Parallel batch processing machines scheduling for SLM additive manufacturing in cloud manufacturing

Shuangshuang Hong, Xiuli Wang*

School of Economics and Management, Nanjing University of Science and Technology, Nanjing, Jiangsu 210000, China

*Corresponding author. Email: wangdu0816@163.com

Abstract. For the scheduling problem of parallel batch processing machines for Selective Laser Melting (SLM) additive manufacturing in cloud manufacturing environments, an Improved Discrete Artificial Bee Colony algorithm(IABC) is proposed by considering the service time of the product as well as introducing the mathematical model of the problem. Firstly, a population initialization strategy is introduced to improve the quality of initial solutions and accelerate the convergence of the population.Secondly, through three neighborhood search operators, the transition from the original nectar source to the new nectar source is achieved for employed bees and following bees. Furthermore, a local search strategy and a modified abandonment pattern are proposed to increase population diversity and prevent the algorithm from falling into local optima.Finally, the performance of the algorithm is analyzed, and the IABC algorithm was applied to 36 examples. The effectiveness and advantages of IABC in solving the studied problem are verified through a large number of simulation experiments.

Keywords: Additive manufacturing; Batch processing machines; Cloud manufacturing; Artificial Bee Colony algorithm.

1. Introduction

Additive manufacturing (AM) is a technology that builds solid parts layer by layer using material incrementally based on CAD design data, which has the characteristics of distributed manufacturing and digital manufacturing, and is suitable for small batch, complex shape, and customized order tasks. Cloud manufacturing is a new production model with advantages such as resource integration, efficient services, and multi-party benefits. It provides an open environment for the efficient integration and sharing of additive manufacturing equipment resources.

In recent years, researchers have been studying scheduling problems in additive manufacturing. Some studies have discussed job formation and determining the orientation of parts on AM machines [1,2,3,4]. The scheduling problem of additive manufacturing unrelated parallel batch processing machines was first introduced by Kucukkoc et al.[5], followed by Li et al. [6] who developed a mixed integer mathematical model to determine the optimal allocation of parts on a group of machines with different specifications. Kucukkoc [7] considered different machine configurations. Li et al. [8] then studied the problem of minimizing job completion time on a single batch processing machine with 2D packing constraints, representing the machine platform as a 2D rectangle, where each job occupies the size of a rectangle. Kim et al. [9] and Toksarı et al. [10] considered multi-material part production, aiming to directly minimize the maximum completion time of multi-material part production plans and indirectly minimize material changeovers. Chergui et al. [11] and Rohaninejad et al. [12] further considered customer order due date problems. Aloui et al. [13] proposed two models, with the objective of minimizing total delay, where the shape of nested parts was projected based on partial dimensions in horizontal and vertical directions.

However, most of the literature sets the scheduling problem of additive manufacturing batch processing machines in a centralized production environment where machines are located in the same geographical location. For the scheduling problem of additive manufacturing in cloud manufacturing environments, only a few studies have explored this issue [14,15,16]. Therefore, this paper

investigates the comprehensive scheduling problem of production and delivery for parallel batch processing machines based on SLM in cloud manufacturing environments.

In the literature on parallel batch processing machine scheduling, heuristic algorithms have been developed [17,18]. The artificial bee colony (ABC) algorithm [19]has advantages such as few control parameters, simple operations, and easy implementation. However, research on the application of ABC in distributed scheduling is still in its infancy [20]. Due to the significant characteristics of ABC and its advantages in solving scheduling problems, this paper attempts to develop an improved discrete artificial bee colony algorithm to address the scheduling problem of identical parallel batch processing machines for SLM additive manufacturing in a cloud manufacturing environment.

2. Problem description and modelling

2.1. Problem description

The scheduling problem of identical parallel batches for SLM additive manufacturing in a cloud manufacturing environment, with the optimization goal of minimizing the maximum service time, can be represented as $DP_m|batch\{AM\}, p_j, s_j|S_{max}$. In the cloud manufacturing environment for identical parallel batch scheduling systems, cloud providers' factories are distributed across different geographical locations, with varying numbers of machines between different factories, but all machines are identical SLM devices with the same processing capabilities. Smax is the optimization goal of the problem, which refers to the service span, representing the total time composed of part processing time and transportation time. This paper simplifies the logistics delivery process of cloud manufacturers delivering orders to customers into logistics delivery time and assumes each order is delivered individually to the cloud customer after production is completed. The schematic diagram of the problem is shown in Fig. 1.

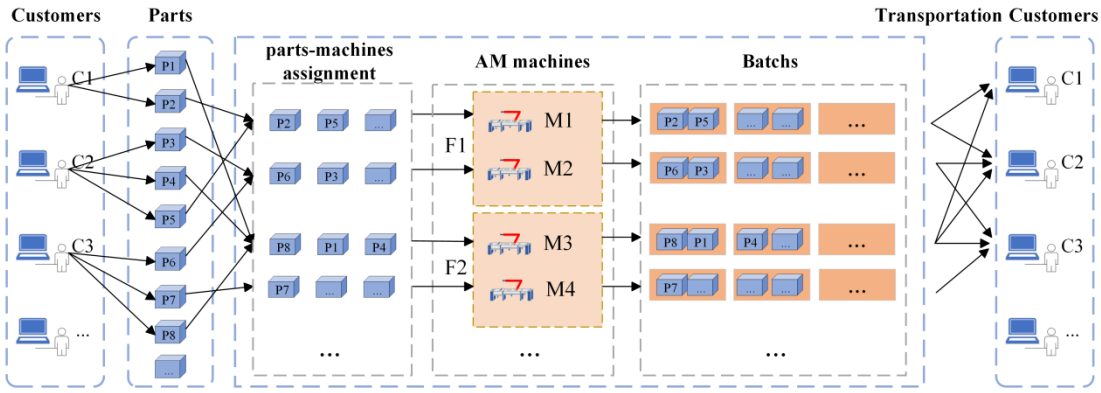


Fig. 1 An illustration of $DP_m|batch\{AM\}, p_j, s_j|S_{max}$

2.2. Formulation

2.2.1 Assumptions

To establish the problem model, this paper makes the following assumptions:

(1) Each AM machine can process one batch at a time. Prior purchasing rights are not allowed.

