete simulation experiments

To verify its effectiveness on minimizing the TFT, the proposed algorithm (the DE-based intelligent optimization algorithm) was compared with five advanced algorithms: discrete artificial bee colony algorithm (DABC_RCT) [19], variable block insertion heuristic algorithm (VBIH) [28], iterated greedy algorithm (IG_RIS) [29], sequence alignment by genetic algorithm (SAGA) [30] and discrete differential evolution (DDE) [31]. All these algorithms were simulated on the MATLAB. The simulation results are compared in Table IV below.

As shown in Table IV, our algorithm achieved the best performance in 7 out of the 11 instances. It is obviously the best algorithm among all comparative methods for minimizing the TFT of the BFSP. There are three possible reasons for the excellent performance of our algorithm.

First, the GSA, as a new population-based evolutionary algorithm, is sensitive to parameters like gravitational coefficient and population size. Second, the algorithm performance hinges on the quality of the initial population. In our algorithm, DABC_RCT, VBIH and IG_RIS, the populations are initialized by heuristic methods. However, the initial

population of SAGA is generated randomly, and that of DDE is generated by a simple heuristic or random method. Third, our algorithm introduces a new variable neighbourhood search strategy, which effectively proves the performance in tackling production scheduling problems.

Table IV: Simulation results of different BFSP algorithms.

$n \times m$	DGSA	DABC_RCT	IG_RIS	VBIH	SAGA	DDE
10×5	0.0378	0.0033	0.0487	0.0002	0.4087	0.5301
10×10	0.0121	0.0056	0.0162	0.0001	0.2846	0.2956
10×20	0.0092	0.0000	0.0110	0.0001	0.1241	0.3011
50×5	0.2479	0.5018	0.6931	0.4649	5.1692	4.0236
50×10	0.3016	0.5379	0.6875	0.6601	4.9892	3.4615
50×20	0.2113	0.3856	0.4235	0.4491	3.4889	2.3698
100×5	0.3702	0.5784	0.4801	0.5732	9.8256	9.5987
100×10	0.3837	0.6425	0.7576	0.6109	9.9905	8.3221
100×20	0.3989	0.6783	0.7109	0.4324	7.5112	6.1008
200×10	0.2824	0.4668	0.5223	0.3568	9.6930	8.8486
200×20	0.3456	0.5179	0.4343	0.2467	9.4413	9.4579

5. CONCLUSIONS

This paper proposes a novel optimization algorithm for the BFSP with the objective of minimizing the TFT. In our algorithm, a high-quality population is initialized through a new heuristic method, which strikes a balance between quality and diversity. Besides, a DE-based method was adopted to update the acceleration, velocity and position of particles. The simulation results on the standard test set proved the superiority and effectiveness of our algorithm.

ACKNOWLEDGEMENT

This paper was supported by National Natural Science Foundation of China (61863005).

REFERENCES

[1] Xu, Z.; Xu, D.; He, J.; Wang, Q.; Liu, A.; Xiao, J. (2018). Mixed integer programming formulations for two-machine flow shop scheduling with an availability constraint, *Arabian Journal for Science and Engineering*, Vol. 43, No. 2, 777-788, doi:10.1007/s13369-017-2763-0

[2] Xuan, H.; Li, B. (2013). Scheduling dynamic hybrid flowshop with serial batching machines, *Journal of the Operational Research Society*, Vol. 64, No. 6, 825-832, doi:10.1057/jors.2012.64

[3] 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

[4] Zhao, F.; Liu, H.; Fan, J.; Chen, C. W.; Lan, R.; Li, N. (2018). Intuitionistic fuzzy set approach to multi-objective evolutionary clustering with multiple spatial information for image segmentation, *Neurocomputing*, Vol. 312, No. 27, 296-309, doi:10.1016/j.neucom.2018.05.116

[5] Kanagasabai, L. (2018). Reduction of real power loss by white male deer mating based optimization algorithm, *Ingénierie des Systèmes d'Information*, Vol. 23, No. 5, 61-68, doi:10.3166/ISI.23.5.61-68

[6] Xu, W.; Yin, Y. (2018). Functional objectives decision-making of discrete manufacturing system based on integrated ant colony optimization and particle swarm optimization approach, *Advances in Production Engineering & Management*, Vol. 13, No. 4, 389-404, doi:10.14743/apem2018.4.298

