Applied Intelligence (2019) 49:3362–3382
https://doi.org/10.1007/s10489-019-01457-w

A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization

Fuqing Zhao[1] · Feilong Xue[1] · Yi Zhang[2] · Weimin Ma[3] · Chuck Zhang[4] · Houbin Song[1]

Published online: 9 April 2019

Abstract
The blocking flow shop problem (BFSP) is one of the key models in the flow shop scheduling problem in the manufacturing systems. Gravitational Search Algorithm (GSA) is an algorithm based on the population for solving various optimization problems. However, GSA is scarcely applied to solve the BFSP as it is designed to solve the continuous problems. In this paper, a Discrete Gravitational Search Algorithm (DGSA) is presented for solving the BFSP with the total flow time minimization. A new variable profile fitting (VPF) combined with NEH heuristic, named $VPF\_NEH(n)$, is introduced for balancing the quality and the diversity of the initial population to configure the DGSA. The three operators including the variable neighborhood operators (VNO), the path relinking and the plus operator are implemented during the location updating of the candidates. The objective of the operation is to prevent the premature convergence of the population and to balance the exploration and exploitation in the process of optimization. The expected runtime of the DGSA is analyzed by the level-based theorem. The simulated results indicate that the effectiveness and superiority of the DGSA.

Keywords Gravitational search algorithm · Blocking flow shop problem · Total flow time · Constructive heuristic · Variable neighborhood search

1 Introduction

The permutation flow shop scheduling (PFSP) is a typical model for various manufacturing and service systems when there are no buffers between consequent machines [1, 2]. In general, the objective of the PFSP is to find the best sequence of jobs on the machines with the minimization makespan. For the PFSP, if the buffer storage capacity among consequent machines is not available, the problem becomes a blocking flow shop scheduling problem [3]. In the BFSP, a machine is blocked by a job that has been processed if the next machine is occupied. Hence, an effective scheduling sequence increases the productivity by minimizing machine blocked and idle time. In the real world, the various industrial production processes are modeled as the BFSP, such as iron and steel industry [4], chemical and pharmaceutical industry [5], and in-line robotic cells [6]. Over the past years, the BFSP has attracted increasing researches and practitioners in endeavoring to construct satisfactory models and solution methods [7].

In Hall, Sriskandarajah [5], the BFSP is proven to be a strongly NP-hard problem when $m > 2$. Various heuristics and meta-heuristics are developed for solving the

✉ Fuqing Zhao
Fzhao2000@hotmail.com

Feilong Xue
767098126@qq.com

Yi Zhang
1049440546@qq.com

Weimin Ma
mawm@tongji.edu.cn

Chuck Zhang
chuck.zhang@isye.gatech.edu

Houbin Song
523712418@qq.com

[1] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China

[2] School of Mechnical Engineering, Xijin University, Xi'an 710123, China

[3] School of Economics and Management, Tongji University, Shanghai 200092, China

[4] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

BFSP by numerous researchers and practitioners. For the heuristic, a profile fitting (PF) heuristic for solving the BFSP with the minimization cycle time was presented in Mccormick et al. [8]. Fernandez-Viagas et al. [9] presented a constructive heuristic based on beam search for solving the BFSP with minimizing the total flow time. In Ribas, Companys [7], the authors proposed two heuristics, named HPF1 and HPF2, for the BFSP with the total flow time minimization. Pan, Wang [10] proposed various heuristics based on the PF method for solving the BFSP with the makespan minimization. Regarding the meta-heuristic, Ribas et al. [11] proposed a discrete artificial bee colony algorithm, named DABC_RCT, for the BFSP with the total flow time minimization. In Nouha, Talel [12], the authors proposed a particle swarm optimization meta-heuristic for the BFSP with the total tardiness minimization. A scatter search algorithm for the mixed BFSP with the makespan minimization is proposed by Riahi et al. [13]. Other meta-heuristics were proposed for solving the BFSP such as a memetic algorithm [14], an artificial immune system [15], a simulated annealing algorithm with three-phase operators [16] and a hybrid harmony search [17].

The evolution criterion, which used as the objective of the BFSP, is the makespan minimization. However, there are other related criteria such as the total tardiness and the total flow time in the application. The makespan minimization criterion is found in various literatures. Two mixed binary integer programming models are presented in Moslehi, Khorasanian [18]. Three versions of the hybrid iterated greedy algorithm for the distributed BFSP are introduced by Ying, Lin [19]. An iterated greedy algorithm is developed in Tasgetiren et al. [20]. A modified fruit fly optimization algorithm is presented in Han et al. [21] and a population-based local search with differential evolution is developed in Tasgetiren et al. [22]. The total tardiness criterion is aimed at minimizing the waiting times of jobs among machines to finish the manufacturing processes as early as possible [23]. Regarding the total tardiness criterion, a multi-objective discrete invasive weed optimization for the multi-objective BFSP is developed in Shao et al. [24]. A self-adaptive discrete invasive weed optimization is presented in Shao et al. [25]. Nouri, Ladhari [26] presented a hybrid meta-heuristic for the BFSP. A branch and bound algorithm is introduced in Toumi et al. [27]. For the total flow time criterion, an iterated greedy algorithm is presented in Khorasanian, Moslehi [28] and other literatures are found in [7, 9, 11].

In the past decades, evolutionary algorithms (EAs) play a significant role in solving optimization problems, especially in complex problems. EAs are inspired by biological systems or physical processes [29]. EAs include particle swarm optimization (PSO) [30], differential evolution algorithm (DE) [31] and other typical hybrid evolutionary algorithms [32–36]. Meanwhile, certain novel algorithms were proposed by different researchers including Quasi-Affine Transformation Evolutionary (QUATRE) algorithm [37] and Jaya algorithm [38]. The experiment results show that the performances of QUATRE algorithm and Jaya algorithm are better than the correspondingly compared algorithms. In numerous evolutionary algorithms, a new population-based intelligent optimization algorithm that has been successfully applied to various optimization problems, named Gravitational Search Algorithm, was proposed by Rashedi et al. [39]. A comprehensive survey of the GSA published up to 2017 and its applications were found in Rashedi et al. [40]. In Zhao et al. [41], a hybrid GSA, named SGSADE, is presented to solve the single-objective real parameter optimization problem. The simulated results of SGSADE showed that the SGSADE outperformed the compared algorithms. Choudhary et al. [42] proposed a hybrid GSA for bi-objective workflow scheduling in cloud computing, named HGSA. The simulations through analysis of variance show that HGSA outperformed the compared algorithms in all cases. Lee et al. [43] applied GSA to the bi-objective flow shop scheduling using the weighted dispatching rules. In Narang [44], an integrated gravitational search algorithm predator-prey optimization technique was presented for solving the hydro-thermal generation scheduling problem. Özyön, Yaşar [45] applied GSA to a fixed head hydrothermal power system with transmission line security constraints. Pelusi et al. [46] presented the design of a Neural and Fuzzy GSA, named NFGSA. The experiment results on benchmark set showed that NFGSA outperformed the compared algorithms. In Mittal, Saraswat [47], the authors presented an exponential *Kbest* GSA for multi-level image threshold segmentation.

In summary, GSA and its variants have been successfully applied to the fields of the real applications and scheduling domains. However, little literature presents GSA or its variants to solve the BFSP. In this paper, a discrete GSA, named DGSA, is proposed for solving the BFSP with the total flow time minimization. Usually, the constructive heuristic method is an effective method of initialing population for solving the BFSP. The excellent initial population enhances the performance of the algorithm. In DGSA, a new variable profile fitting combined with the NEH heuristic, named *VPF_NEH*(*n*), is applied to generate the initial population with balancing the quality and the diversity. The VNO is a significant method which has an effect

on the performance of algorithms. In the current researches, the insert operator and swap operator are the common neighborhood operation for solving the BFSP. However, the different neighborhood structures have the inherent drawback in various instances. In the position update phase of DGSA, three operators (the variable neighborhood operators, the path relinking and the plus operator) are applied to update the position of the candidate to enhance the exploitation and construct the self-improvement method. The simulated results of DSGA on the Taillard's benchmark suite [48] show that the performance of the DGSA outperform the state-of-art algorithms.

The remainder of the paper is organized as follows. The related work is described in Section 2. Section 3 gives a detailed introduction of the proposed DGSA. Section 4 shows the parameter calibration. The experimental results are introduced in Section 5. Section 6 gives the conclusions and the future work.

2 Related work

The notations in this paper are described as follows.

n	the number of jobs
m	the number of machines
π	a processing sequence of jobs on the machine
X_i	the position of the ith agent
xid	the position of ith agent in the dth dimension
vid	the velocity of the ith agent in the dth dimension
$fit_i(t)$, $M_i(t)$	the fitness and the mass of the ith agent at the current tth iteration
$R_{i,\ j}(t)$	the Euclidian distance between agent i and agent j at the current tth iteration
ε	a small constant
$G0$	the initial value of the gravitational constant
$G(t)$	the value of the gravitational constant at tth iteration
aid	the acceleration of ith agent in the dth dimension
num	the number of the initial solutions
$HR_{i,\ j}(t)$	the hamming distance between the sequence i and j in the iteration t
ra	the control parameter of the acceleration update formula
vc	the control parameter of the velocity update formula

2.1 Blocking flow shop scheduling problem

The BFSP is modeled as $Fm|block|\sum C_i$ based on the notation proposed by Graham et al. [49]. In the BFSP, a set of n jobs are processed on m machines with the same order from the first machine to the last machine without an intermediate buffer. Each job $i, i \in \{1, 2, \ldots, n\}$, has a fixed positive processing time on every machine j, $j \in \{1, 2, \ldots, m\}$, which is denoted as $p_{i,\ j}$. In the BFSP, if the job i has been completed on the machine j and the machine $j+1$ is being used, then the job i will be blocked on the machine j until the machine $j+1$ is idle. Only one job is processed on each machine in the same time. In this paper, the objective is to find a permutation of a job that minimizes the total flow time (TFT). TFT is concluded according to the following equations.

