APEM journal
Advances in Production Engineering & Management
Volume 14 | Number 3 | September 2019 | pp 333–342
https://doi.org/10.14743/apem2019.3.331
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Multi-objective scheduling of cloud manufacturing resources through the integration of Cat swarm optimization and Firefly algorithm

Du, Y.[a,*], **Wang, J.L.**[b], **Lei, L.**[c]

[a]Anyang Institute of Technology, Anyang, P.R. China
[b]Guangdong Pharmaceutical University, Guangzhou, P.R. China
[c]Zhengzhou University, Zhenzhou, P.R. China

ABSTRACT

This paper attempts to minimize the makespan and cost and balance the load rate of the process scheduling of cloud manufacturing resources. For this purpose, a multiobjective scheduling model was established to achieve the minimal makespan, minimal cost and balanced load rate. Next, the cat swarm optimization (CSO) and the firefly algorithm (FA) were combined into a hybrid multi-objective scheduling algorithm. Finally, the hybrid algorithm was verified through CloudSim simulation. The simulation results show that the algorithm output the optimal scheduling plan in a short time. This research not only provides an effective way to find the global optimal solution, within the shortest possible time, to the process scheduling problem of cloud manufacturing resources with multiple objectives, but also promotes the application of swarm intelligence algorithms in job-shop scheduling problems.

ARTICLE INFO

Keywords:
Cloud manufacturing;
Multi-objective scheduling;
Cat swarm optimization (CSO);
Firefly algorithm (FA)

**Corresponding author:*
421319725@qq.com
(Du, Y.)

Article history:
Received 2 August 2019
Revised 8 September 2019
Accepted 10 September 2019

1. Introduction

Cloud manufacturing is a web-based, service-oriented intelligent manufacturing paradigm. This concept was proposed by Li Bohu, Academician of Chinese Academy of Engineering, in 2010, with the aim to fully integrate social manufacturing resources, improve resource utilization, lower manufacturing cost and respond faster to market demand. Cloud manufacturing combines such techniques as cloud computing, the Internet of Things (IoT), high-performance computing and intelligent science. In this way, manufacturing resources and capacity can be managed and scheduled in a centralized, uniform and intelligent manner, and the resources can be allocated and scheduled more effectively. In a word, cloud manufacturing pursues "the centralized management of scattered resources and the distribution services of centralized resources". The cloud manufacturing service platform provides users with the required manufacturing resources and their full lifecycle services in real time.

The process scheduling of cloud manufacturing resources depends on makespan, cost and load rate. Therefore, the cloud manufacturing task was decomposed into several processes, the basic units of scheduling. On this basis, a multi-objective scheduling model was established for the minimal makespan, minimal cost and balanced load rate. Meanwhile, a hybrid multi-objective scheduling algorithm was developed, coupling the cat swarm optimization (CSO) and the firefly algorithm (*FA*). CloudSim simulation shows that our algorithm outperformed the con-

trastive algorithms in makespan and search ability. Finally, an example was cited to prove that our algorithm can converge to the optimal scheduling plan in a short time, providing a desirable solution to multi-objective scheduling of cloud manufacturing resources.

2. Literature review

In cloud computing, the scheduling problem involves cost, makespan as well as load balance. The cloud computing system decomposes the computing task into several subtasks, in the light of the massive data on the task. Next, each subtask was further split into processes. The more refined the division, the better the monitoring and control of the processing state. Thus, the process division ensures the professionality of service-oriented cloud manufacturing.

Zhou *et al.* [1] sets up a mathematical model for multi-objective disassembly line balancing problem that minimizes the times of tool replacement, and puts forward a discrete tracking mode based on sequence exchange; Next, the CSO was integrated with the simulated annealing (SA) algorithm to enhance the global search ability; Finally, the proposed model and the hybrid algorithm were applied to design the disassembly line of a printer, creating various balanced solutions for decision makers [2-3].

Zuo *et al.* [4] establishes a cloud computing resource scheduling model based on improved chaotic FA. Firstly, the cloud computing resource scheduling model was built to improve the completion time, efficiency and safety of the task; Then, the chaotic algorithm was introduced to the FA, and the individuals were perturbed to speed up the convergence and avoid the local minimum trap. In addition, the Lagrangian relaxation function was adopted to improve the established model. CloudSim simulation shows that the improved chaotic FA can balance the resource allocation, shorten the completion time and enhance the system capacity.

