Design Engineering

ISSN: 0011-9342
Issue: 1 | Pages: 194 - 206

Lifecycle Multi-colony Bee Algorithm for solving Flexible Job Scheduling Problems

Liling Sun, Yuhan Wu, Xiaodan Liang, Maowei He, Hanning Chen*
School of Computer Science and Technology, Tiangong University, Tianjn, 300387, China
*Corresponding Author: Hanning Chen

Abstract:

We present a Lifecycle Multi-colony Bee Algorithm (LMcBA) for Flexible Job-shop Scheduling Problem (FJSP). This problem has a strong practical background in process industry. Existing approaches considering constant processing time may not be appropriate to address FJSPs. To make evolutionary algorithm applicable in FJSP, a novel Lifecycle Multi-colony Bee Algorithm is chosen as the core optimization algorithm of the FJSP. Experimental results show LMcBA has an outstanding performance in solving FJSP.

Keywords: *Artificial Bee Colony Algorithm, Lifecycle, Swarm Intelligence, Evolutionary Computing, Flexible Job-shop Scheduling.*

I. INTRODUCTION

With the improvement of welding technology, the production of trucks has been fully automated [1]. With the continuous migration of technology, welding process becomes more and more complex. Effective job scheduling has become an important way of improving welding efficiency and reducing total welding time. It also has become the key to improve the interests of enterprises [2]. Welding job scheduling can be classified into job-shop scheduling problem (JSP). More precisely speaking, it is a flexible JSP (FJSP) [3]. For JSP, a series of operations with fixed order is assigned to a group of machines. For FJSP, allocating time and resources to operations should satisfy the priority of each job [4]. Compared with JSP, FJSP is more practical in welding manufacturing [5].

Practically, welding machines can perform a variety of operations. Moreover, different welding machines have different processing time for the same process [6]. In addition, because of advanced process control equipped in the welding machine, the scheduling becomes more and more complicated [7].

In order to solve FJSP, an evolutionary algorithm-based (EA-based) optimization method is adopted. For high search accuracy and efficiency, we present a novel Lifecycle Multi-colony Bee Algorithm (LMcBA). For improving the global search capacity, the proposed algorithm adopts the multi-swarm method and partitions the whole single colony into several sub-colonies with k-means clustering method. To avoid the ineffective search and keep information exchange, the whole colony may re-partition after specified iterations. To enhance the local search capacity, a lifecycle model is set up. In this model, the bee with good performance may reproduce and the bee with bad performance may die. With comparisons of performance of solving FJSP optimized by LMcBA, PSO, GA and ABC algorithms, LMcBA has an outstanding performance in searching scheduling results.

II. FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

2.1 Description

Generally, FJSP can be described as follows: N workpieces with different process routes are processed on M machines, involving parallel machines and multi-functional machines. A parallel machine is defined as a machine that can process the same process. A multifunctional machine is defined as one that can process two or more processes. A flexible process line can be selected in more than one machine to process. Two or more workpieces cannot be processed simultaneously on the same machine. Furthermore, after being processed on a certain machine, the workpiece cannot be processed on other machine.

Every workpiece I is composed of n_i processes. There are technological constraints between processes, and a series of processes of workpiece can be executed on different machines among M machines. $M_{ij} \in \{1, 2, \ldots, M\}$ represents the machine set available for the j_{th} process of the i_{th} workpiece, O_{ijk} represents the j_{th} process of the i_{th} workpiece which can be processed by the M_k machine, p_{ijk} represents the time required for the j_{th} process of the i_{th} workpiece to be processed on the k_{th} machine.

Flexible job scheduling consists of two subproblems:

(1) Machine allocation problem, selecting the appropriate machine for each process;

(2) Arrange N workpiece processing tasks on the selected set of machines, simultaneously optimizing multiple given performance indicators, and meeting certain constraints.

Flexible job scheduling problem (FJSP) can be divided into 2 classes:

(1) Total FJSP: For O_{ij}, the i_{th} process of any J_i workpiece could be processed on every machine, $M_{ij} = M$.

(2) Partial FJSP: For O_{ij}, the i_{th} process of certain J_i workpiece is only processed on specific machines, $M_{ij} \subset M$.

2.2 Assumptions and Nomenclature

Assumptions

(1) Every machines is spare at time $t = 0$;

(2) All workpieces can be processed at time t=0;

(3) Process plan of all the workpieces is predetermined and fixed;

(4) Processing time of the process on the machine available for selection is determined;

(5) For the processing time, both of the workpiece handling time and preparation time are counted;

(6) At the same time, the same workpiece can not have more than one processing operation,

and every workpiece could only be processed on certain machine at a fixed time;

(7) Processing is non-preemptive, which means that a machine is available for other operations only after completing the processed operation;

(8) All workpiece tasks shall be assigned to the machine for processing;

(9) Once the process is processed, it cannot be interrupted;

(10) Damage of the machine is not taken account of;

(11) Machines are independent;

(12) Workpieces are independent;

Nomenclature

Notations Related to Production Order

f_1	completion time of the workpiece
f_2	load time of the key machine
W_1	weight of f_1
W_2	weight of f_2
S_{ijk}	starting time of the jth process of the ith workpiece on the kth machine
E_{ijk}	completion time of the jth process of the ith workpiece on the kth machine
P_{ijk}	time required of the jth process of the ith workpiece on the kth machine
X_{ijk}	decision variable, indicating whether the jth process of the ith workpiece is processed on the kth machine; $X_{ijk} = 1$ means selected, and $X_{ijk} = 0$ means unselected.
R_{ijegk}	decision variable, indicating whether the jth process of the ith workpiece and the gth process of eth workpiece are processed on the same kth machine.