$$D_{1(\pi),0} = 0 \tag{1}$$

$$D_{1(\pi),k} = D_{1(\pi),k-1} + p_{1(\pi),k} \qquad k = 1, \ldots, m-1 \tag{2}$$

$$D_{j(\pi),0} = D_{(j-1)(\pi),1} \qquad j = 2, \ldots, n \tag{3}$$

$$D_{j(\pi),k} = max\left\{D_{j(\pi),k-1} + p_{j(\pi),k}, D_{(j-1)(\pi),k+1}\right\} j = 2, \ldots n; k = 1, \ldots, m-1 \tag{4}$$

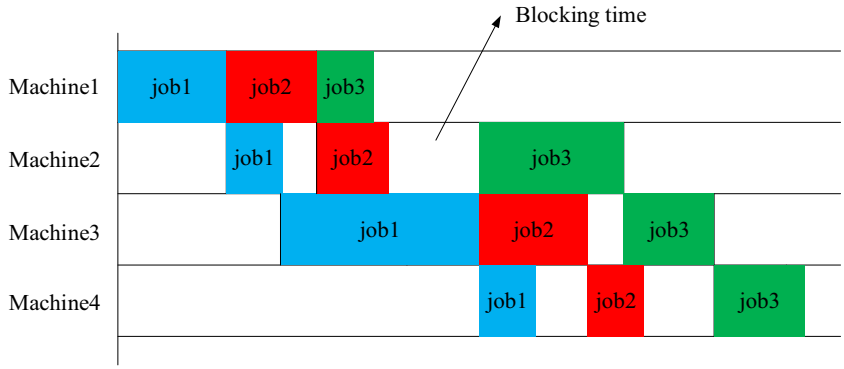
$$D_{j(\pi),m} = D_{j(\pi),m-1} + p_{j(\pi),m} \quad j = 1, \ldots, n \tag{5}$$

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

where $D_{j(\pi),\ 0}, j = 1, \ldots, n$ denotes the starting time of jth job on the first machine in the sequence π, $D_{j(\pi),\ k}, k = 1, \ldots, m$ is the departure time of jth job on kth machine. In this paper, the objective is to find a sequence π^* to satisfy the inequality $TFT(\pi^*) \le TFT(\pi) \quad \forall \pi \in T$, T donates the set of all sequences π. The Gantt chart of the BFSP is show in Fig. 1(An instance of the BFSP is shown in Fig. 1).

In the scheduling problems, insertion and swap neighborhood structures are two basic neighborhood structures and the computational complexity of both neighborhood structures is $O(n^3m)$. Based on Li et al. [50], a speed-up method for the TFT calculation of the BFSP is proposed in this paper. Suppose that a sequence $\pi = \{1, 2, 3, 4, 5\}$. First, the departure time matrix donated as $d_{\pi,\ k}(k = 1, 2, 3, 4, 5)$ is calculated for π. Second, a new sequence $\pi 1 = \{1, 2, 5, 3, 4\}$ is obtained after insert operator. Due to the position of job1 and

Fig. 1 An instance of the BFSP



job2 aren't changed, the departure time until position 3 isn't repeated calculation. Finally, in order to calculate the TFT ($\pi 1$), the departure time starting from position 3 to 5 are calculated. The details about the fast TFT calculations are shown in Algorithm 1.

Algorithm 1 Procedure Fast TFT Calculation($\pi, \pi 1, pos, d_{\pi,m}$)

$\pi 1$ is the changed sequence based on π through insert or swap operator and pos is the changed position. $d_{\pi,m}$ is the departure time matrix for π according to equation 1-equation 5.

if pos = 1 do
  Compute TFT of $\pi 1$ and $d_{\pi 1,m}$
else
  For i = 1 to pos-1 do
    $d_{\pi 1(i),m} = d_{\pi(i),m}$
  End for
  For i = pos to n
    For j = 1 to m-1
      $d_{\pi 1(i),j} \leftarrow$ calculate the departure time of job $\pi 1(i)$ on the machine j
    End for
    $d_{\pi 1(i),m} = d_{\pi 1(i),m-1} + p_{\pi 1(i),m}$
  End for
TFT $\leftarrow d_{\pi 1,m}$
Return TFT and $d_{\pi 1,m}$

2.2 Gravitational search algorithm

The GSA is a novel meta-heuristic algorithm that is inspired by Newton's law of gravity and motion. In the GSA, each agent presents a solution of the problem. The main steps of the standard GSA are described as follows. First, the initialized population is randomly generated.

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^D\right) \quad for\ i = 1 \ldots N; \tag{7}$$

Second, the mass of each agent is calculated by the fitness of the current population.

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{8}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)} \tag{9}$$

For a minimization problem, *worst*(*t*) and *best*(*t*) are defined as follows.

$$best\ (t) = fit_j(t) \tag{10}$$

$$\text{worst}(t) = \max_{j\in\{1\ldots N\}} fit_j(t) \tag{11}$$

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{16}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{17}$$

where $rand_i$is a random variable in the interval [0, 1].

The pseudo code of the standard GSA is given in Algorithm 2.

Algorithm 2 The GSA

```
Randomly initialize the population in the search space
t = 0
While (t < T)
    Evaluate the fitness of all agents
    Update worst (t), best (t) and M_i(t) based on equation 8, equation 9, equation 10, and equation 11
    Update G(t) based on equation 13
    Evaluate the total force in different dimension based on equation 12 and equation 14
    Update acceleration and velocity based on equation 15 and equation 16
    Update agent's position based on equation 17
End While
Output Result
```

Third, at iteration time *t*, the force acting on *ith* agent from *jth* agent in *d*th dimension is defined as follows.

$$F_{ij}^d(t) = G(t)\frac{M_i(t) \times M_j(t)}{R_{i,j}(t) + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{12}$$

$G(t)$ (is)

$$\mathrm{G(t)} = \mathrm{G0} \times e^{-\alpha\frac{t}{T}} \tag{13}$$

The total force of all agents acting on the *i*th agent in the *d*th dimension is defined as follows.

$$F_i^d(t) = \sum_{j\in Kbest, j\neq i} rand_j F_{ij}^d(t) \tag{14}$$

In the beginning, all agents apply the force as time pass, and *Kbest* is linearly decreased. There are only two agents applying force to the others at the end. Finally, the acceleration, the velocity and the position are updated.

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{15}$$

3 The proposed discrete GSA

3.1 Solution representation

In the BFSP, a natural representation is the permutation of *n* jobs. In the permutation, the *ith* number denotes the job arranged at position *i*. In the DGSA, an agent at the *tth* iteration is denoted as $X^t = \left[x_1^t, x_2^t, \ldots, x_n^t\right]$, which x_i^t is the index of the job that arranged at position *i*.

3.2 Population initialization

In this paper, the DGSA starts with the generation of *num* initial solutions. Each agent is donated as a sequence of the job and the quality of the agent is evaluated by TFT. In the past decades, the three methods are proposed for generating initial solutions in the literature. The first, Han et al. [51] and Wu et al. [52] proposed a random generation of the initial solutions to guarantee the diversification of the solutions. The second, one solution with a certain quality is generated by a heuristic procedure [18, 20, 23]. The last, one solution is

obtained by a based problem heuristic search method and the remaining solutions or other methods [11, 21] are randomly generated to guarantee the quality and diversity of the initial solutions.

A new method is proposed for the initial solutions based on the above three methods. The details of the new method are shown as follows.

In this paper, the *VPF_NEH*(n) heuristic is proposed to generate the initial solutions for the DGSA. It is a novel modification of the *tPF_NEH*(x) heuristic that has been presented in Tasgetiren et al. [23]. In the *tPF_NEH*(x), an initial order of jobs based on x jobs as the separate first job is determined in the starting. However, in the *VPF_NEH*(n), the initial order of jobs is different with *tPF_NEH*(x). The following equation is employed to establish the initial order of jobs as in Ribas et al. [11]. In this paper, the constructive procedure to create a solution is named *VPF*.

$$ind(i,k) = \mu * \left(\sum_{j=1}^{m} \left(D_{k+1,j}(\sigma * i) - D_{k,j}(\sigma) - p_{i,j} \right) \right) + (1-\mu) * \left(C_i - C_{[k-1]} \right) \quad (18)$$

The generate steps of the initial order of jobs are described as follows.

Step 1: According to the number of jobs, a sequential integer sequence is generated. Set $t = 1, \ldots, n$, the *tth* job of the sequence is selected as the first position of the new sequence σ. Set $k = 1$.

Step 2: When $k < n$, *ind* is calculated based on eq. 18 for each unscheduled job i. Select the job with minimum *ind*. In case of ties, the job which leads to the partial sequence with the minimum TFT is selected.

After the generation of the initial order of jobs, the NEH heuristic [53] to the partial sequence is applied to generate the initial solutions as in Tasgetiren et al. [23]. The details of *VPF_NEH*(n) are shown in Algorithm 3.

Algorithm 3 The procedure of $VPF\_NEH(n)$

```
Denote π as the initial solution which is a sequential integer sequence from 1 to n.
For (k = 1 to n) do
    Take π□ as the first job of π
    Generate a sequence by VPF heuristic π1 = VPF（π_h）
    For (i = n − δ + 1） to n) do
        Select job π_i from π1
        Insert π_i in all possible position of π1 and evaluate TFT
    End for
    Obtain the sequence π2 with the lowest TFT
    If (TFT(π2) < TFT(π1)) π^k = π2;else π^k = π1
End for
Return a set of sequence {π^1, π^2,…, π^n} and corresponding TFT
```

In the DGSA, the population is constructed based on the *VPF_NEH*(n) heuristic. The number of the population is determined as if ($n \leq 20$), $num = 20$; *else* $num = 50$. Based on the results of the *VPF_NEH*(n), $TFT(\pi^k)(k = 1, \ldots, n)$ is sorted with an increasing order. Top *num* sequences are selected as the initial solutions in this paper. The quality and diversity of the population are guaranteed by the *VPF_NEH*(n) heuristic mechanism.

3.3 New particles

In Wang, Tang [54], the simple principle of the updated model of velocity based on a list of moves is introduced in the discrete particle swarm optimization algorithm to solve the permutation flow shop problem with blocking. According to the mechanism, the updated methods of the acceleration, the velocity and the position in the DGSA are modified in this paper.

