Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Flow shop rescheduling problem under rush orders*

HU Yan-hai (胡燕海)†[1,2], YAN Jun-qi (严隽琪)[2], YE Fei-fan (叶飞帆)[1], YU Jun-he (于军合)[1]

([1]*Faculty of Engineering, Ningbo University, Ningbo 315211, China*)
([2]*Institute of CIM, Shanghai Jiao Tong University, Shanghai 200030, China*)
†E-mail: huyanhai@nbu.edu.cn; huyanhai@sjtu.edu.cn

Received Feb. 27, 2005; revision accepted Apr. 26, 2005

Abstract: In the environment of customization, disturbances such as rush orders and material shortages often occur in the manufacturing system, so rescheduling is necessary for the manufacturing system. The rescheduling methodology should be able to dispose of the disturbance efficiently so as to keep production going smoothly. This aims researching flow shop rescheduling problem (FSRP) necessitated by rush orders. Disjunctive graph is employed to demonstrate the FSRP. For a flow shop processing n jobs, after the original schedule has been made, and z out of n jobs have been processed in the flow shop, x rush orders come, so the original n jobs together with x rush orders should be rescheduled immediately so that the rush orders would be processed in the shortest time and the original jobs could be processed subject to some optimized criteria. The weighted mean flow time of both original jobs and rush orders is used as objective function. The weight for rush orders is much bigger than that of the original jobs, so the rush orders should be processed early in the new schedule. The ant colony optimization (ACO) algorithm used to solve the rescheduling problem has a weakness in that the search may fall into a local optimum. Mutation operation is employed to enhance the ACO performance. Numerical experiments demonstrated that the proposed algorithm has high computation repeatability and efficiency.

Key words: Flow shop rescheduling, Dynamic scheduling, Rush order, Ant colony optimization, Mutation operation
doi:10.1631/jzus.2005.A1040 **Document code:** A **CLC number:** TP182; O223

INTRODUCTION

Rescheduling problem in manufacturing system is considered as a particularly hard combinatorial optimization problem, closely related with uncertainty caused by exterior business environment and interior production conditions. Production rescheduling is a common practice in manufacturing companies all over the world. Four sources of production disturbances have been identified: (1) incorrect work; (2) machine breakdowns; (3) rework due to quality problem; and (4) rush orders (Rangsaritratsamee *et al.*, 2004). This paper deals with the rescheduling problem arising from rush orders. In current customized market, rush orders that often emerge in a manufacturing system should be rescheduled in time so that they would be finished as soon as possible. Besides, some original jobs, which have not been processed yet, can also be processed subject to good criterion.

Some researchers studied rescheduling (dynamic scheduling) problem under rush orders. Li *et al.*(2000) introduced a production rescheduling expert simulation system, which included many techniques and methods, such as simulation technique, artificial neural network, expert knowledge and dispatching rules. In their paper, rush orders do not add to the scheduling system, but their processing time will be added to that of being processed jobs or the waiting time of following jobs. The criterion for the scheduling is maximizing machine utilization. But if rush orders are inserted into the optimized sequence of original jobs, the original jobs may not be processed under the desired criterion. Zhang and Rao (2003) tried to reschedule rush orders with general

*Project supported by the National Natural Science Foundation of China (No. 70171042) and Zhejiang Provincial Natural Science Foundation, China (Nos. Z604342 and M703100), and Ningbo Municipal Youth Foundation, China (No. 2005A620004)

method. If machines have idle periods, the rush orders will be inserted in these periods. If machines are not idle, some jobs with late due dates will be released, and the rush orders will be inserted in the released job periods. Then the released jobs will be rescheduled. With this rescheduling method, the rush orders may not be processed at the earliest time, and the original jobs may not be processed in optimized order. Sousa and Ramos (1999) proposed a negotiation protocol based on the contract net protocol. The negotiation protocol can handle exceptions such as machine breakdowns and rush orders.

Other researchers studied rescheduling problem with new orders arriving randomly (Aydin and Öztemel, 2000; Rangsaritratsamee *et al*., 2004). In fact, the rescheduling problem with new orders arriving randomly is a specialized situation of the rescheduling problem caused by rush orders, when the rush orders are considered as general new orders.

A rescheduling method introduced in this paper is different in that the original jobs will be processed under the criterion of minimum mean flow time, and simultaneously the rush orders are processed at the earliest time.

