Improved Migrating Birds Optimization Algorithm to Solve Hybrid Flowshop Scheduling Problem With Lot-Streaming

PING WANG\(^1\), HONGYAN SANG\(^1\), (Member, IEEE), QIUYUN TAO\(^1\), HENGWEI GUO\(^1\), JUNQING LI\(^1\), (Member, IEEE), KAIZHOU GAO\(^2\), (Member, IEEE), AND YUYAN HAN\(^1\)

\(^1\)School of Computer Science, Liaocheng University, Liaocheng 252059, China
\(^2\)Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China

Corresponding author: Hongyan Sang (sanghongyan@lcu-cs.com)

This work was supported in part by the National Science Foundation of China under Grant 61503170, Grant 61773192, and Grant 61603169, in part by the Special Fund Plan for Local Science and Technology Development Lead by Central Authority, in part by the Open Project of Henan Key Laboratory of Intelligent Manufacturing of Mechanical Equipment, in part by the Zhengzhou University of Light Industry under Grant IM201906, in part by the Shandong Province Colleges and Universities Youth Innovation Talent Introduction and Education Program, and in part by the Research Fund Project of Liaocheng University under Grant 318011922.

Abstract Hybrid flowshop scheduling problem with lot-streaming (HLFS) has played an important role in modern industrial systems. In this paper, we present an improved migrating birds optimization (IMBO) algorithm for HLFS to minimize makespan. To ensure the diversity of initial population, a Nawaz–Enscore–Ham (NEH) heuristic algorithm is used to generate the leader, and the remaining solutions are randomly generated. According to the characteristics of the HLFS problem, we propose a combined neighborhood search structure that consists of four different neighborhood operators. We design effective local search procedure to explore potential promising domains. In addition, a reset mechanism is added to avoid falling into local optimum. Extensive experiments and comparison demonstrate the feasibility and effectiveness of the proposed algorithm.

Index Terms Hybrid flowshop, lot-streaming, migrating birds optimization algorithm, scheduling problem.

I. INTRODUCTION

In the process of production and processing, how to allocate resources reasonably, make full use of resources, and reduce enterprise resource consumption is an important part of enterprise resource allocation [1]. The effective scheduling approaches are beneficial to improve the production efficiency and resource utilization rate of manufacturing enterprises, thus bringing profits to enterprises [2]–[5]. Therefore, the production scheduling problem has always been a focus of research in the field of manufacturing science.

The hybrid flowshop scheduling (HFS) problem is a combination of parallel machines scheduling and flowshop scheduling, also called flexible flow shop scheduling. The HFS problem has important practical applications in industries, including glass [6], steel [7], [8], plastic [9], and pharmaceutical industries [10] et.al. In the HFS problem, the job is not allowed to move to the next stage until the process is completed. It increases the waiting time of the machine. It may have a negative impact on scheduling efficiency, and even cannot adapt to the actual production system. The HFS problem is a classic scheduling model, and many researchers have proposed many solutions to this problem, e.g., branch-and-bound (B&B) [11] and mixed-integer programming (MIP) [12]. Because of the NP-hardness of the HFS problems, these methods can only solve the problems of small scale and simpler.

Lot-streaming refers to dividing a large job into multiple sublots. If the processing of one sublot is completed, the sublots will be transferred to the next stage for processing. Thus, it reduces the idle time of the machine. In this study, we focus on hybrid flowshop scheduling problem with lot-streaming (HLFS). The HLFS problem is more complicated than the HFS problem. So, it is also NP-hardness of the problem. The HLFS problem is an extension of the traditional hybrid flowshop scheduling problem, and is closer to...
a real production environment [13]. It has important practical applications in textile, chemical, steel [14] and many others industries. Li et al. [15] proposed the hybrid flowshop scheduling (HFS) lot-streaming problems, where the variable sub-lots constraint is considered to minimize four objectives. Tzu et al. [16] considered the lot-streaming constraints and applied this to the hybrid flow shop scheduling problem. Dios et al. [9] considered the hybrid flow shop scheduling problem of integrated batching and lot-streaming with variable sublots to reduce the total weighted completion time. Nejati et al. [17] studied a two-stage assembly lot-streaming hybrid flow shop scheduling problem with a work shift constraint to minimize the sum of weighted completion times of products in each shift. Zhang et al. [18] studied a two-stage multi-job hybrid flows shop problem with the lot-streaming.