3.3.1 Acceleration calculation

Based on eq. 12–15, the updated formula of the acceleration is modified.

$$a_i(t) = \sum_{j \in kbest, j \neq i} \frac{HR_{i,j}(t) * rand * G(t) * M(j)}{ra + \frac{HR_{i,j}(t)}{n}} \otimes \left(x_j(t) \ominus x_i(t)\right) \tag{19}$$

where *rand* is a uniformly distributed random variable in the interval [0, 1]. $G(t)$ is the gravitational coefficient. $M(j)$ is the value of the sequence j after normalization.

The definitions of the operators and the notations used in eq. 19 are as follows.

(1) The subtract operator(⊖)

The details of this operator are shown in Fig. 2(a). If the *jth* position values of $x_i(t)$ and $x_j(t)$ are same, the *jth* position values are set to 0; otherwise, it is set to be the *jth* position value of $x_j(t)$. If $HR_{i,j}(t)$ is zero, all values of the array are set to 0.

(2) The multiply operator(⊗)

Let $k = \left\lfloor \frac{HR_{i,j}(t) * rand * G(t) * M(j)}{ra + \frac{HR_{i,j}(t)}{n}} \right\rfloor$ (if $k > HR_{i,j}(t)$, $k = HR_{i,j}(t)$). Then k non–zero elements from $(x_j(t) \ominus x_i(t))$ are randomly selected, and other elements are set as zero. The details are shown in Fig. 2(b).

(3) The summation operator(∑)

According to step 1 and step 2, the *kbest* arrays are obtained. The summation operator is used to combine these arrays into one array as the acceleration. The procedures of combination are shown in Algorithm 4 and Fig. 2(c). In Fig. 2(c), the second, the third and the ninth position is 0 because *job*1 and *job*6 repeatedly appeared.

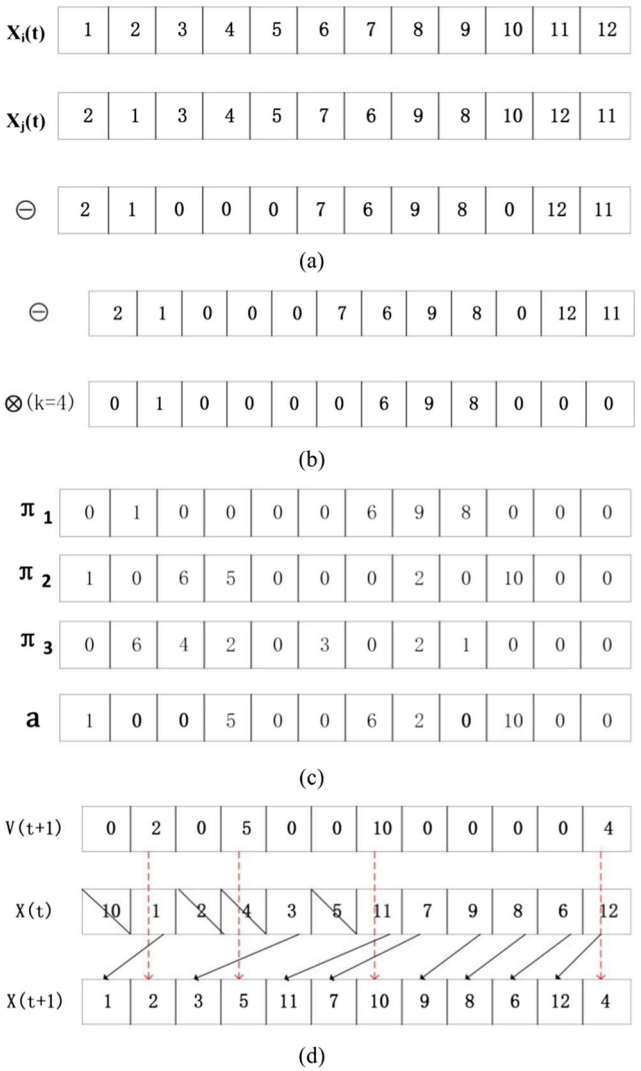


Fig. 2 The illustration of the operators. (a) The subtract operator. (b) The multiply operator. (c) An example of a combination. (d) The position update operator

Algorithm 4 Procedure of combination

A set of sequences $\{\pi_j\}$ $(j = 1, \ldots, kbest)$
For $i = 1$ to n do
    $k \leftarrow$ randomly select an integer from $[1, \ldots, kbest]$
    $a[i] = \pi_k[i]$
End for
If a has multiple identical elements do
    randomly keep one element and set other elements to be zero
End if
Obtain acceleration a

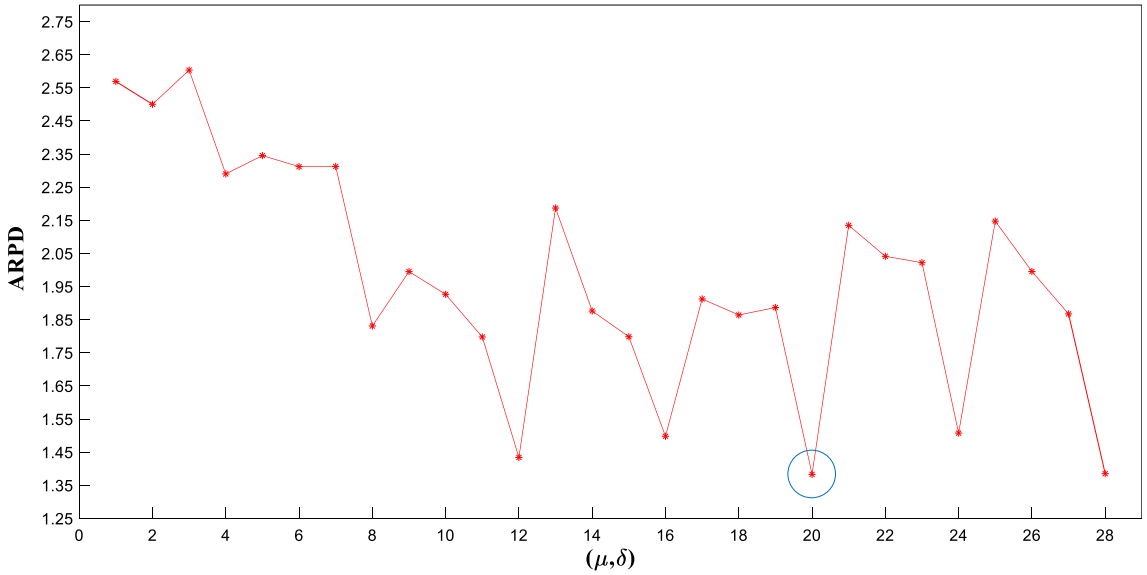


Fig. 3 The ARPD of the combination μ and δ

3.3.2 Velocity calculation

Based on eq. 16, the velocity formula is modified as follows.

$$v_i(t+1) = v_i(t) \odot a_i(t) \tag{20}$$

The notation $\odot$ is used to update the velocity of the agent. Two steps are included in this operator.

Step 1. Set vc is a constant ($0 < vc < 1$). For $j = 1, \ldots, n$, if $rand < vc, v_i(t+1)[j] = v_i(t)[j]$, else $v_i(t+1)[j] = a_i(t)[j]$.

Step 2. If $v_i(t+1)$ has multiple identical elements, the same operation is applied as the summation operator.

3.3.3 Position operation

According to eq. 17, the updated formula is modified.

$$x_i(t+1) = x_i(t) \oplus v_i(t+1) \tag{21}$$

The position update operator ($\oplus$) has three different operations. The operators are shown as follows.

Table 1 The factor level of the parameters

Factor level	1	2	3	4
bs	1	2	3	4
bse	10	12	16	18
msc	0.1	0.2	0.3	0.4
t	1	2	3	4
τP	0.1	0.2	0.3	0.4

(1) A particle is the same as the best particle in the population, namely, two sequences are the same particles. A local search method, named variable neighborhood operators (VNO), is applied in the DGSA. The details about the VNO are introduced in Section 3.4.

(2) A self-adaptive perturbation operator is applied in the position update operator to prevent the DGSA from the premature convergence. The current population is dispersed when necessary. In Wang, Tang [54], the authors proposed the diversity coefficient based on the hamming distance between two sequences, which is a measure of the diversity of the current population. In this paper, a new diversity coefficient based on the DGSA is proposed. The formula is given as follows.

$$div = \frac{\sum_{i=1}^{num}\sum_{j=1}^{j=kbest} HR\left(x_i(t), x_j(t)\right)}{kbest*num*n} \tag{22}$$

where $HR(x_i(t), x_j(t))$ is the hamming distance between the sequence i and j in the iteration t. Since $HR(x_i(t), x_j(t))$

Table 2 The rank of the parameter

Level	Parameters				
	bs	*bse*	*msc*	*t*	τP
1	0.684	0.656	0.631	0.537	0.673
2	0.685	0.684	0.707	0.592	0.661
3	0.708	0.696	0.714	0.696	0.699
4	0.621	0.662	0.647	0.875	0.666
Std.	0.03735	0.018717	0.041892	0.148744	0.016899
Rank	3	4	2	1	5

is within $[0, n]$, the value of the diversity coefficient div is within $[0, 1]$. According to eq. 22, the diversity of the current population keeps as previous status when div exceeds a given threshold. If div is close to zero, the current population are trapped in local optimum or as the destination, and the improved sequence is the path origin. The path is built by inserting movements (include forward insertion and backward insertion) in order to convert the original sequence into the superior sequence. The details of the path relinking are outlined in Algorithm 5.