PROBLEM DESCRIPTION

Provided that n jobs are being processed in a flow shop (FS) including m machines, the schedule is for n jobs and x rush orders to be processed in the FS, so the schedule must be adjusted so that the rush order products will be manufactured as soon as possible, and the original jobs will be processed under some criteria. This flow shop rescheduling problem (FSRP) can be demonstrated by a disjunctive graph. Suppose there are two jobs to be processed in an FS with a couple of machines originally, its disjunctive graph is shown in Fig.1.

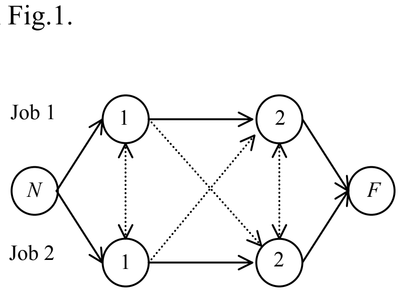


Fig.1 Disjunctive graph for FSP

In Fig.1, N represents the beginning point; F represents the terminal point; node numbers 1, 2 represent machines 1 and 2. For each job, there are single-end arcs pointing to node 2 from node 1. It means that the procedure of jobs is from machine 1 to machine 2. The beginning arc starting from N points to every point of node 1. Every point of node 2 points to F with an ending arc. Points with the same node number have double-end arcs, which means the possible order for jobs. During scheduling, these double-end arcs will become single-end arc to decide the processing order of jobs on the machines. There exist single-end arcs connecting node 1 and node 2 referring to different jobs. During scheduling, only one out of these single-end arcs will be kept, which will connect the last job on the previous machine and the first job on the next machine.

Assume that job 2 is processed first according to the original schedule. When it has been processed on machine 1, a rush order involving job 3 comes. So the manufacturing system must be rescheduled. The task of the scheduling is to determine the orders of jobs 1, 3 on machine 1 and orders of jobs 1, 2, 3 on machine 2. This situation can be demonstrated by a disjunctive graph like Fig.2.

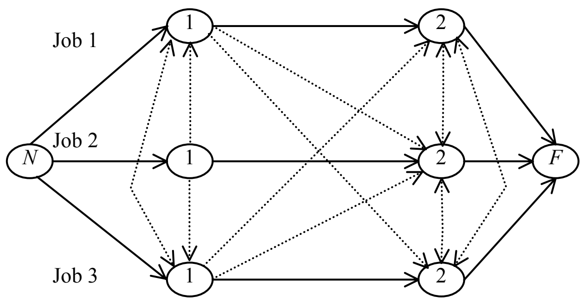


Fig.2 Disjunctive graph for FSRP

In Fig.2, there are only single-end arcs from the node 1 referring to job 2 pointing to node 1 referring to other jobs, since job 2 has already been processed on machine 1. The relationship of other nodes is similar to Fig.1.

The objective function for FSRP is the weighted mean flow time of rush orders and the original jobs. It can be illustrated as Eq.(1).

$$\min f = \frac{\lambda}{x}\sum_{j=1}^{x}(C_j - r_j) + \frac{1-\lambda}{n}\sum_{j=1}^{n}C_j, \qquad (1)$$

where λ is the weight, which is in the area (0,1); x is the number of rush orders; n is the amount of original jobs; C_j is the finishing time of job j; r_j is the arrival time of rush orders.

