An Improved Discrete Migrating Birds Optimization for Lot-Streaming Flow Shop Scheduling Problem with Blocking

Yuyan Han[1], Junqing Li[1,2](✉), Hongyan Sang[1], Tian Tian[3], Yun Bao[1], and Qun Sun[4]

[1] School of Computer Science, Liaocheng University, Liaocheng 252059, China
{hanyuyan, sanghongyan, baoyun}@lcu-cs.com
[2] School of Computer Science, Shandong Normal University, Jinan 252000, China
lijunqing@lcu-cs.com
[3] School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China
tian_tiantian@126.com
[4] School of Mechanical and Automotive Engineering, Liaocheng University, Liaocheng 252059, China
sunqun@lcu.edu.cn

Abstract. Blocking lot-streaming flow shop (BLSFS) scheduling problems have considerable applications in various industrial systems, however, they have not yet been well studied. In this paper, an optimization model of BLSFS scheduling problems is formulated, and an improved migrating birds optimization (iMBO) algorithm is proposed to solve the above optimization problem with the objective of minimizing makespan. The proposed algorithm utilizes discrete job permutations to represent solutions, and applies multiple neighborhoods based on insert and swap operators to improve the leading solution. An estimation of distribution algorithm (EDA) is employed to obtain solutions for the rest migrating birds. A local search based on the insert neighborhood is embedded to improve the algorithm's local exploitation ability. iMBO is compared with the existing discrete invasive weed optimization, estimation of distribution algorithm and modified MBO algorithms based on the well-known lot-streaming flow shop benchmark. The computational results and comparison demonstrate the superiority of the proposed iMBO algorithm for the BLSFS scheduling problems with makespan criterion.

Keywords: Blocking · Lot-streaming flow shop · Migrating birds optimization Estimation of distribution

1 Introduction

In many manufacturing environments, the job often refers to a set of tasks to be carried out by machines over semi-finished goods or raw materials in order to obtain a final product. Lot-steaming is a production layout in which every job can be split into a number of smaller sub-lots. When a sub-lot is completed, it can be immediately transferred to the downstream machine. By this splitting technique, the idle time on successive machines can be reduced, and thereby reducing productive cycle, accelerating the manufacturing process and enhancing the production efficiency. The goal for this problem is to find a solution (or a sequence) that optimizes a given objective function, i.e., maximum completion time minimization or makespan, the total flow time, the tardiness time and the earliness time.

For the lot-streaming problem in flow shop environment, Sarin et al. [1] presented a polynomial-time procedure to determine the number of sub-lots of a single-lot, multiple-machines flow shop lot-streaming problem. In this work, the authors minimized the unified cost-based objective function that comprised criteria pertaining to makespan, mean flow time, work-in-process, sublot-attached setup and transfer times. Pan et al. [2] presented a novel estimation of distribution algorithm (EDA) to minimize the maximum completion time, in which an estimation of a probabilistic model is constructed to direct the algorithm search towards good solutions by taking into account both job permutation and similar blocks of jobs. Defersha and Chen [3] first proposed a mathematical model for the lot-streaming problem in multi-stage flow shops where at each stage there are unrelated parallel machines, and then proposed genetic algorithm based on parallel computing platforms to solve the above problem. Numerical examples showed that the parallel implementation greatly improved the computational performance of the developed heuristic. To minimize the mean weighted absolute deviation from due dates of the lot-streaming flow shop scheduling, Yoo and Ventura developed a heuristic based on pairwise interchange strategy [4]. Chakaravarthy et al. [5] proposed a differential evolution algorithm (DE) and particle swarm optimization (PSO) to evolve the best sequence for makespan/total flow time criterion of the m-machine flow shop involved with lot-streaming and set up time. Following that Chakaravarthy et al. [6] adopted an improve sheep flock heredity algorithm and artificial bee colony algorithm, respectively, to solve lot-streaming flow shop with equal size sub-lot problems. For the same problems, Sang et al. [7] designed an effective iterated local search algorithm, in which an insertion neighborhood and a simulated annealing-typed acceptance criterion are utilized to generate good solutions.