The orientation of parts is predetermined, with allowance for a 90-degree rotation around the Z-axis on the building plane.

(2) All parts are made of a single material.

(3) Assuming that all parts can be produced on the machine.

(4) Parts within the same batch are horizontally placed in the build chamber tray, and stacking between parts is not allowed.

(5) All parts are sliced to the same thickness to determine machine parameters.

(6) The geometry of each part is projected on the XY plane in order to determine the minimum rectangle limits.

(7) All parts are available at time zero.
(8) Each batch requires a setup time independent of the sequence.
(9) Preemption is not allowed; that is, batches that have been started cannot be interrupted.

2.2.2 Symbol definitions

The notation used are as follows:

Sets

M:Set of all available machines
B:Set of all batches
I:Set of parts to be processed

Parameters

L:The effective length of the build platform for machine
W:The effective width of the build platform for machine
H:The effective height of the build platform for machine
X_m:The geographical location's horizontal coordinate of machine *m*
Y_m:The geographical location's vertical coordinate of machine *m*
VT:Time for forming a unit of volume
HT:Time for powder layering a unit height
SET:Setup time
v_i:Material volume of part *i*
h_i:Height of part *i*
l_i:Length of part *i*
w_i:Width of part *i*
lx_i:The geographical location's horizontal coordinate of part *i*
ly_i:The geographical location's vertical coordinate of part *i*
φ:A very large positive number
α:Logistics time per unit distance

Decision variables

x_{ibm}:Binary variable that equals 1 if part *i* is assigned to batch *b* on machine *m*, otherwise 0.
h_{bm}:Height of batch *b* on machine *m*
p_{bm}:Processing time of batch *b* on machine *m*
C_{bm}:Completion time of batch *b* of the machine
S_i:Service time required for part *i*
C_i:Construction completion time of part *i*
S_{max}:Maximum service time of part *i*
x_i:Horizontal coordinate of the bottom left corner of part *i*
y_i:The vertical coordinate of the bottom left corner of part *i*
R_i:Binary variable that equals 1 if part *i* is rotated by 90 degrees, otherwise 0.
PL_{ij}^{bm}:Binary variable that equals 1 if part *i* is placed to the left of part *j* in batch *b* on machine *m*, otherwise 0.
PB_{ij}^{bm}:Binary variable that equals 1 if part *i* is placed below part *j* in batch *b* on machine *m*, otherwise 0.
z_{bm}:Binary variable that equals 1 if there is a part placed on batch *b* of machine *m*, otherwise 0.

2.2.3 mathematical model

The mathematical model of the problem is as follows:

$$\text{Min obj} = S_{max} \tag{1}$$

$$\sum_{m\in M}\sum_{b\in B} x_{ibm} = 1 \quad \forall i \in I \tag{2}$$

$$x_j + l_j R_j + w_j(1 - R_j) \le x_{jbm} L + \varphi(1 - x_{jbm}) \quad \forall j \in I, \forall b \in B, \forall m \in M \tag{3}$$

$$y_j + w_j R_j + l_j(1 - R_j) \le x_{jbm} W + \varphi(1 - x_{jbm}) \quad \forall \mathrm{j} \in I, \forall b \in B, \forall m \in M \tag{4}$$

$$x_i + l_i R_i + w_i(1 - R_i) \le x_j + \varphi(1 - PL_{ij}^{bm}) \quad \forall \mathrm{i,j} \in \mathrm{I, i} \ne \mathrm{j}, \forall \mathrm{b} \in \mathrm{B}, \forall \mathrm{m} \in \mathrm{M} \tag{5}$$

$$y_i + w_i R_i + l_i(1 - R_i) \le y_j + \varphi(1 - PB_{ij}^{bm}) \quad \forall \mathrm{i,j} \in \mathrm{I, i} \ne \mathrm{j}, \forall \mathrm{b} \in \mathrm{B}, \forall \mathrm{m} \in \mathrm{M} \tag{6}$$

$$PL_{ij}^{bm} + PB_{ij}^{bm} + PL_{ji}^{bm} + PB_{ji}^{bm} \ge 3 - (x_{ibm} + x_{jbm}) - 2(1 - x_{ibm}) - 2(1 - x_{jbm}) \quad \forall \mathrm{i,j} \in \mathrm{I, i} \ne \mathrm{j}, \mathrm{b} \in \mathrm{B}, \mathrm{m} \in \mathrm{M} \tag{7}$$

$$\sum_{i \in I} x_{ibm} \le \varphi z_{bm} \quad \forall \mathrm{b} \in \mathrm{B}, \forall \mathrm{m} \in \mathrm{M} \tag{8}$$

$$\sum_{i \in I} x_{ibm} \ge z_{bm} \quad \forall \mathrm{b} \in \mathrm{B}, \forall \mathrm{m} \in \mathrm{M} \tag{9}$$

$$h_{bm} \le H \quad \forall \mathrm{b} \in \mathrm{B}, \forall m \in \mathrm{M} \tag{10}$$

$$h_i \cdot x_{ibm} \le h_{bm} \quad \forall i \in I, \forall \mathrm{b} \in \mathrm{B}, \forall m \in \mathrm{M} \tag{11}$$

$$\sum_{i \in I} x_{i(b+1)m} \le \varphi \cdot \sum_{i \in I} x_{ibm} \quad \forall m \in \mathrm{M}, \forall \mathrm{b} \in \{1, \ldots, \mathrm{B} - 1\} \tag{12}$$