Sun *et al.* [5] mentions that cloud manufacturing resources raise a high requirement on scheduling and its granularity, due to their dispersity, diversity and load rate imbalance. In the light of this, the cloud manufacturing task was divided into processes, which were taken as the basic units of scheduling. Then, a multi-objective cloud manufacturing process scheduling model was constructed to minimize the makespan, minimize the cost and balance the load rate. After that, a hybrid multi-objective scheduling algorithm was designed based on the particle swarm optimization (PSO) and the genetic algorithm (GA) [6-7]. The bi-level coded chromosomes of the GA were taken as the particles of the PSO. Bi-level coding refers to the crossover and mutation of chromosomes through two layers, which speed up the convergence to the global optimum. The first layer is the sequence of processes and the second layer is the number of the resources corresponding to the processes. Finally, the hybrid algorithm was applied to schedule the cloud manufacturing of an elevator. The results show that the algorithm can output the optimal scheduling plan in a short time, and thus effectively solve multi-objective scheduling of cloud manufacturing.

Wu *et al.* [8] designs an algorithm that establishes the mapping relationship between position vector of each particle and the allocated service through integer coding. The crossover and mutation operations of the GA were introduced to update the particle positions by standard PSO. The particle positions were updated by four methods in turns to ensure swarm diversity. Example analysis shows that the algorithm enjoys a high effectiveness and execution efficiency [9-12].

3. Used methods

3.1 Cat swarm optimization

The CSO models the behavior of cats into two modes: seeking mode and tracing mode. The former mainly performs local search and the latter looks for the global optimum. Under the seeking mode, the individuals were perturbed multiple times, such that each can approach the local optimum. Under the tracing mode, each cat traces the target at a certain speed, and updates its position into the better between its current position and the optimal position of the swarm [13].

Here, the tracing mode of the CSO is optimized. The current position of each cat was updated continuously according to the global optimal position [14]. Thus, the current solution can gradu-

ally approximate and reach the optimal solution. The speed and position update strategies can be expressed as:

$$v_i(t+1) = v_i(t) + c_1 \times rand \times (x^* - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where x^* is the current best-known position of the swarm; c_1 is the acceleration coefficient; *rand* is a random number in [0, 1]; $v_i(t)$ and $v_i(t+1)$ are the cat speed at the t-th and $(t+1)$-th iterations, respectively; $x_i(t)$ and $x_i(t+1)$ are the cat position at the t-th and $(t+1)$-th iterations, respectively.

Under the seeking mode, the speed parameter was removed because this mode only performs local search. Then, the position update strategy can be expressed as:

$$x_i(t+1) = x_i(t) + \phi_i \times (x^* - x_i(t)) \quad (3)$$

3.2 Firefly algorithm

The FA treats the particles in the search space as fireflies, and views the search and iteration as the mutual attraction and motion of individual fireflies. The brightness of each firefly is described as the value of fitness function. In the swarm, a firefly with relatively low brightness will approach the relatively bright individuals. Over the time, more and more fireflies gather around the bright ones, creating an extra bright region [15-18]. The brighter this region, the more likely it is to converge to the optimal solution. By contrast, the less bright regions are not likely to obtain the optimal solution. The mathematical description of the algorithm is as follows. Firstly, the brightness of firefly j can be expressed as:

$$I_j = f(X_j), X_j = (x_{i1}, x_{i2}, \ldots, x_{id}) \quad (4)$$

where X_j is the position of firefly j in the d-dimensional space; f is the function of firefly brightness.

The attractiveness of firefly j to firefly i can be described as:

$$\beta_{ji} = \beta e^{-2r_{ji}^2} \quad (5)$$

where β is the attractiveness of a firefly to the other fireflies; $\lambda \in [0.01,100]$ is the light absorption coefficient; r_{ji} is the Cartesian distance between fireflies j and i. If attracted by firefly j at time t, firefly i will update its position by:

$$X_i^{t+1} = X_i^t + \beta_{ji}(X_j^t - X_i^t) \quad (6)$$

As shown in Eq. 6, the position update speed is negatively correlated with the distance between the two fireflies. If the distance is too large, the update proceeds slowly and requires multiple iterations, which slows down the computing speed. Therefore, the FA may easily fall into the local optimal trap, and lose the search ability in the later stage. In this paper, the CSO is introduced to improve the FA*'s* search ability, and the sequential allocation is adopted to integrate the two algorithms.

The sequential allocation is relative to the random allocation of the CSO. In random allocation, the individuals to trace the target are selected randomly, according to the previously determined number of individuals. In sequential allocation, all individuals are ranked in ascending order of fitness, and the top individuals are selected for the tracing task [19-25]. In the CSO-FA algorithm, the firefly distribution area is divided into the concentrated area and the dispersed area. The former is subjected to the seeking mode, and the latter, the tracing mode.

