J Intell Manuf (2018) 29:891–904
https://doi.org/10.1007/s10845-015-1144-3

An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time

W. Qin[1] · **J. Zhang**[1] · **D. Song**[1]

Received: 30 March 2015 / Accepted: 22 August 2015 / Published online: 3 September 2015

Abstract In this article the scheduling problem of dynamic hybrid flow shop with uncertain processing time is investigated and an ant colony algorithm based rescheduling approach is proposed. In order to reduce the rescheduling frequency the concept of due date deviation is introduced, according to which a rolling horizon driven strategy is specially designed. Considering the importance of computational efficiency in the dynamic environment, the traditional ant colony optimization is improved. On the one hand, a strategy of available routes compression to restrict ants' movement is proposed so that the ants' searching cycle for new solutions could be shorten. On the other hand, illuminating function in state transfer possibility is improved to facilitate the exploration of low pheromone trail. Performance of rolling horizon procedure and rescheduling algorithm are evaluated respectively through simulations, the results show the best parameters of rolling horizon procedure and demonstrate the feasibility and efficiency of rescheduling algorithm. An example from the practical production is addressed to verify the effectiveness of the proposed approach.

Keywords Hybrid flowshop · Uncertain processing time · Ant colony algorithm · Rolling rescheduling strategy

Introduction

This article investigates a hybrid flowshop (HFS) scheduling problem, also called a flexible flow line or a flowshop with multiple machines on some or all production stages. The machines of each stage are unrelated and in parallel, which means that a job can be processed on any one of those machines and the processing time at each stage is different. The flow of jobs through the HFS is unidirectional. Each job is processed by one machine in each stage and it must go through all stages (Linn and Zhang 1999).

The scheduling problem for HFS has received extensive study in literatures. The first research paper about hybrid flexible flow shop appeared in the 1970's. Salvador (1973) published one of the pioneer papers on HFS by modelling the production system in the synthetic fibres industry as a no-wait HFS. Garey and Johnson (1979) demonstrated that the HFS problem with makespan objective is NP-complete. Kis and Pesch (2005) reviewed exact methods for the k-stage HFS problem with identical machines to minimize makespan. Wang et al. (2011) studied a hybrid flow shop with multiprocessor tasks, in which a set of independent jobs with distinct processor requirements and processing times must be processed in a k-stage flow shop to minimize the makespan. Zhang et al. 2014 studied a novel three-stage hybrid flow shop problem, in which the first and third stages contain many batching machines and non-batching machines, and the second stage contains non-batching machines. Babu et al. (2014) studied an integrated problem of port operations in the coal import instance of stockyard. The unloading, storing and transferring of coal in different stages are taken into consideration. A two heuristic-based greedy construct algorithms is proposed to solve the problem. Yang (2015) considered a two-stage hybrid flow shop scheduling with dedicated machines at stage 1 with the objective of minimizing the total completion time. There exist two machines at stage 1 and one machine at stage 2. Each job must be processed on one of the two dedicated machines at stage 1 depending on the job type; subse-

✉ J. Zhang
zhangjie@sjtu.edu.cn

1 School of Mechanical Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, China

quently, the job is processed on the single machine at stage 2.

In most research of HFS scheduling, predetermined processing time of various operations is necessary for effective results. However in real-life production with dynamic environment, due to differences between machines, skill levels of workers and so on, it is difficult to designate each processing time a fixed value. Therefore due to such an uncertainty, the scheduling scheme may lead to a large deviation in the actual implementation, and in some case even cannot be executed. Actually, the deviation of processing time of operations is a widespread phenomenon in the workshop and has become the most common dynamic event with uncertainty.

Xu and Gu (2005) use the method of fuzzy theory to handle the uncertainty of system parameters in a no-wait flow shop. But the method does not reflect real-time changes in the characteristics of the plant. Zhao et al. (2008) construct a framework of collaborative production scheduling, in which the fuzzy approach is proposed to judge the probability of production time distribution. They use the proposed method to get feasible solutions for outsourcing production and studied the robustness of the results.

Mehta et al. classify dynamic scheduling into three categories: completely reactive scheduling, predictive–reactive scheduling and robust pro-active scheduling. In completely reactive scheduling no schedule is generated in advance and decisions are made in real-time. Priority dispatching rules are frequently used. Predictive–reactive scheduling is the most common dynamic scheduling approach used in manufacturing systems. It is a scheduling/rescheduling process in which schedules are revised in response to dynamic events. Robust pro-active scheduling approaches focus on building predictive schedules which satisfy performance requirements predictably in a dynamic environment. The main difficult of this approach is the determination of the predictability measures. Therefore, Predictive–reactive scheduling has been widely studied for dynamic scheduling problem. Wu (2007) deal with the dynamic jobshop scheduling problem from the perspective of strategic, but the rolling optimization for better scheduling needs to be further expanded. Liu et al. 2012 propose a concept of critical process set under the frame of rolling scheduling strategy, and take use of hybrid genetic algorithm to determine the critical process set as well as its corresponding optimal scheduling solution.

Although the problem of uncertainty in shop scheduling has attracted some researchers in recent years, most of them rely on probability theory or fuzzy method to evaluate the robustness of the originally generated schedules, which therefore lack of the real-time response to the occurrence of uncertain events. In terms of the research on dynamic scheduling, most of them focus on the analysis and processing for some single, dominant events such as random arrival of urgent orders or machine breakdowns, while the investigation of recessive uncertain processing time is comparatively limited.

In order to fill up with the research gap, this paper investigates the dynamic HFS scheduling problem with uncertain processing time. An effective rescheduling strategy based on the rolling horizon procedure is developed according to the causes of uncertainty. An improved ant colony optimization is proposed to establish the systematic and complete scheduling solution.

The remainder of this paper is organized as follows. We begin in Section "Problem description" by describing the HFS scheduling problem with uncertain processing time. In Section "Rolling rescheduling strategy", we introduce the rolling scheduling strategy. In Section "Improved ant colony algorithm" an improved ant algorithm is addressed. In Section "Numerical results", we give numerical results. The conclusion is drawn in Section "Conclusions".

Problem description

This research for HFS scheduling has been motivated by a real-life problem faced by a semiconductor manufacturer which is specialized in printed circuit board (PCB) assembly. In the PCB assembly shop, all jobs containing lots of PCBs will be processed through 4 shops (stages), which is shown in Fig. 1. The four stages include: surface mount assembly (one side or two sides), inserting, wave soldering and testing/packing. In each stage, a job can be processed by several unrelated machines. In such a typical HFS scheduling problem, it not only needs to sequence all the PCB jobs, but also to take the assignment problem of the parallel machines for each job.

Suppose there are m machines and n jobs in a hybrid flow shop. Each job consists of items within the same category. In each stage the job can be processed by several machines which are differentiated by the processing time of operations. The set up time for changing batches on a machine is required and it is related to the sequence of the two batches.