Algorithm 5 The procedure of path relinking

```
Select the best sequence π* and the original sequence π.
For i = 1 to n do
    If π(i) ≠ π*(i) do
        For k = 1 to n do
            Find π(k) = π*(i), key = k, key value = π(k)
        End for
        If (key > i) ( forward insertion) do
            Generate a sequence π1 by inserting key value into the front of π(k)
            Evaluate TFT of new sequence π1
            If (TFT(π1) < TFT(π) ) do
                π =π1; TFT(π) = TFT(π1)
            End if
        Else（backward insertion）
            Generate a sequence π1 by inserting key value into the back of π(k)
            Evaluate TFT of new sequence π1
            If (TFT(π1) < TFT(π) ) do
                π =π1; TFT(π) = TFT(π1)
            End if
        End if
    End if
End for
Return the best sequence with the lowest TFT
```

all particles are in the same sequence. Based on the diversity coefficient, the self-adaptive perturbation probability sp is calculated by $sp = e^{-K * div}$, in which K is a control parameter. The current population has the tendency to fall in premature convergence, the sp increases as the div decreases. If $rand \leq sp$, a strategy called path relinking is introduced in this phase.

In Glover, Laguna [55], the path relinking is proposed to explore the path between two sets of solutions as a search technique. In the DGSA, the path relinking is used to disperse the current sequence when $rand \leq sp$. Select the best sequence

(3) If $rand > sp$, the position update operator is applied based on Wang, Tang [54]. The procedure of the position update operator is shown in Fig. 2(d).

3.4 The variable neighborhood operators (VNO)

The variable neighborhood search (VNS) is proposed by Mladenović, Hansen [56], and it has been applied to the scheduling problems [34, 57, 58]. The VNS is an effective local search strategy when a sequence is trapped in

a state of stagnation. In Ribas et al. [11], the authors proposed a new scheme named Three Neighborhood Operators (TNO). The results of experiments show that the TNO is a competitive local search strategy. In this paper, a hybrid local search operator based on the TNO and the block insertion move operators (BIM) from Tasgetiren et al. [23] is proposed to enhance exploitation capacity of the DGSA, named VNO. The details of the VNO are shown as follows.

(1) The selected sequence is processed by the BIM in the first. The procedure of operators is described in Algorithm 6.

(2) After the BIM operation, the selected sequence is operated by the TNO. The procedure of operators is shown in Algorithm 7, Algorithm 8 and Algorithm 9.

After the VNO, a simple simulated annealing type of acceptance criterion is applied to determine whether the new sequence is accepted, which is suggested in Tasgetiren et al. [23]. The formula is shown in eq. 23.

$$T = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{i,j}}{10*n*m} * \tau P \tag{23}$$

Algorithm 6 The procedure of BIM for the selected sequence π^*

1 Initial parameters, the block size in the beginning (bs), the block size in the end (bse), the maximum move size ($ms = msc * (n - bse)$) (msc is the move size coefficient)
2 Do
3 $\pi = \pi^*$
4 $temp = MAX\_VALUE$
5 p = choose a position in π randomly (guarantee $a + bs < n$)
6 π^b = remove the block of jobs with size bs start from position p from π
7 For i = 1 to ms do
8 pt = choose a position in π randomly (guarantee $pt < p \,\&\&\, pt > p + bs$, pt isn't in the π^b)
9 $\pi 1$ = insert π^b in position pt of π
10 If (TFT($\pi 1$) < $temp$) do
11 $\pi = \pi 1$
12 $temp$ = TFT($\pi 1$)
13 End if
14 End for
15 TFT(π) = $temp$
16 If (TFT(π) < TFT(π^*)) do
17 $\pi^* = \pi$
18 TFT(π^*) = TFT(π)
19 else
20 $bs = bs + 1$
21 End if
22 While ($bs \leq bse$)
23 Return π^* and TFT(π^*)

Algorithm 7 The three operators

PI (pair exchange)
    Set a sequence π, two positions, k1 and k2 are randomly selected
    Exchange $\pi(k1)$ and $\pi(k2)$
    Obtain a new sequence $\pi 1$
FI (Forward Insertion)
    Set a sequence π, two positions, k1, and k2 are randomly selected (if $k1 < k2$)
    Inserting $\pi(k2)$ into the front of $\pi(k1)$
    Obtain a new sequence $\pi 2$
BI(Backward Insertion)
    Set a sequence π, two positions, k1, and k2 are randomly selected (if $k1 < k2$)
    Inserting $\pi(k2)$ into the back of $\pi(k1)$
    Obtain a new sequence $\pi 3$

Algorithm 8 The two local search operators

LS1(based on the swap neighborhood structure)
Select a sequence π,
Do
    a job is selected randomly as $\pi(a)$
    Obtain $\pi 1$ by swapping $\pi(a)$ with all jobs that follow it in π
    If TFT($\pi 1$) < TFT(π)
        $\pi = \pi 1$
        TFT(π)= TFT($\pi 1$)
    End if
While (all jobs have been considered)
LS2(based on the insert neighborhood structure)
Select a sequence π,
Do
    a job is selected randomly as $\pi(a)$
    Obtain $\pi 2$ by inserting $\pi(a)$ with all possible positions in π
    If TFT($\pi 2$) < TFT(π)
        $\pi = \pi 2$
        TFT(π)= TFT($\pi 2$)
    End if
While (all jobs have been considered)

Algorithm 9 The procedure of TNO

```
Select a sequence π. Set nml = 1.
For j = 1 to t
    k1 and k2 are randomly selected from 1 to n (k1 ≠ k2)
    π is operated by the three operators
    Keep a sequence π with the lowest TFT
End for
If (rand < 0.5) then
    indmet = 0
Else indmet = 1
do
    nml = nml + 1
    If indmet = 0 then
        π is operated by the LS1
    else
        π is operated by the LS1
    End if
    Obtain a new sequence π*
    If TFT(π*) < TFT(π) or nm = 1 then
        indmet = 1 − indmet
    Else exit do
While (true)
```

where τP is a control parameter.

The complete procedure of the DGSA is given in Algorithm 10. The fast TFT calculation method is implemented where the swap operators and insertion operators are executed.

3.5 Expected runtime of the DGSA

The time complexity of the evolutionary algorithm (EA) is the number of fitness evaluation or the number of iteration to find the optimal or approximate solution [59]. In the real world, the time consuming of EA is spent on the fitness evaluation, instead of the evolution operator (crossover, mutation, and selection). In this paper, the upper bound of the expected runtime of the DGSA is shown based on Goryajnov [60] and Sudholt [61]. For convenience, certain definitions are shown as follows.

Definition 1 The population set is defined as $\text{pop} = \{I_1, I_2, \ldots, I_N\}$, the population state $\text{S} = \{x_1, x_2, \ldots, x_N\}$, where $x_1, x_2, \ldots, x_N$ is the state of $I_1, I_2, \ldots, I_N$, respectively. N is the number of the population.

Definition 2 The population state space consists of all possible states of the population, $\mathbf{SP} = \{S_i = (x_{i1}, x_{i2}, \ldots, x_{iN}) | \, i = 1, 2, \ldots, N\}$.

Definition 3 The state transition probability, from x_1 to x_2 in one step, is recorded as $x_1 \rightarrow x_2$, where x_1 and x_2 are random two states in the individual state space.

Definition 4 The state transition probability, from $S_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$ to $S_j = (x_{j1}, x_{j2}, \ldots, x_{jN})$ in one step, is recorded as $\text{P}(S_i \rightarrow S_j)$.

Lemma 1 In DGSA, the population state sequence

$$\{S(t) | t \geq 0\} \tag{24}$$

is the finite homogeneous Markov chains.

Algorithm 10 The procedure of DGSA

According to $VPF\_NEH(n)$, initialize the population of num agents
Set termination condition
While (Not Termination) do
    Set it = current iterations
    Set $t0 = \frac{it}{it+vt*n}$
    Calculate the best value, the worst value, and the corresponding sequence
    Calculate the gravitational coefficient $G(it) = (1 - t0) + G0$
    Calculate the $kbest$ = round $(\frac{2+(1-t0)*(100-2)*num}{100})$
    Calculate the $bestR \leftarrow$ the hamming distance between the best sequence and each sequence
    Based on section 3.3.1 and 3.3.2, calculate the acceleration and the velocity of the population
    For i = 1 to num do
        If $bestR(i) = 0$ do
            Implement VNO, obtain new sequence $\pi(i)^*$
            If TFT($\pi(i)^*$) < TFT($\pi(i)$) do
                $\pi(i) = \pi(i)^*$; TFT($\pi(i)$) = TFT($\pi(i)^*$)
            Else if $rand < exp\{\frac{-(TFT(\pi(i))-TFT(\pi(i)^*))}{T}\}$
                $\pi(i) = \pi(i)^*$; TFT($\pi(i)$) = TFT($\pi(i)^*$)
            End if
        Else if $rand < sp$
                Based on the second part of section 3.3.3, generate a new sequence
            else
                Based on the third part of section 3.3.3, generate a new sequence
            End if
        End if
    End for
End while
Return π_{best} and TFT (π_{best})

Proof In this paper, x_i denotes a sequence on the machines based on Definition 1. Hence, the length of x_i is finite. The population state $S = \{x_1, x_2, \ldots, x_N\}$ consists of the state of N sequences, N is a finite positive integer. Therefore, the population state space **SP** is finite based on Definition 2.

