Paper

Three-Stage Tabu Search for Solving Large-Scale Flow Shop Scheduling Problems

Yuedong Xu* Non-member
Yajie Tian** Non-member
Nobuo Sannomiya* Member

Tabu search is a meta-heuristic approach designed skillfully for finding a suboptimal solution of combinatorial optimization problems. In this paper the tabu search with three stages is proposed for solving large-scale flow shop scheduling problems. In order to obtain a better suboptimal solution in a short computation time, three different candidate lists are used to determine the incumbent solution in the respective search stages. The candidate lists are constructed by restricting the moving of each job. Test problems with four kinds of job data are examined. Based on analyzing the relationship between the candidate list and the suboptimal solution for each job data, a common parameter is given to construct the candidate list during the search process. Comparison of the computation result is made with the genetic algorithm and the basic tabu search, from which it is shown that the proposed tabu search outperforms two others.

Keywords: Tabu search, Flow shop, Tardiness, Candidate list

1. Introduction

In manufacturing systems effective scheduling of jobs has become important for improving the efficiency of operation and decreasing the production cost. Tardiness criterion is of great importance because certain costs are incurred for a job not completed by its due date. These costs include: penalty clause in the contract, if there are any; loss of good will resulting in an increased probability of losing the customer for some or all future jobs, and a damaged reputation which will turn other customers away [1]. Flow shop problems viewed as sequence problems (called permutation flow shop problems) are concerned in this paper and the objective is to minimize the total tardiness $F||\sum T_i$.

Tabu search(TS) was proposed by Glover [2], and aims at guiding the search beyond local optimum in the solution space of a given problem. In the real production field for large-scale and complicated scheduling problems we have to accept a suboptimal solution, because the computation time is limited. According to the paper written by Armentano [3] and Diaz [4], restricting the neighborhood of the current solution leads to reduction of computation time and improvement of the computational efficiency.

TS is one method of neighborhood search, and its performance depends on the neighborhood structure. Different neighborhoods have different properties, which may make them more or less suitable depending on the particular problem at hand. In our earlier paper [5], we proposed a tabu search with restricted neighborhood(TSRN) to solve flow shop problems. The knowledge obtained from the simulation of an adaptive behavior of fish schools has been used to define the neighborhood in TSRN. When the computation time is limited, the suitable neighborhood size has been given in TSRN for different kinds of job data.

We also proposed a tabu search with two stages(TSTS) [6] for solving the parallel machine problems $P||\sum T_i$. In TSTS the heuristic method NEH is used to order the jobs assigned to each machine at the first stage, and the candidate list with small size is employed for reducing the search space at this stage. In order to rise the accuracy of the suboptimal solution obtained by the first stage, the intensive search with large candidate list is performed at the second stage. Because the search route is changed by changing the structure of candidate list, the risk of trapping in local minimum is reduced in TSTS.

In this paper a tabu search with three stages(TS3S) is proposed to minimize $F||\sum T_i$. Because the solution contribution changes along with the search process, the different candidate lists are used in the three search stages for determining the incumbent solution. At the beginning there are many neighboring solutions better than the current solution, the random search is used and the candidate list with small size is employed to reduce the computation time. As the accuracy of solution rises, the number of better solutions included in the neighborhood set is reduced, and the large candidate list is used to improve a suboptimal solution.

The standard approach of intensification and diversification is able to improve the accuracy of the suboptimal solution. But the computational time is not acceptable

* Kyoto Institute of Technology.
Matsugasaki, Sakyo-ku, Kyoto 606-8585
** Kyoto University.
Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501

for large-scale problems. In TS3S new methods are used to determine the intensification and the diversification approach.

2. Flow Shop Problem

The permutation flow shop problem is formulated as follows. n jobs $\{J_1, J_2, \cdots, J_n\}$ have to be processed in the same sequence on m machines. The processing time of job J_i on machine M_j is given by $p_{ij}(i = 1, 2, \cdots, n, j = 1, 2, \cdots, m)$. These times are non negative and some of them may be zero if the job is not processed on the machine. The due date of job J_i is d_i.

For this problem the following assumptions are made:

- Every machine processes only one job at a time.
- Every job is processed on one machine at a time.
- Every job has to be processed at most once on machine $M_1, M_2, \cdots, M_m$(in this order).
- The operations are not preemptive.

The objective function is to minimize the sum of the tardiness of each job. Let s_{ij} and $c_{ij}(c_{ij} = s_{ij} + p_{ij})$ be the start time and the completion time of operation O_{ij} of job J_i processed on machine M_j , respectively.

The problem is to find a sequence $\pi = \{J_{i_1}, \cdots, J_{i_n}\}$ so that the following objective function reaches a minimum value.

$$Z(\pi) = \sum_{i=1}^{n} \max(c_{im} - d_i, 0) \quad \cdots\cdots\cdots\cdots\cdots (1)$$

3. Tabu Search

TS is a meta-heuristic approach which can be designed skillfully for finding a suboptimal solution of combinatorial optimization problems. Basically it consists of several components called the *move, neighborhood, initial solution, attribute selection, tabu list, aspiration criterion,* and *stopping rules.* TS starts from an initial solution. The move is a function which transforms a solution into another solution. The subset of moves applied to a given solution generates a collection of solutions called the neighborhood. The candidate list $C_L(\pi)$ is defined as the list of solutions evaluated in the neighborhood of the correct solution π. At each iteration a move leads to the best solution in $C_L(\pi)$ which may not be an improved solution. To avoid cycling and more generally to add robustness to the search, a recency-based short term memory called a tabu list $T = T_1, T_2, \ldots, T_n$ gives forbidden moves. A tabu list consists of attributes of the latest moves which are not allowed to be repeated during several iterations (the tabu tenure). But a forbidden move can be admitted if an aspiration criterion is satisfied.