Assumptions are listed as follows:

(1) All jobs are available and can be processed at time zero;
(2) Since all jobs are processed in a flow shop, any job cannot enter the next operation before the previous operation is completed;
(3) Processing time is associated with the machine and has a pre-estimated value, which deviates from the actual value;
(4) Set up time for changing batches is related to the sequence of the two batches, which contains the set up time of machines and delivery time of jobs;
(5) One machine can process only one job at a time and one job can be processed by only one machine at any time;

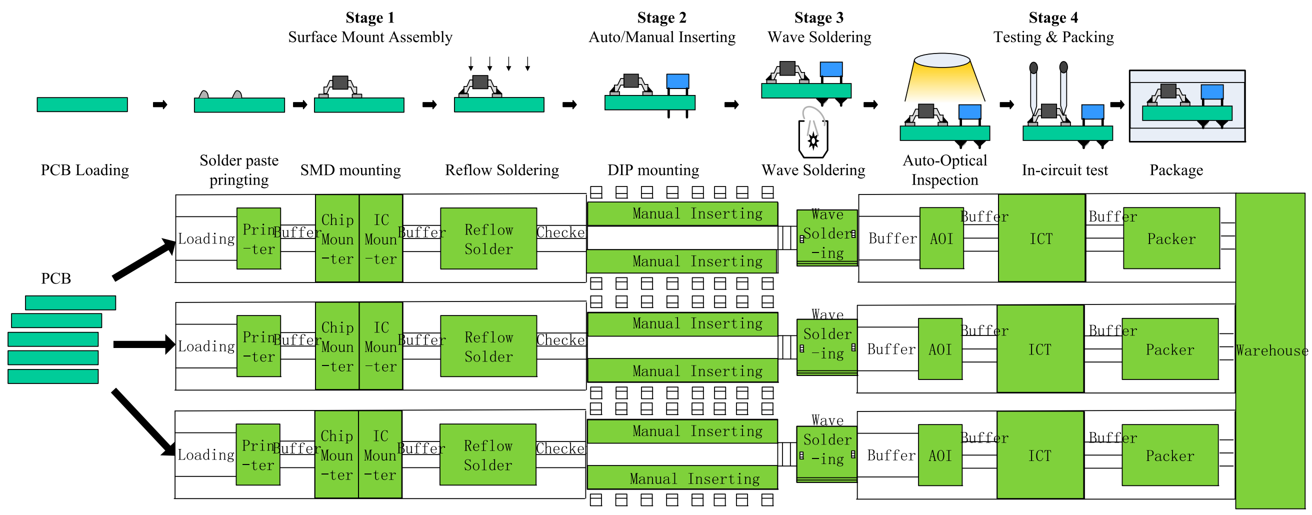


Fig. 1 Layout of PCB assembly shop

(6) Operations of jobs belonging to different batches do not have constraints between each other;
(7) Once the job is processed, it cannot be interrupted until the whole batch is completed.

The processing time is affected by the material type, batch size, manufacturing processes, machine status (new or old) and workers' efficiency. In the actual production process it is difficult to determine a fixed value. According to queuing theory, the normally used distribution model includes the negative exponential distribution, shift negative exponential distribution and Erlang distribution (Bose 2002). However, in the negative exponential distribution the occurrence of small probability events appears a relatively high probability, which implies that such a model is not suitable in the actual production. Thus in this paper, the Erlang distribution model is adopted to simulate the real processing time in the manufacturing system. The basic probability density function is:

$$f(t) = (k\lambda)^k \left[\frac{e^{-k\lambda}}{(k-1)!}\right] t^{k-1} \tag{1}$$

In which t obeys the k-order Erlang distribution, $\mathrm{E}[t] = 1/\lambda$, $\mathrm{var}[t] = 1/(k\lambda^2)$. If the product is designed to go through k operations, and the processing time of each operation is independent, then the processing time obeys the k-order Erlang distribution, its density function is:

$$A(t) = e^{-k\lambda t}\sum_{n=0}^{k-1}\frac{(k\lambda t)^n}{n!}, \quad k > 0 \tag{2}$$

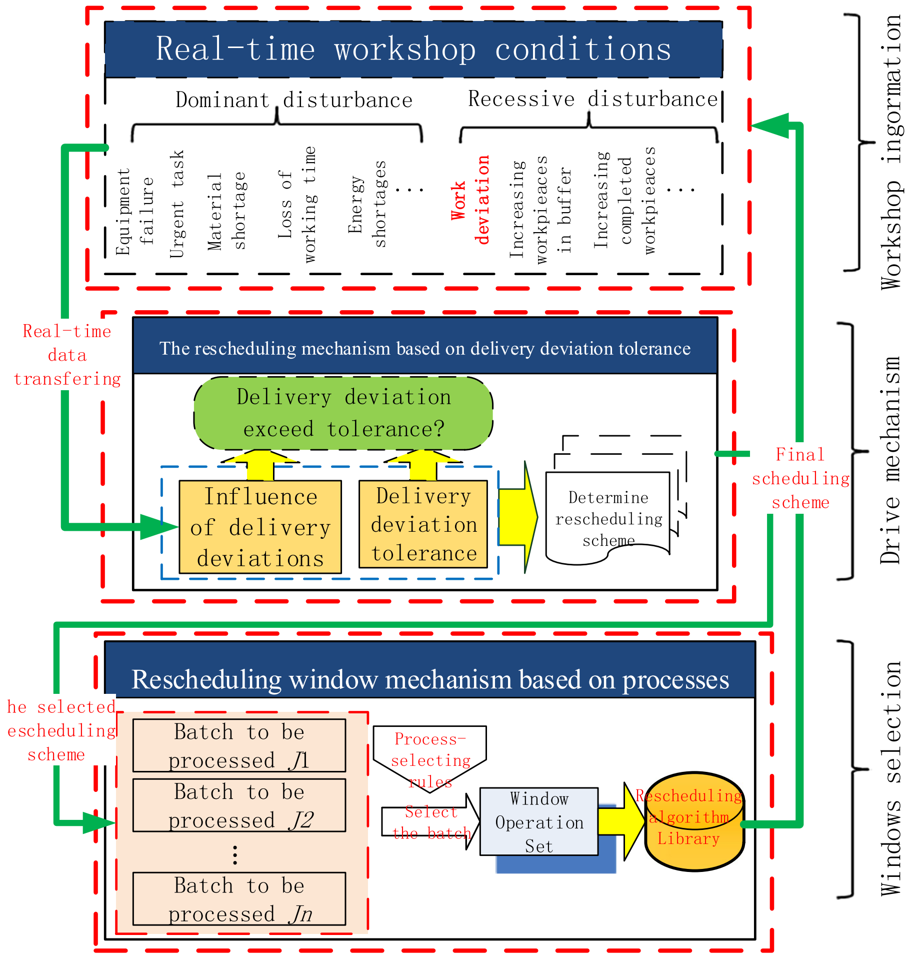
Rolling rescheduling strategy

At the beginning moment of the rolling optimization, it is required to schedule all the tasks to get a pre-scheduling sequence. With the advance of the actual production process, due to changes in the workshop environment, deviations between the actual scheduling program and pre-scheduling scheme occur. Thus the rescheduling is required. That means the completed job will be removed from scrolling window, and select some from operation set to be processed instead. The above processes will be repeated till all jobs are done. Figure 2 shows the rolling strategy map. The processing time deviation information is detected during the processing in the workshop. The detected information is judged by the driven mechanism based on delivery deviation tolerance for whether reschedule or not. If rescheduling is started, its window is selected based on the use of rescheduling process window mechanism. Appropriate algorithm is designed for batch scheduling within the window. The following figure describes two key elements of rescheduling: the rolling rescheduling mechanism and job window mechanism.

Rescheduling mechanism based on delivery deviation tolerance

Currently there are 3 kinds of rolling horizon rescheduling mechanism: event-driven rescheduling, cycle-driven rescheduling and mixed driving rescheduling based on cycle and event. The event-driven mechanism gets more applications since it's capable of respond to the events in real-time production. But if each batch is adjusted according to actual implementation, it will inevitably lead to frequent adjustments of job sequence. A buffer mechanism must be established to avoid the response to the disturbance events with little effect. This buffering mechanism requires not only filtration of small amplitude disturbance event, but also considering the disturbance caused by a large number of small changes.