In most practical manufacturing enterprises, there is no intermediate buffer between machines to store completed jobs. Therefore, these completed jobs have to remain in the current machine, until its following one is available for processing, which increases waiting time and the production period. Previous research has already been done to tackle a blocking flow shop (BFS) scheduling problem [8], so as to improve the production efficiency. Similarly, in a lot-streaming flow shop (LSFS) scheduling problem, each sublot will also be blocked when there is no intermediate buffer to store completed sublots. These practical scenarios encourage us to apply the blocking constraint to a LSFS scheduling problem, and form a blocking LSFS (BLSFS) scheduling problem [9].

Very recently, a new metaheuristic intelligence approach named the Migrating Birds Optimization (MBO) algorithm, which simulates the V flight formation of migrating birds, as the name implies, was presented by Duman [10]. For the scheduling problems, Tongur and Ülker [11] first applied the basic MBO algorithm to optimize the discrete flow shop sequencing problem. Following that Pan and Dong designed an improved MBO (IMBO) algorithm to minimize the total flowtime of the hybrid flow shop scheduling [12]. In this work, the authors presented a diversified method to initialize population with high quality, and constructed a mixed neighbourhood based on insertion and pairwise exchange operators to generate promising neighbouring solutions for the leader and the following birds. Similarly, Niroomand et al. [13] also proposed modified MBO (MMBO) algorithm to optimize closed loop layout with exact distances in flexible manufacturing systems, which are different from IMBO considered by Pan and Dong. In MMBO, the authors employed crossover and mutation operators to yield the neighbor regeneration.

For the above literature, the simulation experimental results have verified that the MBO algorithm is appropriate and competitive for solving continuous and discrete optimization problems. To the best of our knowledge, the MBO algorithm has not been applied to the LSFS scheduling problem with blocking. With the above motivations, we proposed an improved MBO (iMBO) algorithm to solve the BLSFS scheduling problem.

The rest of this paper is organized as follows. After this brief introduction, in Sect. 2, the description of BLSFS scheduling problem is stated. Next, Sect. 3 presents the proposed algorithm. Section 4 provides the experimental results. Finally, the paper ends with some conclusions in Sect. 5.

2 BLSFS Scheduling Problem

For each job, it can be processed at the ith machine after its front job completed at the ith machine, in which all the sub-lots of the same job should be processed continuously. At any time, for each machine, it can process at most a sub-lot, and a sub-lot can be processed on at most one machine at a time.

In the sequel, we assume that there are n jobs and m machines; denote the job sequence (solution) as $\pi = (1, 2, \ldots, n)$; and let lj be the number of sublots in job j. All sublots of the same job have to be processed on each of m machines in the same series. The processing time of each sublot of a job j on machine t is pj, t. We use Sj, t, e and Cj, t, e to represent the start and the completion time of the e-th sublot in job j on machine t, respectively, where e = 1, 2, …, lj; dj refers to the due date of job j.

The completion time of each job on each machine can be calculated using the following equations.

$$\begin{cases} S_{1,1,1} = 0 \\ C_{1,1,1} = S_{1,1,1} + p_{1,1} \end{cases} \tag{1}$$

$$\begin{cases} S_{1,t,1} = C_{1,t-1,1} \\ C_{1,t,1} = S_{1,t,1} + p_{1,t} \end{cases} \quad t = 2, 3, \ldots, m \tag{2}$$

$$\begin{cases} S_{j,1,1} = \max\{C_{j-1,1,l_{j-1}}, S_{j-1,2,l_{j-1}}\} \\ C_{j,1,1} = \mathrm{S}_{j,1,1} + \mathrm{p}_{j,1} \end{cases} \quad j = 2, 3, \ldots, n \tag{3}$$

$$\begin{cases} S_{j,t,1} = \max\{C_{j,t-1,1}, S_{j-1,t+1,l_{j-1}}\} \\ C_{j,t,1} = S_{j,t,1} + p_{j,t} \end{cases} \quad \begin{array}{l} t = 2, 3, \ldots, m-1 \\ j = 2, 3, \ldots, n \end{array} \tag{4}$$

