International Journal of Hybrid Information Technology
Vol. 9, No.7 (2016), pp. 29-42
http://dx.doi.org/10.14257/ijhit.2016.9.7.04

An Improved Multi-Rules-Based ACO Algorithm for FJSS Problem in Cloud Manufacturing Environment

Hongguo Zhang, Linyan He, Chao Ma, Shuli Zhang and Shenghui Liu

School of Software, Harbin University of Science and Technology, China
zhg07@163.com

Abstract

In cloud manufacturing environment, for a scheduling Job, there may be a lot of servicizing manufacturing resources and manufacturing capability that can be used to support for its realizing. Therefore, how to efficiently solve FJSS problem becomes more complex and significant. First, this paper uses disjunctive graph model to analyze the characteristic of FJSS problems, and then, focusing on machine selection sub problem, this paper designs multi-rules to solve machine selection conflicts in different scenarios. Finally, on this basis, an improved multi-rules-based ACO algorithm is proposed. The algorithm is applied to the typical examples of the flexible job-shop scheduling problem. Compared with other algorithms, final experimental results indicate that this algorithm is effective.

Keywords*: Cloud manufacturing environment; Manufacturing resources; Disjunctive graph model; An improved multi-rules-based ACO algorithm*

1. Introduction

Cloud Manufacturing is the integration of multiple techniques including Cloud Computing, Manufacturing, and Internet of Things, *etc.* It is the embodiment of the *Manufacture as a Service* concept. It makes the manufacturing industry be able to provide the products with high value-added, low cost, and also provide the services of global manufacturing. The key point of realizing *Manufacture as a Service* is to transform the existing manufacturing model into cloud manufacturing model. This transformation depends on whether or not the existing environment of manufacturing enterprise can be transformed into cloud manufacturing environment, and also depends on whether or not cloud computing model can be applied to the production and processing in manufacturing shop through manufacturing resources servicizing.

In cloud manufacturing environment, a lot of manufacturing resources and processing capability was carried out servitization, and then was published online. This leads to the situations where these manufacturing resources and processing capability can be shared over the internet. This make the situations in which one machining procedure of workpiece can be processed on multiple machines becomes more common. So it is more meaningful to solve flexible job-shop scheduling (FJSS) problem effectively in cloud manufacturing environment.

In the job-shop scheduling (JSS) problem, n workpieces need to be processed on m machines. It belongs to the resource allocation problem constrained by task sequence and configuration requirement. It is a typical NP-hard problem in combinatorial optimization field. In the traditional JSS problems, one machining procedure of workpiece is only able to be processed on one machine. Compared with this, the FJSS problem is more complex because that it breaks through the limits of machine capability, and then expands the scope of solution. In the FJSS problem, one machining procedure is able to be processed on the multiple related machines. The FJSS problem is more suitable for modeling the actual production plan scheduling in the manufacturing shop.

ISSN: 1738-9968 IJHIT

At present the FJSS problem is generally solved by means of genetic algorithm, particle swarm optimization algorithm, ant colony optimization (ACO) algorithm and other hybrid algorithms [1]. Especially ACO algorithm [2] has been more widely adopted because it has many excellent characters including positive feedback, robustness and so on. But in practical application, ACO algorithm also has some disadvantages, for instance, it is easy to fall into local optimum, and sometimes the convergence rate is very slow.

For overcoming or reducing these disadvantages, Wang [3] presented a new pheromone updating strategy to improve ACO algorithm. Xing [4] added a knowledge model into ACO algorithm so as to improve its efficiency. Liouane [5] put forward a kind of ACO algorithm which is combined with tabu search algorithm, and it may better avoid falling into the local optimum. Taking the deviation between production cycle and delivery date of key workpiece minimum as the optimization goal, Li [6] improve ACO algorithm by adaptively adjusting evaporation rate of pheromone and taking utilization rate of machine as a new heuristic information. Liu [7] modified the updating rule of pheromone. The meaning of this new rule is that the global pheromone should be updated after the current optimal solution is obtained. This rule can make ACO algorithm reduce the running time and improve the efficiency.

By analyzing the existing literature, it is found that the most researchers mainly focus on how to better carry out machining procedure scheduling in the FJSS problem by means of the improved ACO algorithm. And the improvement of ACO algorithm mainly depend on the modification of some rules, for instance, modifying the updating rule of pheromone or adjusting evaporation rate of pheromone. This kind of ACO algorithm can make itself more effectively avoid falling into the local optimum, and then more quickly converge to the global optimal solution.

In view of this case, this paper focuses on how to better solve conflicts in machine selection in the FJSS problem. For different kinds of conflict situation, the multiple corresponding hierarchical rules are given for solving conflicts. Based on these rules, an improved ACO algorithm is presented for better solving the FJSS problem. By selecting machine for the machining procedures of workpiece more reasonably, the efficiency of this improved multi-rules-based ACO algorithm can be enhanced.

2. Problem Description

2.1. Mathematical Model

Compared with the traditional JSS problems, the FJSS problem is more suitable for modeling the actual production plan scheduling in the manufacturing shop. But actually it is also more difficult to solve. In FJSS problem, n workpieces need to be processed on m machines, all the workpieces are composed of one or more machining procedures, these machining procedures need to be processed in specific sequence, each machining procedure can be processed on one or more machines, the processing time of different machines for the same machining procedure may also be different. The mathematical model of FJSS problem is as follows:

Optimized objective:

$$Minimize(C_{max}), \text{ and } C_{max}=Max\{CT_{ijk}\}$$

In which, CT_{ijk} denates the completion time of the jth machining procedures of the ith workpiece on the kth machine.