2.3 Modelling

The important characteristic of FJSP, which is different from classical JSP, is equipment allocation, which is related to the rational utilization of enterprise resources and particularly concerned with top corporate leaders. On the one hand, in order to avoid the blockage of bottleneck links, it is required to allocate resources in a balanced way to smooth the operation of production. On the other hand, it requires that the load of key equipment should be minimized, so that enterprises can produce more products in a limited time. Meanwhile, the shortest completion time is also the most important issue among all time indicators.

Therefore, a meaningful comprehensive goal is given to solve FJSP: 1) shortening the production cycle as much as possible is to improve the work efficiency of the shop; 2) reducing the machine load as far as possible is to improve the utilization of existing resources.

Accordingly, the optimization objectives include two aspects: completion time and equipment utilization rate. Completion time target is measured by the maximum completion time f_1; and machine load (equipment utilization) target is measured by the maximum load of the key machine f_2.

In order to solve FJSP, the sum of two target values with weights can be regarded as the target function.

According to the above analysis of flexible job workshop objective, the objective function is designed:

$$F = \min(w_1 f_1 + w_2 f_2) \quad (1)$$

where $w_1 + w_2 = 1$.

Constraints include:

(1) Time constraints:

$$p_{ijk} \geq 0 \text{ and } p_{i0k} = 0;$$
$$S_{ijk} \geq 0 \text{ and } S_{i0k} = 0;$$

(2) Resource constraints: the k_{th} machine, one task should be finished before next task can be started.

$$E_{egk} - E_{ijk} \geq p_{egk}$$

where $X_{egk} = X_{ijk} = 1$ and $R_{ijegk} = 1$.

(3) Process constraints: for certain workpiece, the adjacent processes in the processing sequence is followed:

$$E_{ijk} - E_{i(j\text{-}1)k} \geq p_{ijk}$$

where $1 \leq i \leq N$, $1 \leq j \leq n_i$, $1 \leq \text{k} \leq M$, and $X_{ijk} = \text{X}_{i(j\text{-}1)k} = 1$.

(4) Other constraints: The completion time of any process shall not be less than its processing time.

$$E_{ijk} \geq p_{ijk}$$

where $X_{ijk} = 1$.

Constraint (1) indicates that the release time of all workpieces is 0; Constraint (2) means that at any moment, the k_{th} machine cannot simultaneously process any two different workpieces or two different processes. Constraint (3) means that the j_{th} process of the i_{th} workpiece can only start after the completion of the $(j$ - $1)_{\text{th}}$ process; Constraint (4) represents the logical constraint of workpiece's process completion time and processing time, which means that the completion time of any process cannot be less than its processing time.

III. THE PROPOSED ALGORITHM

3.1 Artificial Bee Colony Algorithm [8]

First, food sources are generated.

$$x_{i,j} = x_j^{\min} + rand(0,1)(x_j^{\max} - x_j^{\min}) \tag{2}$$

where i = 1, 2, ..., SN, j = 1, 2 , ..., D. SN is the colony size. D is the number of dimensions optimized. $x_j^{\min}$ and $x_j^{\max}$ are the limits of the jth dimension.

In the employed bees' phase, bee learns to its two neighbors and update as followed.

$$v_{i,j} = x_{i,j} + \phi(x_{i,j} - x_{k,j}) \tag{3}$$

where k and i are different, j is a random dimension. ϕ is stochastically generated in [-1, 1]. After comparing the new and original food source, the prior one would be reserved.

In the onlooker bees' phase, they exchange information with employed bees according the probability as followed.

$$P_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j} \tag{4}$$

Then, each onlooker updates according to Eq. (3). The better food source is determined with a greedy selection.

The employed and onlooker bees' phases will repeat.

3.2 Lifecycle Multi-colony Bee Algorithm (LMcBA)

In order to explore the search landscape, the multi-swarm strategy is employed. In our proposed algorithm, the single colony with N_s bees is partitioned into several sub-colonies with the *k*-means clustering method [9] which works according to the nearest distance among the current solution to several cluster centres. The bee exchanges information with the neighbours in its sub-colony. After the specified number of iterations c_d, the whole colony is re-partitioned into several sub-colonies. To avoid to overmuch exploit the local but not global optimal, the number of sub-colonies $c \in \{c_1, c_2, \ldots, c_M\}$ should decrease with search process, where $c_1 > c_2 > \ldots > c_M$.

In order to enhance the local search capability, inspired by the natural phenomena of the biological world, the lifecycle theory [10] is brought into our proposed algorithm. Here, a lifecycle model is set up, which includes grow-up, reproduction, and death status. If one bee finds out a good food source, such bee would obtain enough energy and gradually grow up. If this good status can keep for long time, a new bee would be reproduced near the good food source. If a bee can't find out a suitable food source, it would spend its energy and would not obtain enough energy to subsist. If this bad status lasts for a long time, this bee would dead and remove from the colony. Here, the nutritive value is set up to evaluate the status of each bee in its sub-colony. When the performance of one bee is better than the average of performance of its sub-colony, its corresponding nutritive value increases. Inversely, its corresponding nutritive value decreases. The detailed operation is given as follows:

$$Nutri_i^t = \begin{cases} Nutri_i^{t-1} + \dfrac{\mathrm{sort}^-(f_i, Num^+)}{Num^+} & \text{if} \quad f(i) < f_{\mathrm{mean}} \\ Nutri_i^{t-1} - \dfrac{\mathrm{sort}^+(f_i, Num^-)}{Num^-} & \text{if} \quad f(i) > f_{\mathrm{mean}} \end{cases} \tag{5}$$

where Num^+ and Num^- are the number of bees better and worse than the average performance, $\mathrm{sort}^-(f_i, Num^+)$ means the position of the *i*th bee in descending order.