$$\mathrm{S_{max}} \ge \mathrm{S_i} \quad \forall i \in I \tag{13}$$

$$p_{bm} \ge VT \sum_{i \in I} v_i \cdot x_{ibm} + HT \cdot h_{bm} \quad \forall \mathrm{b} \in \mathrm{B}, \forall m \in \mathrm{M} \tag{14}$$

$$C_{0m} = 0 \quad \forall \mathrm{m} \in \mathrm{M} \tag{15}$$

$$C_{bm} \ge C_{(b-1)m} + p_{bm} + SET z_{bm} \quad \forall m \in \mathrm{M}, \forall \mathrm{b} \in \mathrm{B} \tag{16}$$

$$C_i \ge C_{bm} - \varphi(1 - x_{ibm}) \quad \forall i \in I, \forall \mathrm{b} \in \mathrm{B}, \forall m \in \mathrm{M} \tag{17}$$

$$S_i \ge C_i + \sum_{b \in B} x_{ibm} * \frac{\sqrt{(lx_i - X_m)^2 + (ly_i - Y_m)^2}}{\alpha} \quad \forall i \in I, \forall m \in \mathrm{M} \tag{18}$$

$$C_i 、\ S_i 、\ x_i 、\ y_i \ge 0 \quad \forall i \in I \tag{19}$$

$$C_{bm} 、\ p_{bm} 、\ h_{bm} \ge 0 \quad \forall m \in \mathrm{M}, \forall \mathrm{b} \in \mathrm{B} \tag{20}$$

$$x_{ibm} 、\ z_{bm} 、\ PL_{ij}^{bm} 、\ PB_{ij}^{bm} 、\ R_i \in \{0,1\} \tag{21}$$

The objective function, (1) , minimizes the maximum service time of parts. Constraints (2) ensures that each part is assigned to some batch b on some machine m. Constraints (3) and (4) ensure that the dimensions of the building platforms are not exceeded. Constraints (5) and (6) guarantee that parts assigned to the same batch do not overlap. Constraints (7)guarantee that if two parts are placed in the same batch, they must necessarily be placed one after other. Constraints (8) and (9) ensure that a batch on machine m exists if there are parts placed on it. Constraints (10) ensures that the batch height does not exceed the height of the corresponding machine. Constraints (11) sets the batch height to be greater than or equal to the height of each part assigned to it. Constraints (12) implies that batches must be formed continuously. For instance, after the formation of batch 1, batch 3 cannot start its operation unless batch 2 is formed. Constraints (13) ensures that the service time of part *i* is greater than or equal to the sum of its build time and transportation time. Constraints (14) sets a lower bound for the maximum service time. Constraints (15) computes the production time for each batch on machine *m*. Constraints (16) determines that the production of the first batch on each machine can start at time 0. Constraints (17) calculates the completion time of other batches, considering the completion time of their preceding batches and the required setup time. Constraints (18) is used to obtain the service time for each part. Constraints (19), (20), and (21) define the nature of each decision variable.

3. Problem Solving Based on Artificial Bee Colony Algorithm

3.1. Population Encoding

In the encoding process, the decision variables to be considered are the machine allocation problem and the determination of the machining sequence of parts on each machine. This paper adopts a dual-layer encoding method to represent the solution to the optimization problem, referring to the encoding method in Reference [21]. The first line represents the part sequence, composed of the identification numbers of each part. The second line represents the machine sequence, determining the machine identification number for the corresponding parts in the first line. Fig. 2 illustrates a solution to a problem with 10 parts and 2 machines.

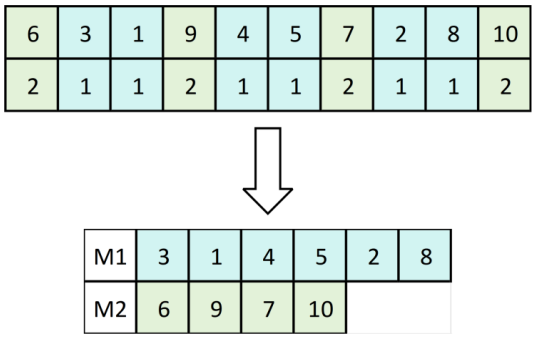


Fig. 2 An illustration of dual-layer encoding

3.2. Population Initialization

In the scheduling of additive manufacturing, existing research has shown that producing parts with similar heights in the same batch can greatly improve production efficiency[25]. Therefore, the height difference between parts is considered as a factor in the initialization strategy. Parts are sorted based on the height decrease criterion, generating partial initial solutions.

3.3. Heuristic placement strategy

This paper adopts a best-fit heuristic based on skyline, where skyline is a dynamically updated structure that describes the layout shape. In the best-fit heuristic based on skyline algorithm, the first step is to select the bottom-left skyline segment *s* as the candidate segment to place a rectangle. Then, using our evaluation function, as shown in Fig. 3, the best-fitting rectangle is selected and placed at *s*. In the heuristic algorithm, it is necessary to select the rectangle with the maximum fitness value. If there is a tie, the first unplaced rectangle is chosen. When calculating the maximum fitness value of a rectangle, rotation by 90 degrees is allowed, i.e., exchanging the length and width of the rectangle. If, under the condition of allowing rotation by 90 degrees, the width or length of all unplaced rectangles is greater than the width of *s*, *s* will be raised to the height of its lower adjacent skyline and merged with it.

