ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.097

Three-Stage-encoding Barnacle Mating Algorithm for Dynamic Manpower Scheduling in Flow Shop

Wushuang Wang [1] [+], Shuyun Luo [1]

[1] Zhejiang Sci-Tech University, Hangzhou, China

Abstract. In actual industrial manufacturing scenarios, the dynamic behavior of workers is widespread. When the production starts for a while, the worker's request for leave and back will break the original production rhythm. This paper concentrates on how to reschedule workers in the flow shop under the consideration of the workers' dynamic behavior to minimize the total time of workers adjustment and production. Inspired by the Self-encoding Barnacle Mating Optimizer (SBMO) to our problem, we design a novel solution mode, offspring generation method, and neighborhood search scheme to propose the Three-stage-encoding Barnacle Mating Algorithm (TBMA). Extensive experiments are conducted to test the performance of TBMA, and the results demonstrate that TBMA has competitive performance.

Keywords: flow shop, three-stage-encoding barnacle mating algorithm, manpower scheduling, dynamic behavior.

1. Introduction

In reality, the industrial background of the flow shop is extremely strong, which is in line with the production characteristics of most shops, such as clothing shops, dyeing & finishing shops, and textile shops. In the flow shop, products need to be processed in orderly stages, like woven cloth demands winding, warping, sizing, reeding, and weaving. In recent years, the manpower of manufacturing continues to decline, leading to a continuous increase in manpower costs [1]. Therefore, it is necessary to solve manpower shortages and improve the manpower utilization of the flow shop.

Some previous studies have addressed the issue of manpower scheduling in the flow shop. Costa et al. [2] proposed a novel discrete backtracking search algorithm powered by tabu search to assign workers to operate the machines, but they did not consider the workers' proficiency for each stage. Workers' proficiency is positively correlated with the efficiency of workers operating machines to produce jobs. In real shops, workers' efficiency is sundry, and workers' assignment considering their proficiency is critical to the production process. There are some researches considering workers' proficiency. Gong et al. [3] presented a hybrid evolutionary algorithm to address this problem. Liu et al. [4] formulated this problem into a mixed-integer linear programming model and adopted the ε-constraint method to solve this model. Matheus et al. [5] proposed a new hybrid algorithm based on tabu search and iterated greedy. To tackle this problem, Ebrahim et al. [6] established a mixed-integer linear programming model and used CPLEX to obtain the optimal solution of the small-scale problems. Moreover, to address the problem on a large scale, they presented a hybrid algorithm based on particle swarm optimization and simulated annealing. Han et al.[7] proposed seven multi-objective evolutionary algorithms with heuristic decoding to address this problem. Luo et al. [8] proposed a novel Self-encoding Barnacle Mating Optimizer (SBMO) to solve the flow shop manpower scheduling problem. However, none of the above studies considered the scenario of workers' dynamic behavior. In real life, workers may need to ask for leave for some urgent reasons. Moreover, it is natural some workers will come back. These situations will inevitably disrupt the original production rhythm in the flow shop. Therefore, it is crucial for production efficiency to restore the original production rhythm as much as possible after workers' dynamic behavior.

[+] Corresponding author. Tel.: + 8615858194081;
E-mail address: wangwushuang_zstu@163.com.

In this paper, we focus on how to reschedule manpower in the flow shop under the consideration of the workers' dynamic behavior, which is a common phenomenon in a real flow shop. Our target is to fine-tune some workers based on the original worker schedule to minimize the completion time of products and the time cost of workers' adjustment. Our research is based on [8], which is not considered the scenario with workers' dynamic behavior. Inspired by the SBMO with outstanding performance, we adopted it to fine-tune workers. However, SBMO can not be applied directly to our problem. Therefore, we design a solution mode, an offspring generation method, and a neighborhood search scheme based on SBMO and present the Three-stage-encoding Barnacle Mating Algorithm (TBMA) to fine-tune workers.