Constraints:

Cons1: At a given time, a machine can process at most one machining procedure of workpiece. It becomes available to other machining procedures only if the processing is completed.

Cons2: At a given time, a workpiece can be processed on at most one machine, and once the processing starts, it cannot be interrupted until it is completed.

Cons3: Different workpieces have the same priority;

Cons4: There are no precedence constraints among the machining procedures of different workpieces;

Cons5: Each workpiece consists of a predetermined sequence machining procedures, *i.e.*, $S_{ijk} \geq CT_{i(j-1)q}$, it denotes the start time of processing the jth machining procedures of the ith workpiece must be greater than or equal to the completion time of the (j-1)th machining procedures of the ith workpiece;

Cons6: The start time of processing each machining procedure must be greater than or equal to zero, *i.e.*, $S_{ijk} \geq 0$;

A 4×3 instance in literature [8] given in Table 1. It is an incomplete FJJS problem. J_i denotes the ith workpiece, O_{ij} denotes the jth machining procedures of the ith workpiece. As can be seen from table 1, the processing time of the machining procedure O_{11} on the machine M_1 is 2, and its processing time on the machine M_2 is 3. But the processing time of the machining procedure O_{11} on the machine M_4 is +∞, which means that the machining procedure can not be processed on M_4.

Table 1. Processing Time of the 12 Machining Procedures on the 6 Machines

Machining procedures		Machines and processing time					
		M_1	M_2	M_3	M_4	M_5	M_6
J_1	O_{11}	2	3	4	+∞	+∞	+∞
	O_{12}	+∞	3	+∞	2	4	+∞
	O_{13}	1	4	5	+∞	+∞	+∞
J_2	O_{21}	3	+∞	5	+∞	2	+∞
	O_{22}	4	3	+∞	+∞	6	+∞
	O_{23}	+∞	+∞	4	+∞	7	11
J_3	O_{31}	5	6	+∞	+∞	+∞	+∞
	O_{32}	+∞	4	+∞	3	5	+∞
	O_{33}	+∞	+∞	13	+∞	9	12
J_4	O_{41}	9	+∞	7	9	+∞	+∞
	O_{42}	+∞	6	+∞	4	+∞	5
	O_{43}	1	+∞	3	+∞	+∞	3

2.2. Disjunctive Graph Model

Avery popular way to depict shop scheduling instances is the disjunctive graph $G = (V, C, D)$, where V is the set of nodes, C is the set of conjunctive (directed) arcs, and E is the set of disjunction (undirected) arcs. Given an instance of the flexible job shop scheduling problem, the disjunctive graph G is obtained as follows: For each procedure $O_{ij} \in O$(all procedures), a node $v_0 \in$ V(the dotted ovals except 0 and 1)is introduced. The solid ovals which are in dotted ovals include the known processing time on optional machine for given procedure. In the following we identify the nodes of G with the corresponding operations.

Furthermore, for each pair of procedure O_{ij}, $O_{it} \in O$ ($j < t$)with $O_{ij} < O_{it}$, a solid line with single arrow arc $a_{\mathrm{Oij,Oit}} \in$ C is introduced. Finally, for each pair of procedure $O_{ij}, O_{st} \in O$ with $m(O_{ij})=m(O_{st})$ and a dotted line arrow arc $e_{\mathrm{Oij,Ost}} \in$ D is introduced. $m(O_{ij})=m(O_{st})$ denotes that two connected procedures are likely to be processed on the same machine at the same time. Figure 1 show the disjunctive graph of a instance with 12 operations partitioned into 4 jobs, 6 machines from table 1:$O=\{O_{11},O_{12},\ldots O_{42},O_{43}\}$,$J=\{J_1=\{O_{11},O_{12},O_{13}\},J_2=\{O_{21},O_{22},O_{23}\},J_3=\{O_{31},O_{32},O_{33}\},J$

$_4=\{O_{41},O_{42},O_{43}\}\}$, $M=\{M_1=\{O_{11},O_{13},O_{21},O_{22},O_{31},O_{41},O_{43}\}$, $M_2=\{O_{11},O_{12},O_{13},O_{22},O_{31},O_{32},O_{42}\}$, $M_3=\{O_{11},O_{13},O_{21},O_{23},O_{33},O_{41},O_{43}\}$, $M_4=\{O_{12},O_{32},O_{41},O_{42}\}$, $M_5=\{O_{12},O_{21},O_{22},O_{23},O_{32},O_{33}\}$, $M_6=\{O_{23},O_{33},O_{42},O_{43}\}\}$. If it's directed acyclic, get an optimal processing sequence on the selected machine. In the same way, for all selected processing machine, get a directed acyclic G', which cover all nodes and $G'=(V,C\cup D')$, achieving the minimum makespan.