Fig. 2 The rolling strategy


Real-time workshop conditions
Dominant disturbance
Recessive disturbance
Equipment failure
Urgent task
Material shortage
Loss of working time
Energy shortages
...
Work deviation
Increasing workpieaces in buffer
Increasing completed workpieaces
...
Workshop ingormation
Real-time data transfering
The rescheduling mechanism based on delivery deviation tolerance
Delivery deviation exceed tolerance?
Influence of delivery deviations
Delivery deviation tolerance
Determine rescheduling scheme
Final scheduling scheme
Drive mechanism
Rescheduling window mechanism based on processes
he selected escheduling scheme
Batch to be processed J1
Batch to be processed J2
Batch to be processed Jn
Process-selecting rules
Select the batch
Window Operation Set
Rescheduling algorithm Library
Windows selection

The deviation of process time will lead to the deviation between actual due date (makespan) and prescheduled due date, and with the accumulation of processing time, the degree of due date deviation changes. The due date of different schedules varies, and the same deviation of due date has different influence on different schedules. Therefore, the concept of delivery deviation is proposed, and is given by:

$$\delta = \frac{|Max(C_i) - Max(C_i')|}{Max(C_i)} \tag{3}$$

in which $Max(C_i)$ and $Max(C_i')$ denote the rescheduling makespan and the makespan when delivery deviation occurs respectively. Before rescheduling, you need to determine the delivery deviation tolerance $\delta_{\max}$. The delivery deviation δ is compared to $\delta_{\max}$ during the scheduling. The rescheduling is triggered once δ exceeds $\delta_{\max}$. If $\delta_{\max}$ is too large, then the occurrence times of rescheduling will be too small or even zero, and it cannot respond effectively to the production environment changes; if $\delta_{\max}$ is too small, then it will cause frequent rescheduling, resulting in a great impact on the orderly production. Besides each Tolerance deviation is specific for a given problem. From the experimental point of view, this article studies the results of the rescheduling under different deviation tolerances and determines the optimal value of the deviation tolerance.

Rescheduling window mechanism based on processes

For the rolling horizon strategy based dynamic scheduling, it is required to determine the appropriate window size initially. Since scrolling window has measurable characteristics, generally workpiece or time scales is used as a measure. The current re-scheduling window mechanisms are: workpiece-based windows, time-based windows and process-based windows.

In this research, the mechanism of process-based windows is adopted due to the advantages of adjusting with high precision and being able to adapt to changes of the processing load. As shown in Fig. 3, ($O(l)$ denotes the l-th predictive window, and contains $S(l)$ jobs; $CS(l)$ is the set of jobs already completed; $ES(l)$ is the set of waiting jobs.), a certain amount of workpieces from operation set to be processed in

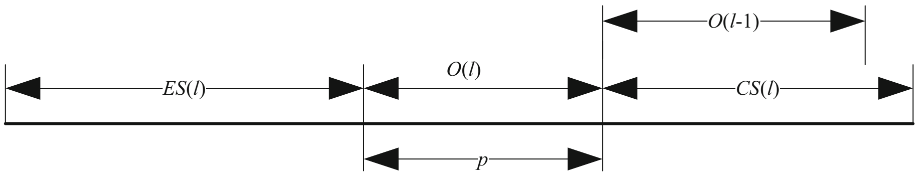


Fig. 3 Rolling window based on operation

rescheduling window. Window size has a great impact on the result of rescheduling. If the window is too small, the amount of computation is reduced, but the global information considered is less, which will result in lower overall performance; conversely, if the window is too large, it is difficult to obtain the optimal schedule due to the hard computation, and only suboptimal solution is obtained which may reduce the performance of the final global solution. Thus this paper determines the optimal window size by analyzing the results of numerical experiments with different window sizes.

Improved ant colony algorithm

The randomness and uncertainty of the dynamic scheduling problem make it more complicated than the static scheduling problem. Ant colony algorithm, due to its simple principle, good versatility and less constraints by the restrictions, is a promising alternative for the dynamic scheduling. Since the processing time fluctuates greatly, frequent rescheduling should be carried out and it is required that the scheduling algorithm has to be of high computational efficiency. However the traditional ant colony algorithm has disadvantages of long searching time and local convergence. In order to solve these problems, some researchers conducted in-depth studies on the ant colony algorithm and proposed many improving strategies. Xiao and Li (2003) compressed ants searching path by setting ants' optional path sets, thus greatly increasing the computational efficiency of ants. However, the size of optional path sets requires a lot of experiments, and fixed ants optional path sets will limit the range of movement of ants, causing it to be difficult to maintain population diversity and easy to fall into local minimum value. For the above drawbacks, Huang et al. (2009) proposed a self-adaptive ants path compression mechanism based on group evolutionary rates, designed optional path sets jumping strategy based on ants polymorphism and improved the heuristic factors of ants state transition. This improved algorithm has been used to solve JSP problem and achieved better results. However its definition on evolutionary rates and ants polymorphism is too simple, and its improvement on heuristic factors does not have the versatility, which still leaves much room for improvement. This paper designed ant path compression strategy by drawing the concept of ants' optional path sets and stimulated the ants to try the less selected path by improving the ants state transition rules. Thus the algorithmic efficiency and solution quality is better.

Ant path compression strategy

The movement range of ants could be limited by compressing the ant path, thus to improve the searching efficiency. However, fixed ants optional path sets makes the searching process of ants in the whole range of movement to be always consistent, which decreases the population diversity and causes difficulty in finding new solutions, and makes the algorithm easily fall into a local optimum. Meanwhile, the size of ants' optional path sets has a great influence on convergence rate of the algorithm. When the size is too small the algorithm is easy to fall into a local optimum; while the size is too large, it is difficult to achieve the purpose of lowering the dimension and saving the computing time (Gholami et al. 2009). The traditional ant colony algorithm requires trial and error to determine the appropriate ant optional path sets size. This paper proposed the concept of node-weighted divergence, designed the path compression strategy based on node-weighted divergence, making the ant optional path sets size be enable to adjust dynamically according to searching conditions.

In the searching process of ants, when all the ants in the distribution of the scattered paths, the pheromone of each path distributes uniformly, which may slow down the searching velocity of ants and prolong the searching cycle. Therefore it is better to focus the ants on some optimal paths. To the contrary, when the distribution of all the ants is centralized, it is prone to cause stagnation and make the algorithm to fall into local optimum. In order to measure the degree of a node distribution of ants in different paths, this paper introduces the concept of node divergence. If there are r nodes to choose after the node i, and the number of ants passed by node i is M_i in the last iteration, and the number of ants on r paths are respectively $\mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_\mathrm{r}$, then the divergence of node i is:

$$E_i = 1 - \frac{1}{M_i}\sqrt{\frac{r\sum_{i=1}^{r}(M_i/r - m_i)^2}{r-1}} \tag{4}$$

Define the ratio between the number of ants passed by node i and the total number of ants as the node i attract rate, namely $A_i = M_i/Numant$. A higher node i attract rate implies a larger number of ants passing node i, more dispersed distribution of ants and more easily falling into local optimum. On the contrary, a lower of the node i attract rate means more uniform distribution of ants and slower searching velocity. Thus, to a certain extent, the attract rate can reduce the effect of the divergence. Through putting it as a measure of ants' node divergence's weights, the node weighted divergence formula is:

$$G_i = A_i E_i = \frac{M_i - \sqrt{\frac{r\sum_{i=1}^{r}(M_i/r - m_i)^2}{r-1}}}{Numant} \tag{5}$$

Based on the ant node weighted divergence, the ant optional path sets size in this node is:

$$Win_i^k = Min\left\{\left[(1-G_i)Win_{allowed}^k\right]+1, Win_{allowed}^k\right\} \tag{6}$$

While [ ] indicates rounding towards nearest integer, and $Win^k allowed$ indicates all possible number of nodes of ant k at node i. From the formula, it is clear that ants distribute more uniformly when the ant node has a higher weighted divergence. Then we can improve search efficiency by reducing the ant optional path sets. While when the ant node has a lower weighted divergence, ants distribute more centralized, and it is needed to find a new solution space by enlarging the ant optional path sets.