The main contributions of this paper are summarized as follows:

- For manpower scheduling in the flow shop, we consider the scenario of workers' dynamic behavior and establish a pure integer nonlinear programming model to solve this problem.
- In order to apply the SBMO to our problem, we design a novel solution mode, offspring generation method, and neighborhood search scheme. Based on the above newly designed modules, we propose the TBMA.
- Compared with novel and classical algorithms, the computational results show that the TBMA could achieve promising results under the scenarios of workers' dynamic behavior.

The remainder of the paper is structured as follows. Section 2 concisely describes the problem and establishes the mathematical model. Next, section 3 and 4 declares the basic idea of SBMO and TBMA. Furthermore, section 5 analyzes the performance results compared with classical and novel algorithms. At last, a conclusion is drawn in section 6.

2. Problem Description and Mathematical Model

In this section, the Dynamic Manpower Scheduling Problem (DMSP) is described first. Then, the notations and the assumptions are listed. Moreover, the manpower scheduling model considering workers' dynamic behavior and proficiency is described.

2.1. Problem Description

Our question is based on the research in [8]. The same *D* products are processed in *N* stages, and the stage order is fixed from 1 to *N*. Each stage needs to arrange at least one worker, and there are a total of *R* workers. The workers' proficiency is diverse, which is quantified to a decimal value between (0,1]. We define unit time as the spent time for a worker with a proficiency value of 1 to process a product, which of each stage is known at first.

We suppose that workers have been assigned to reasonable stages, and production has started for a while. It is a normal phenomenon that some workers ask for leave or/and come back. If the method in [8] is used to rearrange the workers in each stage, it will take extra time cost to the production process. For example, workers take a long time to adapt to new stages. Therefore, our goal is to fine-tune workers' stages to minimize the completion time of *D* products and the time cost of adjusting workers.

The assumptions of this scheduling problem are as follows:

- Production is started at zero time.
- Before workers' leave, workers need to complete the products they have already processed.
- Time cost is required for fine-tuning workers.

In order to express the problem, the following notations are defined:

i: Index of workers, where $i = 1, 2, \ldots, R$;

j: Index of stages, where $j = 1, 2, \ldots, N$;

t_j: The unit time of the j^{th} stage;

k_{ij}: The proficiency of the i^{th} worker to the j^{th} stage;

r_j: The total number of workers arranged to the j^{th} stage;

r_l: The total number of workers who ask for leave;

r_b: The total number of workers who come back;

r_m (α): The maximum number of fine-tunable workers;

r_a: The number of fine-tuning workers;

t_c: The time cost of adjusting one worker;

T_c: The total time cost of adjusting workers;

T_p: The completion time of D products;

T: The production completion time, which is the sum of T_p and T_c.

D: The number of products;

R: The original number of workers before workers' leaving and return;

X: The workers' working status matrix;

K: The workers' proficiency matrix.

$$x_{ij} = \begin{cases} 1, \text{ if the } i^{th} \text{ worker is assigned to the } j^{th} \text{ stage} \\ 0, \text{ otherwise} \end{cases}$$

2.2. Manpower scheduling model

With the notations above, we present the following manpower scheduling model. From assumption (2) in section II-A, it can be seen that the completion time calculation formula in [8] can still be used. The worker's work status matrix and the worker's proficiency matrix are as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1N} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{iN} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{R1} & x_{R2} & \cdots & x_{Rj} & \cdots & x_{RN} \end{bmatrix} \tag{1}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1N} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{iN} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{R1} & x_{R2} & \cdots & x_{Rj} & \cdots & x_{RN} \end{bmatrix} \tag{2}$$

Take the value on the diagonal of $X^T K$ as the sum of the proficiency of all workers in the stage, that is: $\sum_{i=1}^{R} x_{ij} k_{ij}$. Then the processing time required for one product in the j^{th} stage is $t_j / \sum_{i=1}^{R} x_{ij} k_{ij}$. We regard the processing time of each stage as its processing capacity, i.e., they have a negative relationship. It is noted that after adjusting the workers, the matrix X needs to be updated. The completion time is:

$$T_p = \sum_{j=1}^{N-1} Y_j + (D - N + 1) Y_N + \sum_{j=2}^{N} \bar{Y}_j \tag{3}$$

where

$$Y_j = \max(t_1 / \sum_{i=1}^{R} x_{i1} k_{i1}, t_2 / \sum_{i=1}^{R} x_{i2} k_{i2}, \ldots, t_j / \sum_{i=1}^{R} x_{ij} k_{ij}) \tag{4}$$

$$\bar{Y}_j = \max(t_j / \sum_{i=1}^{R} x_{ij} k_{ij}, t_{j+1} / \sum_{i=1}^{R} x_{i(j+1)} k_{i(j+1)}, \ldots, t_N / \sum_{i=1}^{R} x_{iN} k_{iN}) \tag{5}$$