$$\begin{cases} S_{j,m,1} = \max\{C_{j,m-1,1}, C_{j-1,m,l_{j-1}}\} \\ C_{j,m,1} = S_{j,m,1} + p_{j,m} \end{cases} \quad j = 2, 3, \ldots, n \tag{5}$$

$$\begin{cases} S_{j,1,e} = \max\{C_{j,1,e-1}, S_{j,2,e-1}\} \\ C_{j,1,e} = S_{j,1,e} + p_{j,1} \end{cases} \quad \begin{array}{l} e = 2, 3, \ldots, l_j \\ j = 1, 2, 3, \ldots, n \end{array} \tag{6}$$

$$\begin{cases} S_{j,t,e} = \max\{C_{j,t-1,e}, S_{j,t+1,e-1}\} \\ C_{j,t,e} = S_{j,t,e} + p_{j,t} \end{cases} \quad \begin{array}{l} e = 2, 3, \ldots, l_j \\ t = 2, 3, \ldots, m-1 \\ j = 1, 2, 3, \ldots, n \end{array} \tag{7}$$

$$\begin{cases} S_{j,m,e} = \max\{C_{j,m-1,e}, C_{j,m,e-1}\} \\ C_{j,m,e} = S_{j,m,e} + p_{j,m} \end{cases} \quad \begin{array}{l} e = 2, 3, \ldots, l_j \\ j = 1, 2, 3, \ldots, n \end{array} \tag{8}$$

Equations (1) and (2) give the completion time of the first sublot of the first job at *m* machines. Equations (3–5) computes the completion time of the first sublot of job *j* (*j* = 2, 3, …, *n*) at machine *t* (*t* = 1, 2, …, *m*), in which the values of $S_{(j-1),2,l\ (j-1)}$ and $S_{(j-1),t+1,l\ (j-1)}$ are obtained by Eqs. (5 and 6), respectively. Equations (6–8) calculates the completion time of the *e*-th (*e* = 2, 3, …, l_j) sublot of job *j* (*j* = 1, 2, …, *n*) at machine *t* (*t* = 1, 2,…, *m*).

The start time of the first sub-lot of the first job on the first machine is equal to zero, that is, $S_{1,1,1} = 0$. The makespan of the job permutation, π, is equal to the time when the last job in the processing sequence is finished at machine *m*. Its value can be represented according to Eq. (9).

$$C_{\max}(\pi) = C_{n,m,l_n} \tag{9}$$

3 The Proposed iMBO for the BLSFS Scheduling Problem

3.1 Initialization Population

To generate an initial population with a certain level of quality and diversity, many heuristics, i.e., NEH, MME and PFE, have been successfully adapted to initialize the seeds of the population [8]. But, they can only generate a single solution. If some good seeds in the initial population can be generated, the efficiency convergence of the whole algorithm will be enhanced. Therefore, the above idea is employed in this study. That is, a multiple-based MME initial strategy is proposed to yield β solutions with high quality, and a random method is adopted to generate the rest solutions so as to maintain the diversity of the population. The detailed process of generating β solutions is shown in Algorithm 1.

