Int j simul model 13 (2014) 1, 93-101
ISSN 1726-4529 Original scientific paper

BATCH TASK SCHEDULING-ORIENTED OPTIMIZATION MODELLING AND SIMULATION IN CLOUD MANUFACTURING

Jian, C. F. & Wang, Y.
Computer Science and Technology College, Zhejiang University of Technology, Hangzhou, China
E-Mail: jiancf@zjut.edu.cn

Abstract
Batch task scheduling in cloud manufacturing has dynamic, real-time characteristic and the presence of big data concurrency and exchange requirements, while traditional workshop tasks scheduling models and algorithms can't fit. In order to effectively save the time and reduce the cost of workshop production, an optimization model is put forward at first. And then improved cooperative particle swarm optimization algorithm with fast convergence and strong ability to avoid local optimization is used to solve the tasks scheduling problems. At last simulation experiment analysis results prove its effectiveness.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Cloud Manufacturing, Batch Task Scheduling, Improved Cooperative Particle Swarm Optimization

1. INTRODUCTION

Cloud manufacturing, as an application of cloud computing in the manufacturing industries, has become a new production manufacturing mode which consists of new technologies such as the Internet of things (IOT) to provide supports for manufacturing with wide distribution of production resources [1, 2]. Cloud manufacturing can execute real-time and collaborative production tasks [3].The management system of cloud manufacturing workshop production is mainly for tasks allocation, resource scheduling and real-time monitoring, which can also manage the production resources. During the process of cloud manufacturing production, how to allocate tasks dynamically will directly affect the time and cost of production.

Researches on tasks scheduling of traditional manufacturing have been relatively mature. Evolutionary algorithms applied to tasks scheduling in traditional workshop production have a good effect [4, 5], such as Particle Swarm Optimization algorithm (PSO) [6], Simulated Annealing algorithm (SA) [7] and Genetic Algorithm (GA) [8], etc. However, workshop production in cloud manufacturing with vast amounts of data concurrency and exchange is dynamic and real-time, while the tasks scheduling methods for traditional workshop production have been difficult to adapt to the cloud manufacturing. Tasks scheduling platform should fully consider the information exchange between the clouds [9], which is based on the characteristics of data and information dynamic exchange in cloud manufacturing. Some evolutionary algorithms, such as niche immune algorithm [10], were also put forward to solve the cloud service resources allocation optimization problems. When scheduling tasks in the cloud manufacturing environment, the deadline of production [11] and balanced distribution of load [12] on production resources should also be fully considered. How to optimize batch task scheduling with large-scale, high heterogeneous, strong dynamic and other characteristics is the key question in the cloud manufacturing. Task scheduling needs consider the execution time and cost of batch production tasks, as well as production processes. Whether traditional workshop manufacturing or cloud manufacturing, to save the time and reduce the cost, are the two important optimization goals. So a batch task scheduling-oriented optimization model is

put forward in this paper and Improved Cooperative Particle Swarm Optimization (ICPSO) algorithm is used in this model. Compared with the traditional workshop production task scheduling algorithm, it can not only reduce the time and cost of production tasks, but also better meet the large-scale production mission requirements in cloud manufacturing environment.

2. PROBLEM DESCRIPTION

Most of batch tasks in the cloud manufacturing workshop production are the multistep production tasks. In order to complete the batch tasks production in cloud manufacturing, tasks should be analysed firstly to determine the production processes and the corresponding production resources required. For batch tasks production in cloud manufacturing workshops, the corresponding production resources are needed in each production process. The management system of cloud manufacturing determines production processes and selects the corresponding suppliers of production resource according to the number of batch tasks, completion time planed and the budget. The hierarchical model of cloud manufacturing shows as follows:

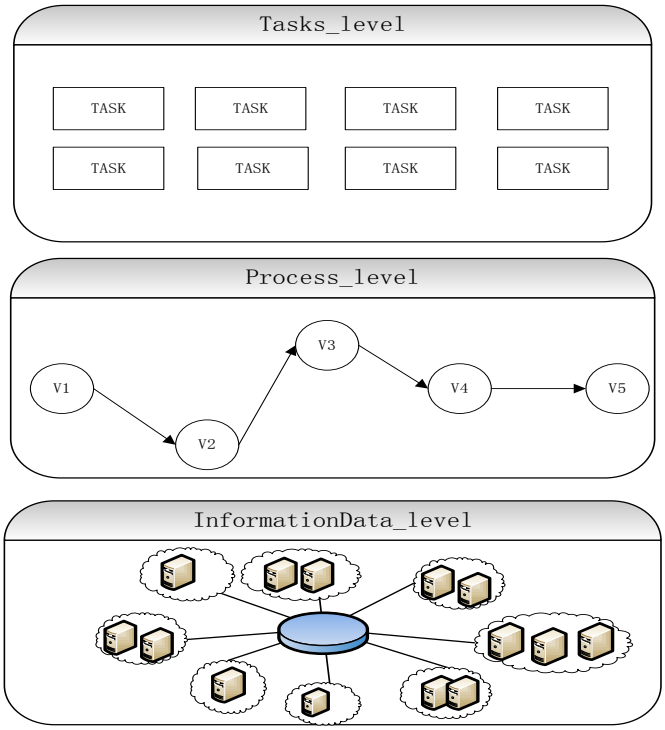

Figure 1: Hierarchical model of cloud manufacturing workshop production.

As shown in Fig. 1, the platform is divided into several levels. The first layer is the task-layer. There are the batch tasks in cloud manufacturing workshop to be produced in this level. The production processes of cloud manufacturing and production process of traditional manufacturing are the same in some ways. For many production tasks with the same characteristics, the production processes to execute them are the same as well. The second level of structure chart in this platform is the process diagram of cloud manufacturing, which can be regarded as a directed acyclic graph (DAG). Production resources $S = \{S_1, S_2, \ldots S_n\}$ are represented by $V = \{V_1, V_2, \ldots V_n\}$ in the DAG, so each node represents a production process. E is set of the connections between nodes representing interdependencies and orders of production processes, namely $E = \{E_{ij} = \mathrm{Re}(V_i, V_j) \mid V_i, V_j \subseteq V\}$. The third layer is for the

transmission data information. Cloud manufacturing mainly consists of widely distributed production resources. There is the need for two-way transmission of data information and management information between different production processes. So problem can be described as: there are m production processes and n batch tasks to be produced in cloud manufacturing workshops. Each task should be carried out in accordance with the order of production process, and production resources of every production process only work for a task in every moment. Given the production time of each task working in each production process and the production cost of each task running on each production process, how to schedule tasks to make the total cost and time minimize.

Sequence array $JQ = \{JQ_1, JQ_2, JQ_3, \ldots JQ_n\}$ records the execution sequence of tasks. Multi-dimensional array $TimeTable_{m\times n}$ records the time of each production task working in each production process. $TimeTable[i, j]$ represents the time task i working in the production process j.

The one-dimensional array $Cost\_h = \{Cost\_1, Cost\_2, Cost\_3, \ldots Cost\_\mathrm{m}\}$ represents executing price of each production process in one minute. $EndTime_{rs_k^i}$ shows the end time of task JQ_i working in the production process j. There are constraints as follows:

$$EndTime_{rs_k^i} \geq EndTime_{rs_{k-1}^i} + TimeTable[i,k] \tag{1}$$

$$EndTime_{rs_k^i} \geq EndTime_{rs_{k+1}^{i\text{-}1}} \tag{2}$$

After executing in production process j, task JQ_i should be executed in process $j + 1$. However, there can be only one task working in one production process in every moment so that task JQ_i can work in the process $j + 1$ depends on whether there is another task working in process $j + 1$. The tasks are performed according to the sequence of the array $JQ = \{JQ_1, JQ_2, JQ_3, \ldots JQ_n\}$. If there is still a task executing in production process, it must be $JQ_{i\text{-}1}$. So the value of the start time of JQ_i executing in the production process $j + 1$ is the lager value of the finish time of JQ_i executing in the *No. j* production process and the finish time of $JQ_{i\text{-}1}$ executing in the production process $j + 1$.