According to assumption (3) in section II-A, fine-tuning a worker requires time cost t_c. Therefore, the time cost of fine-tuning r_a workers is as follows:

$$T_c = r_a \cdot t_c \tag{6}$$

Therefore, in the scenario where workers have dynamic behavior, the production completion time is $T = T_c + T_p$.

To sum up, the manpower rescheduling model is as follows:

$$\min T = T_c + T_p \tag{7}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \tag{8}$$

$$r_j \geq 1 \; with \; r_j \in \mathbf{Z} \tag{9}$$

$$\sum_{j=1}^{N} r_j = R - r_l + r_b \tag{10}$$

$$r_a \le r_m \tag{11}$$

$$r_l + r_b \le R - N \tag{12}$$

As shown in the Eqs. (7), our optimization goal is to minimize the production completion time *T*. Constraint (8) requires that each worker can only be allocated to one stage; constraint (9) ensures that each stage should arrange at least one worker and the number of workers assigned in each stage must be an integer, where $r_j = \sum_{i=1}^{R} x_{ij}$; It is noteworthy that t_j>0 and *D*>>*N*. The above constraints are the same as in [8]. Constraint (10) guarantees the total number of workers to be R-r_l+r_b; constraint (11) restricts the maximum number of workers that can be fine-tuned, where $r_m = round(\alpha R)$. Constraint (12) ensures that after workers' dynamic behavior, the number of remaining workers is not smaller than the number of stages to make sure each stage has at least one worker.

3. The Basic Idea of SBMO

SBMO is a discrete algorithm and was first developed to address the flow shop manpower scheduling problem. The SBMO can be divided into the following modules.

3.1. Solution Mode

In SBMO, it is supposed that the candidate solutions are the barnacles where the row vector of the population is converting from the matrix *X*. The number of row vectors is recorded as the corresponding positions of 1 for each line of matrix *X*. The i^{th} number *j* in the row vector signifies that the i^{th} worker is allocated to the j^{th} stage. The length of the row vector is the total number of workers *R*. Fig. 1 is an example of 4 workers and 4 stages. The first number 2 in the row vector P_1 means the first worker is assigned to the 2^{rd} stage.

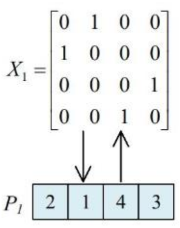
$X_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
P_l
2 1 4 3

Fig. 1: One case of solution mode of SBMO.

3.2. Offspring Generation

The SBMO generates offspring through crossover and mutation operators. The behavior of crossover and mutation depends on the penis length (*pl*) of the barnacles and the distance between them. Specifically, randomly select two barnacles, father and mother. If the distance between them is smaller than or equal to *pl*, they cross; otherwise, the mutation will occur.

1) Crossover operator: The crossover is the primary behavior to produce offspring, which is designed to copy the number of mother and father row vectors, both with 50% probability.

2) Mutation operators: There are three mutation operators used in the SBMO: inversion mutation, insertion mutation, and double-segment swap mutation [8].

3.3. Population Update Rules

After the offspring generation is finished, the new offspring are merged with their parents, and all individuals in the current population are sorted from superior to terrible according to the objective function of the optimization problem. In order to keep population size stable, the individuals corresponding poor solutions will be filtered. The population evolution continues until its stopping criterion is met.