ACO ALGORITHM

Since the FS scheduling is an NP hard problem (Gourgand *et al.*, 2003), the FSRP is NP hard too. It needs heuristic or meta-heuristic algorithm to obtain nearly optimal solution within reasonable time. The ACO algorithm was first introduced by Dorigo *et al.*(1996), which was one of the most hopeful meta-heuristics for combinatorial optimization problems (Ying and Liao, 2004). Researchers had applied the ACO algorithm to shop scheduling problems. Gagne *et al.*(2002) compared ACO algorithm with other heuristics such as genetic algorithm, simulated annealing approach, local search method and branch-and-bound algorithm, for the single machine scheduling problem with sequence-dependent setup times, and found that ACO was competitive and had certain advantage for bigger problems. T'kindt *et al.* (2002) applied ACO algorithm to solve a 2-machine bi-criteria flow shop scheduling problem with the objective of minimizing both the total completion time and the makespan criteria. Computational experiments showed its effectiveness compared to existing heuristics called INSERT and TGB. Wang and Wu (2003) applied the ACO algorithm to no-wait flow line batch scheduling with limited batch sizes, with the jobs being partitioned into groups and jobs of the same group being processed simultaneously as a batch by the batch processing machines. Comparisons with other algorithms such as A and G-NEH on the extended Taillard's benchmark problems showed that the ACO algorithm was very efficient and robust. Kumar *et al.*(2003) applied the ACO algorithm to scheduling of flexible manufacturing systems and found that it stabilized to the solutions with considerably lesser effort and in lesser time. Ying and Liao (2004) applied the ACO algorithm to permutation flow shop scheduling problem and compared its performance with other meta-heuristics such as genetic algorithm, simulated annealing, and neighborhood search. Their computational results demonstrated that ACO was a more effective meta-heuristic for the n/m/P/$C_{\max}$ permutation flow shop scheduling problem.

The ACO algorithm is employed in this paper, because of its good features in scheduling problems as stated above. But searching by ACO may also fall into a local optimum solution, since ACO finds good routes by repeatedly feeding back pheromone to increase the pheromone of these good routes. The mutation operation introduced into ACO may help the algorithm jump out of the local optimization, so ACO with mutation (ACOM) was used to solve the FSRP in this paper.

Description of ACOM

Ant colony optimization (ACO) is an evolutionary meta-heuristic for solving combinatorial optimization problems by using principles of communicative behavior found in real ant colonies (Colorni *et al.*, 1991). When travelling, ants will leave pheromone for others. The following ants may choose route with more pheromones. The possibility for ant k to choose route (i, j) is

$$p_{ij}^{k}(g)=\begin{cases}\dfrac{[\tau_{ij}(g)]^{\alpha}[\eta_{ij}(g)]^{\beta}}{\sum\limits_{l\in allowed_k}[\tau_{ij}(g)]^{\alpha}[\eta_{ij}(g)]^{\beta}}, & \text{if } j\in allowed_k,\\ 0, & \text{else,}\end{cases} \tag{2}$$

where $\tau_{ij}(g)$ is the value of pheromone on route (i,j) at iteration g; $\eta_{ij}(g)$ is the heuristic pheromone on route (i, j) at iteration g, which is calculated by some heuristics, such as earliest due date (EDD) first heuristic and so on; α, β are exponents to adjust the importance of pheromone left by previous ants and the heuristic pheromone. They are decided by simulation ahead of searching. $allowed_k$ contains all possible routes to go for ant k at node i and iteration g.

After one iteration, computation is finished, the pheromone on all routes will be updated as follows:

$$\tau(g+1)=\rho\times\tau(g)+\sum_{k=1}^{h}\Delta\tau_{ij}^{k}, \tag{3}$$

where ρ is a constant within [0, 1]. $(1-\rho)$ means the evaporation speed of pheromone from iteration g to iteration g+1. $\Delta\tau_{ij}^{k}$ is the increment of pheromone for ant k on route (i, j) from iteration g to g+1 and is calculated as follows:

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/L_k, & \text{if ant } k \text{ passes route } (i,j),\\ 0, & \text{else,}\end{cases}$$

where Q is a constant, which means the total pheromone that one ant leaves on a complete route from the beginning point to the terminal point. L_k means the proportion of pheromone of ant k left on route (i, j) and its total pheromone.

Before the rush orders are inserted, the searching procedure of the ACOM is as follows:

(1) Assume there are h ants at the beginning point N; there is unique original pheromone $\tau_{ij}(0)$ on any route (i, j) at iteration 0;

(2) Calculate the heuristic pheromone $\eta_{ij}(g)$ of ant k (k=1, 2, …, h) on route (i, j) at iteration g by some heuristics;

(3) If there is more than one route in the allowed list for ant k, compute their selection possibilities by Eq.(2). Confirm the route that ant k will choose by roulette;

(4) After the route that ant k chooses at point i is confirmed, the possible routes in the allowed list of other points may not be feasible. These infeasible routes must be deleted from the allowed lists of these points. But the pheromones of these routes are kept for future use;