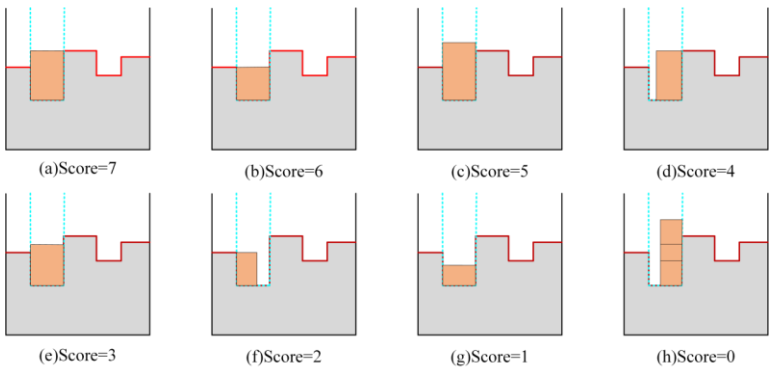


Fig. 3 scoring rule

3.4. The Employed Bee Phase

This section adopts three neighborhood operators to directly manipulate the encoding vectors. The adjacent exchange operator randomly swaps the positions of two adjacent elements in the part encoding row, as shown in Fig. 4. The non-adjacent exchange operator operates on both the part encoding row and the machine encoding row, randomly swapping the positions of two non-adjacent elements to obtain a new legal encoding vector, as shown in Fig. 4 and Fig. 5. The Insert operator randomly deletes an element from the machine encoding row and inserts it into another position, as shown in Fig. 5. Apply these three neighborhood operators to the encoding vectors of the nectar source *x*, compare the generated candidate solutions with the best solution obtained so far, if the best solution in the candidate solutions is better than the original solution of *x*, the original nectar source *x* is replaced with the new nectar source; otherwise, the original nectar source *x* remains unchanged and the exploitation degree is incremented.

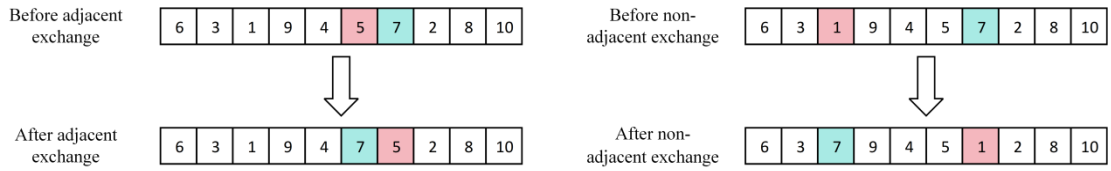


Fig. 4 An illustration of adjacent and non adjacent exchange operators for part sequences

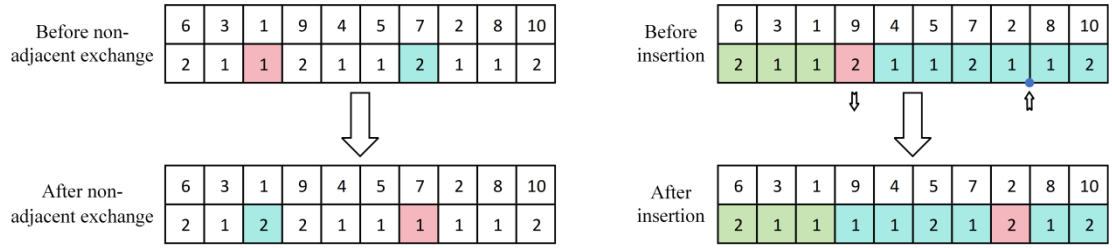


Fig. 5 An illustration of non adjacent exchange and insertion operators in machine sequences

3.5. The Onlooker Bee Phase

This paper adopts a roulette wheel selection strategy for the scout bees to choose high-quality nectar sources. The probability of selecting nectar source X_i is proportional to its fitness value $f(X_i)$, and can be expressed as:

$$\text{prob}(X_i) = \frac{f(X_i)}{\sum_{i \in N} f(X_i)} \tag{22}$$

Where N represents the number of nectar sources. Typically, the number of employed bees and the number of onlooker bees are equal to the number of nectar sources.

3.6. The Scout Bee Phase

If the objective value of the current individual does not improve after a continuous limit of iterations, it is replaced by a randomly generated scout bee. However, because a good food source has more opportunities to be selected and further exploited by the roulette wheel selection method, this abandonment mode may lead to a good food source being more likely to be abandoned. Therefore, a modified abandoning mode is proposed: record the best candidate solution when the exploitation degree *L* of an individual reaches the limit, and compare it with the individual with the lowest fitness in the population. If the best candidate solution is better than that individual, abandon the individual and replace it with the best candidate solution to provide more development opportunities near that food source.

3.7. Local Search Strategy

Due to the significant impact of load balancing on workshop efficiency, a further local greedy search algorithm is set in this chapter to find a better solution. In the local search, first, the part *i* with the longest service time is selected, removed from the original printing machine *m*, and then a random machine *m'* is selected from the machine numbers except the original printing machine. Part *i* is moved to machine *m'* to obtain a new solution. The framework of IABC is shown in Fig. 6.