4. Construction of the objective function

4.1 Assumptions

The following assumptions are undertaken:

- Each process is executed on one manufacturing resource.
- Each process has a fixed makespan and a fixed cost.
- All subtasks have the same priority, and each has its internal process constraint.

4.2 Objective function

It is assumed that a cloud platform receives multiple requests for manufacturing resources, each of which contains N subtasks F_i, $(i = 1, 2, \dots, N)$ to be processed on M manufacturing resources. Each subtask includes P_i processes. There are M_{ij} manufacturing resources available. Then, the objective function can be expressed as:

$$F = \min(f_1, f_2, f_3) \tag{7}$$

Specifically, f_1 is the minimal makespan to process all subtasks on the manufacturing resources. It depends on the subtask with the longest makespan.

$$f_1 = \min\left(\max\sum_{j=1}^{P_i}\sum_{k=1}^{M_{ij}} e_{ijk}\,\eta_i l_{ijk}\right) \tag{8}$$

where e_{ijk} is a variable in [0, 1] (if F_{ij} is processed on manufacturing resource k, e_{ijk} equals 1; otherwise, e_{ijk} equals 0); η_i is the number of jobs corresponding to F_i; l_{ijk} is the makespan of F_{ij} on manufacturing resource k.

f_2 is the minimal cost to process all subtasks on the manufacturing resources:

$$f_2 = \min\left(\sum_{i=1}^{N}\sum_{j=1}^{P_j}\sum_{k=1}^{M_{ij}} e_{ijk}\eta_i c_{ijk}\right) \tag{9}$$

where c_{ijk} is the cost of F_{ij} on manufacturing resource k.

f_3 is the balanced load rate to process all subtasks on the manufacturing resources:

$$f_3 = \frac{1}{\sum_{s=1}^{M} M_s - 1}\sum_{s=1}^{M}\left(\frac{L_s}{Ca_s}\times 100\,\% - \frac{1}{\sum_{s=1}^{M} M_s}\sum_{s=1}^{M}\theta_s\right)^2 \tag{10}$$

where M_s is the s-th manufacturing resource $(s = 1,2, \dots, M)$; L_s, Ca_s and Q_s are the load, available time and load rate of the s-th manufacturing resource, respectively.

4.3 Constraints

For a subtask, the current process should not begin before the completion of the previous process:

$$t_{end_{ij}} \le t_{start_{i(j+1)}} \tag{11}$$

where $t_{end_{ij}}$ is the end time of the previous process; $t_{start_{i(j+1)}}$ is the start time of the current process.

The processes are the basic units of the scheduling plan and each process can only occupy one manufacturing resource.

$$\sum_{k=1}^{M_{ij}} e_{ijk} = 1 \tag{12}$$

All subtasks should be completed no later than the scheduled delivery time.

$$f_1 \ge T_{i\max} \tag{13}$$

where $T_{i\max}$ is the deadline on the delivery time of F_{ij}.

The total cost of all subtasks must be smaller than the project budget.

$$f_2 \ge C_{i\max} \tag{14}$$

where $C_{i\max}$ is the maximum budget of F_{ij}.

5. Design of an integrated Cat swarm optimization and Firefly algorithm

5.1 Coding design

Each of the i subtasks corresponds to I_j processes and k manufacturing resources. Thus, the feasible solutions can be coded with the positions of $i \times j \times k$ fireflies. Then, the subtasks can be arranged into a sequence, in which each subtask must appear for j times. Let the subtask sequence be $M_{(1,1),1}$, $M_{(3,1),1}$, $M_{(2,1),1}$, $M_{1(1,2),2}$, $M_{(4,1),1}$, $M_{(3,2),2}$, $M_{(1,3),3}$, $M_{(3,3),3}$, $M_{(4,2),3}$, where $M_{(i,j),k}$, means subtask i should be processed in process j on manufacturing resource k. This sequence can be interpreted as follows: firstly, the first processes of the first, second and third subtasks should be processed in turns on the first manufacturing resource; next, the second process of the first subtask should be processed on the second manufacturing resource. Finally, the second process of the fourth subtask should be processed on the third manufacturing resource [26].

5.2 Swarm initialization

According to the code design, the proposed CSO-FA algorithm code consists of two parts. The first part explains the processing sequence of subtask processes on the manufacturing resources and the second part specifies the number of the resources corresponding to the processes. The code length is $(i \times j) + k$. Next, a swarm of N fireflies was initialized, and each firefly was assigned a random initial position in the given feasible region.