Algorithm 1. multiple-based MME

```
01: Input: the number of jobs, $n$
02: Output: $\beta$ solutions
03: Begin
04:   Let $\pi_i = \phi$, $\pi' = \phi$, $\pi_i' = \phi$
05:   for $i$=1 to $n$
06:       $\pi'(i) = i$
07:   for $i$=1 to $\beta$
08:       $\pi_i \leftarrow random\_shuffle(\pi'.begin(), \pi'.end())$
09:       Set $\pi_i'(1) = \arg\min_{j \in \pi_i} p_{j,1}$; $\pi_i = \pi_i \setminus (\pi_i(1))$
10:       Set $\pi_i'(n) = \arg\min_{j \in \pi_i} p_{j,m}$; $\pi_i = \pi_i \setminus (\pi_i(n))$
11:       for $k$=3 to $n$
12:           $\theta_j = \varphi \times \sum_{i=1}^{m-1} | p_{j,i} - p_{k-1,i+1} | + (1-\varphi) \times \sum_{q=1}^{m} p_{j,q}$   $\varphi \in [0,1]$ , $j \in \pi_i$   (9)
13:           Set $\pi_i'(k) = \arg\min_{j \in S} \theta_j$; $\pi_i = \pi_i \setminus (\pi_i(k))$
14:       end for
15:   Pick the first two jobs of $\pi_i'$, form two subsequences, $\{\pi_i'(1), \pi_i'(2)\}$ and $\{\pi_i'(2), \pi_i'(1)\}$,
      evaluate the quality of the two subsequences, and select the one with the minimal value of
      $C_{\max}$ as the current sequence, $\pi_i^*$
16:       for $k$=3 to $n$
17:           Pick the $k$th job from $\pi_i'$, obtain $k$ subsequences by inserting it into the current
              sequence, $\pi_i^*$, at $k$ possible positions, and select the subsequence with the minimal
              value of $C_{\max}$ as the current sequence, $\pi_i^*$.
18:       end for
19:   end for
20: End
```

3.2 Improving the Leading Solution

Insertion, swap and inverse operators are commonly used to produce a promising neighboring solution, which can enhance the solution's exploitation ability by slightly disturbing the neighboring solution. For more details about the above operators, please refer to [8].

In this section, three strategies based on insert, swap, and inverse operators are proposed: (1) perform insert once; (2) apply swap one time; (3) conduct inverse once. Generally speaking, more strategies generate different solutions with a larger probability than a single strategy, and avoid the population trapping in local optima.

We randomly chose one of the above four strategies to generate solutions, in which the best neighbor solution is selected to update the leading solution, and the remaining solutions are put into two shared neighbor sets, respectively.

3.3 Improving the Other Solutions in the Population

The process of improving the other solutions in the population plays an important role, whose contribution is that it can lead the offspring to the global good solution, and improve the convergence of the algorithm. The estimation of distribution algorithm (EDA) can utilize the valued information of solutions in the population to construct a probabilistic model, and then estimate the probability distributions of good solution to build new ones. In this paper, the sequence-based discrete EDA is given to generate a number of sequences so as to improve solutions in the population.

The basic EDA mainly includes four steps [14]: First, select PS promising solutions from the original population by computing fitness value of each individual, and then put them into a candidate population $[\eta_{i,j}]_{PS \times n}$; Second, build a probability distribution model $[\xi_{i,j}]_{n \times n}$ based on the candidate population; Third, generate a new solution through learning and sampling from according to the constructed probabilistic model $[\xi_{i,j}]_{n \times n}$. Repeat the above third step for generating some new solutions. Last, update the population by evaluating the objective value of each solution in the population, and delete some bad solutions.

The detailed description of the proposed EDA is given as follows:

Algorithm 2. The estimation of the distribution algorithm

```
Input: the population size, $PS$, the number of jobs, $n$
Output: $\tau$ new solutions
/* establish a candidate population $[\eta]_{PS\times n}$ */
01. Set $[\eta]_{PS\times n} = \phi$
02.   for $v$ =1 to $PS$
03.     Randomly select two solutions from the current population, evaluate their objectives,
        and select the best solution to put into the candidate population, $[\eta_{i,j}]_{PS\times n}$
04:       Let v=v+1
05:   end for
/*Build a probabilistic model $[\xi_{i,j}]_{n\times n}$ */
06: First, two matrixes $[\rho_{i,j}]_{n\times n}$ and $[\beta_{i,j}]_{n\times n}$ are built based on the order of the jobs in the
    permutation and the similar blocks of jobs.
07: Then, the probability $\xi_{i,j}$ of the each job of job sequence $\pi$ is calculated according to
    following formulation,
```