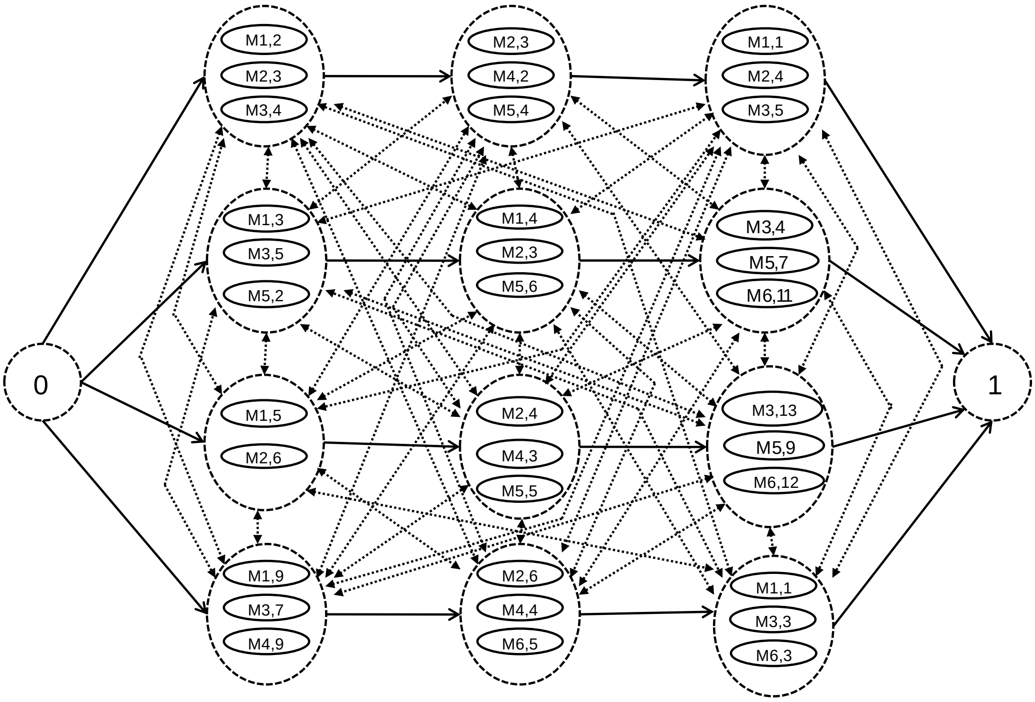


Figure 1. Disjunctive Graph of the Instance in Table 1

3. Improved ACO Algorithm

3.1. Heuristic Rules for Solving Conflicts in Machine Selection

A FJSS problem is composed of two sub-problems: machine selection and machining procedure scheduling, and its difficulty is how to solve the conflicts in machine selection. For solving it, a set of heuristic rules for different scenarios is proposed, which is used to select the most optimized machine to make the completion time of all the machining procedures become shortest.

3.1.1. *Rule1*: Selecting the Machine with the Shortest Processing Time

When selecting a machine from the multiple optional machines to process a machining procedure, the simplest scenario is as follows:

Condition1: At the starting time of this machining procedure can be processed, the state of all the optional machines is idle;

Condition2: For this machining procedure, the completion time of all the optional machines is different;

Condition3: At this time, only one machining procedure needs to carry out machine selection.

In this simplest scenario, a machining procedure can be processed by one or more related machines and the corresponding processing time on each machine is known. The *Rule1* can be described by the formula $Max\{1/T_{ijk}\}$, in which T_{ijk} denotes the processing time of the *j*th machining procedure of the *i*th workpiece on the *k*th machine. The meaning of *Rule1* is that these machines need to be sorted according to the processing time T_{ijk} firstly, and then the machine with shortest processing time should be selected to process this machining procedure.

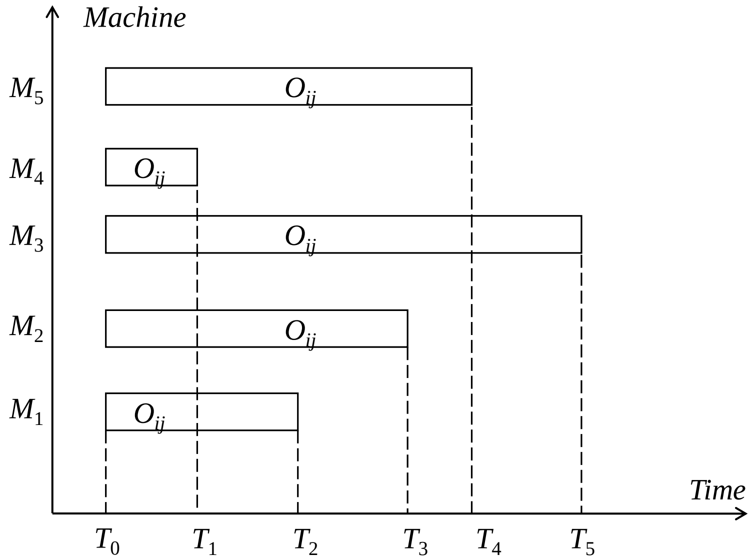


Figure 2. The Gantt Chart of the Machining Procedure O_{ij} Selecting Machine

The formula $Max\{1/T_{ijk}\}$ is suitable for this simplest scenario. An example is given as shown in Figure 2, the machining procedure O_{ij} can be processed on the five machines at T_0 time. The result can be obtained through the concrete computation process: $Max\{1/T_{ijM1}, 1/T_{ijM2}, 1/T_{ijM3}, 1/T_{ijM4}, 1/T_{ijM5}\}$. As can be seen from the Figure 2, T_{ijM4} is the time frame $[T_0, T_1]$, and it is the smallest, so the machine M_4 is the best option.