If the nutritive value of bee is larger than the upper limit, a new bee will be reproduced near it. If the nutritive value of bee is smaller than the lower limit, it will remove.

$$\begin{cases} \text{Reproduction} \quad Nutri_i^t \geq N_{\max} \\ \text{Death} \quad Nutri_i^t \leq N_{\min} \end{cases} \tag{6}$$

The pseudo-code of LMcBA is given in Figure 1.

```
    INITIALIZATION
1   Set $t = 0$, and define iteration cycle of $t_{max}$;
2   Generate initial bee colony $\{b_1^0, b_2^0, \ldots, b_N^0\}$, compute the fitness value $\{f_1, f_2, \ldots, f_N\}$
3   Set the number of cluster $C = \{c_1, c_2, \ldots, c_M\}$
4   Set the nutrition limits $N_{max}$ and $N_{min}$, and generate initial nutrition table $\{N_1, N_2, \ldots, N_N\}$;
5   Partition the colony into $c = c_i$ sub-colony with $k$-means clustering method, where $i = 1$
6   Set the specified number of iterations $c_d$ for re-clustering
    CYCLE
7   while $t < t_{max}$
8       For each sub-colony
9           Update bees as the artificial bee colony algorithm, and compute the fitness value
10          Compute the average fitness value $f_{mean}$ of sub-colony
11          For each bee
12              If it is better than the average performance, the nutritive value increases;
13              If it is worse than the average performance, the nutritive value decreases;
14              If the nutritive value is larger than the upper limit, it reproduces a new bee;
15              If the nutritive value is smaller than the lower limit, it dies;
16          End For
17      End For
18      If $(t, c_d) == 0$
19          $i = i + 1$;
20          $c = c_i$;
21          Partition the colony into $c$ sub-colony with $k$-means clustering method;
22      End If
23      $t = t + 1$;
24  end while
```

Fig 1: Pseudo-code of LMcBA

IV. OPTIMIZATION AND RESULTS

4.1 Encoding and Decoding

Coding is the most important and critical problem in the optimization of evolutionary algorithms. For the classical scheduling problems determined by machines in advance, most of them adopt the process-based coding method. However, the flexible job scheduling problem not only determines the processing sequence of the working procedure, but also needs to select a suitable machine for each working procedure. The solution of the problem cannot be obtained by adopting the coding method based on the working procedure.

In this paper, the coding based on the sequence of workpiece, determining the process of processing sequence, is adopted.

Each individual is composed of N * Max$\{n_i\}$ numbers, each of which represents the workpiece number. Each workpiece number appears max$\{n_i\}$ times. For the case that the workpiece number is less process number than max$\{n_i\}$, the excess part represents the "virtual process", and the processing time is set to 0 when decoding, which does not affect the problem itself.

The sequence of individual's variables determines the sequence of process scheduling. The process of each workpiece is represented by the corresponding workpiece number. The number of the workpiece number appearance is equal to the number of working procedures of the work piece. The k_{th} appearance of the workpiece serial number represents the k_{th} process of the workpiece. According to the sequence of the workpiece number in the individual, the sequence of individual determines the priority of the production process. For example, for the individual (1, 2, 2, 3, 3, 1, 1, 2), the first appearance of "1" represents the 1_{st} process of 1_{st} workpiece, the second appearance of "1" represents the 2_{nd} process of 1_{st} workpiece, and the third appearance of "1" represents the 3_{rd} process of 1_{st} workpiece.

4.2 Fitness Function

According to the above description, for the combination of the objective function f_1 and f_2, F can be expressed:

$$F = \min(w_1 f_1 + w_2 f_2) \tag{7}$$

where $w_1 = w_2 = 0.5$.

4.3 Example and Analysis

In this section, two simulation examples (small-scale one and large-scale one) of flexible job scheduling problems is illustrated first. Then, in order to illustrate the practicability of LMcBA-based flexible job scheduling, a classical practical application of flexible job scheduling for flexible welding production line of truck is given.

Example 1: small-scale flexible job scheduling

In this example, there are 5 workpieces, 6 machines and 12 processes of flexible job scheduling. The detailed processing time is given in TABLE I.

