frontiers | Frontiers in Bioengineering and Biotechnology

ORIGINAL RESEARCH
published: 16 August 2022
doi: 10.3389/fbioe.2022.909548

Research of Flexible Assembly Job-Shop Batch–Scheduling Problem Based on Improved Artificial Bee Colony

Xiulin Li[1]*, *Jiansha Lu*[2], *Chenxi Yang*[1] and *Jiale Wang*[1]

[1]Department of Logistics Management and Engineering, Zhejiang Gongshang University, Hangzhou, China, [2]Institute of Industrial Engineering, Zhejiang University of Technology, Hangzhou, China

This study examined the flexible assembly job-shop scheduling problem with lot streaming (FAJSP-LS), common in multivariety and small-batch production, such as household electrical appliances. In FAJSP-LS, an assembly stage is appended to the flexible job shop, and jobs in the first stage are processed in a large batch to reduce switching costs, while leading to more waiting time, especially during the assembly stage. This article considered splitting the batch into a few sub-batches of unequal and consistent sizes to allow jobs to efficiently pass the two-stage system. With this objective, the problem was modeled as a mixed-integer linear program comprising the following two subproblems: batch splitting and batch scheduling. As the integrated problem is NP-hard, the improved bioinspired algorithm based on an artificial bee colony was proposed, including a four-layer chromosome–encoding structure to describe the solution, as well as an optimization strategy utilizing different bee colonies to synchronously solve this two-stage problem. To examine the algorithm's efficiency, a benchmark case was used to show that better solutions can be acquired with the improved algorithm regardless of whether the batch was split into equal or unequal sizes. To promote practical implementation, the algorithm was applied to a real case refrigerator workshop and showed better performance on time efficiency when jobs were split into unequal sizes compared to jobs without splitting or splitting into equal sizes.

Keywords: flexible assembly job shop, batch splitting, batch scheduling, lot streaming, consistent size, unequal size, artificial bee colony

OPEN ACCESS

Edited by:
Zhihua Cui,
Taiyuan University of Science and Technology, China

Reviewed by:
Robert Ojstersek,
University of Maribor, Slovenia
Surekha Paneerselvam,
Amrita Vishwa Vidyapeetham University, India

***Correspondence:**
Xiulin Li
lxl@zjgsu.edu.cn

Specialty section:
This article was submitted to Bionics and Biomimetics, a section of the journal Frontiers in Bioengineering and Biotechnology.

Received: *31 March 2022*
Accepted: *23 June 2022*
Published: *16 August 2022*

Citation:
Li X, Lu J, Yang C and Wang J (2022) Research of Flexible Assembly Job-Shop Batch–Scheduling Problem Based on Improved Artificial Bee Colony. Front. Bioeng. Biotechnol. 10:909548. doi: 10.3389/fbioe.2022.909548

1 INTRODUCTION

An assembly job shop is a two-stage production structure in that an assembly stage is appended to a job shop. Once the assembly stage is appended, the job-shop scheduling problem becomes the assembly job-shop problem (AJSP) (Wong and Ngan, 2013). In AJSP, some jobs are dependent so that they can be completed directly after the machining stage. In contrast, some jobs are independent and need to enter the assembly buffer waiting for specific jobs with assembly relationships according to the bill of material. A flexible job shop is a generalized version of the job shop and is more common. The AJSP will expand to a flexible assembly jobs-hop scheduling problem (FAJSP) when the assembly stage is appended to a flexible job shop.

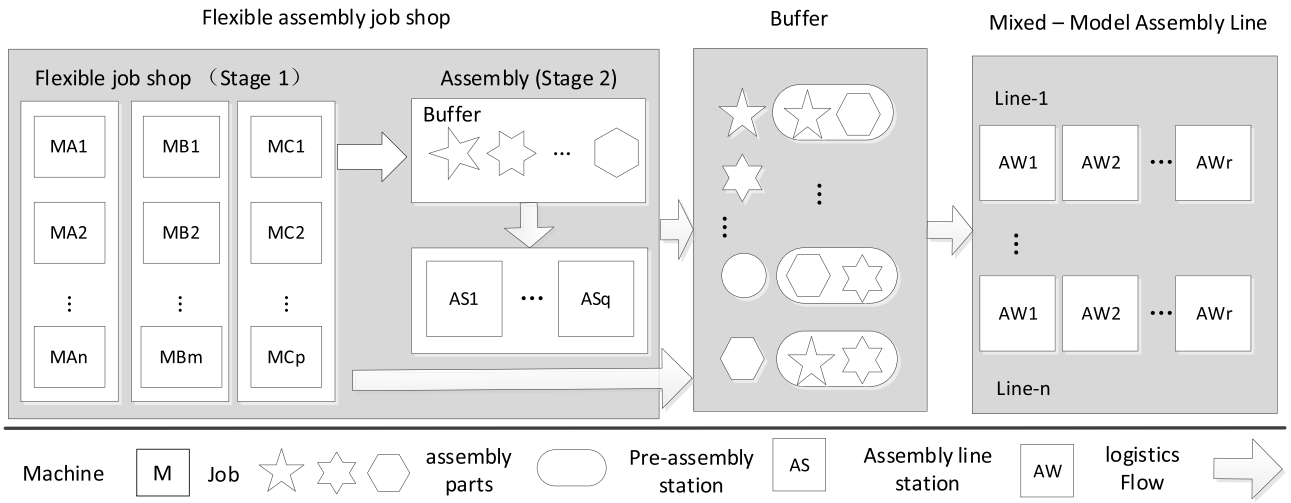

FIGURE 1 | Structure of flexible assembly job-shop with following mixed-model assembly line.

This study was inspired by a real-world case in a refrigerator factory, which has the flexible assembly job-shop structure shown in **Figure 1**. This structure usually exists in the processing and preassembly stage before multivariety and small-batch mixed-model assembly line, such as the household electrical appliance production. As the TAKT time upstream is significantly shorter than the downstream mixed-model line, the job is usually processed in a batch to reduce the total setup time. Large batches are accessible to the backlog of work in process (WIP) among machines if the batch is wholly moved, especially for the independent jobs when waiting for assembly parts. Lot streaming (LS) is a technique that allows splitting a large batch into a few smaller sub-batches and produce on parallel machines to smooth the following demand of multivariety jobs from the assembly line and shorten the flow time of each job to reduce WIP. When LS is applied, the FAJSP expands to FAJSP with LS (FAJSP-LS), which includes the following four decisions: 1) the quantity of sub-batch to each job, 2) the size of each sub-batch, 3) the machine selection for each operation of sub-batch, and 4) sequencing of operations on each machine. Combining with assembly and lot-splitting, the FAJSP-LS is more complicated than FJSP, which was proved to be NP-hard.

As the fundamental problem of FAJSP-LS, AJSP was first studied in the 1960s to solve a multilevel assembly scheduling problem under a random environment (Pereira et al., 2011). Due to the complexity of kit constraints in assembly, heuristic rules of distribution and scheduling are primarily used to solve them in an early stage of research. For example, the production structure of an assembly workshop, including single-part and multipart products, was studied, and a hybrid rule based on the shortest processing time (SPT) and assembly jobs first with SPT as a tie-breaker (Asmf-spt) was proposed to optimize the objectives of tardiness and process time (Huang et al., 1984). Thiagarajan et al. (2005) proposed a series of heuristic rules for distribution scheduling to optimize the comprehensive objectives such as weighted tardiness and process time. Omkumar et al. (2009) proposed a heuristic method based on an ant colony algorithm to solve the multilevel assembly-job-scheduling problem and compared it with various rules to verify the performance advantages of the algorithm. Compared with the existing production system, the classical AJSP problem was simplified in two aspects. First, the flexibility of machine selection and process flow was not considered in the processing stage; second, the impact of a large batch on the operation time and inventory in actual production was not considered.