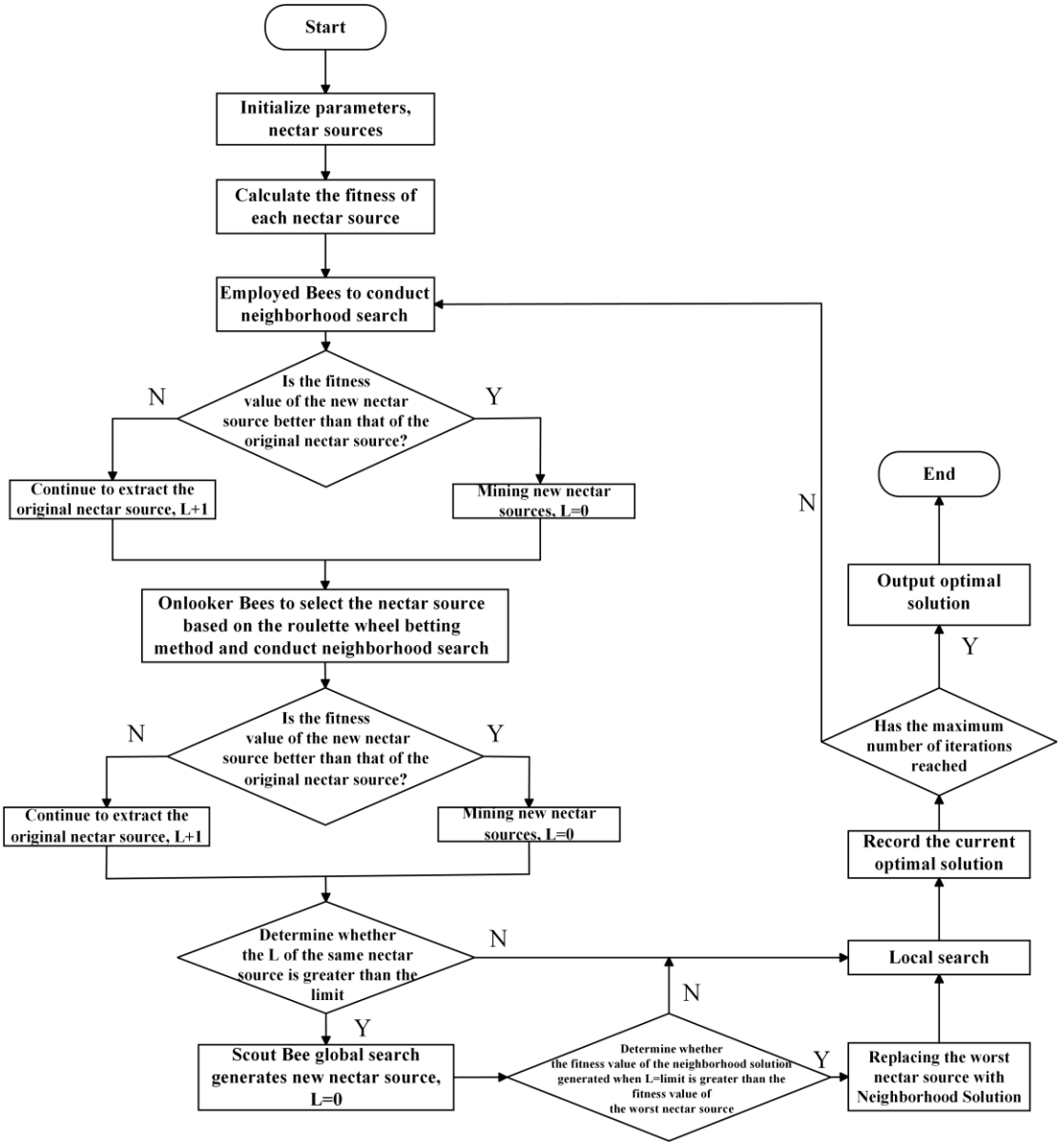


Fig. 6 Framework of IABC

4. Experimental Simulation and Analysis

4.1. Experimental Design

All program codes in this chapter are implemented and tested using Python 3.6 software. The program testing environment is as follows: Operating System: 64-bit Windows 11, Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz, Memory: 8 GB. The SLM machine parameters used in this chapter refer to the equipment parameters in reference [7], and the specific machine parameters are shown in Table 1.

Table 1. Machine Parameter Information

VT(h/cm^3)	HT(h/cm)	L(mm)	W(mm)	SET(h)
0.030864	0.7	250	250	1.2

In view of the digital characteristic of AM, this study uses digital simulation to carry on the experimental verification. The dimensional data of the parts used in this chapter are randomly generated within the range of [1,200], with the unit of part size being mm. The number of machines in the factory is randomly generated within the range of {2,3,4}. The number of parts included in customer task orders is randomly generated within the range of {1,2,3,4,5}. In the x-y coordinate system, the geographical coordinates of the factory and customer locations are randomly generated within the range of [0,200], with the unit being km. The logistics factor is set to 20 km/h.

This section designs 36 instances of different scales to test the performance of the proposed algorithm. Table 2 describes the basic information of the instances. F_M_P is used to represent different instances, for example, F2_M4_P50 indicates that 50 parts are produced in two factories with a total of 4 machines.

Table 2. Instances Information

Instances	Factories	Machines	Instances	Parts
1,2,3,4,5,6	2	4	1,7,13,19,25,31	50
7,8,9,10,11,12	2	6	2,8,14,20,26,32	80
13,14,15,16,17,18	3	8	3,9,14,20,26,32	100
19,20,21,22,23,24	3	10	4,10,15,21,27,33	150
25,26,27,28,29,30	4	12	5,11,16,22,28,34	200
31,32,33,34,35,36	4	14	6,12,18,24,30,36	240

4.2. Experimental Results Analysis

In this section, we analyze the experimental results of IABC and ABC, as well as GA [23] and SA [24]. To ensure fair comparison, the parameter settings of the comparative algorithms are as follows: the original ABC algorithm is set with parameters identical to the IABC algorithm, while GA and SA algorithms use the same search methods for placement strategy as in this paper. Using the Orthogonal experimental method, Population size *pop* and maximum number of nectar source searches *limit* in the IABC algorithm are designed. It is found that the algorithm performs optimally when *pop*=90 and *limit*=50. For the 36 instances, each algorithm runs independently 5 times, and the average, maximum and minimum values of the objective function are calculated. Table 3 records the experimental results, with the better values highlighted in bold.