According to the process of the DGSA, the population state transition probability $P(S_{i,\,(t-1)} \rightarrow S_{i,\,t})$ from time $(t-1)$ to

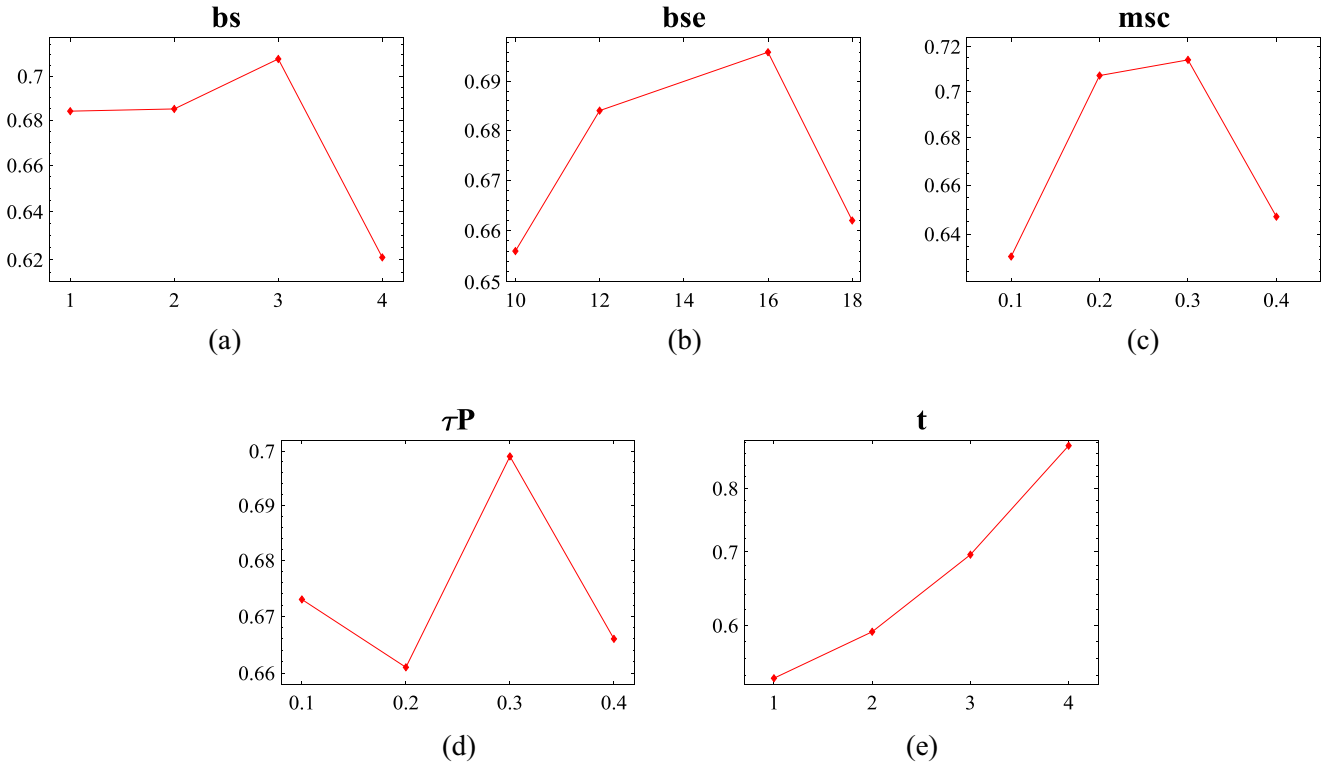


Fig. 4 Changing tendency of the parameters. (a) *bs*. (b) *bse*. (c) *msc*. (d) τP. (e) t

time (t) is only related to the population state $S_{i,\,(t-1)}$ at time ($t-1$) and independent of time ($t-1$).

In summary, the population state sequence has the characteristics of the Markov. The process of population state transfer is a finite homogeneous Markov chain.

From [61], the search space is divided into non-empty sets $A_1, \ldots, A_m$ and A_m only contains global optima. Then, the expected runtime is calculated as follows.

$$\mathrm{E(T)} \le \sum_{\mathrm{i=1}}^{\mathrm{m-1}} \frac{1}{\mathrm{s_i}} \tag{25}$$

where s_i is the probability that the agent in $\mathrm{A_i}$ is mutated to $\mathrm{A_j}$ with $\mathrm{j} > i$.

Theorem 1 The upper bound of the expected runtime of the DGSA for the BFSP is $O(n^2)$.

Proof Due to the elitism strategy of the DGSA, a sequence is changed when the sequence transferred from A_i to A_j with $j > i$. For a sequence with permutation coding, two elements of the sequence have changed at least when the sequence is changed. Hence, the probability that any one element changes for a sequence is $\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}$. For a sequence, the probability that the sequence is unchanged is $\left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right)^{\mathrm{n}}$. For the population, the probability s_i that the sequence reaches a higher fitness level is $num * \left(1-\left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right)^{\mathrm{n}}\right)$ (num is the number of sequence).

$$\begin{aligned} s_i &= num * \left(1-\left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right)^{\mathrm{n}}\right) \\ &= num * \left(1^{n}-\left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right)^{n}\right) \\ &= num * \left(1-1+\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right) * \left(1+1 * \left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right) + , \ldots, + \left(1-\frac{1}{n} * C_{n-1}^{1} * \frac{1}{n}\right)^{n-1}\right) \\ &\ge num * \left(\frac{1}{n}-\frac{1}{n^2}\right) * 1 \end{aligned}$$

Then,

$$E(T) \le \sum_{\mathrm{i=1}}^{\mathrm{m-1}} \frac{1}{\mathrm{s_i}} \le \sum_{i=1}^{\mathrm{n}} \frac{1}{num * \left(\frac{1}{n}-\frac{1}{n^2}\right)} = \frac{1}{num} * \sum_{i=1}^{n} \frac{n^2}{n-1} = \frac{1}{num} * \frac{n^3}{n-1} \le \frac{1}{num} * n^2 = O(n^2)$$

According to the above analysis, the upper bound of the expected runtime is $O(n^2)$.

4 Parameter calibration

4.1 Parameters setting for *VPF _ NEH*(*n*)

In section 3.2, the *VPF _ NEH*(n) heuristic has two important parameters, namely μ and δ. A full factorial design experiments are designed to obtain the best parameter combination

Table 3 The factor level of the parameters

Factor level	1	2	3	4
vt	1	2	3	4
*G*0	0.1	0.5	1	1.5
ra	0.1	0.5	1	1.5
vc	0.1	0.2	0.3	0.4
K	0.5	1	1.5	2

of the *VPF _ NEH*(*n*). In order to test the performance of the parameter combinations, random instances based on the method in Taillard [48] are generated for each combination of $n \in \{20,50,100,200\}$ and $m \in \{5,10,20\}$. Each problem size has 5 instances. At last, a test benchmark including 60 instances is obtained. To be fair, the experiments including the following experiments were coded in Java (jdk 1.8) and carried out on the PC with 3.4 GHz Intel(R) Core™ i7–6700 CPU, 8GB of RAM and 64-bit OS. The evaluated values were $\mu = \{0.5,0.55,0.65,0.7,0.75,0.8,0.85\}$ and $\delta = \{5,10,15,20\}$. Hence, there are 28 groups of experiments. The Relative Percentage Deviation (RPD) is applied to measure the performance of the parameter combination. Equation 26 is the calculation formula of the RPD.

$$RPD = \sum_{i=1}^{28} \frac{TFT(i) - TFT(best)}{TFT(best)} * 100 \tag{26}$$

where *TFT*(*i*) is the total flow time of instance *i* and *TFT*(*best*) is the minimum total flow time in the instance *i* of the combination of parameters. The Average Relative Percentage Deviation (ARPD) is all obtained from RPDs of each instance with each parameter combination. The results of the ARPD are shown in Fig. 3. According to the results of the combination, the best solution is obtained when $\mu = 0.75$ and $\delta = 20$.

4.2 Parameter setting for VNO

To identify the best parameter combination of the VNO, a design of experiments method (DOE) [62] is applied in this paper and 60 instances are employed as the test set in section 4.1. Five parameters are considered in the VNO as the control parameter. These parameters are the block size in the beginning (*bs*), the block size in the end (*bse*), the move size coefficient (*msc*), the number of cycles in the TNO (*t*) and the temperature control parameter τP. According to the number of parameter and the factor levels, the orthogonal array $L_{16}(4^5)$ is selected. The factor levels of the parameters are shown in Table 1.

In the DOE, each parameter combination is run 5 times independently on each instance and the maximum CPU time is set as if $(n \leq 20)$ then $cpuTime = 10 * n * m$ else $cpuTime = 40 * n * m$. For each parameter combination, the average relative percentage deviation of VNO ($ARPD_{VNO}$) is calculated.

In Table 2, the rank of each parameter combination is listed. The $ARPD_{VNO}$ is calculated as follows.

$$ARPD_{VNO} = \left(\sum_{i=1}^{60}\left(\sum_{j=1}^{5}\left(\frac{TFT_i(j) - TFT(min)}{TFT(min)}\right)\right)/5\right)/60 * 100 \tag{27}$$

where $TFT_i(j)$ is the total flow time of the *jth* time on the *ith* instance and the *TFT*(*min*) is the minimum total flow time found among all parameter combination on the test set.

According to Table 2, the main effects plots of the parameters are shown in Fig. 4. From Table 2, *t* is the most significant parameter among the five factors. The reason is that *t* has a significant influence on balancing the time-consuming and the search ability. From Fig. 4, the best parameter combination are $bs = 4$, $bse = 10$, $msc = 0.1$, $t = 1$ and $\tau P = 0.2$.

5 Parameter adjustment of DGSA

After determining the basic structure of the algorithm, the five main parameters are adjusted in Section 4.3. These parameters are *vt*, *G*0 from Algorithm 10, *ra* based on eq. 19, *vc* from Section 3.3.2 and *K* from Section 3.3.3. In the section, the experiment is similar with Section 4.2. The factor levels of the parameters are shown in the Table 3.

In Table 4, the ranks of each parameter combination are listed.

According to Table 4, the main effects plots of the parameters are shown in Fig. 5. From Table 4, *vt* is the most significant one parameter among the five parameters. From Fig. 5, $vt = 3$, $G0 = 1.5$, $ra = 1$, $vc = 0.1$ and $K = 1$ are selected in this paper.

According to the results of the experiments, the final parameters combination of the DGSA is determined.