To solve the flexibility problem, the FAJSP is proposed as an AJSP extension. Benjaafar and Ramakrishnan. (1996) and Zhang et al. (2020) defined and summarized the flexibility in production as operation, process, and sequence flexibility. Among these, process flexibility is the most common; that is, the job process can be processed on multiple optional processing machines. However, before deciding the processing sequence of each process on the machine, it is necessary to allocate the process to a specific machine. Therefore, FAJSP is also regarded as an integration of AJSP and integrated planning and scheduling problem (Nourali et al., 2012; Zhang et al., 2020). Nourali et al. (2012) defined the FAJSP and established a mixed-integer programming model considering makespan, and designed a PSO to solve it. Zhang and Wang (2018) and Zhang et al. (2020) studied FAJSP with component sharing, established a constraint programming model and mixed-integer programming model, and designed scheduling rules and distributed an ant colony algorithm. Wu et al. (2019) considered the FAJSP based on the distributed workshop architecture and proposed a model considering tardiness, production, and transportation costs and designed a genetic algorithm to solve it. Lin et al. (2022) studied FAJSP with a tight job based on a genetic algorithm.

To solve the batch processing problem, LS was used to split the product batch into production/transfer sub-batches and smoothly transition to a multistage production process through sorting and coordination sub-batches to shorten the

Parameters	Description
m	Total number of machines
n	Total number of jobs
n_i	Total number of options of job Ji
Am_i	Total amount of job Ji
Lp	Minimum size of sub-batch
Variables	**Description**
L_i	Total number of sub-batches of job Ji, $(i=1,\ldots, n)$
Am_{ip}	Volume of the pth sub-batch of job Ji
E_{ipjk}	The completion time of operation j of pth sub-batch of job Ji on machine M_k $(i=1,\ldots, n; j=1,\ldots, n_i; p=1,\ldots, L_i; k=1,\ldots, m)$
E_{ip}	The completion time of the pth sub-batch of job Ji
F_{ip}	The flow time of the pth sub-batch of job Ji
F_{ipjk}	The flowtime of the jth operation of pth sub-batch of job Ji on machine M_k
PT_{ijk}	The preparation time for the jth operation of job Ji on the machine M_k
W_{ipjk}	The waiting time of the jth operation of pth sub-batch of job Ji on machine M_k
A_{ipjk}	The time when the jth operation of pth sub-batch of job Ji arrive machineM_k
B_{ipjk}	The time when the jth operation of pth sub-batch of job Ji start processing on machineM_k
P_{ijk}	The processing time of jth operation of job Ji on machineM_k
$\alpha_{ii'}^k$	$\begin{cases} 1 & \text{Job } J_i \text{ and } J_{i'} \text{ processed continuously on } M_k \text{ are of the same type} \\ 0 & \text{else} \end{cases}$
$\gamma_i^{i'}$	$\begin{cases} 1 & \text{job } Ji \text{ and job } Ji' \text{ have assembling relation} \\ 0 & \text{else} \end{cases}$
X_{ipjk}	$\begin{cases} 1 & \text{the } jth \text{ operation of } pth \text{ sub} - \text{batch of job } Ji \text{ is processed on } M_k \\ 0 & \text{else} \end{cases}$

production cycle. In terms of sub-batch size, the batch-splitting strategy can be divided into that of equal and variable size (Trietsch and Baker, 1993) and into consistently sized (the sub-batch size of the different job is different but remains unchanged during processing) and variably sized (the sub-batch size of the different job is different but can be changed when moving to next machine) (Martin, 2009). Low et al. (2004) proved that the equal-sized division method performs better in optimizing the time-related target. Similar studies also show that batch splitting can effectively reduce the system makespan and reduce WIP (Kalir and Sarin, 2000). However, Low (2004) also pointed out that too many batches will lead to a decline in time performance. That means the connection between the batches' quantity and completion time is U-shaped. Both too large and too small quantity will lead to the reduction in time performance. This phenomenon is more pronounced when there are assembly and flexible constraints in the production process. Chan et al. (2008) studied LS with AJSP (AJSP-LS) for the first time and solved equal and unequal problems using a genetic algorithm. Furthermore, Wong and Ngan (2013) continued this study and designed a hybrid genetic algorithm and hybrid particle swarm. Ba et al. (2015) considered taking the equal-size strategy to divide all jobs and use PSO to optimize the scheduling of divided sub-batches.

In summary, among the reviewed literature, most studies focus on the flexibility problem or batch processing separately. However, an integrated problem, flexible assembly job shop with LS (FAJSP-LS) that includes these two problems is closer to the reality in processing-assembly production. Thus, Zeng et al. (2019) studied the FAJSP-LS and designed a hierarchical iteration algorithm that first uses improved GA to solve the batch-splitting subproblem and then distribution rules such as first come first served (FCFS) and operation due date to solve the batch-scheduling subproblem. However, compared with the upper-layer batch–splitting problem, the lower layer batch-scheduling problem, which includes machine assignment and sub-batch sequencing, has a larger search space. Solutions can be acquired quickly with distribution rules, but at the expense of performance. Other hierarchical iteration or integrated algorithms in the reviewed literature had shown good performance on AJSP-LS and found that equal-size LS strategy is better than unequal one ([Chan et al., 2008; Wong and Ngan. (2013)]. However, the conclusion may change under the FAJSP environment, because flexible machine choices are allowed for each operation of the sub-batch. In addition, nearly all studies restricted the assembly to be processed only after all the sub-batches of assembly jobs are ready, which potentially increased the makespan, and WIP then affects the conclusion.

To the best of our knowledge, the FAJSP-LS has been rarely studied. Hence, in this article, the problem was described in detail, and a mathematical model was proposed based on unequal and consistent size that sub-batches of the same job may have unequal size and the size had to be maintained during the entire processing route (Novas, 2019). Considering unequal size, this study also relaxed the restriction that sub-batches can be assembled even if they are not the same size. For example, jobs A and B have assembly connections, and the size of sub-batch A_i of job A and B_j of job B are 100 and 200, respectively. One hundred pieces of both A and B can be assembled directly once these two sub-batches pass through the first stage, and the other 100 pieces of B will be left waiting for more A to assemble. Applying this rule can increase flexibility to batch splitting and significantly reduce flow time and WIP, but at the cost of more complexity to batch scheduling.