$$EndTime_{rs_k^i} = \max(EndTime_{rs_{k-1}^i}, EndTime_{rs_{k+1}^{i-1}}) \tag{3}$$

$$Cost_{rs_k^i} = TimeTable[i,k] \times Cost\_k \tag{4}$$

The total cost can be defined as:

$$Cost\_sum = \sum\sum Cost_{rs_k^i} \tag{5}$$

3. ICPSO DESCRIPTION

Particle swarm optimization algorithm (PSO) is an adaptive heuristic algorithm, which was firstly proposed by Kennedy and Eberhart in 1995. Its basic principle originates from imitation of animal foraging behaviour. Each particle with position and speed represents a solution. In the process of constant iterative, each particle changes its speed and position. Assume that there are n particles and the dimension of the solution space is m. The position and speed of each particle are represented by m dimensional vector $x_i = \{x_{i\_1}, x_{i\_2}, x_{i\_3}, \ldots x_{i\_m}\}$ and $v_i = \{v_{i\_1}, v_{i\_2}, v_{i\_3}, \ldots v_{i\_m}\}$. x_{i_j} and v_{i_j} represent the location and speed of the particle i in the j^{th} dimension. The changes of the speed and position of particles are in compliance with the following equations:

$$v_{i\_j}^{k+1} = \omega v_{i\_j}^k + c_1 * r_1 * (pbest_i - x_{i\_j}^k) + c_2 * r_2 * (gbest - x_{i\_j}^k) \tag{7}$$

$$x_{i\_j}^{k+1} = x_{i\_j}^k + v_{i\_j}^{k+1} \tag{8}$$

x_{ij}^k and v_{ij}^k represents current position and speed of particle i at iteration k. $Pbest_i = \{Pb_{i\_1}, Pb_{i\_2}, Pb_{i\_3}, \ldots Pb_{i\_m}\}$ stores the best position of particle i, and $gbest = \{gb_1, gb_2, gb_3, \ldots gb_\mathrm{m}\}$

stores the position of best particle in the populations. Set the maximum number of iterations before starting this algorithm. Algorithm ends when iteration number reaches the maximum value, and the optimal value required is *gbest* at that moment. On the basis of the basic particle swarm optimization algorithm, Cooperative Particle Swarm optimization algorithm (CPSO) appears. CPSO divided each particle into *k* parts.

$S_{i\_j} = \{x_{j\_(i-1)[m/k]}, x_{j\_(i-1)[m/k]+1}, x_{j\_(i-1)[m/k]+1}.....x_{j\_i[m/k]}\}$ represents the vector group *j* in particle *i*. Assuming that fitness function is $Fitness$, the optimum position of each particle for being updated is as the following equation in the process of iteration:

$$S_{i\_j\_pbest} = \begin{cases} S_{i\_j}, Fitness(M(S_{i\_j})) > Fitness(M(S_{i\_j\_pbest})) \\ S_{i\_j}, Fitness(M(S_{i\_j})) \leq Fitness(M(S_{i\_j\_pbest})) \end{cases}$$

$S_{i\_j\_pbest}$ and S_{i_j} represents the optimal location and current position of the particle *i* in the subgroup *j*. Whether the best location will be updated to the current position depends on whether the current fitness value is larger than the best fitness value.

$M(S_{i\_j}) = \{S_{1\_pbest}, S_{2\_pbest}, S_{3\_pbest}, \ldots, S_{i\_j}, \ldots S_{k\_pbest}\}$ is the new multi-dimensional vector which is consisted of current position vector of the particle *i* in the subgroup *j* and the optimal locations of other subgroups. CPSO algorithm select optimal particle through the collaborative communication between subgroups.

However, selecting the optimal location of each subgroup to compose $M(S_{i\_j})$ may cause to fall into local optimum early. There is the need to put forward a new way to increase the diversity of the subgroups so that the possibility will be lager to obtain the more optimal location and avoid local optimum early. Compared with the basic CPSO, Improved Cooperative Particle Swarm Optimization (ICPSO) provides a new method for $M(S_{i\_j})$ vector composition [13] using the method that combines the way to produce $M\_ram(S_{i\_j})$ randomly and the way to produce $M\_std(S_{i\_j})$ by selecting the optimal value of each subgroup. Calculate the fitness function value of both $M\_ram(S_{i\_j})$ and $M\_std(S_{i\_j})$. The one with larger value is $M(S_{i\_j})$.

$$M(Si\_j) = \begin{cases} M\_ram(Si\_j), Fitness(M\_ram(S_{i\_j})) > Fitness(M\_std(S_{i\_j})) \\ M\_std(Si\_j), Fitness(M\_ram(S_{i\_j})) \leq Fitness(M\_std(S_{i\_j})) \end{cases}$$

ω is set as the inertia weight, and c_i represents acceleration coefficients. The global search ability of the algorithm is relatively stronger if the value of ω in the eq. (7) is large enough [14]. Solving the production optimization problems in cloud manufacturing by ICPSO algorithm needs wide population diversity, and strong global search ability of the algorithm can improve the quality of the solution. Some literatures did researches for the value of ω, such as the conclusion has been proved that the algorithm would be faster to get the optimal value when ω was between 0.8 and 1.2 [15]. To adapt to the mass scale of problems and strength the ability ω is equal to 1.2 in this paper. Both c_1 and c_2 are equal to 0.2.