TABLE I. Machining time of job processes of problem 5×6

6×5		M_1	M_2	M_3	M_4	M_5
j_1	O_{11}	6	5	6	5	5
	O_{12}	7	8	8	7	6
	O_{13}	4	5	4	5	6
j_2	O_{21}	5	4	5	6	5
	O_{22}	4	5	5	5	3
	O_{23}	6	5	6	5	6
	O_{24}	3	5	5	4	5
j_3	O_{31}	7	8	7	6	8
	O_{32}	7	5	6	8	6
j_4	O_{41}	11	14	14	15	13
	O_{42}	12	8	10	7	9
	O_{43}	9	8	6	7	4

j_5	O_{51}	4	5	6	2	3
	O_{52}	7	6	9	10	8
	O_{53}	12	10	9	9	11
j_6	O_{61}	6	8	9	4	6
	O_{62}	4	3	3	5	5
	O_{63}	7	9	8	9	6

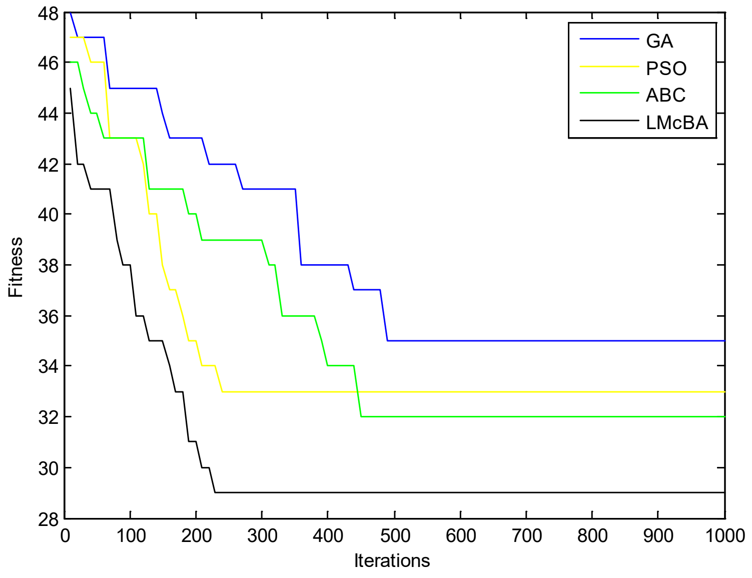


Fig 2: Fitness curves

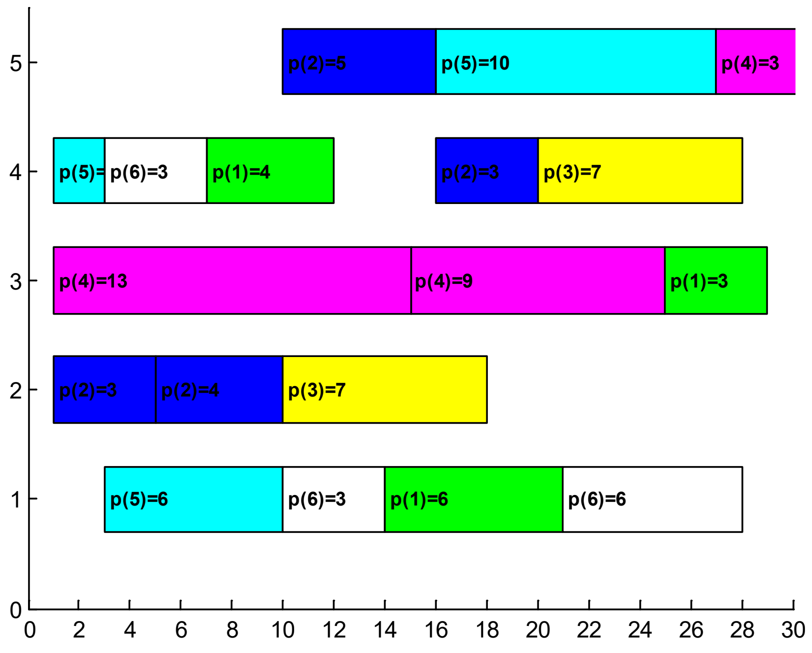


Fig 3: The gantt after evolution algorithm run

From the comparison of search rate shown in Figure 2, it is obvious that the proposed LMcBA obtains excellent results. GA locates the second rank of searching results. But ABC and PSO is unsuitable for searching the results of flexible job scheduling problem. It may be due to that the convergence rates of ABC and PSO are higher than GA, which means that both of them fall into local optimal but not the global optimal. Fortunately, the relatively slower convergence rates of GA and LMcBA can continuously find the well-performance results. The best gantt obtained by LMcBA is shown in Figure 3.

Example 2: large-scale flexible job scheduling

In this example, there are 9 workpieces, 8 machines and 28 processes of flexible job scheduling. The detailed processing time is given in TABLE II.

TABLE II. Machining time of job processes of problem 5×6

9×8		***M*₁**	***M*₂**	***M*₃**	***M*₄**	***M*₅**	***M*₆**	***M*₇**	***M*₈**
j_1	O_{11}	15	15	16	15	16	17	15	10
	O_{12}	38	38	47	35	52	43	43	45
	O_{13}	6	6	7	5	5	4	4	4
j_2	O_{21}	33	19	29	30	34	41	33	33
	O_{22}	6	7	6	5	5	6	5	7
	O_{23}	35	39	45	28	36	28	38	44
j_3	O_{31}	22	20	22	16	17	17	23	23
	O_{32}	12	10	12	9	11	11	12	14
	O_{33}	31	26	39	34	29	27	25	23
j_4	O_{41}	19	12	22	19	17	22	22	22
	O_{42}	36	34	31	42	33	43	46	42
	O_{43}	56	69	56	69	53	58	61	44
j_5	O_{51}	33	22	25	27	33	39	38	33
	O_{52}	21	13	19	16	26	21	21	33
	O_{53}	12	11	14	7	11	15	11	14
	O_{54}	43	31	31	41	42	53	44	53
j_6	O_{61}	9	8	10	10	6	6	6	10
	O_{62}	17	14	15	22	17	19	22	13
	O_{63}	21	22	18	21	21	22	24	13
j_7	O_{71}	30	27	25	27	26	28	19	29
	O_{72}	17	17	9	17	18	19	12	13
	O_{73}	34	36	33	28	32	30	37	24
j_8	O_{81}	24	25	18	24	22	21	23	23
	O_{82}	22	24	14	21	17	24	21	23
	O_{83}	19	17	18	14	22	18	12	27
j_9	O_{91}	16	8	14	14	13	16	16	14
	O_{92}	20	21	18	21	20	24	20	20
	O_{93}	21	17	21	26	21	18	23	31