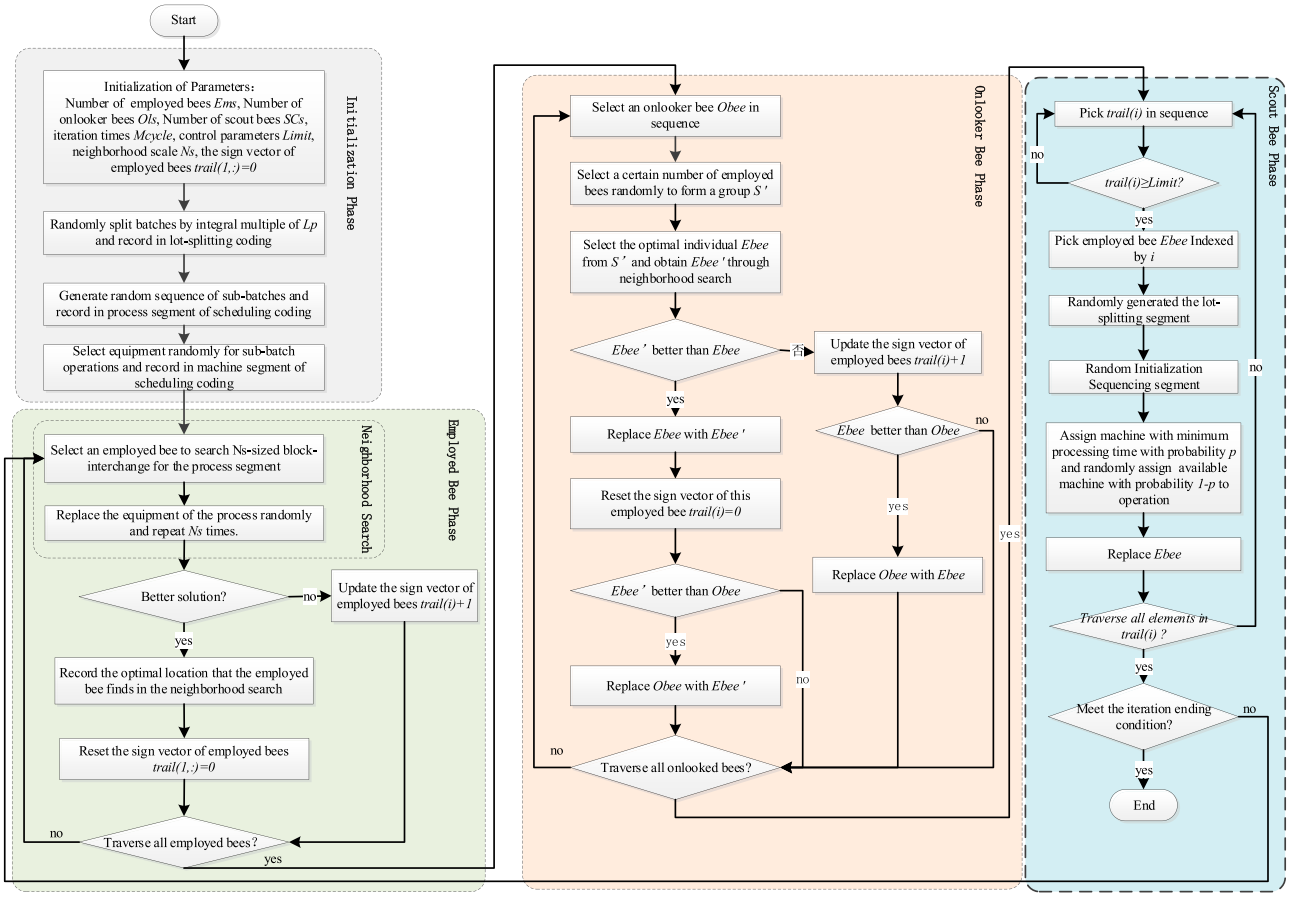

FIGURE 2 | Framework of improved algorithm based on an artificial bee colony.

Hence, in this study, a global optimization algorithm was designed to solve the integrated problem directly based on the multipopulation collaboration mechanism of the artificial bee colony algorithm (ABC). ABC is a bioinspired algorithm that is easily tailored to a new problem and obtains near-optimal solutions (O Kheirandish et al., 2015; Xie et al., 2022). ABC was inspired by the foraging behavior of bee colony and had shown better performance than other bioinspired algorithms such as particle swarm optimization and genetic algorithm (Karaboga and Akay, 2009). Since it was proposed in 2005, ABC has been widely used to tackle combination optimization problems, such as flexible job-shop scheduling problems and flow shop optimization (Gong et al., 2020; Meng et al., 2018; Li et al., 2017). Most of the existing algorithms take distributed strategies for solving these two subproblems separately or iteratively (Chan et al., 2008), or take integration strategies but simplify one subproblem by using heuristic rules or batch division rules (Ba et al., 2015; Zeng et al., 2019). In study article, a four-layer chromosome–encoding method was designed for this algorithm to describe the two-stage problem, and an optimization strategy was designed to assign these two subproblems to different populations and then optimized as a whole to acquire ideal splitting and effective scheduling synchronously. Computational experiments were performed to examine the integrated optimization performance for this kind of two-stage problems and test the performance of splitting batches into equal and unequal sizes. Furthermore, a real refrigerator production case tests the effectiveness of unequal-size batch splitting with a minimum average flow time.

2 PROBLEM DESCRIPTION AND FORMULATION

2.1 Problem Description

The FAJSP-LS can be described as follows. The job shop has m machines that can process n kinds of jobs. Each job contains n_i processes that can be processed on at least one machine with different processing times. There is an assembling relation between at least two or more jobs. Each number of jobs can be split into multiple sub-batches with different sizes and keeping the size during processing and transport. In actual production, assembly jobs are generally transferred in one circulation box or according to the number of KanBan. Therefore, the minimum number of transport units is defined as Lp, and the size of the sub-batch is an integral multiple of Lp. The objective is to minimize each sub-batch's average flow

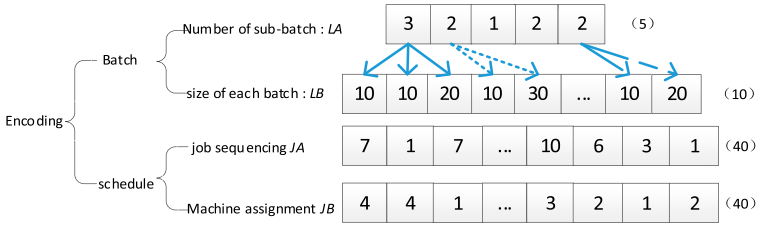

FIGURE 3 | Encoding example.

time, reflecting the sub-batch transfer efficiency and WIP level (Huang et al., 2010). The flow time of a job on a machine mainly includes the waiting, setup, and processing times, and the difference between the completion time and arrival time. When two sub-batches continuously processed on the same machine belong to the same job, the preparation time is not required. Other main assumptions are as follows:

1) The volume of jobs with assembling relation is equal.
2) All sub-batches can be processed at the moment zero.
3) Any machine can only process one sub-batch at a time, and processing cannot be interrupted.
4) There is no transferring time and cost between machines.
5) There is no assembly time during the assembly stage.
6) The buffer between machines or stages is infinite.

2.2 Problem Formulation

Notations:

$$FT = \min\left((F_1 + F_2) \Big/ \sum_{i=1}^{n} L_i \right) \tag{1}$$

$$F_1 = \sum_{i=1}^{n} \sum_{p=1}^{L_i} \sum_{j=1}^{n_i} \sum_{k=1}^{m} F_{ipjk} \tag{2}$$

$$F_2 = \sum_{i=1}^{n} \sum_{i'=1}^{n} \left| \left(\sum_{p=1}^{L_i} E_{ipn_ik} - \sum_{p'=1}^{L_{i'}} E_{i'p'n_{i'}k'} \right) \right| \times \gamma_i^{i'} / 2, \quad k = 1, 2, \ldots m \tag{3}$$

$$\begin{aligned} F_{ipjk} &= W_{ipjk} + P_{ipjk} + PT_{ijk}\left(1 - \alpha_{ii'}^{k}\right) \\ F_{ip1k} &= P_{i1k} \end{aligned} \tag{4}$$

$$\begin{aligned} &i, i' = 1, 2, \ldots, n;\ j = 2, \ldots, n_i;\ p = 1, 2, \ldots, L_i;\ k_1, k_2 = 1, 2, \ldots, m \\ &E_{ipjk} = A_{ipjk} + F_{ipjk} \end{aligned} \tag{5}$$

$$\begin{aligned} &i = 1, 2, \ldots, n;\quad p = 1, 2, \ldots, L_i;\ \ j = 1, 2, \ldots, n_i;\ \ k = 1, 2, \ldots, m \\ &W_{ipjk} = B_{ipjk} - A_{ipjk} \\ &A_{ipjk} = E_{ip(j-1)k'},\quad B_{ip1k} = A_{ip1k}; \end{aligned} \tag{6}$$

$$\begin{aligned} &i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n_i;\ p = 1, 2, \ldots, L_i;\ k, k' = 1, 2, \ldots, m \\ &P_{ipjk} = WT_{ij}^{k} \times Am_p^i \end{aligned} \tag{7}$$

$$\begin{aligned} &i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n_i;\ k = 1, 2, \ldots, m;\ p = 1, 2, \ldots, L_i \\ &Am_i = \sum_{p=1}^{L_i} Am_{ip},\ Am_{ip} = \gamma \times Lp, \\ &\gamma \in N,\ 1 \le \gamma \le (Am_i)/Lp \end{aligned} \tag{8}$$

Eq. 1 is the objective of average flow time minimization. F_1 is the total flow time of all sub-batches on the job-shop stage, and F_2 the total flow time of all sub-batches during the assembly stage. **Eq. 2** is the calculation of F_1; **Eq. 3** is the calculation of F_2; **Eq. 4** indicates that the flow time of sub-batch on machine is equal to the sum of the waiting, setup, and processing times; **Eq. 5** indicates the calculation of completion time of sub-batch; **Eq. 6** defines the waiting time of sub-batch; **Eq. 7** defines the processing time of sub-batch; and **Eq. 8** guarantee that the sum of all sub-batch is an integral multiple of the *Lp*.