The migrating birds optimization (MBO) algorithm is an emerging intelligence optimization algorithm. It simulates the process of migrating birds maintaining a V-shaped flight formation during the migration process to reduce energy loss [19]. The algorithm is widely used, and it was first applied to solve the problem of secondary allocation. With the further research of research scholars, MBO algorithm has been shown very robust and adaptive, and successfully applied to different fields. Ulker and Tongur [20] used the migratory bird optimization algorithm to solve the backpack problem. The performance of the algorithm was tested. Tongur and Ülker [21] proposed an improved MBO algorithm to solve discrete traveling salesman problem. Obtained results showed that proposed method was superior to basic algorithm to solve discrete traveling salesman problem. Obtained results showed that proposed method was superior to basic algorithm to solve discrete traveling salesman problem. Tongur and Ülker [21] proposed an improved MBO algorithm to solve discrete traveling salesman problem. Obtained results showed that proposed method was superior to basic algorithm to solve discrete traveling salesman problem. Zhang et al. [23] proposed a multi-objective MBO algorithm to solve the multi-objective hybrid flowshop scheduling problem related to production efficiency and production environment instability in a dynamic shop environment. However, to the best of our knowledge, the MBO algorithm has not been used in the HLFS problem. As the successful applications of MBO algorithm, this paper proposes an improved migrating birds optimization (IMBO) algorithm for the HLFS problem to minimise the makespan. Compared with the basic MBO algorithm, the improved strategies of IMBO algorithm are as follows. To ensure the diversity of initial population, the leader is generated by the Nawaz-Enscore-Ham (NEH) heuristic algorithm, while the other solutions are randomly generated. According to the characteristics of the HLFS problem, we propose a combined neighborhood search structure that consists of four different neighborhood operators. We design effective local search procedure to explore potential promising domains. In addition, a reset mechanism is added to avoid falling into local optimum.

The main content of this paper is organised as follows. In section II, the HLFS problem is formulated. In Section III, it mainly describes the basic MBO algorithm. In Section IV proposes an IMBO algorithm for the HLFS problem. In Section V, the experimental results and analysis. Section VI is the conclusions.

II. PROBLEM STATEMENT

In a traditional HFS problem, a job is inseparable. If one job is not finished, it is not allowed to transfer to the next machine. Thus, the machine waits for a long time. It is wasting significant resources. However, in an actual production process, a larger job often contains several sublots. That is, the job can be divided into several small sublots. When a sublot is processed on the current machine, it can be moved to the next machine for processing (that is, lot-streaming). Lot-streaming can reduce the waiting time of the machine and avoid waste of resources.

Hybrid flowshop scheduling problem with lot-streaming can be described as: there are \( j \) jobs, which need to go through \( n \) processing stages, each stage has \( m (m \geq 1) \) machines. At any stage, job \( j \) uses the first available machine rule to select a machine. Each job \( j \) can be divided into \( l \) number sublots with equal size. When a sublot of the job is completed at the machines, it will be processed in the next machine. Each machine can only process one job at a time. For example, if the sublots of a job are processed on this machine, the sublots of other jobs cannot be processed on this machine until the processing of the previous job is completed. All sublots of one job must be processed on the same machine. For example, when the first sublot of a job is assigned to a machine, the other sublots of the job are also assigned to the same machine, until all sublots of the job are processed. The optimization objective of the hybrid flowshop scheduling problem without lot-streaming is to find a permutation \( \pi^* \) from the set of all the job permutations \( \Pi \) to get the minimum makespan, i.e.,

\[
C_{\text{max}}(\pi^*) = \min C_{\text{max}}(\pi), \quad \forall \pi \in \Pi
\]  

III. THE BASIC MBO ALGORITHM

The basic MBO algorithm is an algorithm based on neighborhood search technology. It is optimized by simulating a V-shaped flight formation of migratory birds. The V-shaped arrangement of all solutions is shown in Fig. 1. There will be one leader and multiple followers in the V-shaped formation. The MBO algorithm mainly has four stages: algorithm initialization, leader evolution, followers evolution and leader change [24]. The flowchart of Fig. 2 details the process of the MBO algorithm.

A. ALGORITHM INITIALIZATION

Initialization is divided into two parts. The first step is to set the parameters of the MBO algorithm, including the number of initial solutions \( s \), the number of neighboring solutions produced \( k \), the number of neighboring solutions to be shared with the next solution \( x \), and the number of tours \( G \).

Another step is to generate the initial solutions. The performance of the algorithm is affected by the initial solutions. In the basic MBO algorithm, \( s \) solutions are randomly
selected in the feasible solution space as the initial population. Then, the solutions are used to simulate the flight of migratory birds, and placed on a hypothetical V formation. Among the solutions, one solution is chosen as the leader, the remaining solutions are followers, and the followers are evenly divided into two groups. There are \((s-1)/2\) solutions in list \(P_l\) and \((s-1)/2\) solutions in list \(P_r\).