4. SIMULATION EXPERIMENT AND ANALYSIS

Simulations of cloud manufacturing workshop production on the multi-channel procedure processes were done through Matlab. In order to test the performance of ICPSO, it will be compared with PSO and CPSO in the simulations. Matrix $TimeTable_{m \times n}$ stores the execution time of the tasks working on each process in the simulations, which is the random number between1 and 60.

Eight kinds of cloud manufacturing workshop production situations were set up in the simulations to test the performance of the algorithm. Simulations were done 20 times under each condition, and the average values were taken as the results of the simulations.

Table I: Performance comparison of PSO, CPSO and ICPSO with respect to cost criterion.

n	*m*	ICPSO		CPSO		PSO	
		Min	Avg	Min	Avg	Min	Avg
40	15	5.1419	6.1245	5.6524	6.4231	6.1111	7.0230
50	15	5.1454	7.4654	6.9615	8.0185	7.2560	8.4563
60	15	8.1342	9.1133	8.2803	9.4196	9.6513	10.3740
70	15	9.5226	11.1940	10.4860	11.3740	10.2940	12.3590
50	10	3.9039	4.4454	4.0939	4.5143	4.1102	4.9800
60	10	4.4830	5.3098	4.5941	5.4674	5.2921	6.3161
70	10	4.5222	5.9053	5.6512	6.2794	6.7498	7.2037
80	10	5.7879	7.0040	5.8892	7.3157	6.9237	8.1184

Costs of tasks scheduling by ICPSO in various cases are less than the costs by CPSO and PSO, which can be seen from Table I. In this simulation, the numbers of task and processes are the different colocations. However, ICPSO can keep a 2 % - 16 % advantage under each circumstance. The average value of the optimal values obtained in the 20 times simulations by ICPSO is the smallest one, which suggests that solutions of optimization problems solved by ICPSO are stable. At the same time, the scale of problem is gradually increasing with the number of tasks ranging from 40 to 80. Results show that ICPSO, which is keeping the rate of convergence, can also keep the quality and stability of the solution when solving the larger scales of problem.

ICPSO algorithm with fast convergence has strong ability to avoid local optimization. The following Figs. 2 to 4 show the trend of the minimum cost of tasks scheduling solved by PSO, CPSO and ICPSO under the conditions that the number of cloud manufacturing workshop production tasks is 30, 50 and 100 with the number of iterations increasing.