Table 3. Experimental Results of Instances

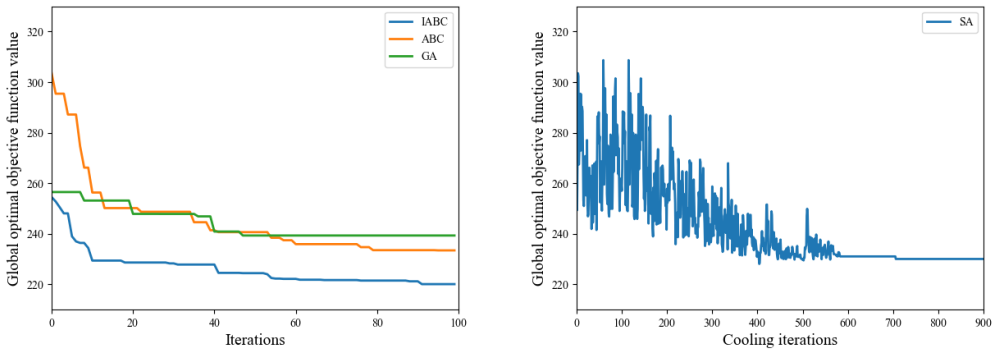
Instances	GA			SA			ABC			IABC		
	AVG	Max	Min	AVG	Max	Min	AVG	Max	Min	AVG	Max	Min
F2_M4_P50	184.4	191.0	179.6	177.8	181.1	174.1	184.4	185.7	182.8	**175.2**	176.1	**173.9**
F2_M4_P80	247.5	252.6	240.3	240.1	243.8	237.5	245.3	247.5	243.7	**237.1**	238.2	**235.9**
F2_M4_P100	338.7	345.5	333.5	331.2	334.3	328.2	334.7	337.4	332.8	**324.2**	326.0	**323.3**
F2_M4_P150	468.8	477.7	460.0	457.1	459.3	453.9	459.2	460.4	457.5	**444.9**	445.5	**443.7**
F2_M4_P200	631.0	650.9	616.5	619.7	621.4	618.1	620.1	622.8	617.8	**602.0**	602.6	**601.3**
F2_M4_P240	756.0	778.2	741.2	748.5	750.8	747.1	755.9	758.5	753.6	**730.8**	731.3	**730.6**
F2_M6_P50	134.4	142.4	127.1	124.9	128.1	120.6	127.5	128.8	126.4	**120.7**	121.7	**119.9**
F2_M6_P80	171.9	174.8	168.2	167.5	170.8	163.1	171.9	173.9	169.8	**162.5**	163.6	**161.2**
F2_M6_P100	248.1	269.9	234.3	236.5	239.2	234.8	234.7	237.7	232.8	**220.0**	220.8	**219.2**
F2_M6_P150	329.2	341.9	312.3	313.5	315.9	310.1	321.5	325.4	318.8	**301.3**	302.3	**300.3**
F2_M6_P200	435.3	441.4	423.3	418.2	420.4	416.3	430.0	432.9	427.3	**406.1**	406.9	**405.2**
F2_M6_P240	527.5	536.5	514.4	509.2	510.4	507.8	514.2	518.4	510.2	**492.0**	493.1	**490.9**

Instances	GA			SA			ABC			IABC		
	AVG	Max	Min	AVG	Max	Min	AVG	Max	Min	AVG	Max	Min
F3_M8_P50	112.4	117.5	109.3	99.7	105.9	**93.9**	106.4	110.4	103.5	**94.6**	94.8	94.2
F3_M8_P80	140.8	149.3	134.7	129.6	132.0	126.6	139.8	142.4	137.4	**125.5**	126.4	**124.5**
F3_M8_P100	199.4	210.7	189.4	179.6	183.5	175.3	184.3	187.4	180.4	**168.9**	169.6	**168.4**
F3_M8_P150	269.8	284.9	260.1	250.1	255.6	244.8	249.5	255.3	246.4	**229.8**	230.6	**229.0**
F3_M8_P200	349.8	358.5	338.2	318.7	322.9	312.1	333.1	338.4	327.4	**308.4**	308.7	**308.1**
F3_M8_P240	410.6	428.0	396.8	382.7	386.1	378.4	404.9	408.0	402.7	**372.4**	372.8	**371.7**
F3_M10_P50	91.9	96.3	87.8	84.4	89.3	78.8	90.6	93.2	88.4	**79.0**	79.4	**78.3**
F3_M10_P80	123.9	131.3	116.6	110.2	116.6	105.2	113.1	117.0	110.4	**103.8**	104.5	**102.8**
F3_M10_P100	171.2	178.6	164.7	154.2	165.3	143.4	157.0	163.0	152.1	**138.5**	139.5	**137.6**
F3_M10_P150	224.1	235.2	216.7	196.9	200.6	193.4	205.0	207.4	199.3	**186.5**	187.5	**185.2**
F3_M10_P200	298.5	309.9	286.5	266.1	273.4	261.4	275.5	279.4	270.4	**249.7**	250.4	**249.1**
F3_M10_P240	342.4	351.8	332.4	322.6	328.5	315.8	331.1	335.2	327.4	**300.8**	301.6	**300.4**
F4_M12_P50	86.0	90.6	82.1	77.7	80.7	73.0	82.1	85.5	79.2	**71.9**	71.9	**71.9**
F4_M12_P80	112.8	119.5	108.8	93.7	98.4	91.1	102.8	110.4	93.3	**89.1**	89.7	**88.5**
F4_M12_P100	155.5	168.5	145.7	131.9	136.8	128.3	140.7	145.3	138.1	**118.1**	118.6	**117.7**
F4_M12_P150	205.8	214.8	193.4	172.8	177.4	168.5	179.5	183.3	175.5	**157.9**	158.8	**157.3**
F4_M12_P200	258.3	268.3	251.2	228.6	234.8	220.2	239.5	245.3	230.5	**211.0**	212.1	**210.4**
F4_M12_P240	303.5	310.4	295.4	273.1	278.5	268.0	286.9	289.4	281.4	**253.5**	254.4	**252.3**
F4_M14_P50	82.5	88.5	74.5	75.0	77.9	**71.9**	77.6	79.4	74.1	**71.9**	71.9	**71.9**
F4_M14_P80	102.5	107.3	99.1	84.5	89.0	79.5	98.0	101.4	94.3	**79.1**	79.8	**78.3**
F4_M14_P100	134.2	138.3	128.1	115.9	117.7	113.9	129.3	133.3	125.4	**103.2**	103.8	**102.7**
F4_M14_P150	173.7	183.3	167.3	152.8	157.4	149.7	172.4	176.7	167.2	**137.7**	138.8	**137.0**
F4_M14_P200	232.6	248.9	221.3	198.7	205.5	193.1	216.3	222.5	213.5	**182.4**	182.8	**181.6**
F4_M14_P240	276.9	291.5	265.9	236.2	238.7	232.9	249.0	253.5	244.4	**219.4**	220.5	**218.8**