3.4. Neighborhood Search

Along with the generation iterations, the updated population tends to be the optimal solution. Hence, the filtered solutions tend to have high similarities. The crossover operation fails to generate new offspring, making the solution locked into a local optimum. To solve the above problem, a neighborhood search algorithm is proposed in SBMO. Expressly, when the feasible solutions tend to be the same, set *pl* as 0 to block the crossover operator. In order to minutely adjust the solutions in population, replace the above three mutation operators with balance mutation, reciprocal exchange mutation, and triplet mutation.

4. Three-stage-encoding Barnacle Mating Algorithm

Inspired by the SBMO with outstanding performance, we applied it to fine-tune workers.However, the solution mode, offspring generation method, and neighborhood search scheme of the SBMO are not suitable for DMSP. Therefore, we design up-to-date modules, and the proposed TBMA is described in algorithm flow chart 2.

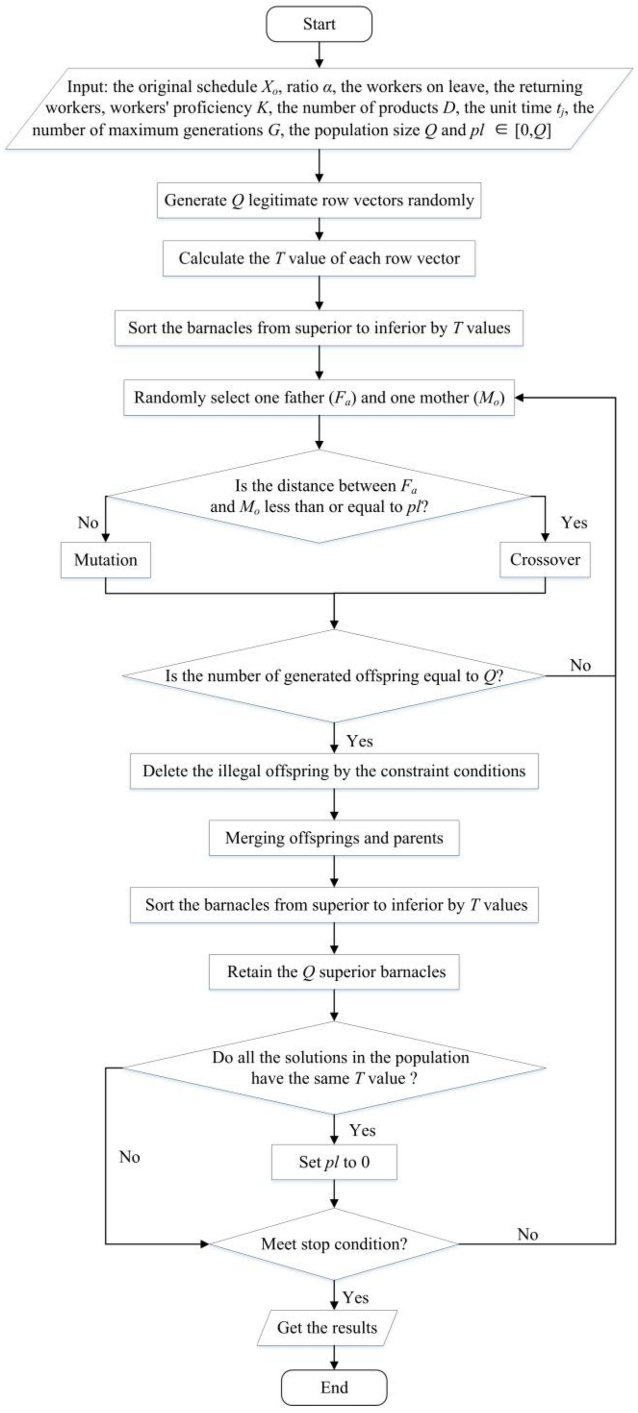
Start
Input: the original schedule X_o, ratio α, the workers on leave, the returning workers, workers' proficiency K, the number of products D, the unit time t_j, the number of maximum generations G, the population size Q and $pl \in [0,Q]$
Generate Q legitimate row vectors randomly
Calculate the T value of each row vector
Sort the barnacles from superior to inferior by T values
Randomly select one father (F_a) and one mother (M_o)
Is the distance between F_a and M_o less than or equal to pl?
No
Yes
Mutation
Crossover
Is the number of generated offspring equal to Q?
No
Yes
Delete the illegal offspring by the constraint conditions
Merging offsprings and parents
Sort the barnacles from superior to inferior by T values
Retain the Q superior barnacles
Do all the solutions in the population have the same T value ?
Yes
No
Set pl to 0
Meet stop condition?
No
Yes
Get the results
End

Fig. 2: Flow chart of the TBMA.