Ant state transition rules

Stagnation is a major flaw of ant colony algorithm. With the evolution of ants, pheromone of paths with better solutions would continue to increase, and the probability of this path being selected during subsequent searches will grow. While paths may be lead to global optimal solution, it may be gradually forgotten due to little ants passing by and ultimately make the algorithm fall into a local optimum. Against the above shortcomings, the ant node state transition rules proposed in this paper stimulates the ants to try the less visited paths, in order to enlarge the global searching capability of the ants. The improved state transition rules are:

$$p_{ij}^k = \frac{\left(\tau_{ij}^k\right)^\alpha \cdot \left(\eta_{ij}^k\right)^\beta .x_{ij}}{\sum_{j\in allowed_k}\left(\tau_{ij}^k\right)^\alpha \cdot \left(\eta_{ij}^k\right)^\beta .x_{ij}} \tag{7}$$

Among $x_{ij} = \frac{\text{Numant}\cdot N}{\text{Numant}\cdot N + \delta\cdot m_{ij}\cdot\eta_{ij}/\max(\eta_{ij})}$, *Numant* represents of the number of ants, *N* represents the current iteration number, m_{ij} represents the total number of ants going through the path (i, j). When the iteration tends to locally optimal, although pheromones is continuously increasing on local optimal path, this effect is on the state transition probabilities is suppressed due to that the number of ants on their path, e.g. m_{ij} for the path (i,j) is also increased, which will lead to reduction of the value of x_{ij}. This proposed rule can improve the algorithm's global searching ability. Because $m_{ij} \le Numant \cdot N, \eta_{ij}/\max(\eta_{ij}) \le 1$, then $1 \ge x_{ij} \ge x_{min} = 1/(1+\lambda)$. Parameter λ can adjust the intensity of x, i.e. the smaller λ is, the larger x_{min} is, and the smaller the weighted-value of the number of ants passed through the path in ant state transition rules is.

$\tau_{i,j}^k$ indicates the pheromone level between batches (i, j); $\eta_{i,j}^k = 1/(Set_{J(i),J(j)} + FT(L_{J(j),p-1,j}) + PT(L_J(j), p, j))$ indicates the heuristic information between batches (*i*, *j*), which considers the processing sequence dependent setup time, the former process completion time, and the current process processing time. *J*(*i*) and *J*(*j*) respectively indicates the workpiece types of batch *i* and *j*; $FT(L_{J(j),p-1,j})$ and $PT(L_{J(j),p,j})$ respectively indicates the former process completion time and the current process processing time. α and β control the relative importance of $\tau_{i,j}^k$ and $\eta_{i,j}^k$ in state transition probabilities.

Algorithm steps

The specific steps of improved ant colony algorithm are as follows (Fig. 4):

STEP1: Initialize the algorithm parameters. Set the ant colony algorithm parameters α, β, ρ, maximum pheromone level $\tau_{\max}$, minimum pheromone level $\tau_{\min}$, maximum number of iterations I_t, ant number N_u, and jump probability P_s. Pheromone levels in all paths are initialized as $\tau_{\max}$.
STEP2: generate an ant *a*, and randomly select a batch *i* among the waiting jobs set as the first node to be visited, then assign the batch *i* to the earliest available machine. Initialize ant node counter $S = 1$, and iteration number counter $N = 1$.
STEP3: Determine the batch set $Win_{allowed}^s = \{1, 2, \ldots, P\} - tabu^a(S)$ that ant *a* can visit at next step, in which $tabu^a(S)$ is the batch set ant *a* has visited at *S*-th step, and *P* is the total batch numbers in the waiting jobs set. Calculate the divergence of this node and determine optional path set $Win_{allowed}^s$ in this node of ant *a*.
STEP4: Select the next node to be visited according to ant state transition rules, and set the earliest available machine to perform this node's processing jobs, refresh ant node counter $S = S + 1$.
STEP5: Determine whether the ants have visited all nodes, namely whether $S = P$ is satisfied, if yes, then go to STEP6, otherwise, go to STEP3.
STEP6: Refresh iteration number counter $N = N + 1$. Determine whether the iterations are complete. If $N =$

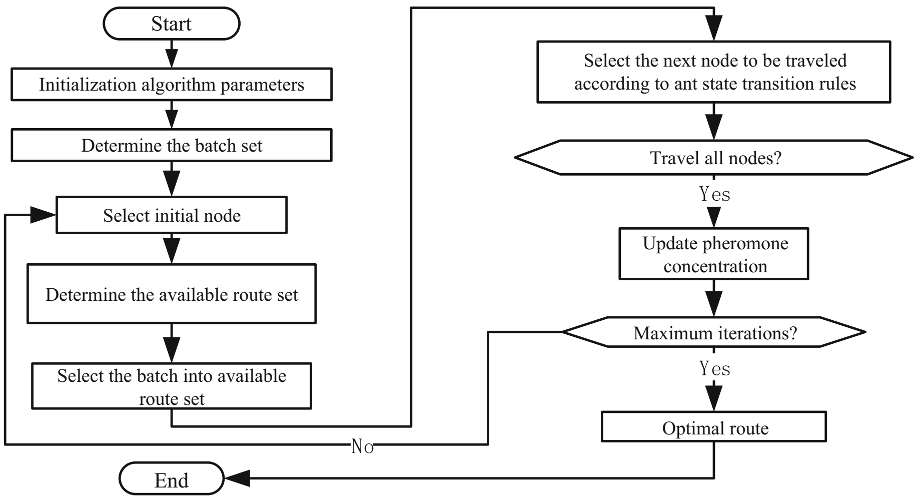

Start
Initialization algorithm parameters
Determine the batch set
Select initial node
Determine the available route set
Select the batch into available route set
Select the next node to be traveled according to ant state transition rules
Travel all nodes?
Yes
Update pheromone concentration
Maximum iterations?
Yes
No
Optimal route
End

Fig. 4 Flow chart of improved ant colony optimization

N_t is satisfied, then finish this algorithm; otherwise, go to STEP7.

STEP7: Update pheromone level. Firstly, calculate the evaporation value of pheromone, $\tau_{i,j}(N+1) = \rho.\tau_{i,j}(N)$. Secondly, increase pheromone level of paths in the solution which obtains the shortest completion time, $\tau_{i,j}^{a(min)} = \tau_{i,j}^{a(min)} + \Delta\tau_{i,j}^{\text{best}}$, in which $a(\text{min})$ is the best solution, C_{max}^{best} is the length of $a(\text{min})$, $\Delta\tau_{n,i}^{\text{best}} = 1/C_{max}^{\text{best}}$. In order to avoid the algorithm trapping into a non-global optimal solution too early, this paper introduces the MMAS (max-min ant system) mechanism to limit each ant pheromone level in $[\tau_{min}, \tau_{max}]$. Specially, the maximum pheromone level $\tau_{\max}$ and minimum pheromone level $\tau_{\min}$ are dynamically changing whenever a new best solution s^{gb} is found, and they are given by formulas (8) and (9), in which p_{best} is set to 0.5 in this paper and n is the total number of job batches to be scheduled. Then go to STEP2.

$$\tau_{\max} = \frac{1}{1-\rho}\frac{1}{f(s^{gb})} \tag{8}$$

$$\tau_{\min} = \frac{\tau_{\max}(1-\sqrt[n]{p_{best}})}{(n/2-1)\sqrt[n]{p_{best}}} \tag{9}$$

Numerical results

Parameters analysis

Determining the tolerance of delivery time deviation and the size of horizon window before the rolling horizon scheduling has a great influence on the scheduling result. However they are difficult to analyze theoretically because of the closely connection with the actual problem. This article analyzes the influences of delivery time deviation tolerance and horizon window size delivery on the scheduling results. The example of the optimization on the tolerance of delivery time deviation and the size of horizon window shows in Table 1.