But sometimes, at the starting time of one machining procedure can be processed, the state of some optional machines is not idle. Especially the state of the machine with shortest processing time may be not idle. For this type of scenario, the machine selection of *Rule1* should be executed by the formula:

$$Max\{1/CT_{ijk}\}= Max\{1/(Max(AT_{ij}, ST_k)+ T_{ijk})\}, k=1, 2, \ldots, N_1 \quad (1)$$

In the formula (1), CT_{ijk} denotes the completion time of the procedure O_{ij} on the machine *k*, AT_{ij} denotes the starting time of the procedure O_{ij} can be processed, and ST_k denotes the starting time of the machine *k* enters the current idle state. N_1 denotes the number of the optional machines that is able to process the procedure O_{ij}.

In this paper the concept of scenario is represented by *Scenario*=<*Condition1*, *Condition2*, *Condition3*>. *Rule1* is proposed for solve the machine selection problem in *Scenario1*, and *Scenario1*=<*false*, *true*, *true* > which means that *Condition1* is not satisfied, but both *Condition2* and *Condition3* are satisfied.

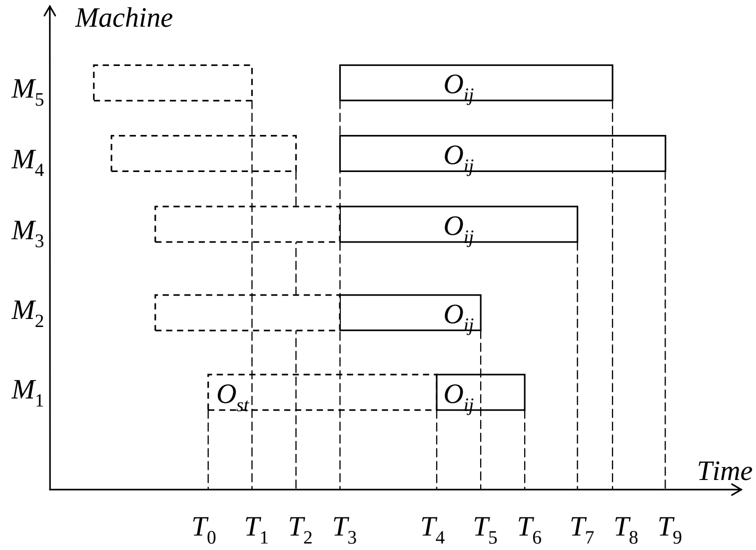


Figure 3. The Gantt Chart of the Machining Procedure O_{ij} Selecting Machine

An example is given as shown in Figure 3, the solid line rectangles represent the time frames in which the procedure O_{ij} is being processed on M_1, M_2, M_3, M_4, M_5, and the dotted line rectangles represent the time frames in which the other machining procedures are being processed on M_1, M_2, M_3, M_4, M_5. It is assumed that machine M_2, M_3, M_4, M_5 are idle at T_3 time, but machine M_1 is not idle until T_4 time. In the time frame [T_0, T_4], the procedure O_{st} is being processed on M_1, and M_1 enters the current idle state at T_4 time.

The result can be obtained by the concrete computation process: $Max\{1/(Max(AT_{ij}, ST_1)+T_{ijM1}), 1/(Max(AT_{ij}, ST_2)+T_{ijM2}), 1/(Max(AT_{ij}, ST_3)+T_{ijM3}), 1/(Max(AT_{ij}, ST_4)+T_{ijM4}), 1/(Max(AT_{ij}, ST_5)+T_{ijM5})\} = Max\{1/(Max(T_3, T_4)+T_{ijM1}), 1/(Max(T_3, T_3)+T_{ijM2}), 1/(Max(T_3, T_3)+T_{ijM3}), 1/(Max(T_3, T_2)+T_{ijM4}), 1/(Max(T_3, T_1)+T_{ijM5})\}$ $=Max\{1/T_6, 1/T_5, 1/T_7, 1/T_8, 1/T_9\}$. As can be seen from the Figure 3, T_5 is the smallest, so the machine M_2 is the best option.

A special case is that T_{ijM1} may be very short, and this case makes $T_4+ T_{ijM1}< T_3+T_{ijM2}$, so the best option become the machine M_1. It means that even one machine is not idle at the starting time of one procedure can be processed, and then this machine also can be the best option for this procedure.

3.1.2. *Rule2*: Selecting the Machine with the Longest Remaining Available Time

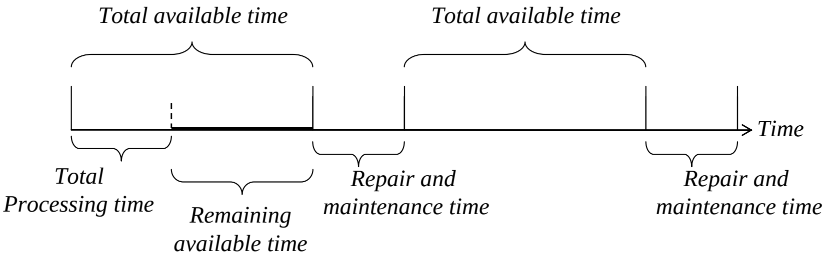
When selecting a machine from the multiple optional machines to process a machining procedure by means of *Rule1*, the best option is more than one. It means that for this machining procedure, the completion time of some optional machines is same (*i.e.* *Condition2* is not satisfied). The *Scenario2* =<*false*, *false*, *true* > is used to describe this case.