(5) After ant k arrives at the final point F, the objective function value of its route is computed. Then a couple of points are selected randomly along this route, the route between the two points is reversed. This is mutation operation. A new route is produced by the mutation operation. Compare the objective function value of the original route and the new route. If the value of new route is less, the new route is accepted for ant k;

(6) After all ants reach the final point F, mutation operation is carried out, and the route for each ant is confirmed, the stopping condition is judged. If the stopping condition occurs, the optimization is stopped, and the best route with the least objective function value is outputted. Otherwise, update the pheromones of all routes by Eq.(3), turn to Step 2;

Jobs are arranged for processing according to the best route brought by the ACOM. If x rush orders come, after z original jobs have entered y machines in the FS, the FS must be rescheduled. Rescheduling can be done by ACOM also. But the first two steps in the above procedure should be changed.

(1) Fix only one route in the allowed list on the points referring to the z jobs and y machines at any iteration. Assume there are h ants at the beginning point N; there is unique original pheromone $\tau_{ij}(0)$ on any route (i, j) at iteration 0.

(2) Calculate the heuristic pheromone $\eta_{ij}(g)$ of ant k (k=1, 2, …, h) on route (i, j) at iteration g by some heuristics, when there is more than one choice in the allowed list.

Allowed list

Before the rush orders are inserted into the processing system, each ant gradually searches the routes through all nodes. Ants travel from point N to any one point of node 1. Before an ant finishes searching all points of node 1, the most routes from node 1 to node 2 in the allowed list are deleted. Only the last point of node 1 that ant travels has routes leading to all points of node 2 in its allowed list. The allowed lists from the previous nodes to next nodes are set the same as from node 1 to node 2. All points of last node m have only one route pointing to the ending point F in their allowed list.

When rush orders are inserted into the system, z out of original jobs had been processed or are being processed on y machines. The ants will travel through the points referring to these z jobs and y machines by fixed routes, since there is only one route in each allowed list of these points. Ants travel through points by node (1, 2, …, m) as above mentioned. In all points referring to one node, if there is a fixed route, ants travel through the fixed route first, and then travel through the other routes. Rush orders bring new nodes to the route which ants will pass, the routes pointing to these new nodes have higher content of heuristic pheromone, so these routes may be chosen at the earliest time after the fixed routes are traveled.

NUMERICAL EXPERIMENTS

To our knowledge, there is no instance especially for FSRP under rush orders, so the FS benchmark instances (Beasley, 1990) *car*2 and *reC*41 are taken as examples. *car*2 is medium-scaled instance, while *reC*41 is large-scaled. The scale of *car*2 is job amount n=13, machine amount m=4. It is assumed that the latter one, two, or three jobs in *car*2 are rush orders respectively, and that in the three cases, the first one, two, or three jobs of the original optimized sequence have entered the FS, so 9 problems are obtained to study for *car*2. The scale of *reC*41 is job amount n=75, machine amount m=20. It is assumed that the last ten jobs are rush orders, while the first twenty jobs are being processed in FS for the originally optimized sequence. Both instances *car*2 and *reC*41 are taken to assess the performance of ACOM for medium-scaled and large-scale problems.

The original sequence is optimized on the criterion of mean flow time of original jobs. After the rush orders come, the sequence of all jobs is optimized according to Eq.(1). The coefficient λ in Eq.(1) is decided according to how urgent the rush orders are. When the rush orders are very urgent, the value of λ has big value (near 1). 0.8 is taken for λ in the study. According to several simulations, the weight in Eq.(3) is taken as α=0.3, β=0.7, the amount of ants h as 10. The amount of searching iteration g is 150 originally. Then the real computation iteration and time for getting the best solutions for each computation are recorded. The computation was run on a computer with Pentium 2.4 GHz CPU on Windows 2000. Every problem is computed 10 times randomly. The results are listed in Table 1. In the table, $f_{\min}$ refers to the least of the 10 times objective function values; f_{ave} refers to the mean value of the 10 times objective function values computed by ACOM; G is the mean computation iterations to get the nearly optimal solution of 10 times computation; T refers to the mean CPU time to get the nearly optimal solution of 10 times computation with ACOM, while $f'_{\min}$, G', T' refers to the corresponding values of 10 times computation calculated by the traditional ACO.