Table 1 The example of the optimization on the tolerance of delivery time deviation and the size of horizon window

Parameter	Value range
Type of production	4
Number of processes	4
Number of the parallel machine for every process	u[2,4]
Processing time	u[20,30]
Set-up time(same products)	u[1,3]
Set-up time(different products)	u[5,7.5]
The quantity of workpieces in each batch	8

The static scheduling result of the above example is shown in Fig. 5. Products are divided into 128 processing batches, the makespan of the schedule is 407. In the figure, number represents product category and batches. For example “3, 2,” means the second batch of the third type of products. Black areas means set-up time.

In order to determine the most appropriate tolerance of delivery time deviation, 4-order Erlang distribution is adopted to simulate the actual processing time distribution. 10 times calculation under different deviation tolerance are taken to solve the above example. The makespan, rolling times and the average computation time are showed in Table 2. From the table, when the delivery time deviation tolerance is greater than 0.25, since the deviation tolerance is large, no rescheduling is triggered and the result is obviously not ideal.

When the delivery time deviation tolerance is too small, the rolling time increases, leading to frequent rescheduling. And since each rescheduling only considers the local information of processing window, frequent rescheduling has a poor performance on global optimization and com-

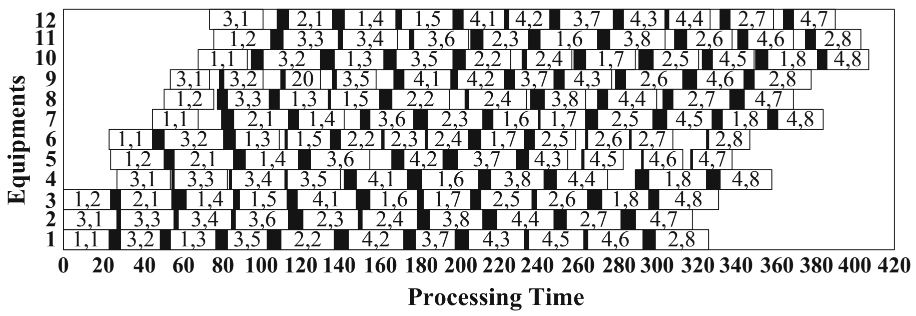


Fig. 5 The gantt diagram of static scheduling

Table 2 The results of the delivery time deviation tolerance experiments

The tolerance of delivery time deviation	Makespan (min)	Rescheduling times	Computation time (ms)
0.025	472.6	20	26432.3
0.05	443.2	11	10674.8
0.075	423.7	6	6416.5
0.1	406.8	4	3763.2
0.125	411.5	3	2836.5
0.15	417.6	3	2753.6
0.2	424.3	2	1521.4
0.25	432.6	1	882.3
0.3	436.5	1	862.1
0.35	439.5	0	532.4
0.4	439.5	0	538.5
0.45	439.5	0	531.2
0.5	439.5	0	528.6

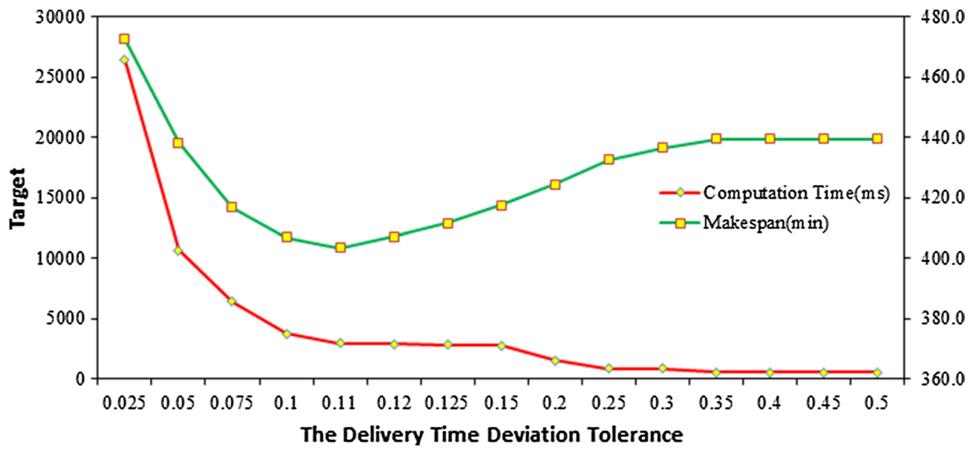


Fig. 6 The results of the delivery time deviation tolerance analysis

putation efficiency. Figure 6 shows the relationship between the delivery time deviation tolerance, scheduling results and computing time. It can be seen that when the error tolerance is 0.1, the rolling time is less, the solution is optimal and the computation time's increasing is not obvious.

In order to determine the most appropriate size of horizon window, 4-order Erlang distribution is adopted to simulate the actual processing time. 10 times calculation under different size of horizon window are taken to solve the above example, in which the delivery time deviation tolerance is 0.1. The makespan, rolling times and the average computation time are showed in Table 3. From the table, when the size of horizon window is too small, also the rolling number increases, leading to frequent rescheduling. In the meantime, too many batches lead to less rescheduling times and longer computation time. Therefore it cannot obviously reduce the total computing time. Figure 7 shows the relationship between the size of horizon window, scheduling results and computation time. It can be seen that when the size of horizon window is between 64 and 90, the rolling time is less, the solution is optimal and the computation time's increasing is not obvious.

Table 3 The results of the size of horizon window experiments

The size of horizon window	Makespan (min)	Rescheduling times	Computation time (ms)
13	541.2	13	56314.2
26	486.5	8	35163.5
39	463.2	4	27327.4
51	432.1	4	25368.4
64	412.4	3	14695.8
77	402.1	2	9532.4
90	408.3	2	9321.4
102	428.6	2	9346.5
115	436.5	1	6672.4
128	452.6	1	6532.8

