ATLANTIS PRESS

*Advances in Intelligent Systems Research, volume 154*

6th International Conference on Measurement, Instrumentation and Automation (ICMIA 2017)

# Migrating birds optimization for hybrid flow shop scheduling problem with makespan

Caile Ren[1, a], Chaoyong Zhang[1,b], Yanbin Zhao[2,c] and Leilei Meng1[1, d]

[1]School of Mechanical Science & Engineering, Huazhong University of Science and Technology

Luoyu Road No.1037, Wuhan, 430074, China

[2] School of Mechanical and Electronic Engineering, WuHan University of Technology

Luoshi Road N0.122, Wuhan, 430070, China

[a]1966146042@qq.com, [b]zcyhust@hust.edu.cn, [c] 364880685@qq.com, [d]1084262877@qq.com

**Keywords:** Hybrid flowshop; Makespan; Metaheuristics; Migrating birds optimization

**Abstract.** This paper use migrating birds optimization algorithm to minimize makespan on hybrid flow shop problem firstly in the literature. According to the characteristics of hybrid flow shop problem, this paper proposed a job sequence adjustment model based on probability. Then combined the adjustment model with migrating birds optimization algorithm to solve standard benchmark problems and proved its validity.

## Introduction

Hybrid flow shop scheduling problem(HFSP for short), first proposed by Salvador in 1973[1]. For HFSP, there is at least one stage having more than one processor. It is due to the existence of HFSP characteristics of parallel machine, HFSP can be used for machinery, logistics, steel, textile and other industries scheduling problem. So the study of HFSP has important significance for the actual production.

Since HFSP arises, many approaches have been proposed in the literature. Ruizab[2] divided these approaches into three categories: exact algorithms, heuristics, and metaheuristics. exact algorithms, such as branch and bound (B&B)[3], solve problems mathematically; Heuristics algorithms refer to some scheduling rules, Panwalkar et al.[4] summarized 113 different rules. Metaheuristics, such as Genetic Algorithm (GA)[5], Artificial Bee Colony (ABC)[6], Ant Colony Optimization (ACO)[7] and so on.

Migrating birds optimization (MBO) algorithm is a new metaheuristic algorithm which is proposed by Duman et al. [8] in 2012 for solving quadratic assignment problems. MBO is inspired from migrating birds' V flight formation, which can reduce energy consumption during the flight.

## Problem Statement

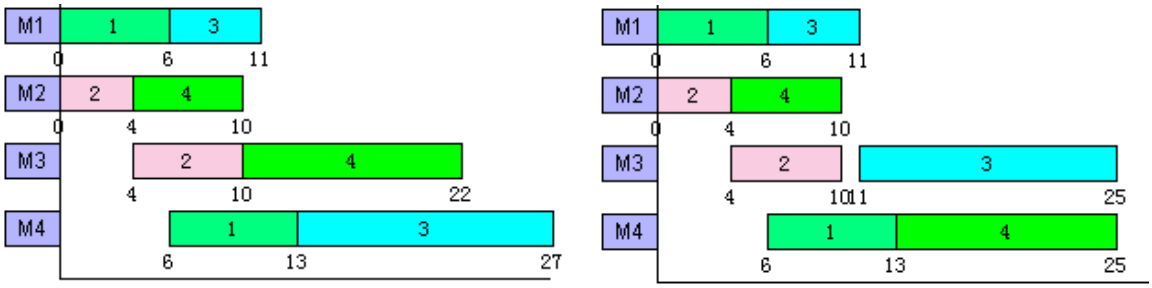
The description of HFSP is as follows: there are n jobs J={1,2,…,N} will be processed through k stages S={1,2,…,K}, each stage has parallel machines. At the same time, following constrains must be satisfied: 1)all jobs go through all stages in the same order; 2)at least one stage has more than one parallel machine; 3)each job only can be processed on one parallel machine at each stage; 4)each machine only can process one job at any time.

## Migrating Birds Optimization Algorithm for HFSP

**Encoding.** The encoding of HFSP is divided into two categories. The first class is the matrix encoding, that is, a matrix represents an individual, the rows and the columns of the matrix represent stages and jobs respectively.The second category is permutation-based encoding, that is, each solution is represented by a sequence of all jobs' index. For HFSP, Matrix encoding is cumbersome to deal with and does not make use of the characteristics of HFSP. Therefore, this paper uses permutation-based encoding.