**B. LEADER EVOLUTION**

Among the \(s\) initial populations, the leader evolution will improve current solution through neighborhood search, and generate \(k\) solutions according to its neighborhood. If there is a better solution, it will be selected to update the leader. The remaining \(x\) better solutions are given to the followers.

**C. FOLLOWERS EVOLUTION**

According to the neighborhood structure, the followers in \(P_l(P_r)\) will generate \(k-x\) neighbors. Find the best solution among the \(k-x\) neighbors and the \(x\) solutions are given by the previous individuals. If there is a better solution, and the solution will be selected to update the current followers. Otherwise, the followers remain unchanged. The remaining \(x\) better solutions will be provided to the next followers.

**D. LEADER CHANGE**

When the evolution cycle is completed, the leader solution will be changed. The first solution of followers in the \(P_l\) or \(P_r\) will becomes the next leader. The former leader will be replaced at the end of the \(P_l\) or \(P_r\). In the same way, all individuals will be having the opportunity to become the next leader.

**IV. THE PROPOSED IMBO ALGORITHM FOR HLFS PROBLEM**

**A. SOLUTION REPRESENTATION**

The proposed algorithm adopts the encoding method of arrangement, that is, all the jobs are numbered. In the first stage, the machine is selected according to the number of each job. For the remaining stages, an idle machine is arranged for the job. An example is given as follow. There are five jobs. The process is divided into two stages, each stage has two parallel machines. The processing sequence is \(\pi = (1, 3, 4, 5, 2)\) at the first stage. Table 1 lists the relevant data about the example. For the first stage, the first available machine is selected and processed in sequence.

As can be seen from Fig. 3, in the HLFS problem, the completion times of the five jobs in the first stage are 8, 9, 14, 19, and 20. These jobs continue to be processed in the next
stage after completion of the first stage. The final completion times of the five jobs are 10, 11, 19, 21, and 23, respectively. Thus, the makespan is 23, and the waiting time of the machine is 17. As can be seen from IV-B, in the HFS problem, the final completion times of the five jobs are 12, 15, 23, 23, and 27. The makespan is 27, and the waiting time of the machine is 23. Comparing the HLFS problem with the HFS problem, it can be seen that the makespan is reduced by 14.8% and the waiting time of the machine is reduced by 26.1%. Therefore, compared with the traditional HFS problem, it can be seen from the example that the HFLS problem can effectively reduce the waiting time of the machine and reduce the waste of resources.

B. POPULATION INITIALIZATION

In order to ensure the quality and diversity of the individual, the NEH algorithm is used to generate an individual, and the remaining individuals are randomly generated. When the search efficiency is ensured, the operability of the algorithm can also be ensured. The literatures prove that the NEH algorithm is a good heuristic algorithm. Especially in the flowshop scheduling problem, optimizing and minimizing the makespan has achieved good results [25], [26]. Therefore, the NEH algorithm is used to initialize the population. And Fig. 5 gives the pseudo code of the NEH algorithm. The NEH algorithm process description is as follows:

**Step 1:** Calculate the completion time $C_j$ for each job on all machines. The sequence $\pi$ is constructed in descending order according to $C_j$.

**Step 2:** Select two elements $\pi[0]$ and $\pi[1]$ from the sequence $\pi$, sort $\pi[0]$ and $\pi[1]$ to generate two new sequences, and calculate the makespan. The order with the smallest makespan is denoted as $\pi^*$. Then the third element $\pi[2]$ is selected from $\pi$, all possible positions in $\pi^*$ are inserted, and the sequence with the smallest makespan is calculated and recorded as a new $\pi^*$.

**Step 3:** Repeat step 2 to sequentially select elements from $\pi$ to insert into $\pi^*$. Until all the jobs have been scheduled, a complete machining sequence is obtained.

The IMBO algorithm uses the solution generated by the NEH algorithm as the leader. The rest of the solutions in the population are divided into two parts as the followers in the left line $P_l$ and $P_r$.

C. NEIGHBORHOOD STRUCTURE

In the MBO algorithm, the individual replaces itself and completes the evolution process by finding an optimal solution in the neighbourhood solution, or a better and unused neighbourhood solution generated by the previous individual in the previous search process. Therefore, the way in which the neighborhood solution is generated directly affects the effect of the algorithm. Common methods for generating neighborhood solutions are insert, exchange and randomly.

The greedy algorithm is a simple algorithm based on random local search. The greedy algorithm is mainly used to solve the problem of damage reconstruction in each iteration process. And the greedy algorithm is often applied to various flowshop scheduling problems. In the improved MBO algorithm, we propose a neighbourhood search strategy composed of four different neighbourhood structures.