4.1. Solution Mode

The solution mode of DMSP is a row vector consisting of three parts, π_1, π_2, and π_3. The length of the row vector is $2r_m+r_b$. Fig. 3 is an example of a solution. The length of π_1 and π_2 is determined by r_m, that is,

r_m workers can be adjusted at most. The length of π_3 is decided by r_b. π_1 indicates which workers are adjusted, and the above numbers correspond to the rows of the matrix X. For example, the first number 10 of π_1 in Fig. 3 indicates that the *10th* worker will be adjusted, and the work status of this worker corresponds to the *10th* row of the matrix X. Therefore, the π_1 in Fig. 3 represents that the *10th*, *5th*, *3rd*, *8th* and *4th* workers will be adjusted. π_1 and π_2 correspond one-to-one, indicating which stage the worker is adjusted to. For instance, the first number 1 of π_2 in Fig. 3 indicates that the *10th* worker will be adjusted to the *1st* stage. Remarkably, there will be the same number in π_1. For example, in Fig. 3, there are two numbers 3 in π_1, corresponding to 7 and 5 in π_2. In the case of multiple identical numbers in π_1, it is considered that the worker has only been adjusted once, and he or she is finally adjusted to the stage matches along with the last corresponding number in π_2. The number of π_3 indicates the stages to which the returning workers are assigned. For instance, in Fig. 3, the returning workers are assigned to the *8th* and *5th* stage, respectively.

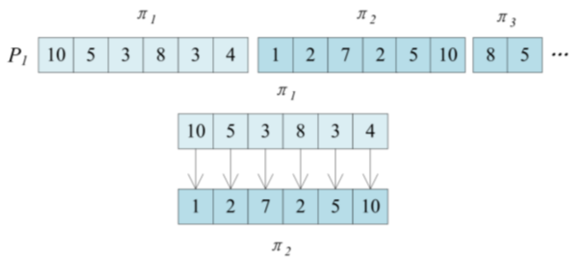

Fig. 3: One case of solution mode of TBMA.

4.2. Offspring generation

In this part, we retain the crossover operator and change the mutation operators in SBMO.

1) Crossover operator: As is shown by the Fig. 4, the row vector P_1 and P_2 cross to produce offspring C_1. The up-to-down arrow means the variable of C_1 copies from the one of P_1. Conversely, the down-to-up arrow represents the variable of C_1 copies from P_2. Each crossover only generates one offspring.

2) Mutation operators: According to the three-segment coding method, there are two mutation methods, and one of them is randomly selected for each mutation.

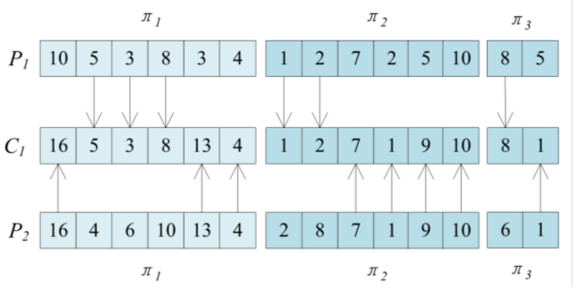

Fig. 4: The illustration of crossover operator.

a) The mutation operators for π_1 (Mu_1): Two positions are selected randomly within the row vector P_1. The number in one position is increased by one, and the other is decreased by one, illustrated as Fig. 5.

b) The mutation operators for π_2 and π_3 (Mu_2): We merge π_2 and π_3 into one segment to mutate. In order to maintain population diversity, we apply three mutation methods, including inversion mutation, insertion mutation, and reciprocal exchange mutation. At the inversion mutation, randomly select a segment of π_2 and π_3 and inverse it, as shown in Fig. 6. At the insertion mutation, one variable of π_2 and π_3 is randomly selected and transferred to another position, seen from Fig. 7. At the reciprocal exchange mutation, two variables of π_2 and π_3 are randomly selected and swapped, illustrated as Fig. 8. When the barnacle goes through mutation, it will choose one of the above randomly.