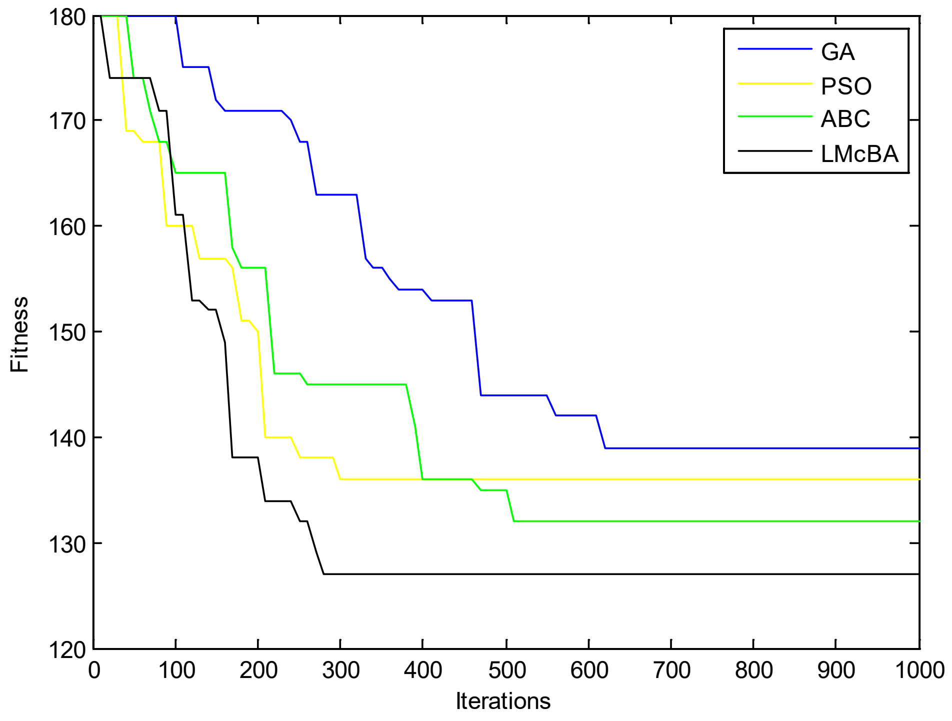


Fig 4: Fitness curves

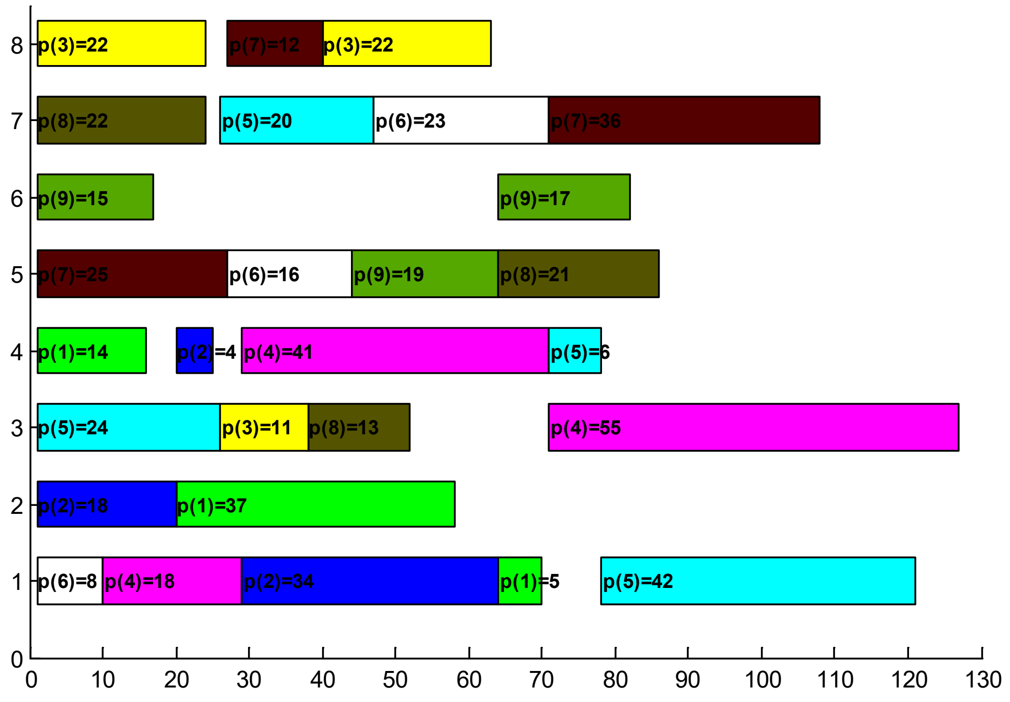


Fig 5: The gantt after evolution algorithm run

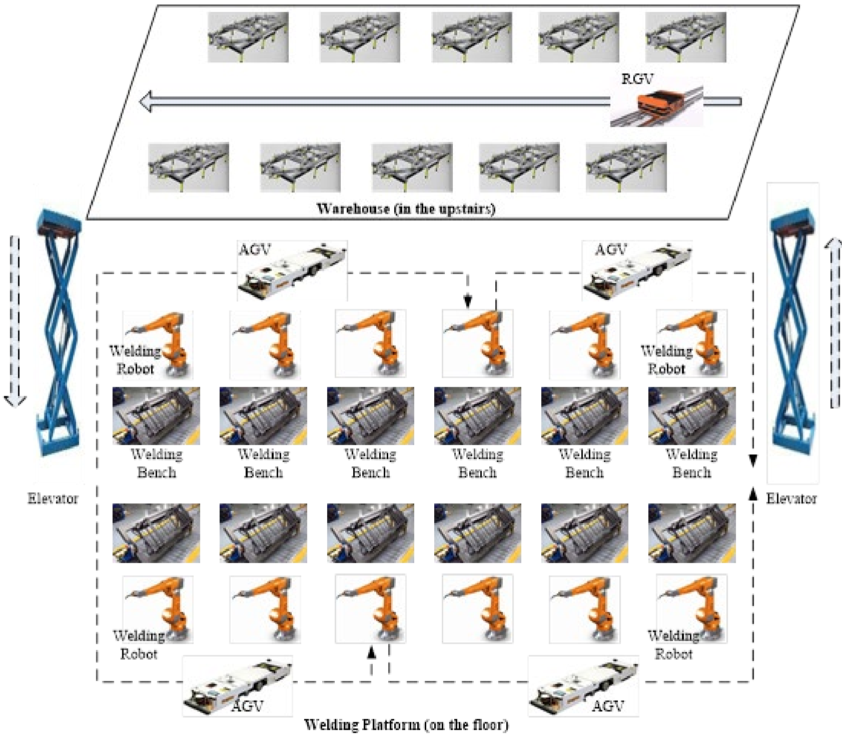
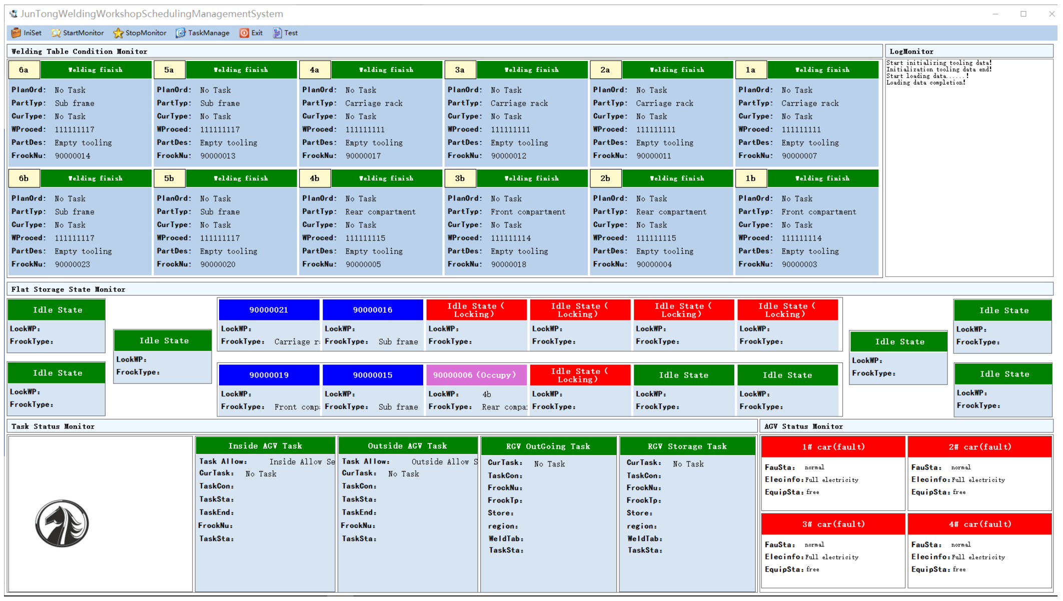


Fig 6: Chart of welding line

Related to small-scale FJSP, the large-scale FJSP is more difficult to find an excellent results. The search rate is shown in Figure 4. As the same as the phenomenon shown in Figure 2, the algorithms with the lower convergence rate can find better results. The lifecycle process of LMcBA can reasonably keep balance of explore and exploit abilities. The best gantt of 9 workpieces, 8 machines and 28 processes of flexible job scheduling obtained by LMcBA is shown in Figure 5.



Welding Table Condition Monitor

	6a	5a	4a	3a	2a	1a
	Welding finish	Welding finish	Welding finish	Welding finish	Welding finish	Welding finish
PlanOrd:	No Task	No Task	No Task	No Task	No Task	No Task
PartTyp:	Sub frame	Sub frame	Carriage rack	Carriage rack	Carriage rack	Carriage rack
CurType:	No Task	No Task	No Task	No Task	No Task	No Task
WProced:	111111117	111111117	111111111	111111111	111111111	111111111
PartDes:	Empty tooling	Empty tooling	Empty tooling	Empty tooling	Empty tooling	Empty tooling
FrockNu:	90000014	90000013	90000017	90000012	90000011	90000007

	6b	5b	4b	3b	2b	1b
	Welding finish	Welding finish	Welding finish	Welding finish	Welding finish	Welding finish
PlanOrd:	No Task	No Task	No Task	No Task	No Task	No Task
PartTyp:	Sub frame	Sub frame	Rear compartment	Front compartment	Rear compartment	Front compartment
CurType:	No Task	No Task	No Task	No Task	No Task	No Task
WProced:	111111117	111111117	111111115	111111114	111111115	111111114
PartDes:	Empty tooling	Empty tooling	Empty tooling	Empty tooling	Empty tooling	Empty tooling
FrockNu:	90000023	90000020	90000005	90000018	90000004	90000003