The neighborhood solution of the population is generated by insertion, exchange, insertion greedy and exchange greedy. We randomly select the four different neighborhood operators to generate a combined neighborhood search strategy.

1) INSERTION

Randomly select a job from a sequence of $j$ jobs, and the selected job is randomly inserted into other positions to obtain a new sequence.

2) EXCHANGE

Randomly select a job from a sequence of $j$ jobs, and exchange the job with other jobs to obtain a new sequence.

3) INSERTION GREEDY

Randomly select a job in a sequence of $j$ jobs, the selected job is inserted into all other positions to obtain $j-1$ new sequences. According to the objective function, select the best sequence of original sequence and the new sequence.

4) EXCHANGE GREEDY

Randomly select a job in a sequence of $j$ jobs, and sequentially exchange the job with other remaining jobs to obtain $j-1$ new sequences. According to the objective function, select the best sequence of original sequence and the new sequence.

In the next section, the mixed neighborhood structure will be compared with the single neighborhood structure. The experimental comparison results will be described in detail.
D. LOCAL SEARCH

After the evolution of the leader and the followers, a local search for the optimal solution is performed in order to enhance the local search capability of the IMBO algorithm. A new solution is generated by randomly inserting or exchanging the initial solutions. If a better solution is obtained, the original solution will be replaced. If a better solution is found, or no improvement is found for $j$ iterations of the loop, the local search will end, otherwise, the local search will continue. And Fig. 6 gives the pseudo code of the local search.

E. RESET MECHANISM

The evolution of the parallel solution and the individual evolution mechanism in the MBO algorithm ensure that the MBO algorithm has a strong decentralized search ability and a centralized search ability, so that the MBO algorithm can quickly search for an approximate optimal solution. However, this method can easily cause the algorithm to converge prematurely and fall into a local optimum. To avoid this, a reset mechanism is designed based on the MBO algorithm.

In the population, we set the age of the individual. Describe the individual’s update process by age of solution, and set an upper limit ($L$) for it. The initial solution age is set to 1. If the optimal solution does not change after evolution, the solution age is increased by 1. If the optimal solution is not changed after $L$ times of evolution, we need to perform the local search of the original solution, and generate a new solution instead of the original solution, that is, update the optimal solution.

F. PROCEDURE OF THE ALGORITHM

To facilitate the description of the algorithm, the following variables are defined: $flag$ is used to mark the follower that replaces the leader, and when $flag$ is 1(0), it means replacing the leader with the first bird in the left (right) list; $k$ is the number of neighbourhood solutions produced by the leader; $x$ is the number of neighbouring solutions to be shared with the next solution; $G$ is the number of tours; $P_l$ ($P_r$) is a set of shared neighbourhood solutions from each individual to the next individual in the left (right) list; $S$ is the neighbour solution set generated by the neighbours in the left (right) list; $L$ is the upper limit of the solution age; $T$ is the termination condition. The complete IMBO algorithm process is as follows:

Step 1: Algorithm initialization. Set parameters $s, k, x, G, T$, and $flag=1$. Generate an initial solution by the NEH algorithm, and this initial solution is the leader of this algorithm. Randomly generate $s-1$ solutions as the followers, and set the lists $P_l$ and $P_r$, and each list contain $(s-1)/2$ solutions.

Step 2: Evolve the leader. The $k$ neighbourhood solutions of the leader are generated by the combined neighbourhood structure, and the best neighboring solution is selected to update the leader. In the remaining solutions, $x$ solutions are selected, and added to the shared neighborhood solution sets $P_l$ and $P_r$, for the evolution with of followers.

Step 3: Evolve the follower. Based on the neighbourhood structures, $k-x$ neighbourhood solutions are generated for each follower in the left lists, and are denoted as $S$. If the optimal solution in $S U P_1$ is superior to the current solution, then the current solution is abandoned, and the new solution becomes the next follower. The $P_1$ is reset to be null, and the other unused and better neighbourhood solutions in $S$ are added to the $P_1$ for the evolution of another follower. The right list the same as the left list.

Step 4: Update the current optimal solution.

Step 5: Local search. Perform local search for the current optimal solution.

Step 6: Update the solution age of the optimal solution, and perform a reset mechanism if no change occurs over $L$ generations.

Step 7: Determine whether to reach the number of tours $G$. If not, perform step 2, otherwise, proceed to step 8.

Step 8: Leader change. If $flag = 1$, the leader solution will move to the end of the left list, becomes a follower. The first follower in the left list moves forward to becomes the new leader, and sets the $flag = 0$. If $flag = 0$, the leader solution will move to the end of the right list, becomes a follower. The first follower in the right list moves forward to becomes the new leader, and sets the $flag = 1$.