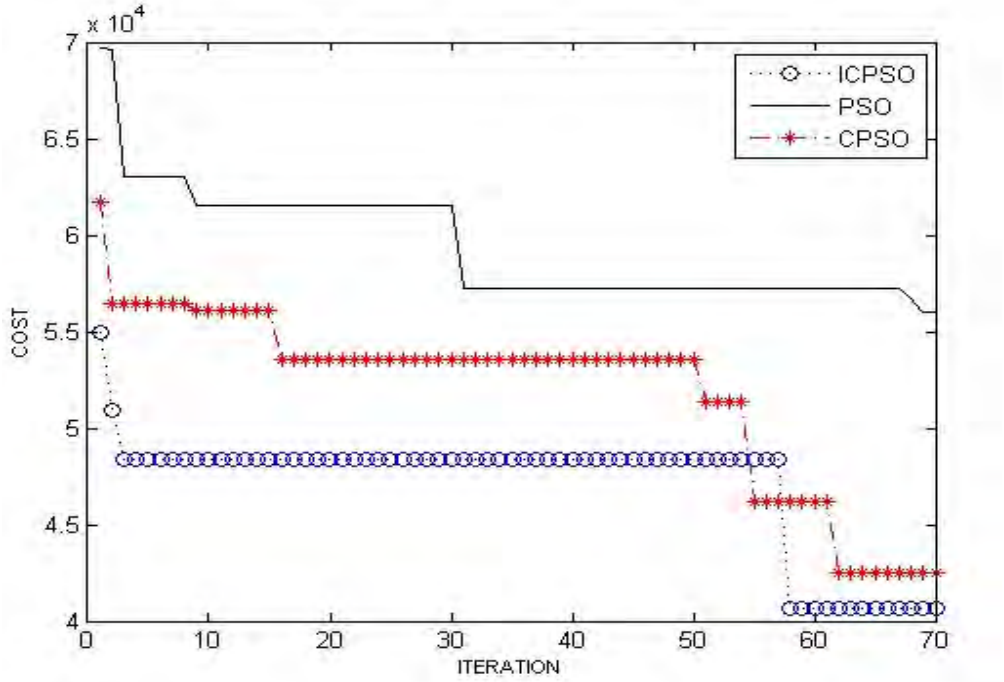

Figure 2: The number of tasks is 30.

ICPSO shows strong ability to jump out of local optimization which can be seen from the figures. ICPSO can strengthen the information exchanging between different species while

keeping the superiority of each species so that it not only extends the diversity of species but also improves the quality of the optimal solution. If stagnation appears in the iterative process with the number of iterations increasing, ICPSO can expand the diversity of populations to jump out of local optimum timely. This is because that when ICPSO is in local search it joins the random mechanism to increase the changes of $M(S_{i\_j})$. In the process of local search ICPSO expands the scope of the candidate set of optimal values, so the quality of optimal value selected by ICPSO is better than the ones selected by PSO and CPSO.

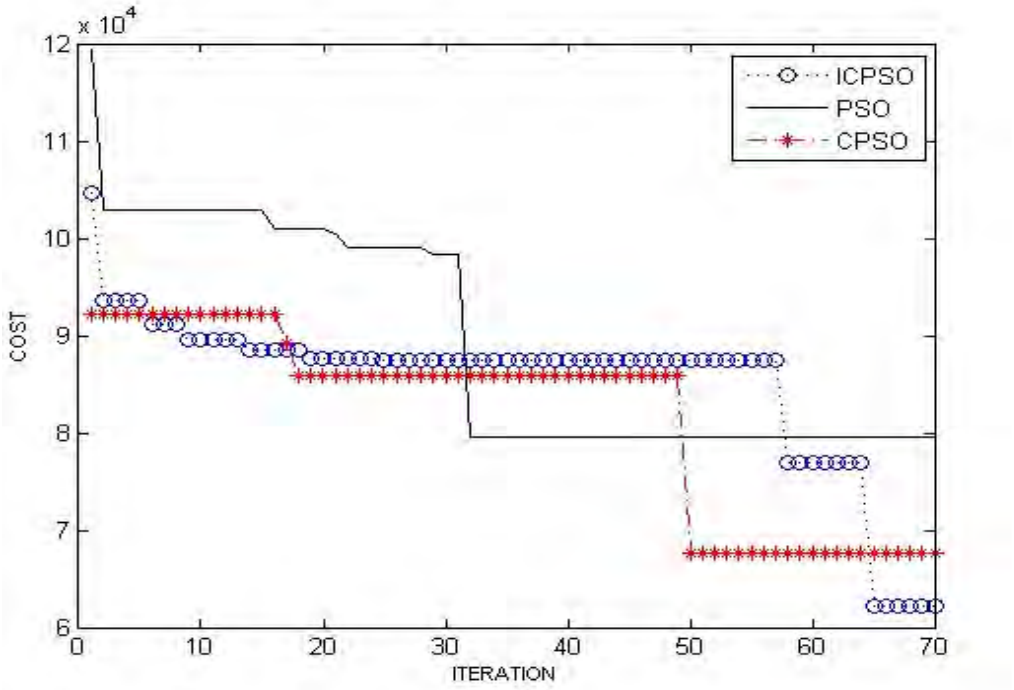

Figure 3: The number of tasks is 50.