In order to test how rapid the rush orders are being processed, the ratio δ_1 of the mean flow time of rush orders and the mean real processing time of the rush orders were calculated:

$$\delta_1 = \frac{1}{x}\sum_{j=1}^{x}\left(L_j \Big/ \sum_{u=1}^{m} q_{ij} \right), \tag{4}$$

where L_j is the flow time of rush order j in the FS for the least f (Eq.(1)). It is the difference of the completing time C_j and ready time r_j of the job. In this paper r_j is taken as C_z, which is the completion time of the zth job being processed in FS. q_{ij} is the real processing time of rush order j on machine i. δ_1 is not less than 1. The nearer it is to 1, the more rapid the rush orders are processed. It can be seen from Table 1 that δ_1 increases as rush orders increase. The result is shown in Fig.3a.

After rush orders are inserted in the original n jobs, the processing of the n jobs will be delayed. In order to test how much the original jobs are delayed, the ratio δ_2 of their mean flow time before and after the insertion of rush orders is calculated.

$$\delta_2 = \sum_{j=1}^{n} C_j \Big/ \sum_{j=1}^{n} F_j, \tag{5}$$

where C_j is the flow time of original job j for the least f (Eq.(1)), after the rush orders are inserted. F_j is flow time of original job j for the least f (Eq.(1)), before the rush orders are inserted. δ_2 is not less than 1 either. The nearer it is to 1, the less the original jobs are delayed.

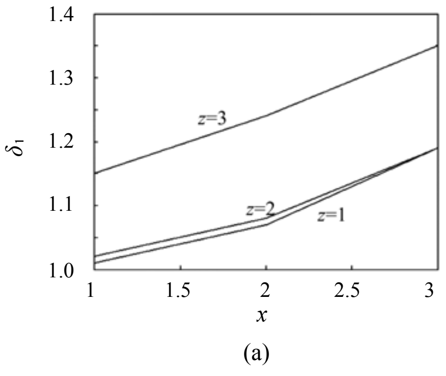
Table 1 Computation results of benchmark instances

Instance	z	x	ACOM						ACO		
			$f_{\min}$	f_{ave}	δ_1	δ_2	G	T'(s)	$f'_{\min}$	G'	T' (s)
*car*2	1	1	1951.7	1959.3	1.02	1.01	47.2	2.41	1951.7	70.6	3.23
		2	1942.1	1942.1	1.01	1.07	50.8	2.24	1942.1	68.3	2.97
		3	2436.9	2440.7	1.15	1.08	46.3	2.20	2436.9	77.1	3.15
	2	1	2776.4	2776.4	1.08	1.12	48.5	2.23	2776.4	80.4	3.46
		2	2766.8	2771.5	1.07	1.12	43.7	2.10	2766.8	74.7	3.34
		3	3006.7	3011.8	1.24	1.16	52.4	2.05	3006.7	67.9	2.85
	3	1	2873.8	2875.2	1.19	1.19	40.3	2.24	2873.8	79.0	3.88
		2	2864.2	2871.9	1.19	1.19	47.7	2.14	2864.2	72.4	3.17
		3	3077.8	3082.6	1.35	1.22	42.0	2.06	3077.8	76.6	3.06
*reC*41	10	20	2193.8	2198.3	1.8339	1.1211	78.5	54.8	2193.8	123.2	114.6

It can be seen from Table 1 that δ_2 decreases as jobs being processed increase. The result is shown in Fig.3b.

It can also been seen that with ACOM the computation time is only about 2 s for *car*2 instance, 55 s for *reC*41 instance, and that the values of $f_{\min}$ and f_{ave} are so near. It means that the repeatability and computation efficiency of ACOM algorithm are good. The computation time is about 115 s for *reC*41 instance with ACO, which is much longer than the time with ACOM. So ACOM is more efficient than ACO, when applying to FSRP.

Table 2 lists the beginning time *t* and completion time *C* of jobs on machines for *car*2 with the least objective function value, when there are 3 rush orders and 2 of original jobs has been processed in the flow shop.