For solving the machine selection problem in *Scenario2*, *Rule2* is proposed. *Rule2* can be represented by the following formula:

$$Max\{RVT_k\}, k=1, 2, \dots, N_2, N_2\leq N_1 \quad (2)$$

In the formula (2), RVT_k denotes the remaining available time of the machine k, and N_2 denotes the number of the optional machines with the same and shortest completion time for the procedure O_{ij}, N_1 denotes the number of the optional machines that is able to process the procedure O_{ij}.

In the actual manufacturing floor shop, the machines need regular repair and maintenance, and they cannot always be in the available state. For modeling this case, the concept of the total available time is defined here. It means that once the total time of one machine process the machining procedures is equals to the total available time, the machine will need to be carried out repair and maintenance.



An example: ***When*** *the procedures O_{ij} and O_{st} have be processed on the machine k*

Then *Total Processing time=$T_{ijk}+T_{stk}$*

Figure 4. Rvt_k, Tvt_k, Tpt_k of the Machine *K*

In the formula (2), $RVT_k=TVT_k-TPT_k$, as show in Figure 4, TPT_k represents the total processing time of the machine *k* have processed the machining procedures, and TVT_k denotes the total available time of machine *k*.

3.1.3. *Rule3*: Selecting the Local Optimal Plan of Machine Selection

When a machine has been selected for one machining procedure by means of *Rule1* and *Rule2*, there may be another machining procedure that also selected this machine at the same time. This type of scenario can be represented by *Scenario3* =<*false*, *false*, *false*>. In *Scenario3*, there may be more than one machining procedures which need to carry out machine selection at the same time. For solving the machine selection conflicts in *Scenario3*, *Rule3* is proposed.

In *Scenario3*, there may be *N* machining procedures which need to carry out machine selection at *T* time. And by means of *Rule1* and *Rule2*, their best options are obtained, it is the machine M_{best}, T_{best} is the time frame [T, $CT_{ijMbest}$]. The selecting the local optimal plan (*SLOP*) algorithm (*i.e*. *Rule3*) is as following:

1. **Sort**(*MP*[*i*]);
2. **For** (i=1, $i \leq N$, i++)
3. **If** ($i \times T_{best} < MP[i].T_{so}$) **Then**
4. **Insert** *MP*[*i*] **into** *SPS*;
5. **Else Insert** *MP*[*i*] **into** *PPS*;
6. **End If**
7. **End For**

In this algorithm, *MP*[*i*] is a set of machining procedures, i=1,...,N. The function **Sort**(*MP*[*i*]) is used to sort *N* machining procedures by the time frame $MP[i].T_{so}$ (largest to smallest). $MP[i].T_{so}$ denotes the time frame [T, $CT_{ijMso[i]}$], where the machine *Mso*[*i*] denotes the suboptimal option of the machining procedures *MP*[*i*].

SPS denotes the set of the machining procedures, and these machining procedures are all assigned to the machine M_{best}, and then are all processed on the machine M_{best} in serial mode. So, a sequence of processing machining procedures on the machine M_{best} can be obtained. *PPS* also denotes the set of the machining procedures, but these machining procedures are all assigned to their corresponding suboptimal option *Mso*[*i*], and they are all processed in parallel mode, together with the sequence of processing machining procedures on the machine M_{best}.