Step 9: The termination checks. If the stop condition $T$ has been satisfied, then perform step 10; otherwise, perform step 2.

Step 10: The algorithm ends and outputs the results.

V. COMPUTATIONAL RESULTS

The algorithms used in the experimental analysis were coded in C++. The experimental programs ran on an advanced micro devices (AMD) A12-9700P CPU @3.4 GHz with 4.0 GB main memory in the WIN 10 OS. During the experiment, we generated a series of instances for the IMBO algorithm, the number of jobs $j = \{20, 40, 60, 80, 100\}$ and the number of stages $n = \{5, 10\}$. We combined $j$ and $n$ in turn to generate ten instances, and performed 100 independent experiments on each instance. The processing time of the jobs was arbitrarily selected in the range $[1, 31]$, the number of sublots was arbitrarily selected in the range $[1, 6]$, and the number of parallel machines in each stage was arbitrarily selected in the range $[1, 5]$. The parameters of the algorithm have a significant impact on the speed of the algorithm and the accuracy of the solution. Therefore, the appropriate setting of parameters is very important. We fix the parameters: the initial solutions ($s$), the number of neighboring solutions produced by the leader ($k$), the number of neighboring solutions to be shared with the next solution ($x$), the number of tours ($G$), termination condition ($T$), and solution age upper limit ($L$).

A. PARAMETER SETTINGS

It can be seen from the reference that if $s$ is too small, the distribution density of initialized individuals in solution space will be too small, and the diversity of population will be
affected. If $s$ is too large, the neighborhood of the individual may overlap and the neighborhood solution of the individual cannot be completely searched for a limited time. $G$ mainly affects the convergence speed of the algorithm. When the $G$ is large, the convergence speed of the algorithm is faster, and when the $G$ is smaller, the convergence speed of the algorithm is slower. Generally, the values of $k$ and $x$ are smaller, and the relationship between $k$ and $x$ must satisfy the following conditions: $k \geq (2 \times x + 1)$. If the value of $k$ or $x$ is too large, it will consume too much time, and it will also cause premature convergence of the algorithm. Under this condition, we also need to ensure that the individual has sufficient neighborhood solutions. In the remaining solutions, choose $x$ best solutions in the shared neighbor sets. Therefore, we set the value of $k$ to 3 and the value of $x$ can only be set to 1.

Then other three parameters can set by orthogonal experiment. Three reasonable levels of the three parameters is listed in Table 2.

Select 60 \times 10 instance from ten instances for experiments. Use the orthogonal array $L_9$ for experimental analysis, and set the $j \times n$ milliseconds as the termination criterion ($T$). The instance performs 100 independent experiments and calculates the average value (AVG). The results are listed in Table 3.

At three different levels in Table 3, $k_1$, $k_2$ and $k_3$ represent different AVG. The level corresponding to the minimum value of $k_1$, $k_2$ and $k_3$ is the level at which the parameter has the greatest influence. The effect of the parameters on the algorithm can be seen by the R (Range). The larger the R, the greater the impact of this parameter on the algorithm. The order of the R is arranged in descending order, and put the sequence numbers in the brackets after R. The trend of the three parameters at different levels is shown in Fig. 7. After the testing in the IMBO algorithm, we set the relevant parameters as follows: $s = 51$, $G = 10$, $k = 3$, $x = 1$, $L = 50$.

### B. EVALUATION OF THE NEH ALGORITHM

In the simulation process, 10 examples were tested. The instance performs 100 independent experiments and
calculates the AVG. The relative percentage increase (RPI) calculation formula is as follows:

$$RPI = \frac{(C - C^*)}{C^*} \times 100\%,$$

Among them, the $C$ represents the makespan generated by each algorithm in the experiment, and the $C^*$ represents the minimum makespan among all algorithms in the experiment. Clearly, lower RPI values are preferred in comparison. Population initialization is the starting point of the algorithm. In order to verify the influence of NEH algorithm, the IMBO algorithm of the initial population solution generated by the NEH algorithm is compared with the MBO algorithm of the randomly generated initial population solution. The comparison results are shown in Table 4.

It can be easily seen from Table 4 that the NEH algorithm has good quality, which can effectively improve the search quality of the MBO algorithm.