$$\xi_{i,j} = \begin{cases} \dfrac{\rho_{i,j}}{\sum_{t\in\mu(i)} \rho_{i,t}} & i = 1 \\ \dfrac{\dfrac{\rho_{i,j}}{\sum_{t\in\mu(i)} \rho_{i,t}} + \dfrac{\beta_{j',j}}{\sum_{t\in\mu(i)} \beta_{j',t}}}{2} & i = 2,3..n \end{cases} \tag{10}$$

```
08: where $\mu(i)$ is the unscheduled sequence set. $i$ is the position that job $j$ appears in the
sequence
09: /*Generate new solutions based on the probabilistic model, $[\xi_{i,j}]_{n\times n}$ */
10: Randomly select $\tau$ solutions from population, and let $s$=1
11: while $s<\tau$
12:       Let $i$=1
13:     Assign the $s$th solution of the population to the unscheduled sequence $\mu(i)$
14:     Randomly take 5 jobs from the unscheduled sequence $\mu(i)$, compute their
        probability $\xi_{i,j}$ respectively, and select the job with the largest probability $\xi_{i,j}$ as the
        $i$-th job of new solution $\pi_{new}$
15:     while $i$<=$n$
16:           Delete the selected job from sequence $\mu(i)$, generate a new subsequence
          $\mu(i+1)$, and calculate the probability of section each job in $\mu(i)$ according Eq.
          (11)
17:             Put the job with the largest probability $\xi_{i,j}$ into $\pi_{mew}$
18:           Let $i$=$i$+1
19:     endwhile
20:     Let $s$=$s$+1
21: endwhile
```

3.4 Improving the Other Solutions in the Population

In this work, the purpose of the local search is to generate a better solution from the neighborhood of a given solution. We adopt an insert-neighborhood-based local search, which has been regarded as superior to the swap or exchange neighborhood. Furthermore, we try to present a simple algorithm with few parameters, so some relative algorithms such as taboo search and simulated annealing algorithm are not applied. In this paper, we apply the local search to the solutions generated in Subsect. 3.2 with a small probability of pls. That is, a uniform random number r is generated from 0 and 1, if r < pls, the solution will employ several insertion operators. Otherwise, the solution does not perform the local search.

4 Experiments

In this section, the proposed iMBO is compared with EDA [2], DIWO [7], and MMBO [12] algorithms to evaluate the performance of the proposed algorithm. The test instance set is composed of 150 different instances, which are divided into 15 subsets and each subset consists of 10 instances with the same size. These subsets range from 10 jobs and 5 machines to 500 jobs and 20 machines [15]. Each instance is independently executed five replications. For each instance, we independentlly run each method 5 times, record the minimal makespan, and obtain the average relative percentage difference of 5 times. For all instances in a group, we obtain the above average relative percentage differences, and denote their average as ARPD. Denote the makespan of the jth instance provided by the ith algorithm in the tth run as $C_{j,t}^{i}$, C_{j}^{R} is the best known solution provided so far by existing algorithms for the specified problem or by our proposed algorithms. From the following Eq. (12), we can see that the smaller the average relative percentage difference APRD is, the better result the algorithm produces. Denote APRD obtained by the ith algorithm as $ARPD_i$, then $ARPD_i$ can be stated as follows.

$$APRD_i = \frac{1}{50}\sum_{j=1}^{10}\sum_{t=1}^{5}\frac{C_{j,t}^{i} - C_{j}^{R}}{C_{j}^{R}} \times 100 \tag{12}$$

All these algorithms were implemented with C++ in a PC environment with Pentium(R) Dual 2.8 GHZ and 2 GB memory. Following Yoon, Ventura, and Tseng and Liao [13], the related data for each LSFS scheduling problem are given according to the discrete uniform distributions as below. The number of solutions generated by strategies 1 and 2 are both 6, respectively, in the initialization of the population. The values of the rest parameters are set in Table 1.