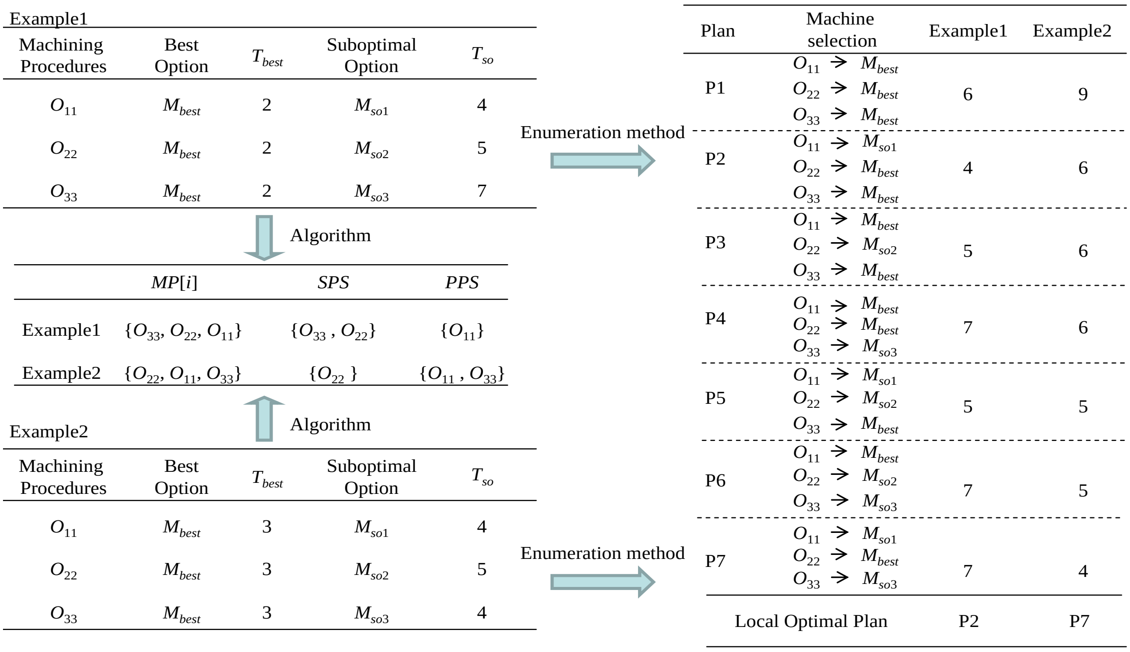


Figure 5. Local Optimal Plan of Machine Selection in *Scenario3*

As shown in Figure 5, there are three machining procedures O_{11}, O_{22} and O_{33} in Example1, and their best options are all M_{best}, and T_{best}=2.Their suboptimal options of O_{11}, O_{22} and O_{33} are M_{so1}, M_{so2}, M_{so3} respectively. By means of *SLOP* algorithm, the intermediate results are obtained: the sorted $MP[i]$ is {$O_{33,}$ O_{22}, O_{11}}, both O_{33} and O_{22} are processed on the machine M_{best} in serial mode, O_{11} is assigned to on M_{so1}, and it is processed in parallel mode, together with he sequence <O_{33}, O_{22} >. So the local optimal completion time is $Max\{O_{11}.T_{so}, (O_{22}.T_{best} + O_{33}. T_{best})\}$=4. By means of enumeration method, all the machine selection plans are listed for Example1, as can be seen from Figure 5, the local optimal plan is P2, its completion time is also 4.

In order to better verify the *SLOP* algorithm, anther example Example2 are given in Figure. It is also executed by means of *SLOP* algorithm and enumeration method respectively. It seems that the function of *SLOP* algorithm and enumeration method is same, but when *N* becomes larger, the enumeration method is different to realize. So, *SLOP* algorithm is necessary.

In summary, for solving the machine selection sub-problems, the multiple rules are proposed can suitable for multiple different scenarios. The mapping relationship between multiple scenarios and multiple rules is given in Table 2.

Table 2. Multiple Rules Match Multiple Scenarios

Scenario=<*Condition1*, *Condition2*, *Condition3*>	Rule
Scenario0= < true, true, true >	*Rule1*
Scenario1= <false, true, true >	*Rule1*
Scenario2= <false, false, true >	*Rule1+ Rule2*
Scenario3= <false, false, false>	*Rule1+ Rule2+Rule3*

3.2. Machining Procedure Scheduling

ACO [9] algorithm is an intelligent bionic optimization algorithm. It has the advantages of distributed computing, strong robustness, positive feedback, and self organization. But it also has the shortcoming of falling into local optimal solution easily and long search time. This paper combines the advantages of the max-min ant system to analyze the ant colony algorithm, and put forward an improved ant colony algorithm to overcome the shortcoming.

3.2.1. State Transition Rules

In the process of scheduling forming, the ant *k* in procedure *i* selects the procedure *j* to move by applying the following state transition rule:

$$P^k{}_{ij}(t_c) = \begin{cases} \dfrac{[\tau_{ij}(t_c)]^\alpha [\eta_{ij}(t_c)]^\beta}{\sum\limits_{s \in Sk} [\tau_{is}(t_c)]^\alpha [\eta_{is}(t_c)]^\beta} & s \in Sk \\ 0 & s \notin Sk \end{cases} \tag{3}$$

Where, $P^k{}_{ij}(t_c)$ is the probability with that ant *k* chooses to move from node *i* to node *j*. $\tau_{ij}(t_c)$ is the pheromone trail on the edge (*i*,*j*); $\eta_{ij}(t_c)$ is the visibility from node *i* to node *j* ; α is a parameter that allow a user to control the relative importance of pheromone trail($\alpha > 0$); β is a parameter that determines the relative importance of heuristic information.($\beta > 0$). Duan [10] and Liu[11] have carried on a large number of experiments in the literature to get a certain scope of α,β which this paper adopts; Visibility $\eta_{ij}(t_c)$ is calculated by the formula (4).

$$\eta_{ij}(t_c) = 1/(T_{ijk} + h) \tag{4}$$

Where T_{ijk} is the processing time when the procedure *j* of the workpiece *i* is processed on the machine *k*. When processing time is shorter, the high visibility, greater attraction to ants, ants select the tendency of the node is higher. But when T_{ijk} equals 1, the parameter of the visibility doesn't work in the formula 4, in order to ensure the α,β can impact in the formula exactly, add constant *h* in the denominator and make *h*=5 to ensure that β has influence on the choice of nodes.

3.2.2. Pheromone Update

This paper adopts the iterative optimal global updating based on Thomas Stuetzle [12], ant-cycle model and traditional ant colony algorithm. This updating is performed after all ants completed their schedules. The pheromone trail level is updated as follows:

$$\tau_{ij}(t+1) = (1-\rho) \bullet \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau^k{}_{ij}(t) \tag{6}$$

$$\Delta\tau^k{}_{ij}(t) = \begin{cases} Q / L_k \\ 0 \end{cases} \tag{7}$$

In formula 5, ρ denotes the pheromone evaporating parameter. $\rho(0 < \rho < 1)$ is from the experiment result of Chunyu Wu [12]; $\Delta\tau^k{}_{ij}(t)$ denotes the pheromone which is left between node *i* and node *j* by ant *k* which get the optimal solution in the current iteration. L_k is the objective function value of the best schedule up to the current iteration; *Q* denotes the pheromone amount, and it is a constant.