**Decoding.** Decoding methods are divided into two categories. The first class is replacement decoding, that is, all stages use the same job sequence. The second type is permutation decoding, of which only the first stage assign jobs according to the given job sequence, the following stage assign jobs according to the completion time of jobs at last stage, that is to say, jobs with earlier finish time at stage s have the priority to be processed at stage *s*+1. This paper uses permutation decoding.

However, if you follow rules of permutation decoding strictly, you may miss a better solution. Fig. 1 is a gantt chart for a HFSP case, which has 4 jobs and 2 stages, each stage has two parallel machines respectively, different colors represent different jobs, the number on rectangles represents job index. As we can see from (a), the completion time of job 4 is 10, which is smaller than the completion time of the job 3 that is 11 at the first stage. According to the principles of permutation decoding , we should first assign job 4 and then job 3 at the second stage, under this circumstance, the makespan is 27. However, if we first assign job 3 on machine 3 and then assign job 4 on machine 4, the makespan will become 25 which is smaller than 27.




(a) (b)

Fig. 1 The gantt chart for one HFSP case

It can be seen from the above that follow the rules of permutation decoding strictly can get a good solution to a large extent, but we may miss a better solution. Therefore, on the basis of permutation decoding, this article proposed an job sequence adjustment model(JSAM for short) based on probability. As shown in Eq.1, where P is the probability, C is a coefficient, e is the natural constant. Firstly, sort job sequence in an ascending order according to jobs' completion time, and then Δ can be calculated as the difference between two adjacent jobs' completion time.

$$P = C * e^{\Delta} \tag{1}$$

Permutation decoding with JSAM described as follows:

At the first stage(s=1), determine the processing order of jobs by a given job sequence. When determine the processing order of jobs at stage s(s=2,…,K), firstly, you need to sort jobs' completion time at stage s-1 in an ascending order, then you will get completion time sequence $FT = \{ft_{a_1}, ft_{a_2}, \ldots, ft_{a_n}\}$ where $ft_{a_1} \le ft_{a_2} \le \cdots \le ft_{a_n}$ and the corresponding new job sequence $\Pi = \{a_1, a_2, \ldots, a_n\}$, calculate $\Delta_j = ft_{a_{j+1}} - ft_{a_j}$, *where* $j = 1,2,\ldots,n-1$, if $\Delta_j \neq 0$, then calculate $P_j = C * e^{\Delta_j}$, generate a random r between 0 and 1, if $r < P_j$, then change the position of job $a_{j+1}$ and job $a_j$ in $\Pi$. Eventually, you get a new job sequence $\Pi'$, and then, you can assign jobs to machines at stage s according to $\Pi'$. The pseudo code of the above steps is shown in Fig. 2.

```
Process jobs according to the given job sequence at stage one
for s=2 to K
    Calculate Π and FT according to finish time of jobs at stage s-1
    for j=1 to n
        Calculate  P_j = C * e^{Δ_j}  where  Δ_j = ft_{a_j} − ft_{a_{j+1}};
        Generating a random number  r;
        If r < P_j
            Change the  position of job a_{j+1} and a_j in sequence Π;
            Change the  position of ft_{a_{j+1}} and ft_{a_j} in sequence FT;
        end if
    end for
    Process jobs according to the new job sequence Π′ at stage s
end for
```

Fig. 2 The Pseudo code for JSAM

**Population Initialization.** Taking the diversity of the population into account is very important in population initialization. Therefore, the population are generated using a random strategy and require the newly generated individual to be different from any individual in the current population.

**Neighborhood Structure.** In the MBO algorithm, the neighborhood of all followers is composed of two parts, that is, the neighborhood solutions generated by itself and the unused neighborhood solutions of the previous individual. The neighborhood structure used in this paper is based on the optimal insertion operation(OIO for short) and the optimal switching operation(OSO for short). For OIO, we remove job at position $p_1$ and reinsert it into position $p_2$; For OSO, we exchange two jobs at position $p_1$ and position $p_2$.

**The Process of MBO Algorithm.** Define symbols as follows: EG is the evolution generation of the population; Psize is the size of population; *k* is the number of neighborhood solutions of the leader; *x* is the number of neighborhood solutions passed to the next bird; G represents tour rounds; 错误!未找到引用源。 represents neighborhood solution set passed to the next bird.