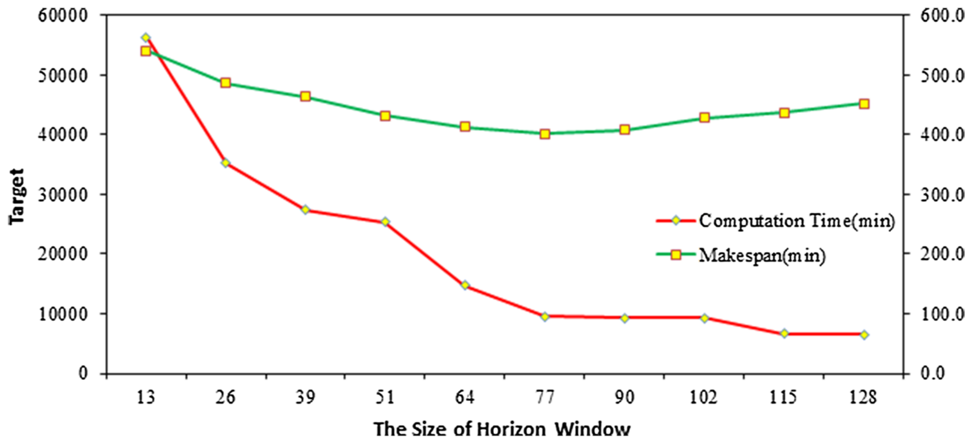


Fig. 7 The results of the size of horizon window analysis

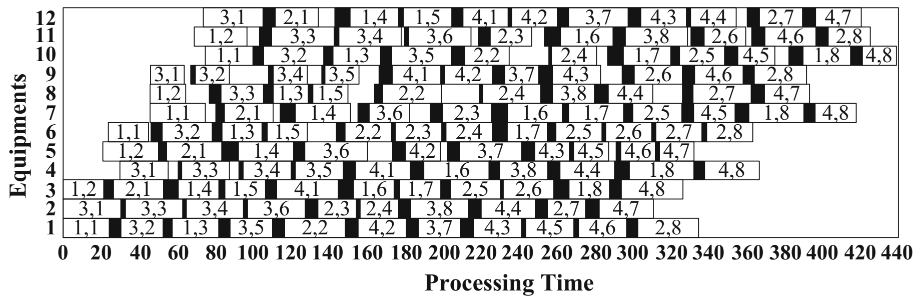


Fig. 8 Schedule gantt chart without researching

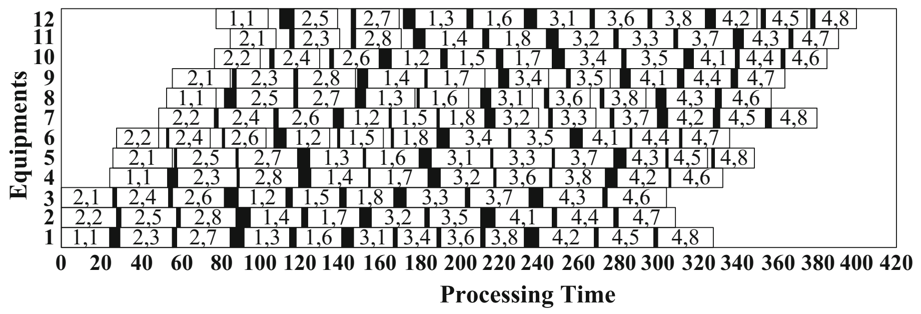


Fig. 9 Rescheduling gantt chart

Compared schedule gantt chart without rescheduling (as shown in Fig. 8) with rescheduling gantt chart (as shown in Fig. 9), and set the delivery time deviation tolerance 0.1, size of horizon window 77. It can be seen from the diagram that the reschedule can obviously shorten the makespan, and make full use of spare equipment, improve the utilization rate of equipment.

Table 4 Benchmarks for dynamic scheduling algorithm

Product name	Number of jobs	Number of processes	Number of the parallel machines for every process	Processing time	Set-up time (same products)	Set-up time (different products)
6 × 2	6	2	U[1,5]	U[50,70]	U[3,5]	U[12,24]
30 × 2	30	2	U[1,5]	U[50,70]	U[3,5]	U[12,24]
100 × 2	100	2	U[1,5]	U[50,70]	U[3,5]	U[12,24]
6 × 4	6	4	U[1,5]	U[50,70]	U[3,5]	U[12,24]
30 × 4	30	4	U[1,5]	U[50,70]	U[3,5]	U[12,24]
100 × 4	100	4	U[1,5]	U[50,70]	U[3,5]	U[12,24]
6 × 8	6	8	U[1,5]	U[50,70]	U[3,5]	U[12,24]
30 × 8	30	8	U[1,5]	U[50,70]	U[3,5]	U[12,24]
100 × 8	100	8	U[1,5]	U[50,70]	U[3,5]	U[12,24]

Table 5 Results of different algorithms

Experiments number	Makespan (min)			Computing time (ms)		
	IACO	IACS	SDWACO	IACO	IACS	SDWACO
6 × 2L	247	247	248	182.3	202.4	187.6
30 × 2L	1130	1135	1137	632.5	868.9	651.5
100 × 2L	3769	3779	3776	9245.6	10102.3	9326.3
6 × 4L	589	591	593	463.8	472.1	451.3
30 × 4L	2302	2312	2317	4320.5	4625.7	4448.2
100 × 4L	7746	7759	7751	48632.3	50136.5	49186.1
6 × 8L	836	843	845	973.2	1003.2	951.3
30 × 8L	2709	2728	2726	16382.4	17568.5	16466.1
100 × 8L	7964	7981	7971	203126.7	221415.3	208929.6

Table 6 The processing times of PCB in each production lines

Different kinds of PCBA	S1	S2	S3	M1	M2	A1	A2	T1	T2
3ET0435TEK	0.923	0.923	0.882	0.6	1.5	0.6	0.4	0.6	0.4
3ET0374TEK	1.428	0.923	1.2	0.4	0.3	0.6	0.3	0.6	0.55
3ET0141CET	1	1.428	1	0.4	0.24	0.6	0.3	0.55	0.5
3ET0630CET	1	1	1	0.06	0.24	0.75	0.6	0.55	0.5
3ET0100CET	1	1	0.6	0.06	1.5	0.75	0.75	0.4	0.55
3ET0349CET	1	1	0.6	0.4	0.3	0.75	0.75	0.6	0.6
3ET0741CET	1	4.615	0.293	0.6	0.3	0.75	0.6	0.4	0.4
3ET0630CET	1	1	0.6	0.06	0.3	0.6	0.75	0.55	0.4
3ET0321AF	4.615	1	1.2	0.3	0.24	0.6	0.75	0.6	0.6
3ET0322AF	4.615	1	0.293	0.4	0.3	0.6	0.75	0.6	0.55

Performance evaluation

In order to verify the effectiveness of the improved ant colony algorithm, this paper takes advantage of GHOLAMI's benchmark (2009) and generates nine benchmarks (as shown in Table 4).

This paper takes use of the Erlang distribution model to simulate the actual processing time, and respectively uses three methods to solve those benchmarks, including improved ant colony algorithm (IACO), improved ant colony system (IACS) proposed by Liu et al. 2012, adaptive ant colony algorithm based on different size window (SDWACO) proposed by Gong and Ruan (2004). Due to the randomness of ant colony algorithm, all those experiments are respectively calculated 10 times to get the best value and the average value. Parameters of IACO are as follows: $\alpha = 1, \beta = 2, \rho = 0.3, P_s = 0.5, \delta = 0.5, Numant = 20, Iter = 200, \tau_{max} = 10, \tau_{min} = 0.5$. The results are shown in Table 5.

From the comparison of IACO and IACS, it can be seen that due to the adoption of ant path compression strategy, IACO is obviously superior than IACS in computing time, i.e. the computing efficiency of IACO is significantly improved. At the same time, the improved ant state transition rules help to improve the global convergence performance of the algorithm, so IACO is better than IACS in computing results. From the comparison of IACO and SDWACO, it can be seen that these two algorithms have little difference in computing time. While solving large scale problems, ICAO can reduce search window's size gradually during the search process, and performs better in adaptability of the problem and can effectively control the size of search window compared with SDWACO. Thus IACO has higher search efficiency than SDWACO. From the results, it also shows that IACO performs better than SDWACO in global searching capability and can acquire superior results. From the comparison of IACS and SDWACO, it can be seen that there is little difference in the quality of results between them. However, when solving large scale problems, SDWACO can compress the search space effectively. On the contrary, it is difficult for IACS to converge to optimal solution in high-dimensional space, and therefore SDWACO is slightly superior to IACS in terms of effectiveness. From the aspect of computational efficiency, the use of rescheduling window mechanism can effectively compress the ant search path of SDWACO. Thus it has better searching efficiency. Overall, IACO performs better in both the effectiveness of the results and computational efficiency in line with the requirements of dynamic scheduling.