3.3. The Pseudo Code and Flow Chart of Algorithm

There is a parallel mechanism when ant chooses its path, it not only chooses the procedure, but also the procedure chooses machine. According to this situation, an ant colony algorithm is adopted in this paper. Its pseudo code is shown as follows:

```
Initialize parameters: Tmax, Gk, Sk, Jk;
For (T=0, TOP=∞; T≤Tmax; T++)
    For(i=1, AS=0; i≤N; i++)
        While (Gk=ϕ)
            Select the next node by formula (3), and AS=AN;
            Insert AN into the set Sk;
            Delete AN from the set Gk;
            Select the best machine through Multi-rules, and ASM=Mbest;
            Add AN into the set Jk;
        End While
        If(TOP＞Sp) Then
            TOP=Sp;
        End If
    End For
    Output: TOP;
End For
```

where, T_{max} denotes maximum iteration; G_k denotes the set of procedures which have not been processed; S_k denotes the set of procedures which is allowed to processed in next step; J_k denotes the set of procedures which have been processed; *TOP* denotes the optimal solution; *N* denotes a constant which can be changed by the size of the example.

4. Experimental Result

In order to evaluate the performance of the proposed algorithm, a numerical experiment is carried out. Instances in this paper and the data in literature [13] are used as test data. In the experiment, JAVA is used to implement the code of the algorithm. All simulations were run on an Inter Pentium CPU at 2.6 GHz, 512 MB RAM running Microsoft Windows 2000.

In the processing system, the parameter in the literature [1] used for the test runs are: Q_m=40,T_{max}=200,α=[10,30],β=[5,10],ρ=[0.15,0.4].The instance on the Table 1 has been experimented for 200 times based on the algorithm in the paper. Its optimal solution is 17 and it is superior to the literature [8]. Gantt chart is shown in Figure 6 and Convergence curve of different algorithms is shown in Figure 7.

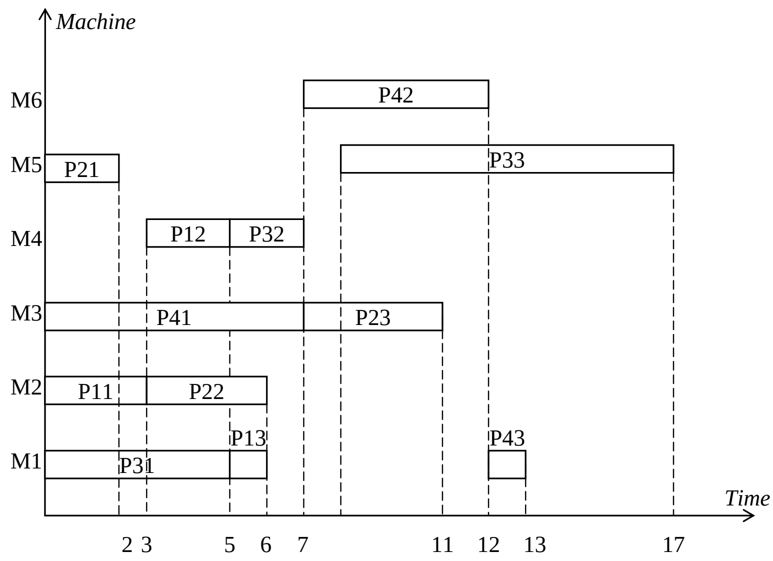


Figure 6. Gantt Chart of the Optimal Solution

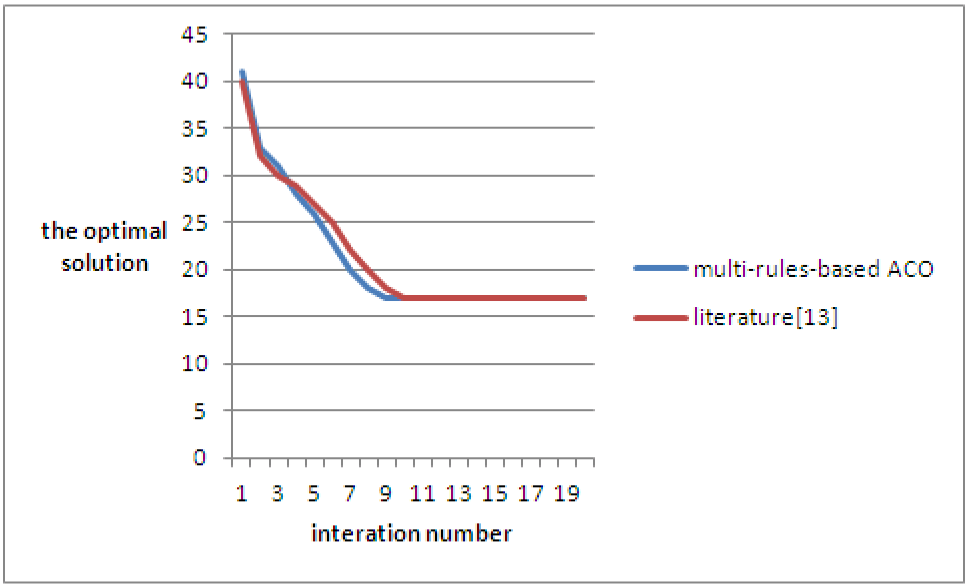


Figure 7. Convergence Curves

In Figure 6, every machine participates in the processing, the utilization rate is maximized as possible. The proposed algorithm can obtain the optimal solution in a relatively short time and relatively less iterations. It can also shorten the average convergence time. In order to evaluate the effectiveness of the proposed algorithm more exactly, three instances including 8×8, 10×10, 15×10 of the Kacem benchmark problem are adopted. Experimental environment and parameter are the same as those in the preceding example. The results compared with literature [7], literature [14] and literature [15] is shown in Table 3.