3 PROPOSED IMPROVED ARTIFICIAL BEE COLONY ALGORITHM

3.1 Framework of the Algorithm

As a bioinspired algorithm, ABC has been studied and applied widely, and it shows a better performance in the combination optimization problem (Karaboga et al., 2014; Meng et al., 2018; Gong et al., 2020). Based on the multipopulation collaboration mechanism of the ABC algorithm, the batch-scheduling subproblem, which has a more extensive solution space, was assigned to employ bee and onlooker bee to optimize, while the batch-splitting subproblem was optimized by the scout bee. The critical control parameter *limit* was designed for employer bee and onlooker bee and storage in vector *trail* to enlarge the global optimization capability. A neighborhood search is designed for the employed bee to strengthen the local optimization of the employed bee. Moreover, a similar design is adopted by the onlooker bee to choose an employed bee from a random temp group *S'* instead of an employed bee. **Figure 2** shows the algorithm architecture.

3.2 Detailed Design of the Algorithm

This part includes a detailed encoding, initialization, and scout bee phase design. The employed bee and onlooker bee design can be found in former research (Li et al., 2017).

3.2.1 Chromosome Encoding

FAJSP-LS is typical discrete optimization problem, and it is necessary to put up an encoding method to structure its solution. It includes two subproblems with four decisions: quantity of sub-batches for each job and lot size for each sub-batch in the batch-splitting subproblem, machine arrangement and

TABLE 1 | Parameters table.

Parameter	Level			
	1	2	3	4
Ps	TL	1.5 × TL	2 × TL	2.5 × TL
limit	4	6	8	10
p	0.2	0.4	0.6	0.8

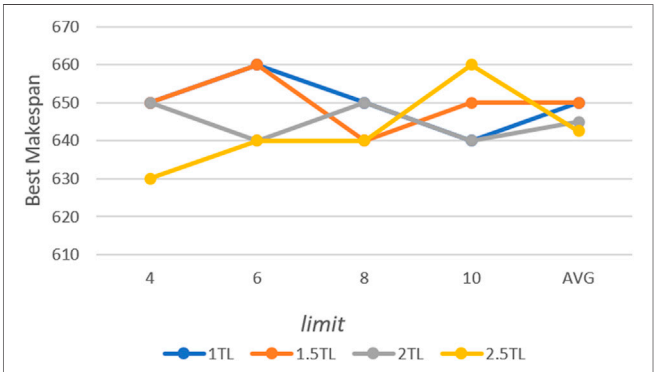

FIGURE 4 | Trend of best solution with *Ps* and *limit*.

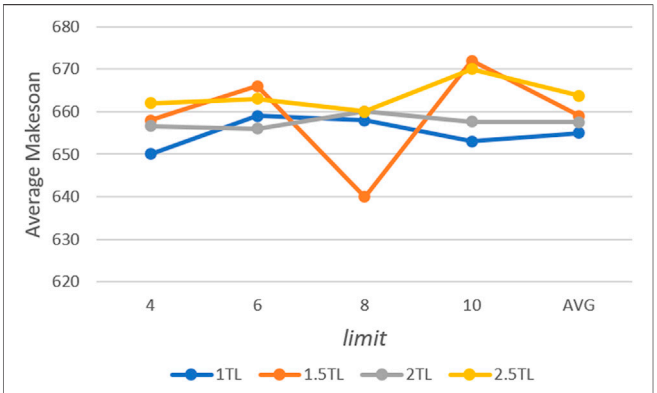

FIGURE 5 | Trend of average solution with *Ps* and *limit*.

processing sequence on specific machine in sub-batch-scheduling subproblem. To encode the solution completely, a four-layer chromosome–encoding structure was proposed to describe the solution, and each segment was encoded using positive integers that record the real value of decision variable (xie, 2022). These segments are recorded as a one-dimensional array of a different size. The LA and LB segments in the first two layers are for the batch-splitting subproblem. LA is the sub-batch quantity segment with n elements, n is the number of jobs, and the ith element $\{L_i, i = 1, 2, ..., n\}$ brepresents the sub-batch quantity of job J_i. The LB segment represents the size of each sub-batch, which can be indexed according to the cumulative number of sub-batches. Assume SL_q is the qth sub-batch in all sub-batches, $q = 1, 2, ..., \sum_{i=1}^{n} L_i$. The qth element Am_{ip} in LB represents the quantity of SL_q, which is also the

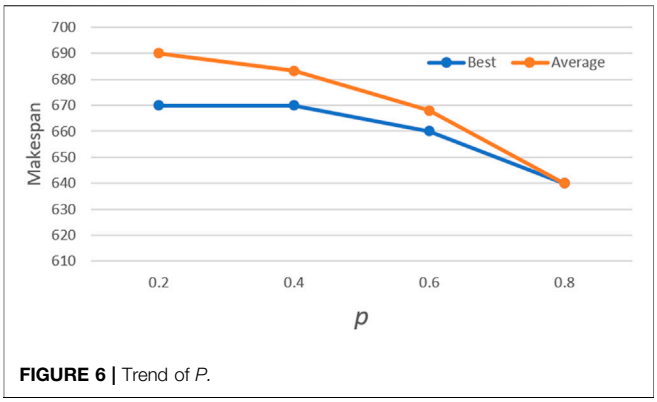

FIGURE 6 | Trend of *P*.

pth sub-batch of job J_i, The correlation of q and p is described as **Eq. 9**:

$$q = \sum_{i=1}^{i-1} L_{i,} + p,\ 1 \le p \le L_i \tag{9}$$

The JA and JB segments in the last two layers are for sub-batch scheduling. JA represents the operations' sequence of each sub-batch, and JB represents the machine assignment for each operation. When each sub-batch is regarded as a job, then the regular encoding method for FJSP (Li et al., 2011) could be adopted in these two segments. Assume the size of JA *is* L_{JA} as shown in **Eq. 10**:

$$L_{JA} = \sum_{i=1}^{\mathbf{n}} \sum_{p=1}^{L_i} n_i * p \tag{10}$$

Figure 3 shows an example of problem coding. The number of machines $m = 4$, the type of jobs $n = 5$, and each job has four n_i operations. The LA segment represents the number of sub-batches of each job. The LB segment represents the size of each sub-batch, which can be indexed according to the job number. For example, job J_1 has 40 pieces to be processed, the number of sub-batches is three, and the sizes are 10, 10, and 20 pieces, respectively. The encoding of job sequencing JA shows the operations sequence of each sub-batch. In this segment, the first element seven is the first operation of the 1st sub-batch of J_4, and the second element one is the first operation of the 1st sub-batch of J_1. They are all assigned to machine four according to the machine assignment segment JB, and the operation of J_4 is processed ahead of the operation of J_1. The second seven in JA is the second operation of the 1st sub-batch of J_4.