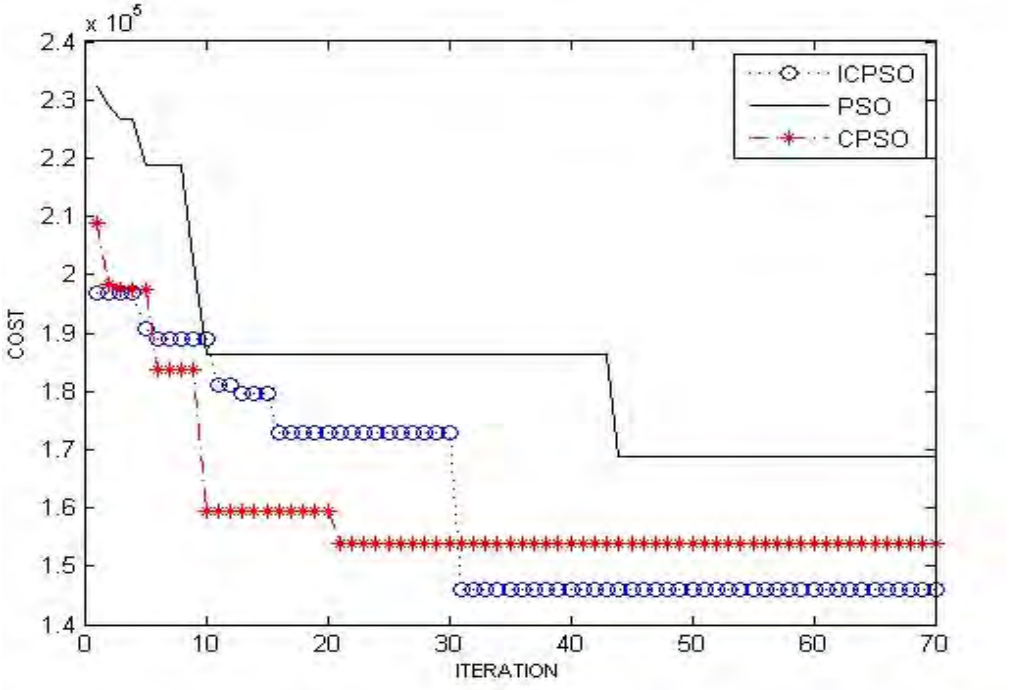

Figure 4: The number of tasks is 100.

The scheduling time of cloud manufacturing batch tasks is also an important reference factor. Experiments simulate the scheduling time for different scales of the tasks production, and the number of production process is set as 10.

Table II: Performance comparison of PSO, CPSO and ICPSO with respect to time criterion.

n	Time (/min)		
	ICPSO	CPSO	PSO
40	1542.3	1543.9	1620.1
50	1978.8	1994.9	2024.2
60	2208.6	2215.3	2269.9
70	2689.8	2689.1	2755.7
80	2933.1	2939.4	3036.0
90	3242.2	3249.5	3382.6
100	3439.5	3445.4	3616.0

The results of simulations show that the discrepancy of time spent on batch tasks scheduling in cloud manufacturing by these three algorithms is not bigger while the number of tasks ranging from 40 to 100. In addition to the number of batch production tasks is 70, other cases the time spent on scheduling by ICPSO is less than the ones by PSO and CPSO. It suggests that ICPSO can not only reduce the production cost of batch cloud manufacturing workshop tasks, also has a certain advantage in the time spent on tasks scheduling at the same time. The less time spent in using the resources of production makes the cost less in the cloud manufacturing environment, which means algorithm used for tasks scheduling is on higher degree of optimization.

As to an algorithm, the critical factor to judge it good or not is whether it can fit into solving large scale of problems. For algorithms solving the cloud manufacturing tasks production problems CPU operation time under different scales of problems is also an important evaluation standard. The CPU time spent in solving problems with different scales by ICPSO is compared. Experiment simulated a variety of production situations, and the results show in the following table. CPU execution time by using ICPSO increases gradually as the problem size increasing. When the number of batch cloud manufacturing tasks is less than 50, the CPU time will be less than 3 seconds. Under the condition that task number is less than 100 and number of production is less than 20, CPU running time will be within 10 seconds. The experiment result shows that ICPSO is highly efficient and applicable to cloud manufacturing production problems with large scales.

Table III: Performance comparison of ICPSO with respect to CPU time criterion.

n	*m*	CPU time (/s)		*n*	*m*	CPU time (/s)		*n*	*m*	CPU time (/s)
30	5	1.308		70	10	2.560		90	20	5.646
40	5	1.419		70	15	3.979		90	25	6.472
40	10	1.861		70	20	4.771		100	15	4.615
50	10	2.450		80	10	2.936		100	20	6.124
50	15	2.852		80	15	4.063		150	25	10.826
60	10	2.299		80	20	5.288		150	30	12.649
60	15	3.626		90	15	4.576		200	30	15.479