Table 3. The Comparison Results between Proposed Algorithm and Other Algorithm

Method	8×8		10×10		15×10	
	optimal solution	average solution	optimal solution	average solution	optimal solution	average solution
Multi-rules	14	15.6	7	7.8	11	12.5
AL+CGA	15	-	7	-	23	-
literature [5]	14	15.5	7	7.8	12	12.8
literature [12]	14	17.6	7	8.1	-	-

As can be seen from Table 3, the proposed algorithm on completing the examples get the optimal solution C_{max}, showing the effectiveness of the proposed algorithm in solving the flexible job shop scheduling problem.

5. Conclusion

In view of the flexible job-shop scheduling problem, we use disjunctive graph model to perform the characteristics of flexible job shop scheduling problem. According to the machine selection, we adopt the method of heuristic rules to solve this problem, because this method is simple and effective. At the same time, adopt the updating pheromone for the optimal solution in the current iteration to reduce the time complexity and improve the efficiency of solving the problem. Finally the proposed algorithm is applied to several typical examples in the flexible job shop scheduling problem, and compared with the results of other algorithms in the literature. The experimental results indicate the effectiveness of the proposed algorithm.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.51375128) and Science and Technology research project of Heilongjiang Provincial Education Department (12541159).

References

[1] Chakravorty, "Neuro Inspired Genetic Hybrid Algorithm for Active Power Dispatch Planning Problem in Small Scale System", International Journal of Hybrid Information Technology, vol.8, no.9, **(2015)**, pp. 171-184.

[2] X. Wei and P. Zhang, "Study on Thinking Evolution based ant Colony Algorithm in Typical Production Scheduling Application", International Journal of Hybrid Information Technology, vol.8, no.6, **(2015)**, pp.125-134.

[3] L. Wan, C. Zhao and X. Jing, "The solving method based on improved ant colony algorithm of flexible job shop scheduling problem", Journal of System Simulation, vol. 20, no. 16, **(2008)**, pp. 4326-4329.

[4] L.-N. Xing, Y.-W. Chen and P. Wang, "A Knowledge-based Ant Colony Optimization for Flexible Job Shop Scheduling Problems", Applied Soft Computing, vol. 10, no. 3, **(2010)**, pp. 888-896.

[5] N. Liouane, I. Saad and S. Hammadi, "Ant System & Local Search Optimization for Flexible Job Shop Scheduling Problems", International Journal of Computers, Communications & Control, vol. 2, no. 2, **(2007)**, pp. 174-184.

[6] Y. Li, H. Chen and S. Wang, "The adaptive ant colony algorithm in the application of two-way production workshop scheduling", Journal of Management and Management, no. 3, **(2008)**, pp. 160-163.

[7] Z. Liu, W. Lv and Q. Xie, "Applicating the improved ant colony algorithm to solve the flexible job-shop scheduling problem", Journal of Industrial Engineering and Management, vol. 15, no. 3, **(2010)**, pp. 115-119.

[8] X. Yang and J. Zeng, "Genetic algorithm to solve the flexible job shop scheduling problem", Control and Decision, vol. 19, no. 10, **(2004)**, pp. 1197-1200.

[9] H. Xie, H. Chen and C. Wang, "A Priority-driven ACO Algorithm for DAG Task Scheduling in Cloud Environment", vol.8, no.6, **(2015)**, pp. 205-216.
[10] H. Duan, D. Wang and X. Yu, "Research on the Optimum Configuration Stategy for the Adjustable Parameters in Ant Colony Algorithm", Journal of Communication and Computer, vol. 2, no. 9, **(2005)**, pp. 32-35.
[11] L. Liu, Y. Dai and L. Wang, "Ant colony algorithm parameters optimization", Computer Engineering, no. 11, **(2008)**, pp. 208-210.
[12] T. Stutzle and H.H. Hoos, „MAX-MIN Ant System", Future Generation Computer Systems, vol. 16, no. 8, **(2000)**, pp. 889-914.
[13] C. Wu, Z. Chen and M. Jiang, "The system initialization and system parameters research in the ant colony algorithm", Journal of Electronics, no. 8, **(2006)**, pp. 1530-1533 .
[14] Q. Liu, C. Zhang and Y. Rao, "Improved genetic algorithm to solve the flexible job-shop scheduling problem", Journal of Industrial Engineering and Management, vol. 14, no. 2, **(2009)**, pp. 59-66.
[15] I. Kacem, S. Hammadi and P. Botne, "Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems", IEEE Transactions on Systems, Man and Cybernetics, Part C, vol. 32, no. 1, **(2002)**, pp. 1-13.