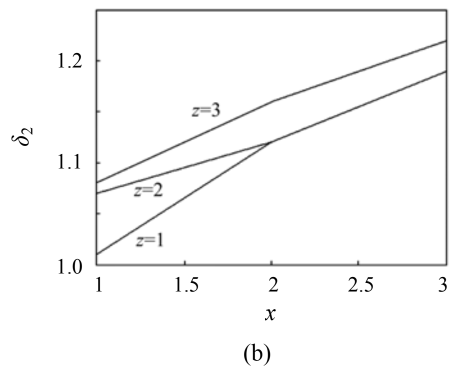


Fig.3 The effects of *x*, *z* on δ_1 (a) and on δ_2
x: Rush orders; *z*: Processing jobs

CONCLUSION

The rescheduling problem for flow shop under rush orders was studied in this work. Other kinds of rescheduling problem, such as order cancelling, material shortage, and machine breakdown, can be studied with similar methodology. If the disjunctive graph model is changed to describe open shop scheduling problem (OSP), job shop scheduling problem (JSP), or hybrid flow shop scheduling problem (HFSP), the methodology proposed in this paper can also be used to solve the rescheduling problem for these kinds of shops. If the rush orders are treated as new jobs coming randomly, the weight λ in Eq.(1) becomes 0.5. That means the dynamic scheduling problem with new jobs coming randomly is a specialized case of FSRP. The method used for FSRP is also suitable for dynamic scheduling problem with new jobs coming randomly.

Table 2 Beginning and ending time of jobs on machines

Job	Machine 1		Machine 2		Machine 3		Machine 4	
	t	*C*	*t*	*C*	*t*	*C*	*t*	*C*
1	1916	2570	2570	2717	3152	3497	3721	4168
2	3992	4313	4578	5098	5098	5887	5887	6589
3	25	37	49	196	196	826	826	1081
4	3647	3992	3992	4578	4578	4792	4934	5800
5	1238	1916	2022	2554	2877	3152	3389	3721
6	2684	3647	3647	3792	4133	4435	4709	4934
7	0	25	25	49	49	191	191	780
8	5098	5972	5972	6489	6489	6513	6823	7819
9	2570	2684	2717	3613	3613	4133	4168	4709
10	4313	5098	5098	5641	5887	6223	6589	6823
11	339	542	851	1061	1072	1771	1771	2555
12	542	1238	1238	2022	2022	2877	2877	3389
13	37	339	339	851	851	1072	1081	1426

t: beginning time of jobs; *C*: completion time of jobs

References

Aydin, M.E., Öztemel, E., 2000. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, **33**:169-178.

Beasley, J.E., 1990. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, **41**(11):1069-1072.

Colorni, A., Dorigo, M., Maniezzo, V., 1991. Distributed Optimization by Ant Colonies. Proceedings of First European Conference on Artificial Life, Elsevier Publishing, Amsterdam, p.134-142.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics*, **26**(1):24-29.

Gagne, C., Price, W.L., Gravel, M., 2002. Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, **53**:895-906.

Gourgand, M., Grangeon, N., Norre, S., 2003. Markovian analysis for performance evaluation and scheduling in *m* machine stochastic flow-shop with buffers of any capacity. *European Journal of Operational Research*, **12**:1-22.

Kumar, K., Tiwari, M.K., Shankar, R., 2003. Scheduling of flexible manufacturing systems: an ant colony optimization approach. *Proceedings of the Institution of Mechanical Engineers Part B—Journal of Engineering Manufacture*, **217**:1443-1453.

Li, H., Li, Z.C., Li, L.X., Hu, B., 2000. A production rescheduling expert simulation system. *European Journal of Operational Research*, **124**:283-293.

Rangsaritratsamee, R., Ferrell, W.G.J., Kurz, M.B., 2004. Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers and Industrial Engineering*, **46**:1-15.

Sousa, P., Ramos, C., 1999. Distributed Task Scheduling. Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning, p.436-441.

T'kindt, V., Monmarché, N., Tercinet, F., Laügt, D., 2002. An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, **142**:250-257.

Wang, X.R., Wu, T.J., 2003. An Ant Colony Optimization Approach for No-wait Flow-line Batch Scheduling with Limited Batch Sizes. Proceedings of 42nd IEEE Conference on Decision and Control, Maui, Hawaii, USA, p.2959-2964.

Ying, K.C., Liao, C.J., 2004. An ant colony system for permutation flow-shop sequencing. *Computers and Operations Research*, **31**:791-801.

Zhang, Q., Rao, Y.Q., 2003. Study on dynamic scheduling method for job shop. *Mechanical Manufacturing*, **41**(461):39-41 (in Chinese).