4.3. Neighborhood Search

With iterations, the solutions in the population have a high degree of resemblance. Almost no new offspring will probably be produced through the crossover in this event. Therefore, the crossover operator is abandoned by setting pl to 0, and the generation of offspring is mainly through mutation. It is remarkable that we do not change the mutation operators after setting *pl* to 0 because using the previous mutation operators is enough for the population to maintain diversity and the balance mutation operator in [8] is not suitable for DMSP.

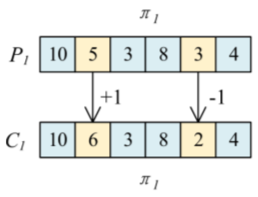

Fig. 5: The illustration of mutation operator of π_1.

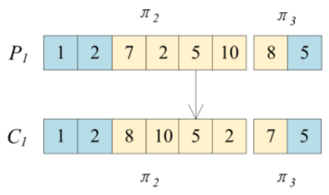

Fig. 6: The illustration of inversion mutation of π_2 and π_3.

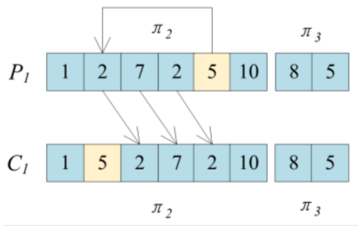

Fig. 7: The illustration of insertion mutation of π_2 and π_3.

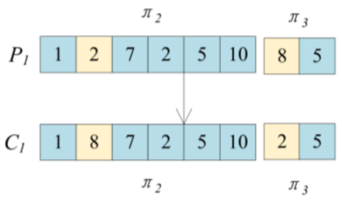

Fig. 8: The illustration of reciprocal exchange mutation of π_2 and π_3.

5. Experimental Results

Extensive experiments are designed to certificate the effectiveness of TBMA for DMSP. All experiments are executed using Pycharm and run on an 8.0 G RAM 3.20 GHz CPU PC.

5.1. Algorithms and Parameter Setting

To analyze the performance of the TBMA, simulation results are compared with novel and classical algorithms, Improved Adaptive Genetic Algorithm (IAGA) [9] and Multi-objective Whale Swarm Algorithm (MOWSA) [10]. For accuracy, each simulation is repeated 20 times individually, and the average results are recorded for comparison.

The IAGA is a classical algorithm to solve the combinational optimization problem, such as job shop scheduling problem, flow shop scheduling problem, and open shop scheduling problem. The MOWSA was first proposed to address the multi-objective distributed permutation flow shop scheduling problem, which can be applied simply for TBMA by revising the encoding and sorting method of solutions in population. The MOWSA was first used to solve multi-objective problems, so it adopted a non-dominated sorting method to sort the solutions in the population. However, DMSP is a single-objective optimization problem. Therefore, we changed the non-dominated sorting method used in MOWSA to a simple sorting method from good to flawed based on the value of objective function. Since the neighbor search algorithm adopted in the MOWSA is absorbed in arranging jobs to factories, the scenario is inappropriate for DMSP. So, we execute the MOWSA only without its neighbor search algorithm. To make it fair, all comparison algorithms use the same crossover and mutation methods.

Table 1: 9 comprehensive cases

Case	Number of stages	Number of workers before workers' dynamic behavior	Number of workers ask for leave	Number of returning workers
Case 1	10	20	2	2
Case 2	10	20	3	2
Case 3	10	30	3	0
Case 4	10	30	4	0
Case 5	10	40	4	3
Case 6	10	40	5	3
Case 7	10	50	5	4
Case 8	10	50	6	4
Case 9	10	32	5	0

The TBMA needs to define three parameters: the population size Q, the number of maximum generations G, and pl. Extensive preliminary experiments show that the three parameters have significant influence, and the algorithm generally performs well with Q=200, G=300, and pl=170. Hence, we fix these values in the following experiments. All the other parameters involved in compared algorithms are referred to the settings in their literature.