[7] Jiang, C.; Zhang, C.; Zhang, Y.; 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
[8] Xue, P.; Jiang, C. H.; Wei, W.; Lin, J. (2018). Optimization of the intelligent workshop control based on the improved group leadership optimization algorithm, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 690-701, doi:10.2507/IJSIMM17(4)CO16
[9] Zhang, H. P.; Ye, J. H.; Yang, X. P.; Muruve, N. W.; Wang, J. T. (2018). Modified binary particle swarm optimization algorithm in lot-splitting scheduling involving multiple techniques, *International Journal of Simulation Modelling*, Vol. 17, No. 3, 534-542, doi:10.2507/IJSIMM17(3)CO13
[10] Song, D.-L.; Zhang, J. (2013). Batch scheduling problem of hybrid flow-shop based on ant colony algorithm, *Computer Integrated Manufacturing Systems*, Vol. 19, No. 7, 1640-1647
[11] Wang, L.; Pan, Q.-K.; Suganthan, P. N.; Wang, W.-H.; Wang, Y.-M. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems, *Computers & Operations Research*, Vol. 37, No. 3, 509-520, doi:10.1016/j.cor.2008.12.004
[12] Li, X.; Zhang, Y. (2012). Adaptive hybrid algorithms for the sequence-dependent setup time permutation flow shop scheduling problem, *IEEE Transactions on Automation Science and Engineering*, Vol. 9, No. 3, 578-595, doi:10.1109/TASE.2012.2192729
[13] Bozorgirad, M. A.; Logendran, R. (2016). A comparison of local search algorithms with population-based algorithms in hybrid flow shop scheduling problems with realistic characteristics, *The International Journal of Advanced Manufacturing Technology*, Vol. 83, No. 5-8, 1135-1151, doi:10.1007/s00170-015-7650-9
[14] Song, X.; Gao, S.; Chen, C. (2018). A novel vehicle feature extraction algorithm based on wavelet moment, *Traitement du Signal*, Vol. 35, No. 3-4, 223-242, doi:10.3166/TS.35.223-242
[15] Shao, W.; Pi, D. (2016). A self-guided differential evolution with neighborhood search for permutation flow shop scheduling, *Expert Systems with Applications*, Vol. 51, No. 1, 161-176, doi:10.1016/j.eswa.2015.12.001
[16] Fernandez-Viagas, V.; Leisten, R.; Framinan, J. M. (2016). A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime, *Expert Systems with Applications*, Vol. 61, 290-301, doi:10.1016/j.eswa.2016.05.040
[17] Ribas, I.; Companys, R.; Tort-Martorell, X. (2015). An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flowtime minimization, *Expert Systems with Applications*, Vol. 42, No. 15-16, 6155-6167, doi:10.1016/j.eswa.2015.03.026
[18] Han, Y.; Gong, D.; Jin, Y.; Pan, Q.-K. (2016). Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time, *Applied Soft Computing*, Vol. 42, 229-245, doi:10.1016/j.asoc.2016.01.033
[19] Pan, Q.-K.; Tasgetiren, M. F.; Suganthan, P. N.; Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, *Information Sciences*, Vol. 181, No. 12, 2455-2468, doi:10.1016/j.ins.2009.12.025
[20] Liao, C.-J.; Tjandradjaja, E.; Chung, T.-P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem, *Applied Soft Computing*, Vol. 12, No. 6, 1755-1764, doi:10.1016/j.asoc.2012.01.011
[21] Dai, M.; Tang, D.; Giret, A.; Salido, M.; Li, W. D. (2013). Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm, *Robotics and Computer-Integrated Manufacturing*, Vol. 29, No. 5, 418-429, doi:10.1016/j.rcim.2013.04.001
[22] Trabelsi, W.; Sauvey, C.; Sauer, N. (2012). Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems, *Computers & Operations Research*, Vol. 39, No. 11, 2520-2527, doi:10.1016/j.cor.2011.12.022
[23] Tasgetiren, M. F.; Kizilay, D.; Pan, Q.-K.; Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion, *Computers & Operations Research*, Vol. 77, 111-126, doi:10.1016/j.cor.2016.07.002
[24] Li, J.-Q.; Pan, Q.-K.; Mao, K.; Suganthan, P. N. (2014). Solving the steelmaking casting problem using an effective fruit fly optimisation algorithm, *Knowledge-Based Systems*, Vol. 72, 28-36, doi:10.1016/j.knosys.2014.08.022

[25] Sun, G.; Bin, S. (2017). Router-level internet topology evolution model based on multi-subnet composited complex network model, *Journal of Internet Technology*, Vol. 18, No. 6, 1275-1283, doi:10.6138/JIT.2017.18.6.20140617

[26] Precup, R.-E.; David, R.-C.; Petriu, E. M.; Preitl, S.; Radac, M.-B. (2012). Novel adaptive gravitational search algorithm for fuzzy controlled servo systems, *IEEE Transactions on Industrial Informatics*, Vol. 8, No. 4, 791-800, doi:10.1109/TII.2012.2205393

[27] Zeng, R.-Q.; Basseur, M.; Hao, J.-K. (2013). Solving bi-objective flow shop problem with hybrid path relinking algorithm, *Applied Soft Computing*, Vol. 13, No. 10, 4118-4132, doi:10.1016/j.asoc.2013.05.018

[28] Valente, J. M. S.; Alves, R. A. F. S. (2008). Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups, *Computers & Operations Research*, Vol. 35, No. 7, 2388-2405, doi:10.1016/j.cor.2006.11.004

[29] Li, X.; Ma, S. (2017). Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times, *IEEE Transactions on Engineering Management*, Vol. 64, No. 2, 149-165, doi:10.1109/TEM.2016.2645790

[30] Mansour, N.; El-Fakih, K. (1999). Simulated annealing and genetic algorithms for optimal regression testing, *Journal of Software Maintenance: Research and Practice*, Vol. 11, No. 1, 19-34, doi:10.1002/(SICI)1096-908X(199901/02)11:1<19::AID-SMR182>3.0.CO;2-M

[31] Deng, G.; Gu, X. (2012). A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion, *Computers & Operations Research*, Vol. 39, No. 9, 2152-2160, doi:10.1016/j.cor.2011.10.024