5. CONCLUSIONS

Cloud manufacturing is a new kind of manufacturing mode which is based on network and is service-oriented. Traditional manufacturing workshop production is changing forward to cloud manufacturing workshop production mode. The key problem of the cloud manufacturing workshop production is how to reduce production time and cost by reasonable tasks scheduling. In this paper, a dual-objective optimization model is established based on the purpose of reducing time and cost. Solving this problem, in fact, is to solve a multi-objective optimization problem, and the evolutionary algorithm has a very good effect on solving such problems. Improved Cooperative Particle Swarm Optimization (ICPSO) algorithm is used in this paper. Compared with the traditional Particle Swarm Optimization algorithm (PSO) and Cooperative Particle Swarm Optimization (CPSO), ICPSO with fast convergence has a wide population and is easy to jump out of local optimization. Through a lot of simulations it proved that the model and algorithm proposed in this paper are applicable to the mass scale of tasks scheduling in cloud manufacturing, and they can effectively reduce time and cost.

REFERENCES

[1] Xu, X. (2012). From cloud computing to cloud manufacturing, *Robotics and Computer-Integrated Manufacturing*, Vol. 28, No. 1, 75-86, doi:10.1016/j.rcim.2011.07.002

[2] Li, B. H.; Zhang, L.; Wang, S. L.; Tao, F.; Cao, J. W. (2011). Cloud manufacturing: a new service-oriented networked manufacturing model, *Computer Integrated Manufacturing Systems*, Vol. 16, No.1, 1-7

[3] Valilai, O. F.; Houshmand, M. (2013). A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm, *Robotics and Computer-Integrated Manufacturing*, Vol. 29, No. 1, 110-127, doi:10.1016/j.rcim.2012.07.009

[4] Pan, Q.; Wang, L.; Sang, H.; Li, J.; Liu, M. (2013). A high performing memetic algorithm for the flowshop scheduling problem with blocking, *IEEE Transactions on Automation Science and Engineering*, Vol. 10, No. 3, 741-756, doi:10.1109/TASE.2012.2219860

[5] Onwubolu, G.; Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research*, Vol. 171, No. 2, 674-692, doi:10.1016/j.ejor.2004.08.043

[6] Liao, C.-J.; Tseng, C.-T.; Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems, *Computers & Operations Research*, Vol. 34, No. 10, 3099-3111, doi:10.1016/j.cor.2005.11.017

[7] Naderi, B.; Tavakkoli-Moghaddam, R.; Khalili, M. (2010). Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan, *Knowledge-Based Systems*, Vol. 23, No. 2, 77-85, doi:10.1016/j.knosys.2009.06.002

[8] Ishibuchi, H.; Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 28, No. 3, 392-403, doi:10.1109/5326.704576

[9] Buyya, R.; Yeo, C. S.; Venugopal, S.; Broberg, J.; Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Generation Computer Systems*, Vol. 25, No. 6, 599-616, doi:10.1016/j.future.2008.12.001

[10] Laili, Y.; Tao, F.; Zhang, L.; Sarker, B. R. (2012). A study of optimal allocation of computing resources in cloud manufacturing systems, *The International Journal of Advanced Manufacturing Technology*, Vol. 63, No. 5-8, 671-690, doi:10.1007/s00170-012-3939-0

[11] Van den Bossche, R.; Vanmechelen, K.; Broeckhove, J. (2013). Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, *Future Generation Computer Systems*, Vol. 29, No. 4, 973-985, doi:10.1016/j.future.2012.12.012

[12] Fang, Y.; Wang, F.; Ge, J. (2010). A task scheduling algorithm based on load balancing in cloud computing, *Web Information Systems and Mining*, Lecture Notes in Computer Science, Vol. 6318, 271-277, doi:10.1007/978-3-642-16515-3_34

[13] Yu, B.-N.; Jiao, B.; Gu, X.-S. (2009). An improved cooperative particle swarm optimization and its application to flow shop scheduling problem, *Journal of East China University of Science and Technology* (*Natural Science Edition*), Vol. 35, No. 3, 468-474

[14] Liu, D.; Yuan, S.; Zhang, J.; Wu, T. (2008). Optimization design of particle swarm with self-adaptive parameter adjusting, *Transactions of the Chinese Society of Agricultural Machinery*, Vol. 39, No. 9, 134-137

[15] Shi, Y.; Eberhart, R. (1998). A modified particle swarm optimizer, *Evolutionary Computation Proceedings*, *IEEE International Conference on Evolutionary Computation*, *IEEE World Congress on Computational Intelligence*, 69-73, doi:10.1109/ICEC.1998.699146