The advantages of the proposed four-layer chromosome encoding are as follows: 1) high flexibility to meet the uncertain size of LB, JA, and JB segments. The size of LA is known as the number of jobs, once the LA segment is determined, the size of the other three segments can be acquired. 2) The structure can ensure the whole search space, and any solution can be encoded into only one chromosome; correspondingly, any one chromosome can also be decoded into only one legal solution. 3) The four-segment structure shows the relevance between segments in a more intuitive way.

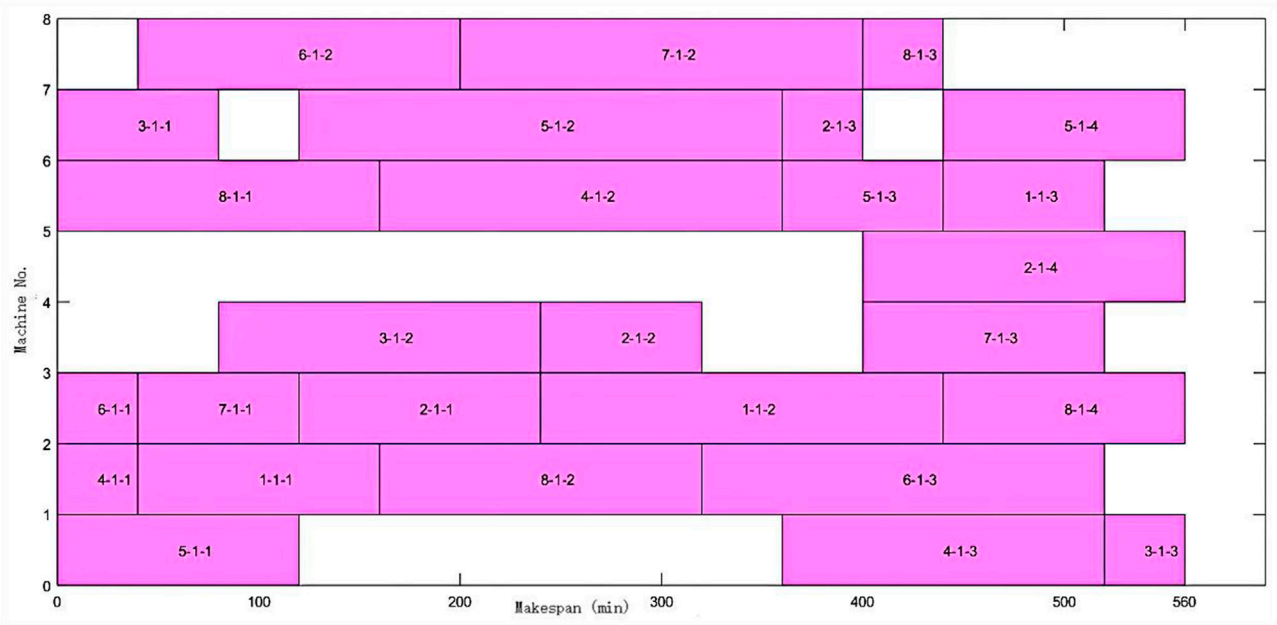

FIGURE 7 | Kacem 8 × 8 Gantt chart without batch splitting.

TABLE 2 | 4 × 6 Comparison of batch splitting and scheduling results.

	No. of sub-batch (equal size)	Makespan (equal size)	No. of sub-batch (unequal size)	Makespan (unequal size)
Sun et al. (2008)	13	87	/	/
Bai et al. (2010)	8	90	8	84
Xu et al. (2016)	10	86	9	83
This article	8	80	10	78

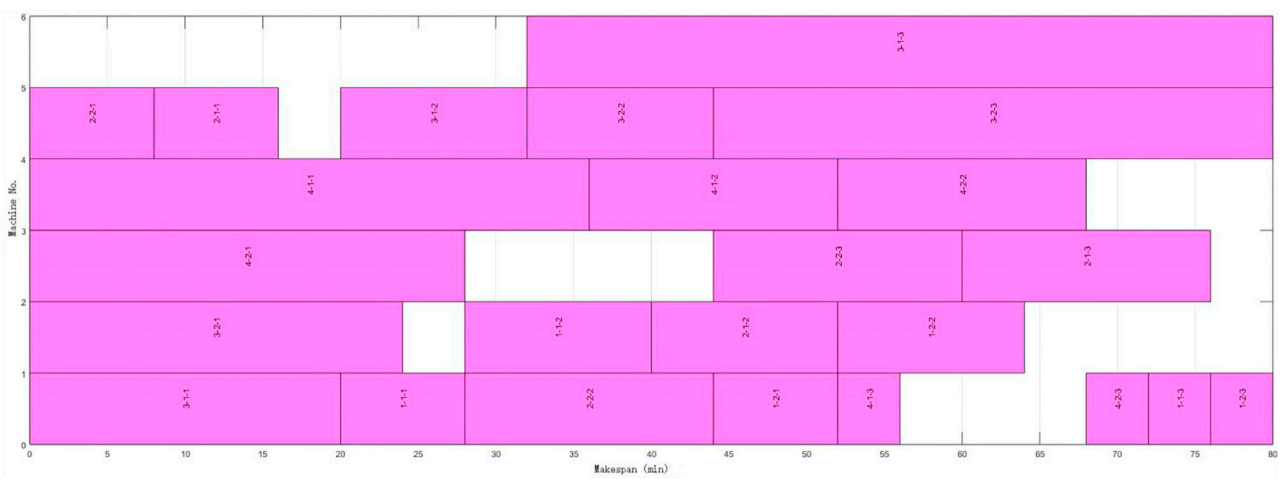

FIGURE 8 | 4 × 6 Gantt chart of equal-size batch splitting and scheduling.

FIGURE 9 | 4 × 6 Gantt chart of unequal-size batch splitting and scheduling.

TABLE 3 | Production plan.

Model	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11
Quantity	400	400	300	500	400	200	200	400	400	600	500

3.2.2 Population Initialization of Algorithm

The initialization phase includes batch segment and scheduling segment initialization as follows:

1) Generate the batch segment code; the size of every sub-batch is the integral multiple times of Lp. Then the maximum number of sub-batches is Am_i/Lp for the job i. Then, a random L_i is generated with the constraint,$L_i = 1, 2, ..., Am_i/Lp]$ and it generates the coding segment LA.
2) In random split the job batch into L_i sub-batches, make sure the size of each sub-batch is an integral multiple of Lp, and generate the coding segment LB.
3) Regard the sub-batch as a new job and randomly generate the scheduling sequence of operations of these new jobs and encoding segment JA.
4) In random assign operations to available machines and generate encoding segment JB.
5) Repeat the above steps $\sum_{i=1}^{n} Am_i/Lp$ times to finish the initialization of the employed and onlooker bees.

3.2.3 Scout Bee Algorithm

The scout bee algorithm undertakes the optimization for batch splitting. The employed and onlooker bees whose *trail* fit the control parameter *limit* will trigger the scout bee algorithm and generate a new batch-splitting solution. The steps, which are similar to initialization, are as follows:

1) Follow the first two steps of population initialization that randomly generate the number of sub-batches of each job.
2) Follow the third step of population initialization to structure the job sequencing segment, assign a machine with minimum processing time with probability p, and randomly assign an available machine with probability $1\text{-}p$ to the corresponding operation per batch.

4 EXPERIMENTS AND RESULTS

In this section, the performance of the proposed algorithm is separately tested on FJSP, FAJSP, and FAJSP-LS cases. Furthermore, a design of the experimental method is used to optimize the parameters.