5.3 Fitness function

The computer algorithm for cloud manufacturing should be designed in view of the specific demand of users. If the products are urgently needed (e.g., receiving a rush order with delay penalty), there is no need to think too much about cost and balance of load rate. The only goal to be pursued is to minimize the makespan. If the order is not urgent but with a small amount, it is only necessary to consider the cost. If the order faces limited manufacturing resources, the balance of load rate should be the top priority. Of course, overall consideration should be given to cost, makespan and load rate for most orders. In this paper, two types of fitness functions are discussed: the total fitness function and the sub-fitness function [27-28].

The total fitness function provides an evaluation criterion for scheduling plans. Each plan was represented as a firefly, whose brightness reflects the quality of the current position. The relationship between firefly position and brightness was established. Then, the brightness was taken as the value of the fitness function to evaluate each scheduling plan. The greater the fitness, the brighter the firefly, the more suitable the position, and the better the scheduling plan.

The value of the total fitness function for multiple objectives can be derived from the random weighting of the single objectives:

$$fitness(k) = \omega_1 \frac{\min T - T_{\min}}{T_{\min}} + \omega_2 \frac{\min C - C_{\min}}{C_{\min}} + \omega_3 \frac{\min L - L_{\min}}{L_{\min}} \tag{15}$$

where $\omega_i = \frac{r_i}{\sum_{i=1}^{3} r_i}$ is the weight (r_i is a non-negative random number); $T_{\min}$, $C_{\min}$ and $L_{\min}$ are the minimal makespan, minimal cost and balanced load rate of the three single-objective functions, respectively.

The sub-fitness functions can be expressed as:

$$T_{\min} = fitness_1(k) = \min T[(i,j),k], k = 1,2,\ldots,K \tag{16}$$

$$C_{\min} = fitness_2(k) = \min C[(i,j),k], k = 1,2,\ldots,K \tag{17}$$

$$L_{\min} = fitness_3(k) = \min L[(i,j),k], k = 1,2,\ldots,K \tag{18}$$

5.4 Algorithm flow

Based on the CSO-FA algorithm, a cloud computing task can be scheduled in the following steps:

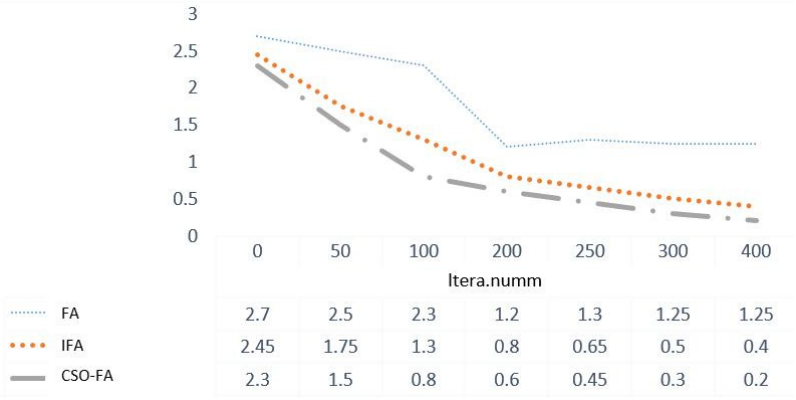
Step 1: Initialize the firefly swarm, determine the positions of M fireflies as per the problem scale, and set the algorithm parameters.

Step 2: Judge if the maximum number of iterations has been reached. If so, end the iterative process; otherwise, proceed with the following steps.
Step 3: Decode the coded fireflies, find the mapping relationship between manufacturing resources and the subtasks, and compute the minimum makespan, minimum cost and optimal load rate.
Step 4: Randomly assign weight to each single-objective function, and compute the value of the total fitness function, i.e. the firefly brightness.
Step 5: Let all fireflies move towards the nearby brighter individuals. Divide the search space into small regions. Apply the seeking mode of the CSO in the relatively bright regions, and the tracing mode of the CSO in the relatively dark regions. Update the position of each firefly to speed up the search.
Step 6: Control the swarm diversity with equations.
Step *7*: Decode the global optimal firefly and output the result as the optimal scheduling plan.