6 Experiment and performance analysis

In this paper, the proposed algorithm, named DGSA, is compared with the state-of-art algorithms including the

Table 4 The rank of the parameter

Level	Parameters				
	vt	*G*0	*ra*	*vc*	*K*
1	0.423	0.415	0.415	0.387	0.387
2	0.396	0.403	0.400	0.390	0.381
3	0.371	0.386	0.379	0.401	0.409
4	0.384	0.380	0.380	0.395	0.396
Std.	0.02216	0.01598	0.01729	0.006131	0.012176
Rank	1	3	2	5	4

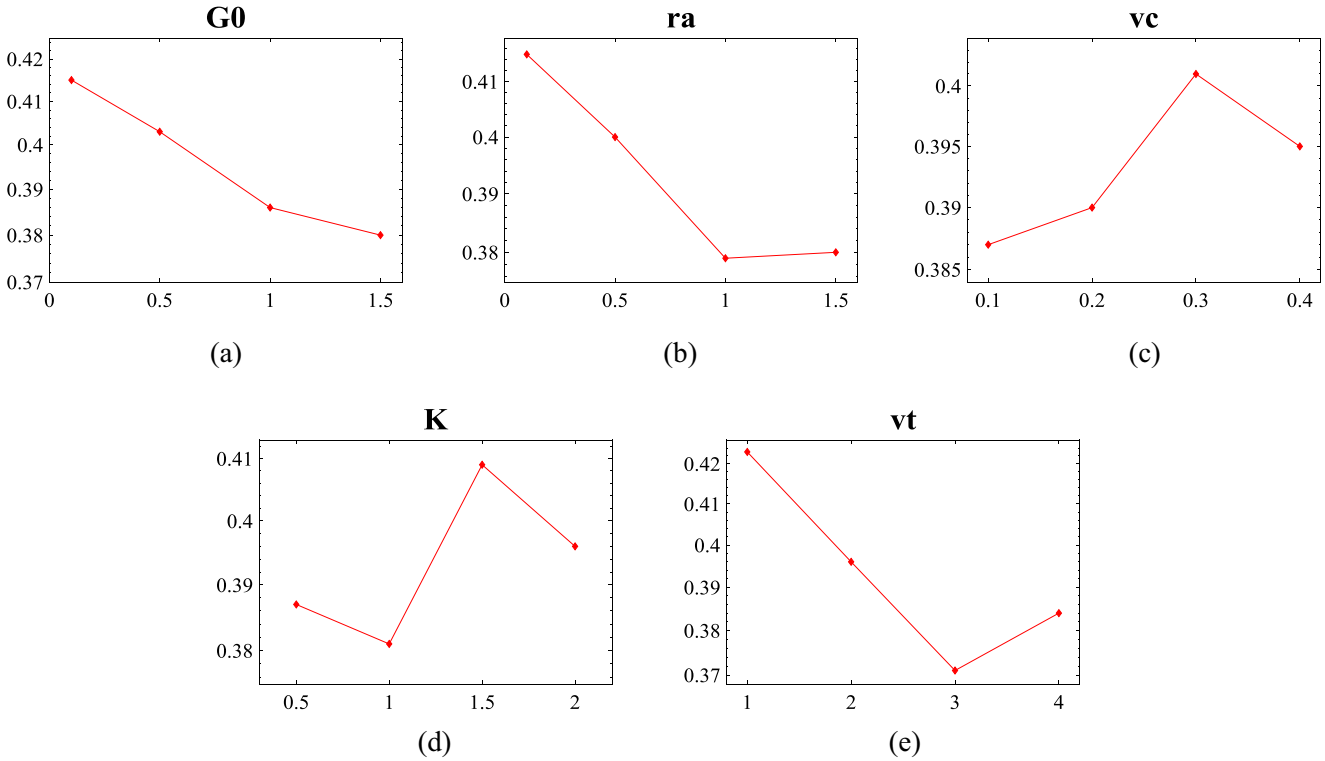


Fig. 5 Changing tendency of the parameters. (a) *G*0. (b) *ra*. (c) *vc*. (d) *K*. (e) *vt*

Discrete Artificial Bee Colony algorithm [11] (denoted as DABC_RCT), the Variable Block Insertion Heuristic [23] (denoted as VBIH), the IG_RIS algorithm from Tasgetiren et al. [23], the Simulated Annealing Genetic algorithms [63] (denoted as SAGA), and the Discrete Differential Evolution algorithm [64] (denoted as DDE). DABC_RCT, VBIH and IG_RIS are the the effective methods for the BFSP with the total flow time minimization. In addition, the SAGA and the DDE, which are the advanced algorithms for the BFSP or the distributed BFSP with the different criteria, are also carried to compare with DGSA. In the VBIH, the μ value with 0.35 is selected as the compared algorithm. To make a fair comparison, the compared algorithms are reprogrammed according to the details in the original papers and all algorithms were coded in Java (jdk1.8) and tested on the same computer, a PC with 3.4 GHz Intel(R) Core™ i7–6700 CPU, 8GB of RAM and 64-bit OS. The simulation is carried out based on the well-known flow shop benchmark suite of Taillard [48] and using the total flow time criterion. For all the compared algorithms, the maximum CPU time is fixed at $80 * m * n$ milliseconds. In each test, all algorithms are run 5 times independently for all 120 instances. The results of the DGSA and the compared algorithms are represented by the average relative percentage deviation (ARPD) which is the average RPD of the 10 instances of each $n * m$ group. The RPD is computed as follows.

$$RPD = \sum_{i=1}^{R} \left(\frac{TFT(i) - TFT(min)}{TFT(min)} \right) * 100/R \tag{28}$$

where $TFT(i)$ is the total flow time generated by the algorithms in *ith* run, R is the number of runs, and

Table 5 The ARPD of the compared algorithms

n × m	DGSA	DABC_RCT	IG_RIS	VBIH	SAGA	DDE
20 × 5	0.0389	0.0031	0.0494	0.0003	0.4192	0.5285
20 × 10	0.0118	0.0058	0.0163	0	0.2850	0.2952
20 × 20	0.0091	0	0.0105	0	0.1235	0.3007
50 × 5	0.2481	0.5017	0.6942	0.4655	5.1186	4.0125
50 × 10	0.3004	0.5453	0.6889	0.6562	4.9866	3.4543
50 × 20	0.2102	0.3811	0.4216	0.4323	3.4971	2.3788
100 × 5	0.3696	0.5874	0.4756	0.5618	12.2256	10.2988
100 × 10	0.3832	0.6408	0.7415	0.6081	10.4904	8.3198
100 × 20	0.3849	0.6810	0.7051	0.4290	7.5039	6.0118
200 × 10	0.2735	0.4999	0.5115	0.3492	15.6995	13.6886
200 × 20	0.3470	0.5151	0.4330	0.2457	11.213	9.4556
500 × 20	0.2557	0.4379	0.2461	0.0981	16.2227	15.2911
Average	0.2360	0.3999	0.4161	0.3205	7.3154	6.1696

TFT(*min*) is the minimum total flow time generated by all algorithms in *ith* run.

The computational results of the algorithms are shown in Table 5. The bold values are the minimum ARPD values which indicate the best algorithm among the compared algorithms. Table 5 shows that the DGSA is the best algorithm among the compared algorithms. Of the total 12 categories, the 7 best results are generated by the DGSA and the average is the smallest among the compared algorithms.

To further demonstrate the performance of the DGSA and the compared algorithms, certain representative cases of convergence curves are shown in Fig. 6. From Table 5, the performance of the DGSA is better than the compared algorithms from 50×5 to 200×10. From Fig. 6(b) and (d), the DGSA achieved the fastest convergence speed and the best convergence accuracy on Ta 64 and Ta 83. For Ta 44 and Ta 76, although the convergence speed of the DGSA isn't the fastest, the best results are generated in the end. The reason of getting high-quality solutions with the DGSA on most of the problem instances is that the DGSA achieves balance exploration and exploitation between the compared algorithms.

The boxplots of all algorithms are shown in Fig. 7 for Ta 44, Ta 64, Ta 76 and Ta 83. From the above pictures, the stability and performances of the DGSA outperform the compared algorithms.

The statistical tests are implemented to illustrate that the DGSA has a statistically significant difference between the compared algorithms. Two nonparametric statistical tests are