5.2. Test Data and Performance Criteria

To the best of our knowledge, no benchmark instances are available for the considered problem. Therefore, we generate 9 synthetic instances. A summary is given in Table 1. In all cases, we set t_c=30, α=0.2, D=100.

In this paper, two performance criteria are chosen. They are the value of T and execution time. The value of T is calculated by Eqs. (7) according to the changed worker schedule X and the number of adjusted workers. The execution time of the algorithm is the total time cost to find the optimal solutions.

5.3. Evaluation of TBMA

We compare the performance of the TBMA with IAGA and MOWSA. The comparison is given thoroughly from three aspects: T, execution time and convergence performance.

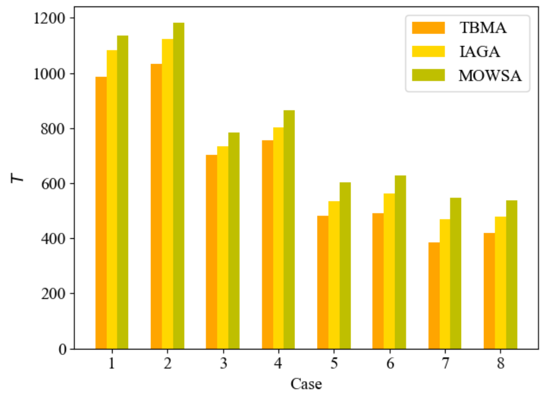

Fig. 9: The comparison results of T values.

1) Evaluation T value: From Fig. 9, it is observed that the performance of TBMA is the best of all top 8 cases, no matter the scale of the cases.

2) Evaluation of computation efficiency: According to literature [8], the time complexity of SBMO, IAGA and MOWSA is $O(GQR)$, $O(GQR)$ and $O(GQ^2R)$, where R is the total number of workers before workers' dynamic behavior. In literature [8], the code length of the solution is determined by R, but in this paper, the code length of the solution is $2r_m+r_b$. Typically, $2r_m$ is greater than r_b. Therefore, the time complexity of TBMA, IAGA and MOWSA are: $O(GQr_m)$, $O(GQr_m)$ and $O(GQ^2r_m)$.

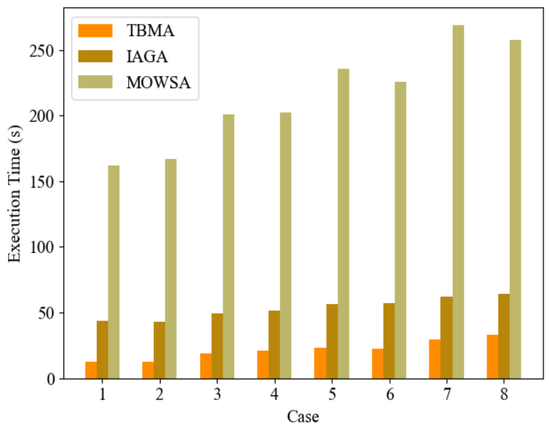

Fig. 10: The comparison results of execution time.

As shown in Fig.10, TBMA has the supreme computation efficiency. Although IAGA has the same time complexity as TBMA, it has additional tournament selection and elite retention operations. Moreover, the execution time increases with the enlargement of rm. Therefore, the simulation results fit the theoretical analysis.

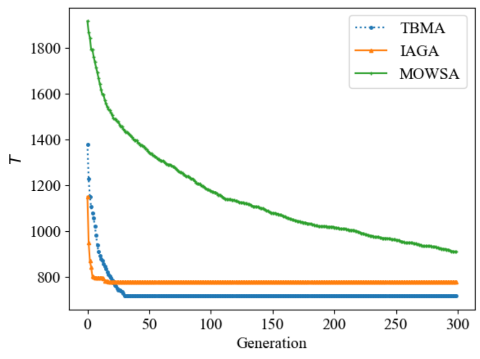

Fig. 11: The mean value of T in the population.