6. Results and discussion

The proposed CSO-FA algorithm was verified through simulation on CloudSim 4.0. CloudSim is a cloud computing simulation software released on April 8, 2009 by the Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, in association with the Gridbus Project. It is a function library developed based on SimJava, a discrete event simulation package. The software can operate on both Windows and Linux. CloudSim inherits from GridSim the support to the *R&D* of cloud computing, and provides two distinctive new features: (1) the ability to model and simulate large cloud computing infrastructure; (2) the provision of a self-sufficient platform supporting data centers, service agents, as well as scheduling and allocation strategies. CloudSim also boasts many unique functions. For example, a virtualization engine is designed to help data centers provide multi-layered virtualization services both independently and collaboratively, and the processors assigned to visualization services can switch flexibly in time and space.

The CSO-FA, the FA and the improved FA (IFA) were separately applied to simulate the scheduling of 5 manufacturing resources and 1,000 cloud computing tasks. The convergence curves of the three algorithms are shown in Fig. 1. Obviously, the FA saw a gradual slowdown of convergence speed, and converged prematurely in the global search, owing to its weak search ability. Compared with the FA, the IFA converged to the optimal solution rapidly. However, the fastest convergence and optimal solution were achieved by the CSO-FA. This is because the CSO-FA introduces the seeking mode and tracing mode to different regions, which speeds up the search for the global optimum and prevents the local optimal trap (the Y-axis is the convergence of form Fig. 1 to Fig. 4).



	0	50	100	200	250	300	400
FA	2.7	2.5	2.3	1.2	1.3	1.25	1.25
IFA	2.45	1.75	1.3	0.8	0.65	0.5	0.4
CSO-FA	2.3	1.5	0.8	0.6	0.45	0.3	0.2

Fig. 1 The convergence curves of different algorithms

Table 1 Requested subtasks and their processes

Subtask number	Subtask name	Number of processes	Subtasks correspond to processes	T_{imax} (s)	C_{imax} (Yuan)
F_1	floor	6	101,102,103,104,105,106	336	158
F_2	Mounting plate	6	201,202,203,204,205,206	288	208
F_3	Shock absorber	5	301,302,303,304,305	240	350
F_4	Limit board	4	401,402,403,404	200	340
F_5	Licating piece	5	501,502,503,504,505	192	258
F_6	Card	6	601,602,603,604,605,606	160	267