4.1 Parameter Setting

The proposed ABC algorithm has the following three main parameters: population size (*Ps*), the maximum number of trials *limit*, and probability p that assigns the machine with minimum processing time to an operation in scout bee stage. Among them, *Ps* and *limit* are intrinsic parameters of ABC, and limit is the key parameter that decides the global performance of the algorithm and impacts the optimization capability of the batch-splitting subproblem. P is a secondary parameter adopted to improve local optimization capability on batch-scheduling subproblem.

Population size *Ps* is usually set as a specific number (Gong et al., 2020; Li et al., 2020). However, it should be associated with the scale of problem to be solved. The maximum number of sub-batches TL can be described as $TL = \sum_{i=1}^{n} Am_i/Lp$ and the *Ps* is designed to four levels; *limit* is set as (4, 6, 8, 10) according to the

TABLE 4 | Processing time matrix.

Job	Operation	Processing machine									
		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	O_{11}	6.5	7.2	6.5	8	—	—	—	—	—	—
	O_{12}	—	—	—	—	7	6	6	6.5	—	—
	O_{13}	—	—	—	—	7.5	7	8	6	—	—
J_2	O_{21}	7	7	7	6	—	—	—	—	—	—
	O_{22}	—	—	—	—	8	8.4	7.6	8	—	—
	O_{23}	—	—	—	—	7	5	6	7	—	—
J_3	O_{31}	5.5	5.5	7	6	—	—	—	—	—	—
	O_{32}	—	—	—	—	4	4	5	4	—	—
	O_{33}	—	—	—	—	5	5.5	6	6	—	—
J_4	O_{41}	8	7.7	8	7.5	—	—	—	—	—	—
	O_{42}	—	—	—	—	5	5	7	6	—	—
	O_{43}	—	—	—	—	6	6	5	5	—	—
J_5	O_{51}	6	7	7	6	—	—	—	—	—	—
	O_{52}	—	—	—	—	—	—	—	—	8	7
	O_{53}	—	—	—	—	8	7	7.7	8	—	—
J_6	O_{61}	6.8	6	7	7	—	—	—	—	—	—
	O_{62}	—	—	—	—	—	—	—	—	6	6
	O_{63}	—	—	—	—	8	8.4	7.6	8	—	—
J_7	O_{71}	7	5	6	7	—	—	—	—	—	—
	O_{72}	—	—	—	—	—	—	—	—	6	6
	O_{73}	—	—	—	—	8	7	8	8	—	—
	O_{74}	—	—	—	—	6.8	6	7	7	—	—
J_8	O_{81}	5	5	4	5	—	—	—	—	—	—
	O_{82}	—	—	—	—	—	—	—	—	7	7
	O_{83}	—	—	—	—	8	7.7	8	7.5	—	—
	O_{84}	—	—	—	—	7	7	7.5	6.5	—	—
J_9	O_{91}	8	7	8	8.2	—	—	—	—	—	—
	O_{92}	—	—	—	—	—	—	—	—	6	6
	O_{93}	—	—	—	—	5	6	6.5	7.2	—	—
	O_{94}	—	—	—	—	5.8	7	6.6	7.2	—	—
J_{10}	O_{101}	7.5	9	8	6	—	—	—	—	—	—
	O_{102}	—	—	—	—	7.2	8.5	8	8	—	—
	O_{103}	—	—	—	—	8	8	8	8	—	—
J_{11}	O_{111}	6.5	7.2	6.5	8	—	—	—	—	—	—
	O_{112}	—	—	—	—	7.6	9	8	7.5	—	—
	O_{113}	—	—	—	—	7.8	8	7.5	8	—	—

"—" means that the operation cannot be processed on that machine.

existing research (Li et al., 2011; Xie et al., 2020). The parameters are shown in **Table 1**.

The middle scale 10 × 10 case from Kacem benchmark (Kacem et al., 2002) is used to test the parameters. The max iteration times are 500. First, the *Ps* and *limit* are tested. For each possible configuration, the proposed algorithm is run 10 times independently. The best and average solutions' trends are show in **Figures 4** and **5**. Avg is the average value of different *limit* for a specific *Ps*.

According to the results, the population size has an important impact on performance. Better solutions could be acquired with the increase in population size, however, more computation time is needed. The similar result can be acquired when *Ps* is 1.5 × TL, and it is also the most stable choice for the solution. Moreover, a trend can be observed that the lower *limit* shows better performance when *Ps* is larger, and vice versa. That means it needs more chances to avoid falling into local optimal when it has a larger population. For *limit*, eight is the most appropriate value. The algorithm is run 10 times for each value of *P with Ps* = 1.5 × TL and *limit* = 8. Furthermore, *P* is set as 0.8 according to the result in **Figure 6**.

4.2 Algorithm Performance Test

FAJSP-LS with the unequal and consistent batch size is a new problem, and there is no benchmark for testifying available. As the fundamental problem of FAJSP-LS, the FJSP benchmark can be used to verify the integrated performance for splitting and scheduling of the proposed algorithm. The 8 × 8 case from Kacem benchmark (Kacem et al., 2002) was used. Without batch splitting, the near-optimal result obtained by the proposed algorithm in this article is 560, the same as the best results obtained in former research (Yazdani et al., 2010; Lu et al., 2012). **Figure 7** shows the scheduling Gantt Chart.

Besides, 4 × 6 FJSP from Kacem is used to verify the algorithm's equal and unequal splitting performance, respectively. The results are shown in **Table 2**.

The proposed algorithm performs well in both equal and unequal-size splitting compared with the above studies. Better performance can be obtained with a smaller number of sub-batches. **Figures 8** and **9** are Gantt charts of equal size and unequal-size scenarios, respectively.

4.3 Case Analysis

The proposed algorithm is applied to a confirmed case of a refrigerator shell parts production workshop, a typical flexible job-shop combined with an assembly stage. The flexible job-shop produced the upper bar A and B, the lower bar A and B, the U-shell A and B, the backplate A and B, the front shell A, the left plate A, and the right plate A. The job set is denoted as

TABLE 5 | 4 × 6 lot-splitting scheduling results comparison.

Scenarios	Without assembly constraints			With assembly constraints		
	Average maximum completion time	Average flow time	Optimal maximum completion time	Average maximum completion time	Average flow time	Optimal maximum completion time
Full batch	16848	12590.1	16700	17544	14550.3	17280
Batch splitting (equal size)	15712	11823.6	15604	16361	13754.3	16147
Batch splitting (unequal size)	15433	10578.3	15320	15433	12102.7	15320
Improvement (with full batch)	8.40%	15.98%	8.26%	12.03%	16.82%	11.34%