### C. EVALUATION OF THE COMBINED NEIGHBORHOOD STRATEGY

We first verify the effectiveness of the combined neighborhood strategy presented above. The insertion neighborhood strategy (IMBO$_I$ for short), exchange neighborhood strategy (IMBO$_E$), insertion greedy neighborhood strategy (IMBO$_{IG}$), exchange greedy neighborhood (IMBO$_{EG}$) and combined neighborhood strategy (IMBO$_M$) are applied to the IMBO algorithm respectively. The results are given in Table 5. From the data we can observe that the IMBO$_M$ algorithm has a relatively low RPI value of 0.50% when comparing with IMBO$_I$(0.79%), IMBO$_{IG}$(0.60%), IMBO$_E$(0.79%) and IMBO$_{EG}$(0.61%) in most instances. Therefore, the combined neighborhood strategy is more effective in the IMBO algorithm. As the number of jobs increases, the IMBO$_M$ algorithm is more superior than the other four single neighborhoods.

To further verify whether the difference is statistically significant, we also take a boxplot. The compared algorithms are considered as factors. It can accurately describe the distribution of data and facilitate the screening of data. The compared algorithms are considered as factors. The results are shown in Fig. 8.

As can be seen from Fig. 8, there are significant differences between the neighborhood strategy. IMBO$_I$ algorithm and IMBO$_E$ algorithm have abnormal values. The median of IMBO$_I$ algorithm and IMBO$_E$ algorithm are basically the same, indicating that the RPI of the two neighborhood structures are basically the same. Based on the upper and lower limits of the boxplot, the RPI range of IMBO$_{IG}$ algorithm and IMBO$_{EG}$ algorithm is less than IMBO$_I$ algorithm and IMBO$_E$ algorithm. From the position and height of the quartile of the boxplot, the RPI of IMBO$_{IG}$ algorithm is more concentrated than 9 IMBO$_{EG}$ algorithm. The median of IMBOM algorithm is smaller than IMBO$_{IG}$ algorithm. And the position and height of the quartile of IMBO$_M$ algorithm are both smaller than IMBO$_{IG}$ algorithm. Indicating that the RPI of IMBO$_M$ algorithm is smaller and more stable.
TABLE 6. Comparison of IMBO\textsubscript{R1}, IMBO\textsubscript{R2}, IMBO\textsubscript{R3} and IMBO\textsubscript{R}.

| Instances | IMBO\textsubscript{R1} | IMBO\textsubscript{R2} | IMBO\textsubscript{R3} | IMBO\textsubscript{R} |
|-----------|------------------------|------------------------|------------------------|------------------------|
| 20×5      | 0.14                   | 0.14                   | 0.14                   | 0.14                   |
| 20×10     | 0.38                   | 0.38                   | 0.37                   | 0.36                   |
| 40×5      | 0.30                   | 0.30                   | 0.29                   | 0.29                   |
| 40×10     | 0.62                   | 0.62                   | 0.62                   | 0.53                   |
| 60×5      | 0.57                   | 0.56                   | 0.56                   | 0.47                   |
| 60×10     | 0.77                   | 0.75                   | 0.64                   | 0.61                   |
| 80×5      | 0.59                   | 0.60                   | 0.58                   | 0.48                   |
| 80×10     | 0.88                   | 0.88                   | 0.89                   | 0.75                   |
| 100×5     | 1.27                   | 1.45                   | 1.45                   | 0.61                   |
| 100×10    | 2.16                   | 2.16                   | 2.15                   | 0.77                   |
| Mean      | 0.77                   | 0.78                   | 0.75                   | 0.50                   |

FIGURE 9. Analysis of local search and reset mechanism.

D. EVALUATION OF LOCAL SEARCH AND RESET MECHANISM

First, we excluded the reset mechanism from the IMBO algorithm to prove the superiority of the local search. We apply two single local search methods (e.g., insertion local search, exchange local search) and the mixed local search to the solutions. The three methods are named as follows: IMBO\textsubscript{R1} (the insertion local search), IMBO\textsubscript{R2} (the exchange local search), and IMBO\textsubscript{R3} (the mixed local search). Then in order to prove the superiority of the reset mechanism, the IMBO\textsubscript{R3} algorithm adds the reset mechanism, that is IMBO\textsubscript{R}.

The results are reported in Table 6.

From the data in Table 6, it can be analyzed that the RPI value of IMBO\textsubscript{R3} (0.75%) algorithm is lower than IMBO\textsubscript{R1} (0.77%) and IMBO\textsubscript{R2} (0.78%). This finding indicates that the IMBO\textsubscript{R3} algorithm performs better results than IMBO\textsubscript{R1} algorithm and IMBO\textsubscript{R2} algorithm. The effectiveness of the local search is proved. The RPI generated by the IMBO\textsubscript{R} algorithm is smaller than the IMBO\textsubscript{R3} algorithm, which is equal to 0.50%. The reset mechanism in IMBO\textsubscript{R} algorithm improves the performance of IMBO\textsubscript{R3} algorithm. Therefore, the effectiveness of the reset mechanism is proved.