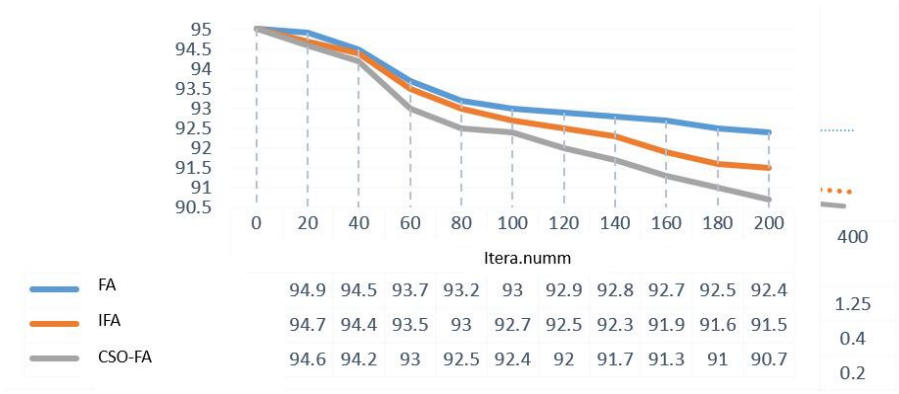


Fig. 2 The makespans of different algorithms with 50 tasks

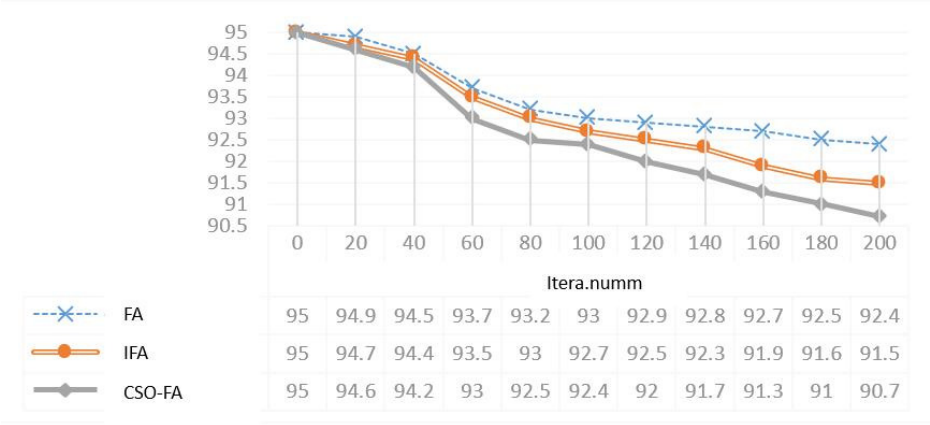


Fig. 3 The makespans of different algorithms with 100 tasks

Table 2 Available manufacturing resources

	F_{i1}			F_{i2}			F_{i3}			F_{i4}			F_{i5}			F_{i6}		
	M	t_{ijk}	c_{ijk}	M	t_{ijk}	c_{ijk}	M	t_{ijk}	c_{ijk}	M	t_{ijk}	c_{ijk}	M	t_{ijk}	c_{ijk}	M	t_{ijk}	c_{ijk}
F_1	5	3	11	6	10	25	4	9	14	[2,9]	[5,4]	[9,11]	[3,7]	[3,3]	[7,6]	10	4	8
F_2	4	6	15	[2,9]	[8,6]	[12,9]	8	4	8	[6,7]	[2,6]	[5,8]	5	3	4	[1,10]	[3,3]	[7,7]
F_3	3	4	8	[6,8]	[5,7]	[10,9]	7	7	11	[2,1]	[5,5]	[15,16]	[4,10]	[9,11]	[21,21]			
F_4	5	7	16	2	3	7	[4,7]	[4,6]	[8,11]	10	3	8						
F_5	[4,5]	[6,4]	[9,14]	5	10	18	[9,10]	[7,9]	[17,19]	6	8	16	2	5	13			
F_6	[2,6]	[3,7]	[8,13]	4	10	20	[6,9]	[8,7]	[13,15]	7	9	15	8	4	7	[3,9]	[9,4]	[15,13]

Table 3 Load rate of each manufacturing resource

M_s	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
L_S	0.91	0.89	0.83	0.94	0.70	0.79	0.93	0.71	0.88	0.95

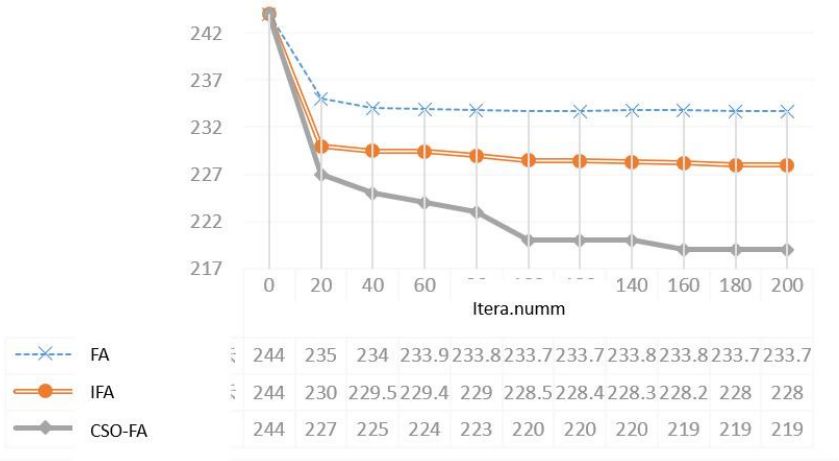


Fig. 4 Iterative process of each algorithm

6.1 Comparison of makespans

Without changing the number of manufacturing resources, the three algorithms were further compared through the simulation on 50 and 100 cloud computing tasks. The computing power of each manufacturing resource was randomly determined in the interval of [400, 200], and the subtask length was controlled within [400, 1,000]. The swarm size and number of iterations of the three algorithms were kept the same. The other parameters were configured in reference to related literature. Each algorithm was run 20 times and the mean value was taken as the final result. The simulation results are plotted as Figs. 2 and 3.

As shown in Figs. 3 and 4, the proposed CSO-FA had a much shorter makespan than the FA and the IFA. This advantage is attributable to the strategy to control swarm diversity, which protects the search ability of the swarm and reduces the chance of falling into the local optimum trap. Hence, our algorithm is more suitable for cloud manufacturing scheduling than the contrastive algorithms.

6.2 Example analysis

The proposed algorithm was applied to schedule the manufacturing subtasks of an elevator enterprise on the cloud platform. There are a total of six manufacturing subtasks. The processes, delivery time $T_{i\max}$ and budget $C_{i\max}$ of each subtask are listed in Table 1. For each subtask, the manufacturing resources suitable for its processed were searched for in the resource pool on the cloud platform. The pool provides 10 manufacturing resources for these subtasks. Table 2 describes the manufacturing resources M available for the processes of each subtask, and the makespan t_{ijk} and cost c_{ijk} of each process on different manufacturing resources. Table 3 specifies the load rate of each manufacturing resource. The iterative process of each algorithm is recorded in Fig. 4.