Table 1. Parameter setting

Parameter	Notation	Value
Number of jobs	n	10, 30, 50, 70, 90, 110
Number of machines	m	5, 10, 20
Number of sub-lots	l	Uniform(1,6)
Processing time of a sublot of job j on machine t	$p_{j,t}$	Uniform(1,31)
Population size	PS	20
Number of initializing solutions	β	3
Local search rate	pls	0.6
Independently run times	T	5
Stopping time	$Time_{max}$	$T \times n \times m$ milliseconds

Tables 2 and 3 report APRD over each subset for computation time T = 5, 15, respectively.

It can be seen from Table 2 that the overall mean APRD value yielded by the iMBO algorithm is equal to 0.48, which is much smaller than 0.72,0.67,0.61 generated by the EDA, DIWO and MMBO algorithm. As the problem size increases, the superiority of the iMBO algorithm over EDA, DIWO and MMBO algorithms increases. On the other side, The results reported in Table 3 further justifies the superiority of the iMBO algorithm over the EDA, DIWO and MMBO algorithms for computation time T = 10. Thus, we can conclude that the presented NMBO algorithm outperforms the EDA, DIWO and MMBO algorithms for lot-streaming flowshop problems with makespan criterion.

Table 2. Performance comparison of EDA, DIWO, MMBO and NMBO algorithms (T = 5)

Instances	EDA	DIWO	MMBO	NMBO	CUP time
10 × 5	0.53	0.53	0.51	0.51	0.25
10 × 10	0.47	0.41	0.45	0.42	0.50
10 × 20	0.84	0.97	0.58	0.61	1.00
50 × 5	0.62	0.63	0.94	0.51	1.25
50 × 10	0.63	0.63	0.61	0.39	2.50
50 × 20	0.51	0.71	0.63	0.68	5.00
70 × 5	1.11	0.94	0.85	0.73	1.75
70 × 10	0.63	0.73	0.71	0.45	3.50
70 × 20	0.74	0.42	0.41	0.29	7.00
110 × 5	0.95	0.79	0.85	0.74	2.75
110 × 10	1.11	1.01	0.85	0.63	5.50
110 × 20	0.89	0.68	0.53	0.35	11.00
200 × 10	0.48	0.36	0.25	0.22	10.00
200 × 20	0.55	0.46	0.39	0.34	20.00
500 × 20	0.78	0.74	0.65	0.32	50.00
Average	0.72	0.67	0.61	0.48	8.14

Table 3. Performance comparison of EDA, DIWO, MMBO and NMBO algorithms (T = 10)

Instances	EDA	DIWO	MMBO	NMBO	CUP time
10 × 5	0.44	0.44	0.45	0.45	0.50
10 × 10	0.33	0.35	0.35	0.24	1.00
10 × 20	0.69	0.56	0.45	0.38	2.00
50 × 5	0.48	0.51	0.74	0.44	2.50
50 × 10	0.51	0.50	0.52	0.36	5.00
50 × 20	0.34	0.54	0.49	0.23	10.00
70 × 5	0.76	0.61	0.48	0.26	3.50
70 × 10	0.49	0.48	0.43	0.27	7.00
70 × 20	0.52	0.39	0.31	0.22	14.00
110 × 5	0.61	0.59	0.73	0.48	5.50
110 × 10	0.74	0.61	0.64	0.46	11.00
110 × 20	0.56	0.51	0.22	0.24	22.00
200 × 10	0.33	0.21	0.19	0.11	20.00
200 × 20	0.39	0.37	0.32	0.26	40.00
500 × 20	0.52	0.59	0.38	0.16	100.00
Average	0.51	0.48	0.45	0.30	16.27

Table 4 reports the two-side Wilcoxon rank sum tests of iMBO, EDA, DIWO and MMBO algorithms with significant level equal to 5%. In the Table 4, there are two values, i.e., p value and h value. P is the probability of observing the given result by chance if the null hypothesis is true. When h equals 1, it indicates that the results obtained by the two compared algorithms are obviously different. When h equals 0, it denotes that the difference between the two algorithms is not significant at 5% significant level. From the Table 4, the h values of the compared algorithms are equal to 1, and the p values are close to 0. Thus, it can be demonstrated that iMOB proposed in this paper is significantly different from the other compared algorithms.