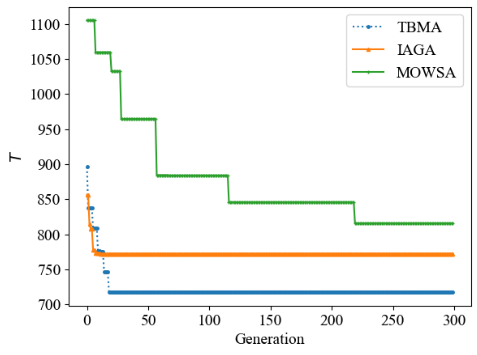

Fig. 11: The optimal value of T in the population.

3) Evaluation of convergence: We use case 9 to compare the convergence performance of the three algorithms. As shown in Fig.11 and Fig.12 that all three algorithms can converge. IAGA converges very quickly at first, but it is liable to fall into a local optimum. TBMA has a slow convergence speed and unsatisfactory results. Overall, TBMA can obtain higher-quality solutions.

6. Conclusion

Concentrating on the scenario of workers' dynamic behavior in the flow shop, we first define a new problem, DMSP. To address the DMSP, we modify the solution mode, offspring generation method, and neighborhood search scheme of SBMO and propose the TBMA. Finally, the results of experiments denote that TBMA is superior to IAGA and MOWSA when solving the DMSP, verifying the effectiveness of proposed optimization strategy to improve the solution quality.

7. Acknowledgments

This research was supported in part by the Fundamental Research Funds of Zhejiang Sci-Tech University (No. 2021Q026).

8. References

[1] S. Perini, M. Oliveria, J. Costa, D. Kiritsis, P. H. K. Hansen, and L. Rentzos et al, Attracting Young Talents to Manufacturing: A Holistic Approach. In IFIP International Conference on Advances in Production Management Systems, Springer, 2014, pp. 626-633.

[2] A. Costa, V. Fernandez-Viagas, and J. M. Framinan, Solving the Hybrid Flow Shop Scheduling Problem with Limited Human Resource Constraint. Computer & Industrial Engineering, 146: 106545, 2020.

[3] G. Gong, R. Chiong, Q. Deng, W. Han, L. Zhang, and W. Lin et al, Energy-efficient Flexible Flow Shop Scheduling with Worker Flexibility. Expert Systems with Applications, 141: 112902, 2020.

[4] M. Liu, and X. Yang, Bi-objective Optimization for Scheduling and Multi-skilled Worker Assignments in the Hybrid Flow Shop. IFAC-PapersOnLine, 52(13):2128-2133, 2019.

[5] M. De Freitas Araujo, J. E. C. Arroyo, and L. B. Fialho, Tabu Search and Iterated Greedy for a Flow Shop Scheduling Problem with Worker Assignment. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, 2020, pp. 1661-1668.

[6] E. A. Gangraj, F. Bozorgnezhad, and M. M. Paydar, Developing Metaheuristic Approaches to Solve Flow Shop Scheduling Problem with Worker Assignment. Journal of Modeling in Engineering, 16(54): 283-293, 2018.

[7] Han W, Deng Q, Gong G, Like Z, and Qiang L. Multi-objective Evolutionary Algorithms with Heuristic Decoding for Hybrid Flow Shop Scheduling Problem with Worker Constraint. Expert Systems with Applications, 168: 114282, 2021.

[8] S. Luo, W. Wang, M. Fang, and W. Xu, Self-encoding Barnacle Mating Optimizer Algorithm for Manpower Scheduling in Flow Shop. unpublished.

[9] Z. Ren, and Y. San, Improved Adaptive Genetic Algorithm and its Application Research in Parameter Identification. Journal of System Simulation, 1, 2006.

[10] G. Wang, L. Gao, X. Li, P. Li, and M. F. Tasgetiren, Energy-efficient Distributed Permutation Flow Shop Scheduling Problem Using a Multi-objective Whale Swarm Algorithm. Swarm and Evolutionary Computation, 57: 100716, 2020.