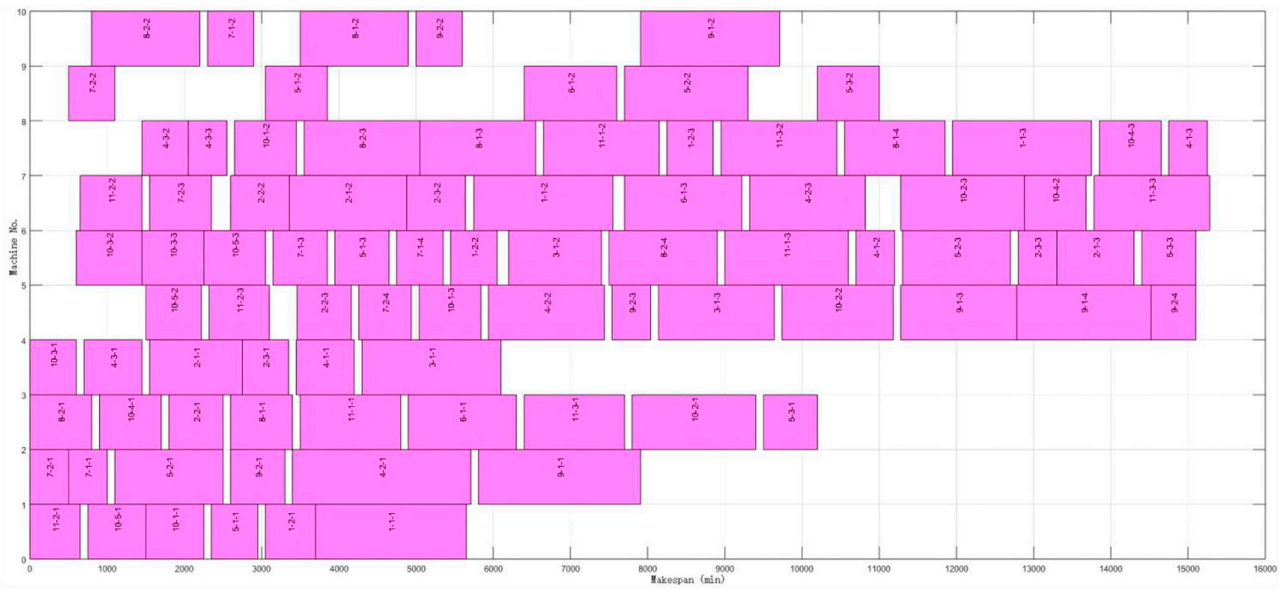

FIGURE 10 | Gantt chart of lot-splitting scheduling with assembly constraints.

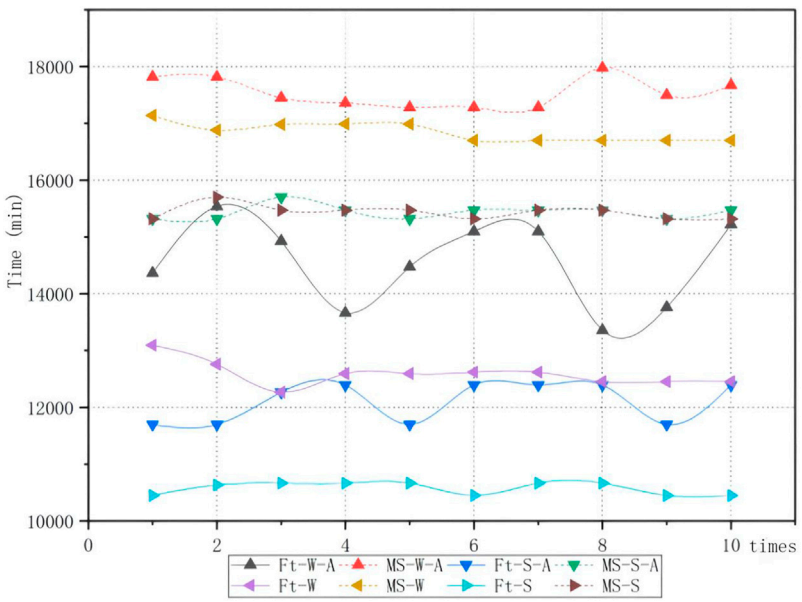

FIGURE 11 | Comparison of different scenarios.

$J = \{J_i\}, i = 1, 2...11$. Machine set is denoted as $M = \{M_j\}, j = 1, 2...10$. The jobs with assembly constraints are (J_1, J_2), (J_3, J_4), and (J_{10}, J_{11}). The minimum size of the sub-batch is $Lp = 100$, and the preparation time is 100. **Table 3** shows the production plan. **Table 4** is the processing time matrix of operations on each machine.

The proposed algorithm was applied to solve the problem in four scenarios classified by whether to split the job and append the assembly stage. The results are shown in **Table 5**. It can be found that the average flow time and makespan increase significantly when the assembly is appended. However, compared with the full batch production, batch splitting can effectively reduce the makespan and average flow time, and average efficiency can be improved by 12.14%. At the same time, the optimal makespan can be acquired even considering the assembly constraints. The Gantt chart of the optimal solution is

shown in **Figure 10**. A comparison of experimental results is shown in **Figure 11**. Ft indicates that the optimization objective is an average flow time. MS indicates that the optimization objective is makespan. A stands for the assembly constraint. W and S represent scheduling with whole or splitting batches, respectively. At last, the Ft-S-A is the problem with the assembly stage and batch splitting.

4.4 Discussion

In our study, an improved algorithm based on an ABC was proposed to solve the FAJSP-LS. To the best of our knowledge, it is the first time to use the multipopulation collaboration mechanism of ABC to solve this two-stage integrated optimization problem. In examining the performance, first, the benchmark case with the minimization of makespan as the objective is used to show the optimization power on classic FJSP. Second, another modified benchmark case is used to show the algorithm's equal and unequal splitting performance respectively. It shows unequal-size batch splitting works better than equal-size batch splitting under the flexibility situation. In addition, the proposed algorithm shows better performance on makespan compared with algorithms mentioned in the literature, because the design of a multipopulation collaboration mechanism that scouts the bee colony is in charge of batch splitting that has lower search space, and employed bee colony is in charge of batch scheduling that has a greater search space, and onlooker bee is in charge of maintaining the better solutions combined with these two subproblems. Third, the algorithm is applied to the real case in a refrigerator shell parts production workshop that produces jobs in a full batch. The experiment shows a positive impact on average flowtime and makespan by adopting LS. Compared with the original strategy, makespan can be improved by 12.03%, and average flowtime can be improved by 16.82%, significantly reducing WIP in the production and increasing the production efficiency. For this real case, unequal-size splitting also shows better performance than equal-size splitting when considering flexibility and the assembly stage.

5 CONCLUSION

This study examined an extension problem of FJSP and presented a flexible assembly job-shop scheduling problem with unequal and consistent size batch splitting, in brief, FAJSP-LS. This model comprises two subproblems: batch splitting and batch scheduling. An improved algorithm considering the multipopulation collaboration mechanism of an ABC was proposed to solve it, and it shows good performance on the problems mentioned above. Overall, it is recommended to adopt unequal-size splitting in FAJSP to optimize time-correlated objectives such as average flowtime and makespan. Furthermore, the inherent properties of multipopulation in ABC allow algorithm to solve two-stage problem without adopting distributions rules or other algorithms. A limitation of the research is that it focuses on the upper stage of the whole structure in **Figure 1**, and only considers the time-correlated objectives. Once the downstream assembly line stage is considered, more factors such as level scheduling that consider average demand for each component, due date on the specific workstation, transfer cost, and energy-consuming cost should be studied in the subsequent research. Furthermore, more bioinspired algorithms will be studied to optimize these kinds of multistage problems, and more neighborhood searching methods will be adopted to further the current algorithm.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material. Further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

XL structured the research plan and proposed the model of FAJSP-LS and the improved algorithm, JL supported the case of the refrigerator factory and provided help with data analysis, CY drafted part of the manuscript, and JW provided simulation experiments.

FUNDING

The Humanity and Social Science Foundation of Ministry of Education of China (19YJC630086), Zhejiang Provincial Natural Science Foundation (No. LQ14E050001).