As can be seen from Fig. 9, there are significant differences between the local search. The IMBO\textsubscript{R1}, IMBO\textsubscript{R2} and IMBO\textsubscript{R3} algorithm have abnormal values. The median of IMBO\textsubscript{R1} algorithm and IMBO\textsubscript{R2} algorithm are basically the same, indicating that the RPI of the two neighborhood structures are basically the same. The median of IMBO\textsubscript{R3} algorithm is slightly smaller than IMBO\textsubscript{R1} algorithm and IMBO\textsubscript{R2} algorithm. But the median of IMBO\textsubscript{R} algorithm is smaller than IMBO\textsubscript{R3} algorithm. And the position and height of the quartile of IMBO\textsubscript{R} algorithm are also both smaller than other algorithms. Indicating that the RPI of IMBO\textsubscript{R} algorithm is smaller and more stable. The IMBO\textsubscript{R} algorithm with the reset mechanism further improves the results of IMBO\textsubscript{R3} algorithm.

E. EVALUATION OF THE PROPOSED IMBO ALGORITHM

This section intends to compare the IMBO algorithm with other intelligent algorithms. By comparing the effectiveness of the IMBO algorithm is proved. The other three algorithms are widely used intelligent optimization algorithms, especially in the application of flowshop scheduling. Next, we evaluate the effectiveness of the IMBO algorithm to minimize the makespan in the HLFS problem. The other three algorithms are applied to the HLFS problem and compared with the IMBO algorithm. They are MBO algorithm [27], extended genetic algorithm (EGA) [29], discrete invasive
Weed optimization (DIWO) algorithm [31], and artificial bee colony (ABC) algorithm [32]. According to the characteristics of the HLFS problem, we need to make necessary adaptations to these three algorithms to make comparison possible. For the method of generating the initial population, all algorithms are unified as described in Section V. The parameters of each algorithm are unchanged and using the parameter settings in the literature. In order to ensure the accuracy of the algorithm comparison, we also used the NEH algorithm to initialize the population in all algorithms. The algorithms used in the experimental analysis are encoded in C++. Then, setting the $j \times n$ milliseconds as the termination criterion. The calculation results of different algorithms are shown in Table 7.

Table 7 shows that the solution quality of the IMBO algorithm is better. The average RPI produced by the proposed IMBO algorithm is the smallest, which is 0.50%. The average RPI values of the IMBO algorithm is smaller than the EGA algorithm (3.40%), ABC algorithm (2.51%), MBO algorithm (0.78%) and DIWO algorithm (0.73%). Furthermore, for the ten instances, the IMBO algorithm mostly achieves the minimum RPI value, and even surpassed other algorithms by a considerable amount. Therefore, it can be concluded that the proposed IMBO algorithm is very effective for HLFS problem.

In order to verify the convergence ability of the IMBO algorithm, Fig. 10 presents the convergence curves of MBO algorithm and IMBO algorithm for one instance of $60 \times 10$. It can be seen from the convergence trend of the curve in the figure. Compared with the MBO algorithm, the IMBO algorithm has better convergence performance and can converge to a better result.

Finally, we performed a multifactor analysis of variance (ANOVA) on several algorithms. In Fig. 11, for different algorithms, the least significant difference (LSD) intervals (at the 95% confidence level) are plotted. Comparing the results of the algorithms in Fig. 11, it can be seen that the performance of the IMBO algorithm is significantly better than other algorithms. Therefore, it can be proven that the IMBO algorithm is the most effective for minimizing makespan in HLFS problem.

VI. CONCLUSION

The hybrid flowshop scheduling problem with lot-streaming (HLFS) is often applied to various modern practical production systems. In order to solve the HLFS problem, an improved migratory birds optimization (IMBO) algorithm is proposed to optimize the makespan. According to the characteristics of the problem, to ensure the diversity of initial population, a Nawaz–Enscore–Ham (NEH) heuristic algorithm is used to generate the leader, and the remaining solutions are randomly generated. In order to ensure the solution quality and convergence speed of the algorithm, the combined neighborhood search strategy of the individual is constructed by four different neighborhood operators. And then a local search method is designed to enhance the local search ability of the algorithm. In addition, a reset mechanism is added to avoid falling into local optimum. In the randomly generated example of HLFS problem, the simulation experiment shows that the IMBO algorithm is more effective and feasible than other algorithms in the literature.

The research on HLFS problem has practical significance. In addition, future research directions will involve more complex scheduling problems. In particular, the dynamic constraint problem in actual production scheduling will be the focus of the next research.