7. Conclusion

Taking processes as the basic scheduling unit, this paper establishes a multi-objective optimization model for cloud manufacturing resource scheduling, in the light of the main influencing factors of cloud manufacturing scheduling. Besides, the CSO-FA for subtask scheduling in cloud computing was designed to rationalize the resource allocation in the cloud environment. Specifically, the CSO was introduced to the FA to accelerate the search process, without sacrificing the search ability of the swarm. The CSO-FA was applied to the established model, minimizing the time to converge to the global optimum. Compared with the FA and IFA, the proposed algorithm converged to the optimal solution in a short time. Finally, the algorithm was proved suitable to solve the multi-objective scheduling of cloud manufacturing resources, through the simulation of manufacturing order processing in an elevator enterprise.

Acknowledgement

Fund support: Anyang science and technology research project [2018] No. 66. Key scientific research projects of institutions of higher learning in Henan province in 2020 Item no.20B460002.

References

[1] Zou, B.S., Zhang, Z.Q., Cai, N., Zhu, L.X. (2018). Cat swarm simulated annealing algorithm for disassembly line balancing problem under tool constraints, *Computer Integrated Manufacturing Systems*, Vol. 24, No. 9, 82-94, doi: 10.13196/j.cims.2018.09.009.

[2] Huang, X., Zhang, T., Deng, Z., Li, Z. (2018). Design of moving target detection and tracking system based on cortex-A7 and openCV, *Traitement du Signal*, Vol. 35, No. 1, 61-73, doi: 10.3166/TS.35.61-73.

[3] Zhang, D. (2017). High-speed train control system big data analysis based on fuzzy RDF model and uncertain reasoning, *International Journal of Computers, Communications & Control*, Vol. 12, No. 4, 577-591, doi: 10.15837/ijccc.2017.4.2914.

[4] Zuo, Z.-L., Guo, X., Li, W. (2018). An improved swarm optimization algorithm, *Microelectronics and Computer*, Vol. 35, No. 2, 61-66, doi: 10.19304/j.cnki.issn1000-7180.2018.02.013.

[5] Sun, W., Wu, H., Lv, W., Gao, Y. (2017). Research on multi-objective process scheduling of cloud manufacturing resources, *Journal of Nanjing University of Aeronautics and Astronautics*, Vol. 49, No. 6, 773-778, doi: 10.16356/j.1005-2615.2017.06.003.

[6] Li, B.-H., Zhang, L., Wang, S.-L., Tao, F., Cao, J.-W., Jiang, X.-D., Song, X., Chai, X.-D., (2010). Cloud manufacturing: A new service-oriented networked manufacturing model, *Computer Integrated Manufacturing System*, Vol. 16, No. 1, 1-7, doi: 10.13196/j.cims.2010.01.3.libh.004.

[7] Wang, S.-L., Song, W.-Y., Ling, K, Qiang, L.-I., Liang, G., Chen, G.-S. (2012). Manufacturing resource allocation based on cloud manufacturing, *Computer Integrated Manufacturing System*, Vol. 18, No. 7, 1396-1405.

[8] Wu, Q., Ma, Y.-M., Li, L., Qiao, F. (2015). Data driven dynamic scheduling method for semiconductor production line, *Control Theory and Application*, Vol. 32, No. 9, 1233-1239.

[9] Liu, W., Liu, B., Sun, D. (2013). Multi-task oriented service composition in cloud manufacturing, *Computer Integrated Manufacturing System*, Vol. 19, No. 1, 199-209.

[10] Yin, C., Zhang, Y., Zhong, T. (2012). Optimization model of cloud manufacturing services resource combination for new product development, *Computer Integrated Manufacturing System*, Vol. 18, No. 7, 1368-1378.

[11] Kshirsagar, P., Jiang, D., Zhang, Z. (2016). Implementation and evaluation of online system identification of electromechanical systems using adaptive filters, *IEEE Transactions on Industry Applications*, Vol. 52, No. 3, 2306-2314, doi: 10.1109/TIA.2016.2515994.

[12] Puttonen, J., Lobov, A., Cavia Soto, M.A., Martinez Lastra, J.L. (2015). Planning-based semantic web service composition in factory automation, *Advanced Engineering Informatics*, Vol. 29, No. 4, 1041-1054, doi: 10.1016/j.aei.2015.08.002.