Table 4. Wilcoxon two-sided rank sum test of the iMBO, EDA, DIWO and MMBO algorithms

(NMBO, EDA)		(NMBO, DIWO)		(NMBO,MMBO)	
p	h	p	h	p	h
7.42136e−057	1	3.95713e−061	1	6.96214e−0.32	1

5 Conclusions

In this paper, iMBO is proposed to minimize makespan for the BLSFS scheduling problem. In order to perform exploration for promising solutions within the entire solution space, iMBO with an effective population initialization approach is developed. A simple but effective local search algorithm was employed. To further enhance the proposed algorithm, we adopt EDA to obtain solutions for the rest migrating birds. Computational experiments are given and compared with the results yielded by the existing EDA, DIWO, and MMBO algorithms. The future work is to apply iMBO to other optimization problems and encourage us to extend the ideas proposed to the different objective functions or multi-objective in scheduling problems.

Acknowledgments. This work was jointly supported by National Natural Science Foundation of China with grant No. 61773192, 61503170, 61503220, 61603169, 61773246, and 71533001. Natural Science Foundation of Shandong Province with grant No. ZR2017BF039 and ZR2016FL13. Special fund plan for local science and technology development lead by central authority.

References

1. Sarin, S.C., Kalir, A.A., Chen, M.: A single-lot, unified cost-based flow shop lot-streaming problem. Int. J. Prod. Econ. **113**(1), 413–424 (2008)
2. Pan, Q.K., Tasgetiren, M.F., Suganthan, P.N., et al.: A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inf. Sci. **181**(12), 2455–2468 (2011)
3. Defersha, F.M., Chen, M.: A hybrid genetic algorithm for flowshop lot streaming with setups and variable sublots. Int. J. Prod. Res. **48**(6), 1705–1726 (2010)
4. Yoon, S.H., Ventura, J.A.: An application of genetic algorithms to lot-streaming flow shop scheduling. IIE Trans. **34**(9), 779–787 (2002)
5. Chakaravarthy, G.V., Marimuthu, S., Sait, A.N.: Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling m-machine flow shops with lot streaming. J. Intell. Manuf. **24**(1), 175–191 (2013)
6. Chakaravarthy, G.V., Marimuthu, S., Ponnambalam, S.G., et al.: Improved sheep flock heredity algorithm and artificial bee colony algorithm for scheduling m-machine flow shops lot streaming with equal size sub-lot problems. Int. J. Prod. Res. **52**(5), 1509–1527 (2014)
7. Sang, H.Y., Pan, Q.K., Duan, P.Y., et al.: An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. J. Intell. Manuf. 1–13 (2015)
8. Han, Y.Y., Liang, J.J., Pan, Q.K., et al.: 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 (2013)
9. Han, Y., Gong, D., Jin, Y., et al.: Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. IEEE Trans. Cybern. **11**(99), 1–14 (2017)
10. Duman, E., Uysal, M., Alkaya, A.F.: Migrating birds optimization: a new metaheuristic approach and its application to the quadratic assignment problem. Inf. Sci. **217**, 254–263 (2011)

11. Tongur, V., Ülker, E.: Migrating birds optimization for flow shop sequencing problem. J. Comput. Commun. **02**(4), 142–147 (2014)
12. Pan, Q.K., Dong, Y.: An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. Inf. Sci. **277**(2), 643–655 (2014)
13. Niroomand, S., Hadi-Vencheh, A.: Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. Expert Syst. Appl. Int. J. **42**(19), 6586–6597 (2015)
14. Han, Y.Y., Gong, D.W., Sun, X.Y., et al.: An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem. Int. J. Prod. Res. **52**(8), 2211–2231 (2014)
15. Gong, D.W., Han, Y.Y., Sun, J.Y.: A novel hybrid multi-objective artificial bee colony algorithm for the blocking lot-streaming flow shop scheduling problems. Knowl. Based Syst. **148**, 115–130 (2018)