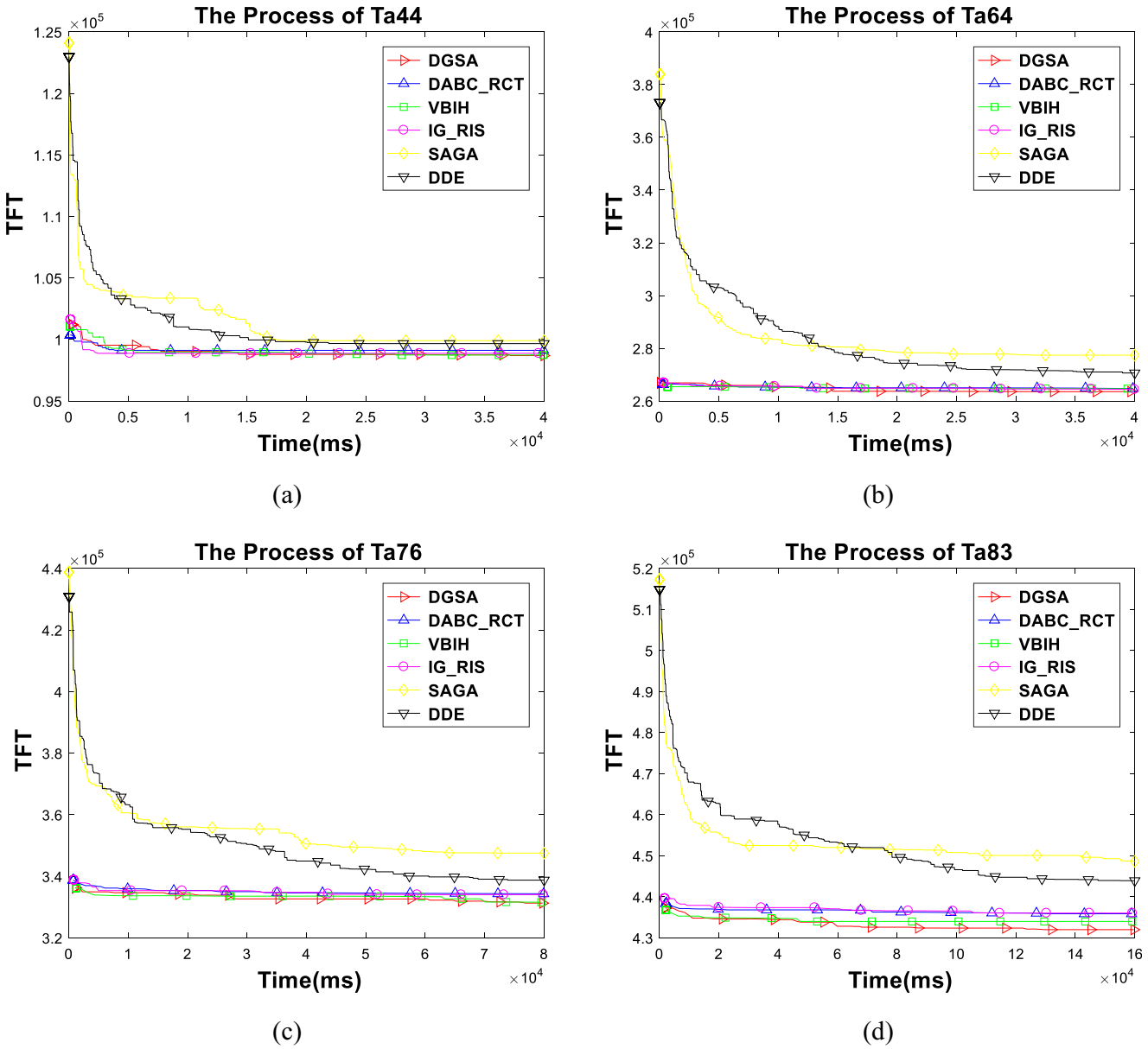


Fig. 6 The convergence curve of Ta instances

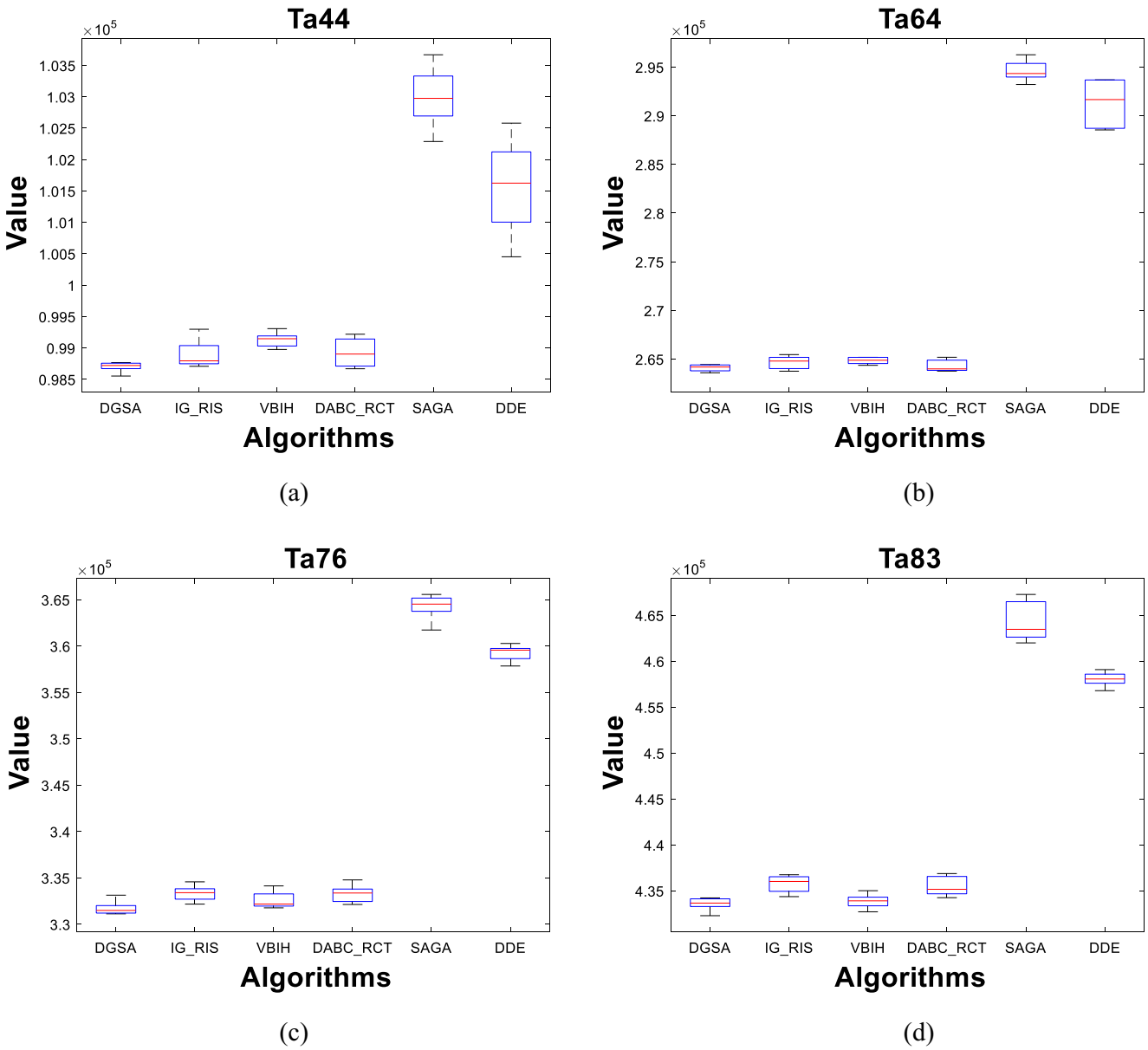


Fig. 7 The boxplot of Ta instances

selected in this paper, namely the Wilcoxon's sign rank test [65] with 99% confidence intervals and the Friedman test.

The Wilcoxon rank-sum test is a paired related data comparison method. The *p*-values of the compared algorithms are shown in Table 6. In the Wilcoxon rank-sum test, the null hypothesis H0 is that the *p* value which is more than 0.01 is considered as the strong evidence to accept the null hypothesis H0 that the two algorithms come from distributions with equal means. If the *p*-value is less than 0.01, the null hypothesis is rejected and the right-tailed hypothesis H1 is accepted that the two algorithms come from distributions with different means, and it is determined which algorithm is better than another based on symbol rank sum. From Table 6, the performance of the DGSA outperforms the compared algorithms.

"Yes" indicates that statistically significant is observed under the corresponding α.

The mean rank of the Friedman test is shown in Table 7 and the graphical representation of the Bonferroni-Dunn's test is shown in Fig. 8. The statistical tests indicated that the performance of DGSA outperformed the compared algorithms for solving the BFSP with the total flow time minimization. First, the GSA algorithm is a classical population-based algorithm. It has been confirmed to be an effective framework in various domains. The performances of the GSA are sensitive to the parameters such as the gravitational coefficient and population size. Therefore, the parameter combinations of DGSA are analyzed by the orthogonal

Table 6 The *p*-value of Wilcoxon's rank-sum test for the compared algorithms

DGSA vs.	R+	R-	Z	*p*-value	$\alpha = 0.01$
DABC_RCT	4450	803	−6.087	1.15E-9	Yes
IG_RIS	4828	843	−6.28	3.38E-10	Yes
VBIH	3606	1750	−3.053	0.008	Yes
SAGA	7140	0	−9.467	2.88E-21	Yes
DDE	7140	0	−9.467	2.88E-21	Yes

experiment. From the experimental results of Section 4, the optimal parameter combinations are determined. Second, the performance of algorithm is affected by the quality of the initial population. For the DGSA, DABC_RCT, VBIH, and IG_RIS, the initial solution is generated by the heuristic methods. However, the initial solution of the SAGA is randomly generated. For the DDE, the initial solution is generated by the simple heuristic method or random method. The results are shown in Fig. 6 due to the differences of initialization methods. Third, the neighborhood search is a significant operator which affects the performance of algorithms. In the DGSA, a new variable neighborhood search, named VNO, is implemented in the process of the search. The VNO is a significant reason which enhances the performance of the DGSA. Finally, although the SAGA and the DDE are excellent algorithms for the BFSP with the makespan criterion or the distributed BFSP, the proposed DGSA outperforms the SAGA and the DDE for solving the BFSP with the total flow time minimization duo to the no-free-launch theorem. From Table 6, Table 7 and Fig. 8, the DGSA outperforms the compared algorithms.

From the analysis of the experimental results, the following conclusions are obtained.

(a) The performance of DGSA is the best among the compared algorithms. 7 out of the 12 categories are better than other algorithms, and the ARPD is the smallest among the compared algorithms.

Table 7 The mean rank of the algorithms

Algorithms	Mean rank
DGSA	1.86
DABC_RCT	2.80
IG_RIS	3.09
VBIH	2.30
SAGA	5.78
DDE	5.17

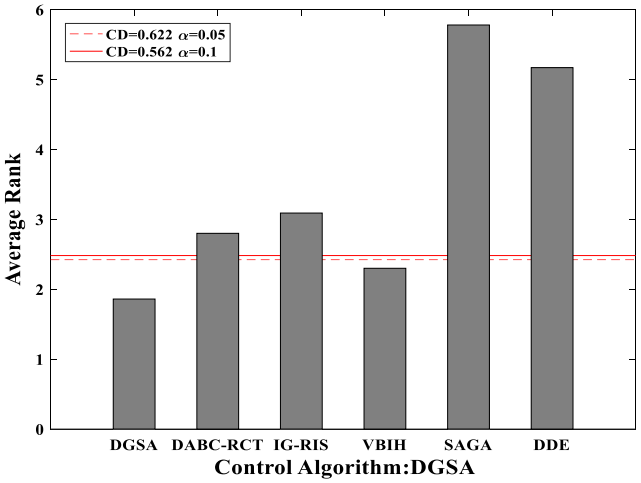


Fig. 8 The graphical representation of the Bonferroni-Dunn's test

(b) The convergence ability of the DGSA is stable. From Fig. 6, especially Fig. 6(d), the DGSA produces the best results with fast convergence speed among the compared algorithms. The stability of the algorithms is shown in Fig. 7. According to the experiment results, the DGSA is the extreme robust method among the compared algorithms.

(c) According to Table 6, Table 7 and Fig. 8, DGSA has significant differences with the compared algorithms and it is better than the compared algorithms through the results of two nonparametric statistical tests.