REFERENCES

[1] L. E. I. Deming, “Novel teaching-learning-based optimization algorithm for low carbon scheduling of flexible job shop,” Control Decis., vol. 32, no. 9, pp. 1621–1627, 2017.
[2] Q.-K. Pan and Y. Dong, “An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation,” Int. J. Prod. Res., vol. 57, no. 9, pp. 3388–3398, 2019.
[3] Q.-K. Pan, L. Wang, J.-Q. Li, and J.-H. Duan, “A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation,” Omega, vol. 45, pp. 42–56, Jun. 2014.
[4] N. Zhao, S. Ye, K. Li, and S. Chen, “Effective iterated greedy algorithm for flow-shop scheduling problems with time lags,” Chin. J. Mech. Eng., vol. 30, no. 3, pp. 652–662, May 2017.
[5] I. Ribas and R. Companys, “Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization,” Comput. Ind. Eng., vol. 57, pp. 30–39, Sep. 2015.
[6] S. Wang, X. Wang, F. Chu, and J. Yu, “An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production,” Int. J. Prod. Res., vol. 56, no. 8, pp. 2283–2314, Apr. 2020.
[7] J. Long, Z. Zheng, X. Gao, and P. M. Pardalos, “Scheduling a realistic hybrid flow shop with stage skipping and adjustable processing time in steel plants,” Appl. Soft Comput., vol. 46, pp. 536–549, Mar. 2018.
[8] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and F. Duan, “An efficient optimization algorithm for resource-constrained steelmaking scheduling problems,” IEEE Access, vol. 6, pp. 33883–33894, 2018.
[9] M. Dios, V. Fernandez-Vigas, and J. M. Framinan, “Effective heuristics for the hybrid flow shop scheduling problem with missing operations,” Comput. Ind. Eng., vol. 115, pp. 88–99, Jan. 2018.
[10] Z.-W. Sun and X.-S. Gu, “A novel hybrid estimation of distribution algorithm for solving hybrid flowshop scheduling problem with unrelated parallel machine,” J. Central South Univ., vol. 24, no. 8, pp. 1779–1788, Aug. 2017.
QIUYUN TAO received the bachelor’s degree from the School of Computer, Liaocheng University, in 2014, where she is currently pursuing the degree. Her main research interest includes intelligent optimization and scheduling.  

HONGYAN SANG (Member, IEEE) received the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2010, and the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013. Since 2003, she has been with the School of Computer Science, Liaocheng University, where she became an Associate Professor, in 2016. She has authored more than 60 refereed articles. Her current research interests include intelligent optimization and scheduling.  

PING WANG was born in Liaocheng, Shandong, China, in 1992. She is currently pursuing the M.S. degree in computer science with Liaocheng University. Her main research interest includes intelligent optimization. In 2019, she participated in the 2019 Intelligent Simulation Optimization and Scheduling Academic Conference and won the third prize for symposium papers. She also participated in the 2019 5th International Conference on Control Science and Systems Engineering (ICC-SSE) international conferences, and published articles.
HENGWEI GUO received the bachelor’s and master’s degrees from the School of Computer Science, Liaocheng University, in 2015. His research interest includes intelligent optimization algorithms and applications.

JUNQING LI (Member, IEEE) received the master’s degree in computer science and technology from Shandong Economic University, Shandong, China, in 2004, and the Ph.D. degree from Northeastern University, Shenyang, China, in 2016. Since 2004, he was with the School of Computer, Liaocheng University, where he became a Professor, in 2017. He has been with the School of Information Science and Engineering, Shandong Normal University. He has authored more than 40 refereed articles. His current research interests include intelligent optimization and scheduling.

KAIZHOU GAO (Member, IEEE) received the B.Sc. degree from Liaocheng University, China, in 2005, the master’s degree from Yangzhou University, China, in 2008, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2016. From 2008 to 2012, he was with the School of Computer, Liaocheng University. From 2012 to 2013, he was a Research Associate with the School of Electronic and Electrical Engineering, NTU, where he was a Research Fellow, from 2015 to 2018. He is currently an Assistant Professor with the Macau Institute of Systems Engineering, Macau University of Science and Technology. His research interests include intelligent computation, optimization, scheduling, intelligent transportation, smart buildings, and next-generation port. He has published over 70 refereed articles. He serves as an Associate Editor for *Swarm and Evolutionary Computation* journal.

YUYAN HAN received the M.S. degree from the School of Computer Science, Liaocheng University, China, in 2012, and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, China, in 2016. She has been an Associate Professor with the School of Computer Science, Liaocheng University. She has authored more than 30 refereed articles. Her current research interests include evolutionary computation, multi-objective optimization, and flow shop scheduling.

---

VOLUME 8, 2020