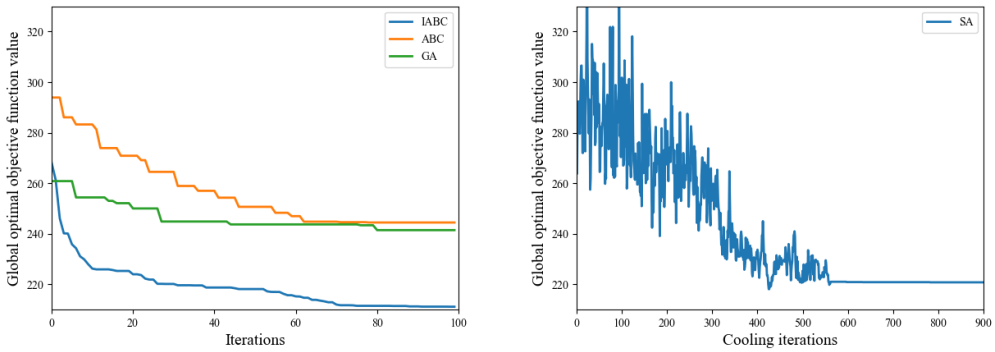
Based on the experimental data in Table 3, it can be observed that for all 36 instances, the average and maximum values obtained by the IABC algorithm are superior to the other three algorithms. The IABC algorithm achieves the best performance in terms of optimal value in 35 out of 36 instances. Among the comparative instances, the SA algorithm performs the best. For instance, in instance F3_M8_P50, the optimal value of the SA algorithm is 93.9, which is better than the optimal value of 94.2 obtained by the IABC algorithm. In instance F4_M14_P50, both the IABC algorithm and the SA algorithm achieve the same optimal value. These instances belong to medium-scale instances. In large-scale instances, the IABC algorithm consistently obtains the optimal value.

Furthermore, to fully consider the stability of the proposed algorithm, one example each from large and medium-scale instances is selected. The convergence speeds of the four algorithms are compared under 100 iterations, as shown in Fig. 7. Specifically, the small-scale instance selected is case F2_M6_P100, and the large-scale instance selected is case F4_M12_P200.

Regarding the convergence speed, it is evident from the solution curves in Figure 7 that the convergence speed of the IABC algorithm is significantly faster than the other three algorithms. Based on the convergence plots of these cases, it can be seen that compared to GA and SA algorithms, the proposed algorithm in this chapter can obtain better initial solutions. Moreover, within the same number of iterations, the proposed algorithm consistently yields higher-quality solutions. By observing these convergence plots, it can be noticed that the possibility of GA algorithm finding better solutions diminishes around 50 iterations, resulting in overly smooth convergence plots after 50 iterations. On the other hand, the IABC algorithm maintains good search capability from the beginning to the end of the iterations. Through these experimental results, it is demonstrated that the IABC algorithm proposed in this paper outperforms the other three algorithms in solving the problem. It exhibits strong search capability, as well as good stability and convergence in instances of different scales.



(a) Convergence Plot for Instance F2_M6_P100



(b) Convergence Plot for Instance F4_M12_P200

Fig. 7 Convergence Plot for Instances

Fig. 8 illustrates the Gantt chart corresponding to each machine in case F2_M4_P50. Each row in the chart represents a machine device, with the x-axis representing the processing time for batches. The continuous rectangles in each row represent different processing batches, with each rectangle indicating a production batch composed of multiple parts. The part numbers produced in each batch are marked within the corresponding rectangles. From the figure, it can be visually observed that the processing times on the four machines in the case are relatively close, indicating a balanced workload among the machines.