7 Conclusions

In this paper, a discrete gravitational search algorithm is proposed, named DGSA, for solving the BFSP with the total flow time minimization. In order to obtain an effective initial population, a new initialization method, named *VPF _NEH*(*n*), is introduced in the DGSA. *VPF _NEH*(*n*) is applied in the obtained initial solutions to balance the quality and diversity. After the initial population constructed, a new update method of the acceleration and the velocity based on the movement of the sequence element when the process enters into the main loop. In the process of position update, three operators are designed to balance exploration and exploitation. The variable neighborhood operator, named VNO, is applied to enhance the exploitation capacity and the solution accuracy. The path relinking operator based on insertion is executed to prevent the DGSA from premature convergence when the diversity of the population declines. The comparison of DGSA with the compared algorithms show that the effectiveness and superiority of the DGSA for solving the BFSP with the total flow time minimization.

In the future research, there is a potential direction of applying the DGSA to solve the BFSP with other evaluation criteria, such as the total tardiness minimization or the total completion time minimization. For solving the scheduling problem, efficient heuristics is significant for generating the initial population by balancing the diversity and quality. Therefore, it is necessary to research in this direction. It is another research direction that the algorithm is used to solve other scheduling problems in the future, such as the permutation flow shop problem or the distributed flow shop problem.

Acknowledgements This work was financially supported by the National Natural Science Foundation of China under grant numbers 61663023. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Zhejiang Provincial Natural Science Foundation (LGJ19E050001), Wenzhou Public Welfare Science and Technology project (G20170016), respectively.

References

1. Pan QK, Ruiz R (2012) An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. Omega 40(2): 166–180
2. Ruiz-Torres AJ, Ho JC, Ablanedo-Rosas JH (2011) Makespan and workstation utilization minimization in a flowshop with operations flexibility. Omega 39(3):273–282
3. Ronconi DP, Henriques LRS (2009) Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. Omega 37(2):272–281
4. Gong H, Tang L, Duin CW (2010) A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Comput Oper Res 37(5):960–969
5. Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44(3): 510–525
6. Sethi SP, Sriskandarajah C, Sorger G, Blazewicz J, Kubiak W (1992) Sequencing of parts and robot moves in a robotic cell. Int J Flex Manuf Syst 4(3–4):331–358
7. Ribas I, Companys R (2015) Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. Comput Ind Eng 87:30–39
8. Mccormick ST, Pinedo M, Wolf B, Wolf B (1989) Sequencing in an assembly line with blocking to minimize cycle time. Oper Res 37(6):925–935
9. Fernandez-Viagas V, Leisten R, Framinan JM (2016) A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. Expert Syst Appl 61:290–301
10. Pan QK, Wang L (2011) Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. Omega 40(2):218–229
11. 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 Syst Appl 42(15–16): 6155–6167
12. Nouha N, Talel L (2015) A particle swarm optimization metaheuristic for the blocking flow shop scheduling problem: Total tardiness minimization. Multi-agent systems and agreement technologies. Springer: 145–153
13. Riahi V, Khorramizadeh M, Newton MAH, Sattar A (2017) Scatter search for mixed blocking flowshop scheduling. Expert Syst Appl 79(C):20–32
14. Pan QK, Wang L, Sang HY, Li JQ, Liu M (2013) A high performing memetic algorithm for the Flowshop scheduling problem with blocking. IEEE Trans Auto Sci Eng 10(3):741–756
15. Lin SW, Ying KC (2013) Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. Omega 41(2):383–389
16. Wang C, Song S, Gupta JND, Wu C (2012) A three-phase algorithm for flowshop scheduling with blocking to minimize makespan. Comput Oper Res 39(11):2880–2887
17. Wang L, Pan QK, Tasgetiren MF (2011) A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. Comput Ind Eng 61(1):76–83
18. Moslehi G, Khorasanian D (2013) Optimizing blocking flow shop scheduling problem with total completion time criterion. Comput Oper Res 40(7):1874–1883
19. Ying KC, Lin SW (2017) Minimizing Makespan in distributed blocking Flowshops using hybrid iterated greedy algorithms. IEEE Access PP (99):1–1
20. Tasgetiren MF, Kizilay D, Pan QK, Suganthan PN (2017) Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput Oper Res 77(C):111–126
21. Han Y, Gong D, Li J, Zhang Y (2016) Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int J Prod Res 54(22):6782–6797
22. Tasgetiren MF, Pan QK, Kizilay D, Suer G (2015) A populated local search with differential evolution for blocking flowshop scheduling problem. IEEE Congress on Evolutionary Computation (CEC): 2789–2796
23. Tasgetiren M, Pan QK, Kizilay D, Gao K (2016) A variable block insertion heuristic for the blocking Flowshop scheduling problem with Total flowtime criterion. Algorithms 9(4):71
24. Shao Z, Pi D, Shao W (2018) A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem. Expert Syst Appl 113:77–99
25. Shao Z, Pi D, Shao W (2017) Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. Comput Ind Eng 111:331–351
26. Nouri N, Ladhari T (2015) Minimizing regular objectives for blocking permutation flow shop scheduling: heuristic approaches. 441–448
27. Toumi S, Jarboui B, Eddaly M, Rebai (2013) A solving blocking flowshop scheduling problem with branch and bound algorithm. International Conference on Advanced Logistics and Transport: 411–416
28. Khorasanian D, Moslehi G (2012) An iterated greedy algorithm for solving the blocking flow shop scheduling problem with Total flow time criteria. Int J Indust Eng 23(4):301–308
29. Yang X-S (2018) Mathematical analysis of nature-inspired algorithms. Nature-inspired algorithms and applied optimization. Springer: 1–25
30. Kennedy J, Eberhart RC (1995) Particle swarm optimization. IEEE Int Conf Neural Netw 4:1942–1948
31. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
32. Zhao F, Qin S, Zhang Y, Ma W, Zhang C, Song H (2019) A two-stage differential biogeography-based optimization algorithm and its performance analysis. Expert Syst Appl 115:329–345
33. Zhao F, Liu Y, Zhang C, Wang J (2015) A self-adaptive harmony PSO search algorithm and its performance analysis. Expert Syst Appl 42(21):7436–7455

34. Zhao F, Liu Y, Zhang Y, Ma W, Zhang C (2017) A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. Eng Appl Artif Intell 65:178–199
35. Zhao F, Shao Z, Wang J, Zhang C (2017) A hybrid differential evolution and estimation of distribution algorithm based on neighbourhood search for job shop scheduling problems. Int J Prod Res 54(4):1–22
36. Meng Z, Pan JS, Kong L (2018) Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution. Knowl-Based Syst 141:92–112
37. Meng Z, Pan JS, Xu H (2016) QUasi-affine TRansformation evolutionary (QUATRE) algorithm: a cooperative swarm based algorithm for global optimization. Knowl-Based Syst 109:104–121
38. Rao RV (2016) Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput 7:19–34
39. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
40. Rashedi E, Rashedi E, Nezamabadi-pour H (2018) A comprehensive survey on gravitational search algorithm. Swarm Evol Comput 41:141–158
41. Zhao F, Xue F, Zhang Y, Ma W, Zhang C, Song H (2018) A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. Expert Syst Appl 113:515–530
42. Choudhary A, Gupta I, Singh V, Jana PK (2018) A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Futur Gener Comput Syst 83:14–26
43. Lee T, Loong Y, Moslemipour (2017) G gravitational search algorithm optimization for bi-objective flow shop scheduling using weighted dispatching rules. 2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE) IEEE: 127–132
44. Narang N (2018) Hydro-thermal generation scheduling using integrated gravitational search algorithm and predator–prey optimization technique. Neural Comput & Applic 30(2):519–538
45. Özyön S, Yaşar C (2018) Gravitational search algorithm applied to fixed head hydrothermal power system with transmission line security constraints. Energy 155:392–407
46. Pelusi D, Mascella R, Tallini L, Nayak J, Naik B, Abraham A (2018) Neural network and fuzzy system for the tuning of gravitational search algorithm parameters. Expert Syst Appl 102:234–244
47. Mittal H, Saraswat M (2018) An optimum multi-level image thresholding segmentation using non-local means 2D histogram and exponential Kbest gravitational search algorithm. Eng Appl Artif Intell 71:226–235
48. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285
49. Graham RL, Lawler EL, Lenstra JK, Kan AHGR (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5(1):287–326
50. Li X, Wang Q, Wu C (2009) Efficient composite heuristics for total flowtime minimization in permutation flow shops. Omega 37(1): 155–164
51. Han Y-Y, Quan-Ke L, Qing J, Cao NN, Liang JJ (2013) Effective hybrid discrete artificial bee colony algorithms for the total;flowtime minimization in the blocking flowshop problem. Int J Adv Manuf Technol 67(1–4):397–414
52. Wu B, Qian C, Ni W, Fan S (2012) Hybrid harmony search and artificial bee colony algorithm for global optimization problems. Comput Math Applic 64(8):2621–2634
53. Nawaz M, Jr EEE, Ham I (1983) A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem. Omega 11(1):91–95
54. Wang X, Tang L (2012) A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. Appl Soft Comput J 12(2):652–662
55. Glover F, Laguna M (1998) Tabu search. Handbook of combinatorial optimization. Springer: 2093–2229
56. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2
57. Ribas I, Companys R, Tort-Martorell X (2017) Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst Appl 74:41–54
58. Zhao F, Liu H, Zhang Y, Ma W, Zhang C (2018) A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. Expert Syst Appl 91:347–363
59. He J, Yao X (2001) Drift analysis and average time complexity of evolutionary algorithms. Artif Intell 127(1):57–85
60. Goryajnov VV (1996) Evolutionary families of analytic functions and time-nonhomogeneous Markov branching processes. Dokl Math 53 (2)
61. Sudholt D (2010) General lower bounds for the running time of evolutionary algorithms. International conference on parallel problem solving from nature. Springer: 124–133
62. Montgomery DC (2006) Design and analysis of experiments. Technometrics 48(1):158–158
63. Lebbar G, Barkany AE, Jabri A, Abbassi IE (2018) Hybrid metaheuristics for solving the blocking Flowshop scheduling problem. Int J Eng Res Afr 36:124–136
64. Zhang G, Xing K, Cao F (2018) Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Eng Appl Artif Intell 76:96–107
65. Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively. Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technolgy, Lanzhou, China, where he became a Full Professor in 2012. He has been as the post Doctor with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China in 2009. He has authored two academic book and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.