LogMonitor

```
Start initializing tooling data!
Initialization tooling data end!
Start loading data......!
Loading data completion!
```

Flat Storage State Monitor

Idle State	Idle State	90000021	90000016	Idle State（Locking）	Idle State（Locking）	Idle State（Locking）	Idle State（Locking）	Idle State	Idle State
LockWP:	LockWP:	LockWP:	LockWP:	LockWP:	LockWP:	LockWP:	LockWP:	LockWP:	LockWP:
FrockType:	FrockType:	FrockType: Carriage r	FrockType: Sub frame	FrockType:	FrockType:	FrockType:	FrockType:	FrockType:	FrockType:
Idle State		90000019	90000015	90000006（Occupy）	Idle State（Locking）	Idle State	Idle State		Idle State
LockWP:		LockWP:	LockWP:	LockWP: 4b	LockWP:	LockWP:	LockWP:		LockWP:
FrockType:		FrockType: Front comp	FrockType: Sub frame	FrockType: Rear compa	FrockType:	FrockType:	FrockType:		FrockType:

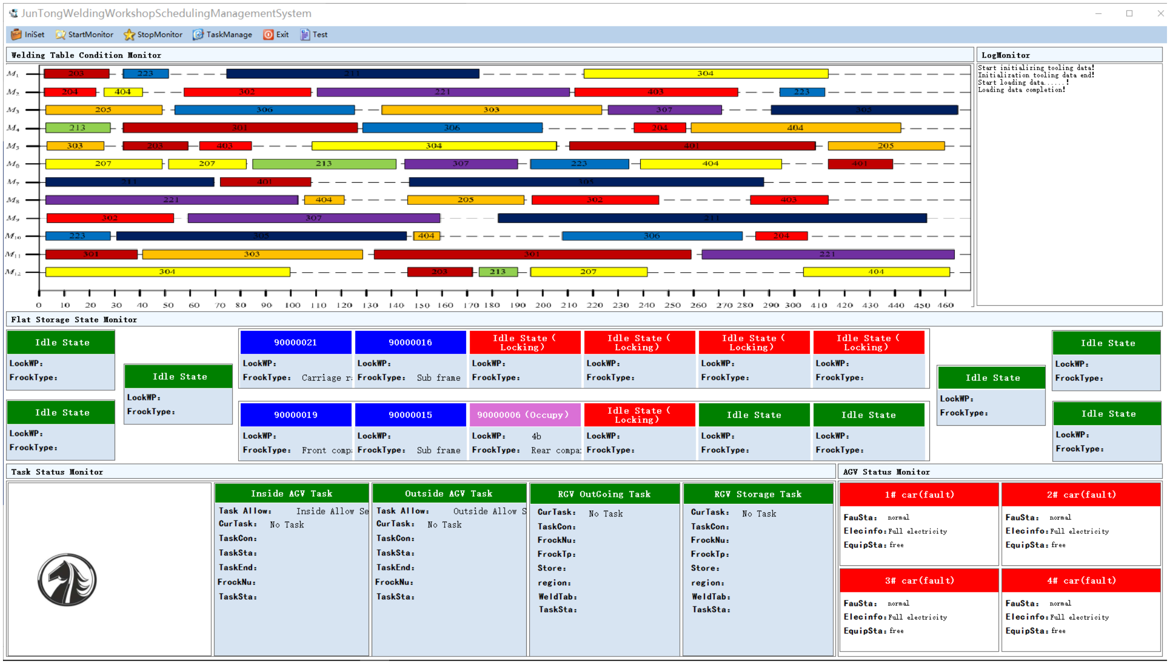
Task Status Monitor

Inside AGV Task	Outside AGV Task	RGV OutGoing Task	RGV Storage Task
Task Allow: Inside Allow Se	Task Allow: Outside Allow S	CurTask: No Task	CurTask: No Task
CurTask: No Task	CurTask: No Task	TaskCon:	TaskCon:
TaskCon:	TaskCon:	FrockNu:	FrockNu:
TaskSta:	TaskSta:	FrockTp:	FrockTp:
TaskEnd:	TaskEnd:	Store:	Store:
FrockNu:	FrockNu:	region:	region:
TaskSta:	TaskSta:	WeldTab:	WeldTab:
		TaskSta:	TaskSta:

AGV Status Monitor

1# car(fault)	2# car(fault)	3# car(fault)	4# car(fault)
FauSta: normal	FauSta: normal	FauSta: normal	FauSta: normal
Elecinfo: Full electricity	Elecinfo: Full electricity	Elecinfo: Full electricity	Elecinfo: Full electricity
EquipSta: free	EquipSta: free	EquipSta: free	EquipSta: free