Case study

The PCB assembly shop in our collaborating company has four processing shops (SMT chip processing zone, plug-processing zone, welding processing zone, and test zone) and 9 production lines. The productions lines are 3(S1,S2,S3), 2(M1,M2), 2(A1,A2) and 2(T1,T2). There are 10 kinds of PCB to be produced, and the number of each PCB is 1000. Every PCB has to be processed in those 4 zones. The processing time of PCB in each production line is shown in Table 6, the set-up time of different PCB are shown in Table 7.

The gantt chart of static scheduling results can be seen in Fig. 10, in which the makespan is 3615.7 and all PCB products are divided into 300 batches. Furthermore, compare the SDWACO dynamic scheduling algorithm with IACO in real-life application. The gantt chart of SDWACO can be seen in Fig. 11, in which the makespan is 3528.3 and the average production line utilization rate is 86.2 %. The gantt chart of IACO can be seen in Fig. 12, in which the makespan is 3496.5 and the average production line utilization rate is 90.3 %. It can be seen that IACO performs better than static scheduling algorithm. While compared with other dynamic scheduling

Table 7 The set-up time of different PCB

Different kinds of PCBA	3ET0435TEK	3ET0374TEK	3ET0141CET	3ET0630CET	3ET0100CET	3ET0349CET	3ET0741CET	3ET0630CET	3ET0321AF	3ET0322AF
3ET0435TEK	1.52	7.85	13.34	13.01	12.52	10.28	6.14	10.72	6.44	7.89
3ET0374TEK	5.03	1.38	10.12	5.85	14.34	7.54	6.82	8.44	13.17	7.75
3ET0141CET	8.02	9.95	1.28	5.97	11.62	12.96	5.69	8.6	8.93	8.27
3ET0630CET	10.12	10.88	10.79	1.56	10.36	14.15	11.8	14.95	8.93	10.83
3ET0100CET	6.54	12.98	6.34	14.63	1.23	11.22	11.09	12.2	9.81	7.23
3ET0349CET	9.29	7.07	7.79	14.26	10.05	1.32	14.6	9.83	14.12	10.73
3ET0741CET	11.83	14.09	8.99	14.33	14.22	9.52	1.22	14.43	12.19	9.81
3ET0630CET	5.48	13.43	9.99	7.51	14.64	9.37	10.37	1.75	9.14	12.32
3ET0321AF	9.2	9.51	13.12	7.28	8.37	10.83	7.1	8.37	1.56	12.58
3ET0322AF	9.82	12.13	7.3	10.57	9.64	9.53	10.67	5.77	5.73	1.32

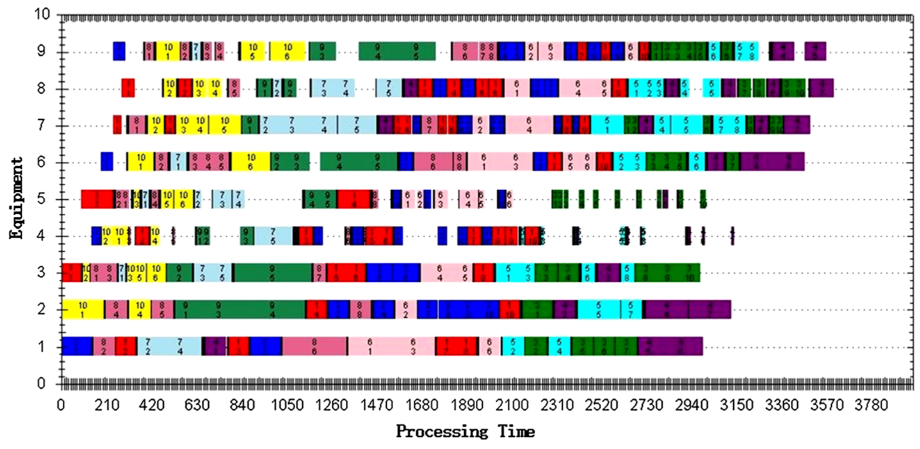


Fig. 10 Static scheduling gantt chart

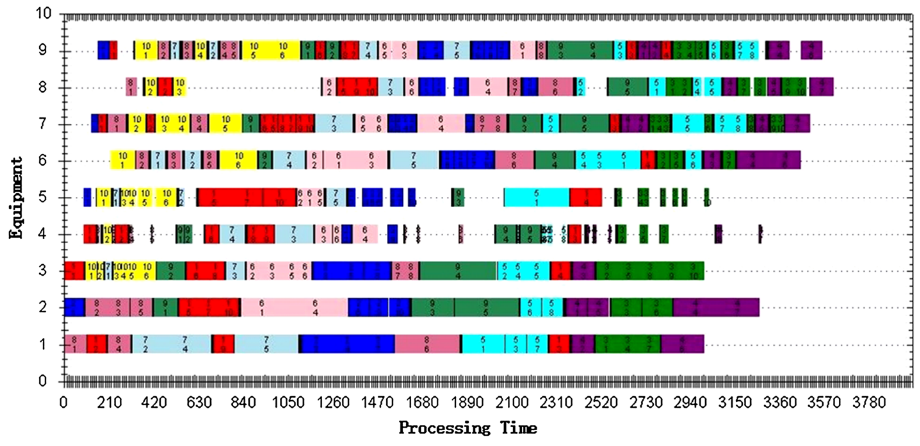


Fig. 11 SDWACO dynamic scheduling gantt chart

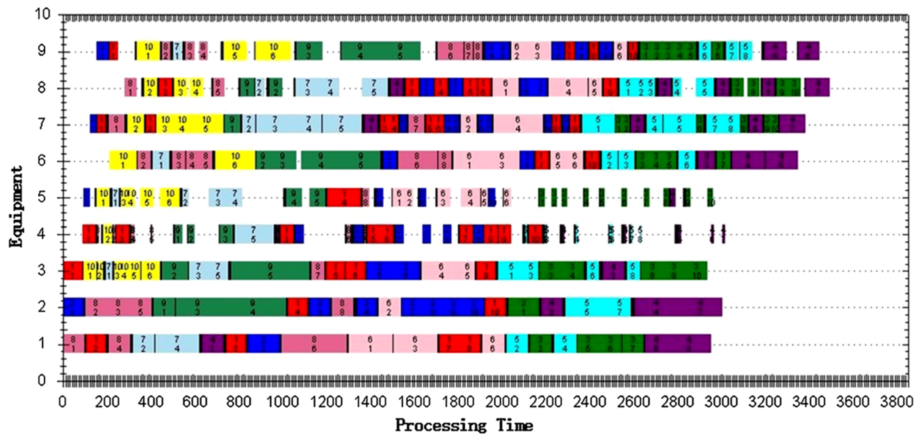


Fig. 12 IACO dynamic scheduling gantt chart

algorithms, it can also effectively solve the uncertain processing time problem in actual production process and obtain good scheduling result.

During the former production of this PCB manufacturing, it always schedules according to previous experience firstly. Then as the production parameters (processing time, machine state, orders, and so on) changes, especially the advance and delay of processing time occurs frequently, all of those cause the entire PCB production to become extremely confusing. It leads the static scheduling result obtained previously to be no longer effective, or even infeasible. This issue has become one of the bottlenecks restricting the enterprise to further improve its production efficiency. The dynamic scheduling method proposed in this paper plays an important role in helping the enterprise to effectively achieve precise control of PCB production process. What's more, in order to guarantee the effectiveness of this dynamic scheduling algorithm, the accurate and real-time information of the workshop's production status is a prerequisite. How to strike a balance between throughput increase and operation costs reduction in the workshop becomes a crucial question for this PCB manufacturing enterprise.