Step 1: initial all parameters, initial population;
Step 2: perform OIO and OSO on the leader to generate *k* neighborhood solutions, select the best neighborhood solution to replace the leader and keep the unused *x* neighborhood solutions to $PN_L(PN_R)$ .
Step 3: perform OIO and OSO on left queue followers to generate *k*-*x* neighborhood solutions, select the best neighborhood solution among *k*-*x* and $PN_L$ to replace the corresponding follower. Clear $PN_L$ and keep the unused *x* neighborhood solutions in *k*-*x* to $PN_L$ ;
Step 4: a similar operation like step 3 is conducted to followers in right queue;
Step 5: if step2 to steps 4 have conducted G times, replace the leader with the best follower in left or right queue;
Step 6: conduct step 2 to step 5 until the population have developed EG times.

## Computational results

The MBO algorithm was coded by C++ on an Intel Core i5 3210M PC with 4GB of memory. According to Duman, The parameters of MBO were set as follows: EG=200, Psize=25,G=10,*k*=3,*x*=1.

Using JMBO to solve 10 large scale benchmark problems. Result is shown in Table 1. We can see that JMBO gets better solutions than HVNS, HDBAC, PSO, AIS [9], thus we can come to the conclusion than JMBO is effective to solve makespan minimisation problem in HFSP.

Table 1 Comparison results for ten large scale problems

| | makespan | | | | | %Deviation | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | JMBO | HVNS | HDBAC | PSO | AIS | IMBO | HVNS | HDBAC | PSO | AIS |
| j30c5e1 | 463 | 464 | 467 | 471 | 479 | 0.00 | 0.22 | 0.86 | 1.73 | 3.46 |
| j30c5e2 | 616 | 616 | 616 | 616 | 619 | 0.00 | 0.00 | 0.00 | 0.00 | 0.49 |
| j30c5e3 | 594 | 595 | 595 | 602 | 614 | 0.00 | 0.17 | 0.17 | 1.35 | 3.37 |
| j30c5e4 | 565 | 566 | 568 | 575 | 582 | 0.00 | 0.18 | 0.53 | 1.77 | 3.01 |
| j30c5e5 | 600 | 601 | 603 | 605 | 610 | 0.00 | 0.17 | 0.50 | 0.83 | 1.67 |
| j30c5e6 | 602 | 603 | 605 | 605 | 620 | 0.00 | 0.17 | 0.50 | 0.50 | 2.99 |
| j30c5e7 | 626 | 626 | 626 | 629 | 635 | 0.00 | 0.00 | 0.00 | 0.48 | 1.44 |
| j30c5e8 | 674 | 674 | 675 | 678 | 686 | 0.00 | 0.00 | 0.15 | 0.59 | 1.78 |
| j30c5e9 | 642 | 643 | 645 | 651 | 662 | 0.00 | 0.16 | 0.47 | 1.40 | 3.12 |
| j30c5e10 | 573 | 577 | 580 | 594 | 604 | 0.00 | 0.70 | 1.22 | 3.66 | 5.41 |

**Acknowledgements**

The authors would like to thank Jasques Carlier and Emmanuel Neron for their benchmark problems. This research is supported by Cooperation and Exchange of the National Natural Science Foundation of China(no.51561125002), the National Natural Science Foundation of China (nos. 51275190, 51575211).

**References**

[1] A solution to a special class of flow shop scheduling problems. Symposium on the Theory of Scheduling and its Applications (North Carolina State Univ., Raleigh, N. C., 1972), pp. 83–91. Lecture Notes in Econom. and Math. Systems, Vol. 86, Springer, Berlin, 1973.

[2] Ruizab R. The hybrid flow shop scheduling problem[J]. European Journal of Operational Research, 2010, 205(1):1-18.

[3] Carlier, Neron J, Emmanuel. An Exact Method for Solving the Multi-Processor Flow-Shop[J]. RAIRO - Operations Research, 2000, 34(1):1-25.

[4] Panwalkar S S, Iskander W. A Survey of Scheduling Rules[J]. Operations Research, 1977, 25(1):45-61.

[5] Xiao W, Hao P, Zhang S, et al. Hybrid flow shop scheduling using genetic algorithms[C]// Intelligent Control and Automation, 2000. Proceedings of the, World Congress on. IEEE, 2000:537-541 vol.1.

[6] Cui Z, Gu X. An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems[J]. Neurocomputing, 2015, 148(148):248-259.

[7] Alaykýran K, Engin O, Döyen A. Using ant colony optimization to solve hybrid flow shop scheduling problems[J]. International Journal of Advanced Manufacturing Technology, 2007, 35(5-6):541-550.

[8] Duman E, Uysal M, Alkaya A F. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem[J]. Information Sciences, 2012, 217(24):65-77.

[9] Li J Q, Pan Q K, Wang F T. A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem[J]. Applied Soft Computing Journal, 2014, 24(24):63-77.