Fig 7: Software interface of manufacturing execution system


JunTongWeldingWorkshopSchedulingManagementSystem
IniSet
StartMonitor
StopMonitor
TaskManage
Exit
Test
Welding Table Condition Monitor
LogMonitor
Flat Storage State Monitor
Task Status Monitor
AGV Status Monitor

```
Start initializing tooling data!
Initialization tooling data end!
Start loading data......!
Loading data completion!
```

```
Idle State           90000021                      90000016                 Idle State (Locking)  Idle State (Locking)  Idle State (Locking)  Idle State (Locking)  Idle State
LockWP:              LockWP:                       LockWP:                  LockWP:               LockWP:               LockWP:               LockWP:               LockWP:
FrockType:           FrockType:  Carriage r        FrockType:  Sub frame    FrockType:            FrockType:            FrockType:            FrockType:            FrockType:

Idle State           90000019                      90000015                 90000006 (Occupy)     Idle State (Locking)  Idle State            Idle State            Idle State
LockWP:              LockWP:                       LockWP:                  LockWP:     4b        LockWP:               LockWP:               LockWP:               LockWP:
FrockType:           FrockType:  Front comp        FrockType:  Sub frame    FrockType:  Rear compa FrockType:           FrockType:            FrockType:            FrockType:

Idle State                                                                  Idle State
LockWP:                                                                     LockWP:
FrockType:                                                                  FrockType:
```

```
Inside AGV Task                Outside AGV Task                RGV OutGoing Task       RGV Storage Task
Task Allow:   Inside Allow Se  Task Allow:   Outside Allow S   CurTask:   No Task      CurTask:   No Task
CurTask:   No Task             CurTask:   No Task              TaskCon:                TaskCon:
TaskCon:                       TaskCon:                        FrockNu:                FrockNu:
TaskSta:                       TaskSta:                        FrockTp:                FrockTp:
TaskEnd:                       TaskEnd:                        Store:                  Store:
FrockNu:                       FrockNu:                        region:                 region:
TaskSta:                       TaskSta:                        WeldTab:                WeldTab:
                                                               TaskSta:                TaskSta:
```

```
1# car(fault)                  2# car(fault)
FauSta:   normal               FauSta:   normal
Elecinfo:Full electricity      Elecinfo:Full electricity
EquipSta:free                  EquipSta:free

3# car(fault)                  4# car(fault)
FauSta:   normal               FauSta:   normal
Elecinfo:Full electricity      Elecinfo:Full electricity
EquipSta:free                  EquipSta:free
```

Fig 8: Software interface of flexible job shop scheduling system

Practical application

The developed flexible job-shop scheduling module, as a significant part of Manufacturing Execution System (MES), has been successfully applied to China Henan Juntong Vehicle Co., Ltd. The structure of truck welding line is show in Figure 6, which is constructed with warehouse (in the upstairs) and welding platform (on the floor). Four AGVs and one RGV are used for transportation on the warehouse and in the welding platform, respectively. Elevators is responsible for the transportation between the two floors. On the welding platform, there are 12 welding robots and 12 welding bench. Each robot is corresponded to its own bench. The software interface of Manufacturing Execution System is shown in Figure 7. And the software interface of flexible job-shop scheduling module is demonstrated in Figure 8, which can effectively improve the production efficiency.

V. CONCLUSION

We have designed a Lifecycle Multi-colony Bee Algorithm (LMcBA), which is inspired by the multi-swarm strategy and biological lifecycle model. In LMcBA. The initial colony is divided into sub-colonies with k-means clustering method, and each sub-colony is optimized by classical artificial bee colony. Borrowing lifecycle model, the local search capacity is enhanced. Then, this proposed LMcBA is adopted for solving FJSPs. FJSP scheduling should obtain a solution within an approximately minimized processing time. LMcBA may own the potential ability of solving FJSP. The experimental results reveal LMcBA obtains the outstanding results of minimizing processing time and total workload for small-scale, large-scale and practical FJSPs.

ACKNOWLEDGEMENTS

This research is supported by National key R&D Program of China (2016YFB1100501), National Natural Science Foundation of China (41772123, 61802280, and 61806143), Tianjin Province Science and Technology Projects (17JCYBJC15100).

REFERENCES

[1] Wang P.F, Chen X.Z., Pan Q.H., Madigan B., and Long J.Q. (2016) Laser welding dissimilar materials of aluminum to steel: an overview. The International Journal of Advanced Manufacturing Technology 87: 3081–3090

[2] Lu C., Xiao S.Q., Li X.Y., and Gao L. (2016) An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. Advances in Engineering Software 99: 161-176

[3] Wu R., Li, Y., Guo S. and Li X. (2018) An Efficient Meta-Heuristic for Multi-Objective Flexible Job Shop Inverse Scheduling Problem. IEEE Access 6: 59515-59527

[4] Yuan Y., and Xu H. (2013) An integrated search heuristic for large-scale flexible job shop scheduling problems. Computers and Operations Research 40(12): 2864–2877

[5] Xu Y., Wang L., Wang S.Y, and Liu M. (2015) An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. Neurocomputing 148: 260-268

[6] Li X.Y., Xiao S.Q, Wang C.Y., and Yi J. (2019) Mathematical modeling and a discrete artificial bee colony algorithm for the welding shop scheduling problem. Memetic Computing 11: 371–389

[7] Mehta K. (2017) Advanced Joining and Welding Techniques: An Overview. Advanced Manufacturing Technologies 101-136

[8] Karaboga D. and Akay B. (2009) A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214: 108–132

[9] Hartigan J.A. and Wong M.A. (1979) A K-means Clustering Algorithm: Algorithm AS 136. Applied Statistics 28(1): 100-108

[10] Kühnert D., Stadler T., Vaughan T.G., and Drummond, A.J. (2014) Simultaneous reconstruction of evolutionary history and epidemiological dynamics from viral sequences with the birth–death SIR model. Journal of The Royal Society Interface 11: 6-17