Conclusions

Aiming to overcome the HFS scheduling problem with uncertain processing time, this paper proposes an improved ant colony algorithm based on rolling rescheduling strategy. Putting forward the concept of delivery time deviation and designing the rolling horizon strategy based on the tolerance of delivery time deviation, this paper improves the traditional event-driven rolling mechanism to reduce the frequency of dynamic events and avoid frequent rescheduling. A new process based rescheduling window measuring mechanism is designed in order to realize the buffer of dynamic event. To guarantee the computational efficiency of the rolling rescheduling algorithm, based on the rolling horizon procedure, an improved ant colony optimization is proposed. The algorithm can improve searching efficiency by designing an ant path compression strategy and enhancing the global convergence ability by introducing an ant state transition rule to stimulate ants to try the path being less searched. Through simulation experiments, this paper analyzes parameters of the rolling scheduling strategy and verified the effectiveness of the dynamic scheduling algorithm by comparison experiments. The results show that IACO performs better both in computational time and solution quality. Finally, this paper conducts the practical enterprise's application, and the results show that the method proposed in this paper has played a certain guiding role in actual production.

Although the research has dealt with several academically challenging issues, further work is still needed in order to be used on a daily basis. The first extension is that multiple objectives especially related with the due date, such as total tardiness, need to be considered. Another extension is that appropriate facilities must be developed to obtain real-time availability shopfloor information for rescheduling, together with the development and implementation of the HFS scheduling system.

Acknowledgements The authors would like to acknowledge the financial support of the National Science Foundation of China (No. 51435009).

References

Arnaout, J. P., Rabadi, G., & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *21*, 693–701.

Babu, S. A. K. I., Pratap, S., Lahoti, G., Fernandes, K. J., Tiwari, M. K., Mount, M., et al. (2014). Minimizing delay of ships in bulk terminals by simultaneous ship scheduling, stockyard planning and train scheduling. *Maritime Economics and Logistics*. doi:10.1057/Mel.20.

Baker, K. R. (1995). Lot streaming in the two-machine flow shop with set-up times. *Annals of Operations Research*, *57*, 1–11.

Bose, S. K. (2002). *An introduction to queuing systems*. New York: Kluwer Academic/Pelenum Publishers.

Chang, P. C., Hsieh, J. C., & Wang, C. Y. (2007). Adaptive multi-objective genetic algorithm for scheduling of drilling operation in printed circuit board industry. *Applied Soft Computing*, *7*, 800–806.

Dorigo, M. (1992). Optimization, learning and natural algorithms. PhD thesis, Politecnico di Milano, Italie.

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*, 53–66.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystem*, *43*, 73–81.

Drótos, M., Erdős, G., & Kis, T. (2009). Computing lower and upper bounds for a large-scale industrial job shop scheduling problem. *European Journal of Operational Research*, *197*, 296–306.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: Freeman.

Gholami, M., Zandieh, M., & Tabriz, A. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *International Journal of Advanced Manufacturing Technology*, *42*(1–2), 189–201.

Gong, D. X., Ruan, X. G. (2004). A hybrid approach of GA and ACO for TSP [C] // Proceedings of the 5th world congress on intelligent control and automation. Washington, D. C., USA: IEEE, 3: 2068–2072.

Gunther, H. O., Gronalt, M., & Zeller, R. (1998). Job sequencing and component set-up on a surface mount placement machine. *Production Planning & Control*, *9*, 201–211.

Hall, N. G., Laporte, G., Selvarajah, E., & Sriskandarajah, C. (2003). Scheduling and lot streaming in flow shops with no-wait in process. *Journal of Scheduling*, *6*, 339–354.

Huang, R. H. (2010). Multi-objective job-shop scheduling with lot-splitting production. *International Journal of Production Economics*, *124*, 206–213.

Huang, M., Liu, P. F., & Liang, X. (2009). Ant colony algorithm oriented to Job Shop scheduling problem based on self-adaptive and uneven windows. *Computer Integrated Manufacturing Systems*, *15*(10), 1973–1978.

Ji, P., Sze, M. T., & Lee, W. B. (2001). A genetic algorithm of determining cycle time for printed circuit board assembly lines. *European Journal of Operational Research*, *128*, 175–184.

Kaczmarczyk, W., Sawik, T., Schaller, A., & Tirpaks, T. (2004). Optimal versus heuristic scheduling of surface mount technology lines. *International Journal of Production Research*, *42*, 2083–2110.

Kis, T., & Pesch, E. (2005). A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research*, *164*, 592–608.

Lamothe, J., Marmier, F., Dupuy, M., Gaborit, P., & Dupont, L. (2012). Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints. *Computer & Operation Research*, *39*, 1236–1244.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering*, *37*(1–2), 57–61.

Liu, G. B. (2012). *Research on production scheduling methods of complex unrelated parallel machines*. Shanghai: Shanghai Jiao Tong University.

Liu, S. C. (2003). A heuristic method for discrete lot streaming with variable sub-lots in a flow shop. *International Journal of Advanced Manufacturing Technology*, *22*, 662–668.

Marimuthu, S., Ponnambalam, S. G., & Jawahar, N. (2007). Tabu search and simulated annealing algorithms for scheduling in flow shops with lot streaming. *Proceedings of the Institution of Mechanical Engineers Vol. 221 Part B: Journal of Engineer Manufacture*, 317–331.

Mehta, S. V., & Uzsoy, R. (1999). Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing*, *12*(1), 15–38.

Potts, C. N., & Baker, K. R. (1989). Flow shop scheduling with lot streaming. *Operations Research Letters*, *8*, 297–303.

Quadt, D., & Kuhn, H. (2005). Conceptual framework for lot-sizing and scheduling of flexible flow lines. *Interational Journal of Production Research*, *43*, 2291–2308.

Ruiz, R. (2010). Lot-streaming for sequence dependent setup time flowshop problems. In *Proceedings of 4th international conference on industrial engineering and industrial management*. Donostia, Spain.

Salvador, M. S. (1973). A solution to a special class of flow shop scheduling problems. In S. E. Elmaghraby (Ed.), *Symposium on the theory of scheduling and its applications* (pp. 83–91). Berlin: Springer.

Sawik, T. (2001). Mixed integer programming for scheduling surface mount technology lines. *International Journal of Production Research*, *39*, 3219–3235.

Truscott, W. (1986). Production scheduling with capacity constrained transportation activities. *Journal of Operation Management*, *6*, 333–348.

Wang, H. M., Chou, F. D., & Wu, F. C. (2011). A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan. *International Journal of Advanced Manufacturing Technology*, *53*, 761–776.

Wu, K. (2007). *Study on manufacturing execution process monitoring and control for multi type and volume production*. Beijing: Beijing Institute of Technology.

Xiao, Y., & Li, B. (2003). Ant colony algorithm based on little window. *Computer Engineering*, *20*, 056.

Xu, Z., & Gu, X. (2005). Immune scheduling algorithm for flow shop problems under uncertainty. *XITONG GONGCHENG XUEBAO*, *20*(4), 374.

Yang, J. (2015). Minimizing total completion time in a two-stage hybrid flow shop with dedicated machines at the first stage. *Computers & Operations Research*, *58*, 1–8.

Zhang, Y., Rong, Z. J., M J. (2014). Hybrid flow shop problem with batching machines and multi-jobs families. *Computer Integrated Manufacturing Systems*, 20(2): 407–413.

Zhao, H., Tang, L., & Zhang, Y. (2008). Simulation and analysis on dynamic job shop scheduling toward networked manufacturing. *Journal of System Simulation*, *11*, 057.