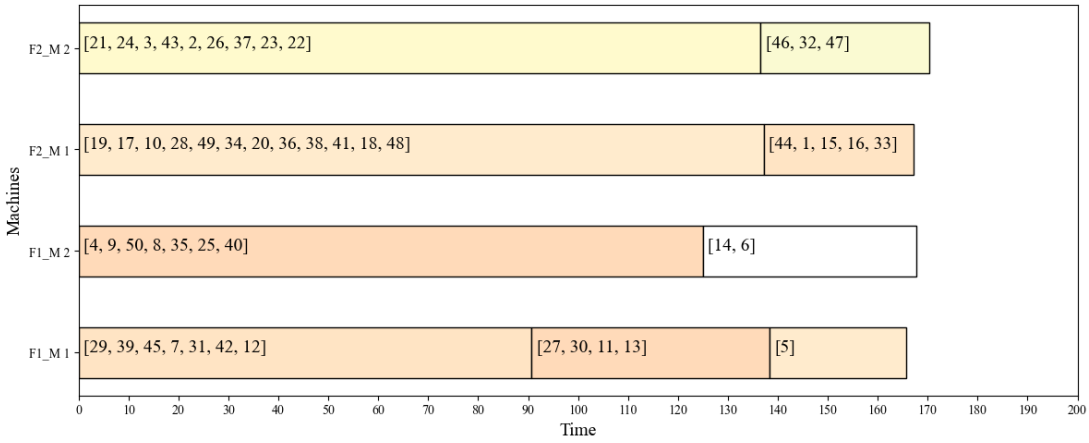


Fig. 8 Gantt Chart for Instance F2_M4_P50

5. Conclusion

To address the issues and deficiencies in the field of parallel batch processing machine scheduling in cloud manufacturing environments, we investigated $DP_m|batch\{AM\}, p_j, s_j|S_{max}$. We establish a mathematical model with the objective of minimizing the maximum service time and design an improved artificial bee colony algorithm for its solution. Firstly, to improve the initial solution quality of the algorithm, a criterion of decreasing part height is added during population initialization.Then, by introducing a dual-layer encoding to represent feasible solutions of the scheduling problem, overall optimization of the scheduling problem is achieved. In addition, a modified abandoning mode and local search strategy further improve the convergence accuracy and performance of the algorithm.Experimental results demonstrate that the proposed algorithm achieves higher solution quality and convergence speed when solving instances of different scales compared to other algorithms. Therefore, the IABC algorithm can effectively solve $DP_m|batch\{AM\}, p_j, s_j|S_{max}$.

This study only considered one type of the parallel machine scheduling problem that minimizes the maximum service time. In future research, more application scenarios should be considered, and the artificial bee colony algorithm should be further optimized to improve its optimization performance.

References

[1] Zhang Y, Bernard A, Harik R, et al. Build orientation optimization for multi-part production in additive manufacturing. Journal of Intelligent Manufacturing, 2017, 28: 1393-1407.

[2] Zhang Y, Bernard A, Harik R, et al. A new method for single-layer-part nesting in additive manufacturing. Rapid Prototyping Journal, 2018, 24(5): 840-854.

[3] Lee S J, Kim B S. Two-stage meta-heuristic for part-packing and build-scheduling problem in parallel additive manufacturing. Applied Soft Computing, 2023: 110132.

[4] Griffiths V, Scanlan J P, Eres M H, et al. Cost-driven build orientation and bin packing of parts in Selective Laser Melting (SLM). European Journal of Operational Research, 2019, 273(1): 334-352.

[5] Kucukkoc, Ibrahim, Qiang Li, and David Z. Zhang. Increasing the utilisation of additive manufacturing and 3D printing machines considering order delivery times. 19th International working seminar on production economics. 2016.

[6] Li Q, Kucukkoc I, Zhang D Z. Production planning in additive manufacturing and 3D printing. Computers & Operations Research, 2017, 83: 157-172.

[7] Kucukkoc I. MILP models to minimise makespan in additive manufacturing machine scheduling problems. Computers & Operations Research, 2019, 105: 58-67.

[8] Li X, Zhang K. Single batch processing machine scheduling with two-dimensional bin packing constraints. International Journal of Production Economics, 2018, 196: 113-121.

[9] Kim Y J, Kim B S. Part-group and build-scheduling with sequence-dependent setup time to minimize the makespan for non-identical parallel additive manufacturing machines. The International Journal of Advanced Manufacturing Technology, 2022: 1-12.

[10] Toksarı M D, Toğa G. Single batch processing machine scheduling with sequence-dependent setup times and multi-material parts in additive manufacturing. CIRP Journal of Manufacturing Science and Technology, 2022, 37: 302-311.

[11] Chergui A, Hadj-Hamou K, Vignat F. Production scheduling and nesting in additive manufacturing. Computers & Industrial Engineering, 2018, 126: 292-301.

[12] Rohaninejad M, Hanzálek Z, Tavakkoli-Moghaddam R. Scheduling of parallel 3D-printing machines with incompatible job families: A matheuristic algorithm. IFIP International Conference on Advances in Production Management Systems. Cham: Springer International Publishing, 2021: 51-61.

[13] Aloui A, Hadj-Hamou K. A heuristic approach for a scheduling problem in additive manufacturing under technological constraints. Computers & Industrial Engineering, 2021, 154: 107115.

[14] Rudolph J P, Emmelmann C. A cloud-based platform for automated order processing in additive manufacturing. Procedia Cirp, 2017, 63: 412-417.

[15] Qian C, Zhang Y, Liu Y, et al. A cloud service platform integrating additive and subtractive manufacturing with high resource efficiency. Journal of Cleaner Production, 2019, 241: 118379.

[16] Wu Q, Xie N, et al. Integrated cross-supplier order and logistic scheduling in cloud manufacturing. International Journal of Production Research, 2020, 60: 5, 1633-1649

[17] Chang, P.-Y., P. Damodaran, and S. Melouk. Minimizing Makespan on Parallel Batch Processing Machines. International Journal of Production Research 2004, 42:(19): 4211-4220.

[18] Zehetner D, Gansterer M. The collaborative batching problem in multi-site additive manufacturing. International Journal of Production Economics, 2022, 248: 108432.

[19] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[20] Yazdani M, Gohari S, Naderi B. Multi-factory parallel machine problems: Improved mathematical models and artificial bee colony algorithm. Computers and Industrial Engineering, 2015, 81:36-45.

[21] Rohaninejad M, Tavakkoli-Moghaddam R, Vahedi-Nouri B, et al. A hybrid learning-based meta-heuristic algorithm for scheduling of an additive manufacturing system consisting of parallel SLM machines. International Journal of Production Research, 2022, 60(20): 6205-6225.

[22] Baumers M, Beltrametti L, Gasparre A, et al. Informing additive manufacturing technology adoption: total cost and the impact of capacity utilisation. International Journal of Production Research, 2017, 55(23): 6957-6970.

[23] Zhang J, Yao X, Li Y. Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing. International Journal of Production Research, 2020, 58(8): 2263-2282.

[24] Yeh W C, Lai P J, Lee W C, et al. Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects. Information Sciences, 2014, 269: 142-158.