REFERENCES

Ba, L., Li, Y., Cao, Y., Yang, M. S., and Liu, Y. (2015). Research on Flexible Job Shop Scheduling Problem Considering Batch Assembly. *China Mech. Eng.* 26 (23), 3200–3207. doi:10.3969/j.issn.1004-132X.2015.23.014

Bai, J. J., Gong, Y. J., Wang, N. S., and Tang, D. B. (2010). Batch Scheduling Optimization of Multi-Objective Flexible Job Shop. *Computer-integrated Manuf. Syst.* 16 (02), 396–403. doi:10.1063/1.3442632

Benjaafar, S., and Ramakrishnan, R. (1996). Modelling, Measurement and Evaluation of Sequencing Flexibility in Manufacturing Systems. *Int. J. Prod. Res.* 34 (5), 1195–1220. doi:10.1080/00207549608904961

Chan, F. T. S., Wong, T. C., and Chan, L. Y. (2008). Lot Streaming for Product Assembly in Job Shop Environment. *Robotics Computer-Integrated Manuf.* 24 (3), 321–331. doi:10.1016/j.rcim.2007.01.001

Gong, G., Chiong, R., Deng, Q., and Gong, X. (2020). A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling with Worker Flexibility. *Int. J. Prod. Res.* 58 (14), 4406–4420. doi:10.1080/00207543.2019.1653504

Kacem, I., Hammadi, S., and Borne, P. (2002). Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems. *IEEE Trans. Syst. Man. Cybern. C* 32 (1), 1–13. doi:10.1109/TSMCC.2002.1009117

Kalir, A. A., and Sarin, S. C. (2000). Evaluation of the Potential Benefits of Lot Streaming in Flow-Shop Systems. *Int. J. Prod. Econ.* 66 (2), 131–142. doi:10.1016/S0925-5273(99)00115-2

Karaboga, D., and Akay, B. (2009). A Comparative Study of Artificial Bee Colony Algorithm. *Appl. Math. Comput.* 214 (1), 108–132. doi:10.1016/j.amc.2009.03.090

Karaboga, D., Gorkemli, B., Ozturk, C., and Karaboga, N. (2014). A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications. *Artif. Intell. Rev.* 42 (1), 21–57. doi:10.1007/s10462-012-9328-0

Kheirandish, O., Tavakkoli-Moghaddam, R., and Karimi-Nasab, M. (2015). An Artificial Bee Colony Algorithm for a Two-Stage Hybrid Flowshop Scheduling Problem with Multilevel Product Structures and Requirement Operations. *Int. J. Comput. Integr. Manuf.* 28 (5), 437–450. doi:10.1080/0951192X.2014.880805

Li, X. L., Fu, P. H., Lu, J. S., and Li, J. (2017). Research on the Serial-Parallel Assembly Line Association Sequencing Problem Based on Artificial Bee Colony Optimization. *Computer-integrated Manuf. Syst.* 23 (3), 567–574. doi:10.13196/j.cims.2017.03.014

Li, X. L., Lu, J. S., Chai, G. Z., and Tang, H. T. (2011). Hybrid Bee Colony Algorithm for Flexible Job Shop Scheduling Problem. *Computer-integrated Manuf. Syst.* 17 (07), 1495–1500. doi:10.13196/j.cims.2011.07.153.lixl.020

Li, Y., Huang, W., Wu, R., and Guo, K. (2020). An Improved Artificial Bee Colony Algorithm for Solving Multi-Objective Low-Carbon Flexible Job Shop Scheduling Problem. *Appl. Soft Comput.* 95, 106544. doi:10.1016/j.asoc.2020.106544

Lin, W., Deng, Q., Han, W., Gong, G., and Li, K. (2022). An Effective Algorithm for Flexible Assembly Job-shop Scheduling with Tight Job Constraints. *Intl. Trans. Op. Res.* 29, 496–525. doi:10.1111/itor.12767

Low, C., Hsu, C.-M., and Huang, K.-I. (2004). Benefits of Lot Splitting in Job-Shop Scheduling. *Int. J. Adv. Manuf. Technol.* 24 (9), 773–780. doi:10.1007/s00170-003-1785-9

Lu, H. D., He, W. P., Zhou, X., and Li, Y. J. (2012). Flexible Job Shop Batch Scheduling Based on Tabu Search. *J. Shanghai Jiaot. Univ.* 46 (12), 2003–2008. doi:10.16183/j.cnki.jsjtu.2012.12.024

Martin, C. (2009). A Hybrid Genetic Algorithm/mathematical Programming Approach to the Multi-Family Flowshop Scheduling Problem with Lot Streaming. *Omega* 37 (1), 126–137. doi:10.1016/j.omega.2006.11.002

Meng, T., Pan, Q.-K., and Sang, H.-Y. (2018). A Hybrid Artificial Bee Colony Algorithm for a Flexible Job Shop Scheduling Problem with Overlapping in Operations. *Int. J. Prod. Res.* 56 (16), 5278–5292. doi:10.1080/00207543.2018.1467575

Nourali, S., Imanipour, N., and Shahriari, M. R. (2012). A Mathematical Model for Integrated Process Planning and Scheduling in Flexible Assembly Job Shop Environment with Sequence Dependent Setup Times. *Int. J. Math. Analysis* 6 (41-44), 2117–2132.

Novas, J. M. (2019). Production Scheduling and Lot Streaming at Flexible Job-Shops Environments Using Constraint Programming. *Comput. Industrial Eng.* 136, 252–264. doi:10.1016/j.cie.2019.07.011

Sun, Z. J., An, J., and Huang, W. Q. (2008). Batch Operation Plan Optimization of Multi-Process Routes in Workshop. *China Mech. Eng.* 19 (02), 183–187. doi:10.3321/j.issn:1004-132X.2008.02.014

Trietsch, D., and Baker, K. R. (1993). Basic Techniques for Lot Streaming. *Operations Res.* 41 (6), 1065–1076. doi:10.1287/opre.41.6.1065

Wong, T. C., and Ngan, S. C. (2013). A Comparison of Hybrid Genetic Algorithm and Hybrid Particle Swarm Optimization to Minimize Makespan for Assembly Job Shop. *Appl. Soft Comput.* 13 (3), 1391–1399. doi:10.1016/j.asoc.2012.04.007

Wu, X., Liu, X., and Zhao, N. (2019). An Improved Differential Evolution Algorithm for Solving a Distributed Assembly Flexible Job Shop Scheduling Problem. *Memetic Comp.* 11 (4), 335–355. doi:10.1007/s12293-018-00278-7

Xie, Y., Gui, F.-X., Wang, W.-J., and Chien, C.-F. (2022). A Two-Stage Multi-Population Genetic Algorithm with Heuristics for Workflow Scheduling in Heterogeneous Distributed Computing Environments. *IEEE Trans. Cloud Comput.* 2022, 1. doi:10.1109/TCC.2021.3137881

Xie, Y., Sheng, Y., Qiu, M., and Gui, F. (2022). An Adaptive Decoding Biased Random Key Genetic Algorithm for Cloud Workflow Scheduling. *Eng. Appl. Artif. Intell.* 112, 104879. doi:10.1016/j.engappai.2022.104879

Xu, B. Y., Fei, X. L., and Zhang, X. L. (2016). Batch Division and Parallel Scheduling Optimization of Flexible Job Shop. *Computer-integrated Manuf. Syst.* 22 (08), 1953–1964. doi:10.13196/j.cims.2016.08.014

Yazdani, M., Amiri, M., and Zandieh, M. (2010). Flexible Job-Shop Scheduling with Parallel Variable Neighborhood Search Algorithm. *Expert Syst. Appl.* 37 (1), 678–687. doi:10.1016/j.eswa.2009.06.007

Zeng, C. F., Liu, J. J., Chen, Q. X., and Mao, N. (2019). Batch Scheduling of Assembly Workshop for Optimizing Delivery Reliability. *Ind. Eng.* 22 (06), 45–56. doi:10.3969/j.issn.1007-7375.2019.06.007

Zhang, S., Li, X., Zhang, B., and Wang, S. (2020). Multi-objective Optimisation in Flexible Assembly Job Shop Scheduling Using a Distributed Ant Colony System. *Eur. J. Operational Res.* 283 (2), 441–460. doi:10.1016/j.ejor.2019.11.016

Zhang, S., and Wang, S. (2018). Flexible Assembly Job-Shop Scheduling with Sequence-dependent Setup Times and Part Sharing in a Dynamic Environment: Constraint Programming Model, Mixed-Integer Programming Model, and Dispatching Rules. *IEEE Trans. Eng. Manage.* 65 (3), 487–504. doi:10.1109/TEM.2017.2785774

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.