[13] Cao, Y., Wu, Z., Zhao, Q., Yan, H., Yang, J. (2015). Study on resource configuration on cloud manufacturing, *Mathematical Problems in Engineering*, Vol. 2015, Article ID 635602, 13 pages, doi: 10.1155/2015/635602.

[14] Cao, Y., Wang, S., Kang, L., Gao, Y. (2016). A TQCS-based service selection and scheduling strategy in cloud manufacturing, *The International Journal of Advanced Manufacturing Technology*, Vol. 82, No. 1-4, 235-251, doi: 10.1007/s00170-015-7350-5.

[15] Xiong, Y., Wang, J., Wu, M., She, J. (2015). Virtual resource scheduling method of cloud manufacturing oriented to multi-objective optimization, *Computer Integrated Manufacturing System*, Vol. 21, No. 11, 3079-3087, doi: 10.13196/j.cims.2015.11.029.

[16] Ge, P., Yang, X., Xiao, X., Qiu, Y.-Q., Ren, P.-Y., Liu, Z.-S., Zheng, W.-M. (2012). Task allocation in cloud manufacturing based on multi-resolution clustering, *Computer Integrated Manufacturing System*, Vol. 18, No. 7, 1461-1468.

[17] Wu, S.-Y, Zhang, P., Li, F. (2015). Multi-task scheduling based on particle swarm optimization in cloud manufacturing systems, *Journal of South China University of Technology (Natural Science Edition)*, Vol. 43, No. 1, 105-110, doi: 10.3969/j.issn.1000-565X.2015.01.017.

[18] Gao, Y., Tao, L., Ma, M. (2018). Multilevel-thresholding image segmentation based on cat swarm optimization, *Chinese Stereology and Image Analysis*, Vol. 23, No. 2, 125-132, doi: 10.13505/j.1007-1482.2018.23.02.001.

[19] Du, Y., Shi, F., Chen, Q., Wang, Y., Zhao, J., Li, Q. (2018). An improved particle swarm scheduling algorithm based on batch changing production time, *European Journal of Electrical Engineering,* Vol. 20, No. 4, 439-453, doi: 10.3166/ejee.20.439-453.

[20] Du, Y., Chen, Q.X. (2012). The model and simulation of SMT production, *Applied Mechanics and Materials*, Vol. 217-219, 1493-1496, doi: 10.4028/www.scientific.net/AMM.217-219.1493.

[21] Pajoutan, M., Golmohammadi, A., Seifbarghy, M. (2014). CMS scheduling problem considering material handling and routing flexibility, *The International Journal of Advanced Manufacturing Technology,* Vol. 72, No. 5-8, 881-893, doi: 10.1007/s00170-014-5696-8.

[22] Capps, O., Seo, S.-C., Nichols, J.P. (1997). On the estimation of advertising effects for branded products: An application to spaghetti sauces, *Journal of Agricultural and Applied Economics*, Vol. 29, No. 2, 291-302, doi: 10.1017/S1074070800007793.

[23] Zacharia, P.T., Nearchou, A.C. (2016). A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem, *Engineering Applications of Artificial Intelligence*, Vol. 49, 1-9, doi: 10.1016/j.engappai.2015.11.007.
[24] Miyake, D.I. (2006). The shift from belt conveyor line to work-cell based assembly systems to cope with increasing demand variation in Japanese industries, *International Journal of Automotive Technology and Management*, Vol. 6, No. 4, 419-439, doi: 10.1504/IJATM.2006.012234.
[25] Sakazume, Y. (2004). Is Japanese cell manufacturing a new system? A comparative study between Japanese cell manufacturing and cellular manufacturing, *Journal of Japan Industrial Management Association*, Vol. 55, No. 6, 341-349.
[26] Cui, J. (2018). The application of improved bacteria foraging algorithm to the optimization of aviation equipment maintenance scheduling, *Tehnički Vjesnik – Technical Gazette,* Vol. 25, No. 4, 1103-1109, doi: 10.17559/TV-20171214025731.
[27] Liu, Y., Wang, L., Wang, X.V., Xu, X., Zhang, L. (2019). Scheduling in cloud manufacturing: State-of-the-art and research challenges, *International Journal of Production Research*, Vol. 57, No. 15-16, 4854-4879, doi: 10.1080/00207543.2018.1449978.
[28] Dalfard, V.M., Ranjbar, V. (2012). Multi-projects scheduling with resource constraints & priority rules by the use of simulated annealing algorithm, *Tehnički Vjesnik – Technical Gazette,* Vol. 19, No. 3, 493-499.