The basic tabu search(BTS) is given as follows.

BTS

Step1: Set $t \leftarrow 1$ as an iteration number. Select the number x from $[n/2, n]$ randomly. Set the initial tabu list $T_j = 0; j = 1, 2, \cdots, n$. $t_n \leftarrow 0$.

Step2: Generate the initial solution π. $\pi^* \leftarrow \pi$ and $\pi_{best} \leftarrow \pi$.

Step3: Construct the candidate list $C_L(\pi^*)$.

Step4: Obtain $Z(\pi_l) = \min\{Z(\pi_i); \pi_i \in C_L(\pi^*)\}$.

Step5: If $Z(\pi_l) < Z(\pi_{best})$ then $\pi_{best} \leftarrow \pi_l$, $t_n \leftarrow 0$, $T_l \leftarrow x$, and go to Step 7. Otherwise find the best π_l in $C_L(\pi^*)$ subject to $T_l = 0$. $t_n \leftarrow t_n + 1$.

Step6: If $Z(\pi_l) < Z(\pi^*)$, then $T_l \leftarrow x - 1$. If $Z(\pi_l) \geq Z(\pi^*)$, then $T_l \leftarrow x + 1$.

Step7: Renew the tabu list, $T_i \leftarrow T_i - 1$; $i = 1, 2, \ldots, l-1, l+1, \ldots, n$. $\pi^* \leftarrow \pi_l$.

Step8: If $t_n = n$, π_{best} is adopted as the suboptimal solution and stop the computation. Otherwise $t \leftarrow t + 1$.

Step9: If the value of $t/20$ is an integer, update x by an integer selected randomly from $n/2$ to n. Go to Step 3.

4. Components of the Proposed Tabu Search

The following components are adopted for the proposed TS.

Initial solution: At the initial sequence π_s^1, jobs are arranged in the non-decreasing order of due dates called EDD (earliest due date).

Move Among many types of moves considered in the literature for the flow shop problem, the following two types are used prominently:

Swapping Swap jobs placed at the ath and bth positions.

Insertion Remove a job placed at the ath position and insert it in the bth position.

In the proposed TS both types of moves are applied at the identical probability.

Neighborhood: For a given solution π, the solutions generated by swapping and inserting compose the swapping neighborhood $SN(\pi)$ and the insertion neighborhood $IN(\pi)$, respectively. For flow shop scheduling problems with n jobs, the size of $SN(\pi)$ is $n(n-1)/2$, and that of $IN(\pi)$ is $(n-1)^2$. If the value of n is large, we have heavy computation burden on checking the whole neighborhood to determine the incumbent solution.

Candidate List: The candidate list$C_L(\pi)$ is applied in order to reduce the effort for evaluating the neighborhood at each iteration of the search process. How to select the neighboring solutions from the neighborhood, and how many neighboring solutions should be selected to construct $C_L(\pi)$ are difficult problems for us. In the proposed TS3S three different $C_L^i(\pi)(i = 1, 2, 3)$ are used at the respective search stages.

Tabu list: The tabu list is defined as the set $T = T_1, T_2, \ldots, T_n$. T_i is the tabu tenure of job J_i. J_i can not be chosen for move until $T_i = 0$.

Tabu tenure: The tabu tenure is set dynamically. For every 20 iterations a new value for the tabu tenure is generated from a uniform distribution between $n/2$ and n, and at each iteration this value is adjusted. This adjustment is carried out according to the value of the selected solution(the incumbent solution) compared with the solution (the current solution)that generated the move. If the objective value of the incumbent solution is larger than that of the current solution, the tabu tenure of J_i, whose moving leads to the current solution translated to the incumbent solution,

is increased by one unit. Otherwise the tabu tenure of J_i is decreased by one unit. A similar procedure was proposed by Dell'Amico et al. [7], and used by Armentano [3].

Aspiration criterion A move can be released from the tabu condition if it has a better solution than the best solution found by the search until that moment.

Stop rule When the suboptimal solution is not improved after continuing n iterations, the search process is stopped at each stage.

5. Design of the Tabu Search with Three Stages

TS is one of neighborhood search algorithms. The starting point for the neighborhood search method is the choice of neighborhood structure; this determines which pairs of solutions are regarded as adjacent to each other. Consequently allowable moves are determined.

Small neighborhoods are preferable from the viewpoint of the speed at which the local search proceeds. If the neighborhood is too small, however, the final solution may be of poor quality. Neighborhood search techniques should be done in such a way that a move to an adjacent solution leads to a rapid update of the objective function value, rather than a complete recalculation based on the whole neighborhood [8].

In this paper our purpose is to propose a tabu search for solving large-scale flow shop problems. Instead of the whole neighborhood the candidate list is used for determining the incumbent solution. The construction of the candidate list is different for the respective stages.

5.1 **The First Stage Process TS3S-I** At the first stage, the search process begins from the initial solution which is obtained by EDD. Since at the beginning search stage we can easily find better neighboring solutions than the current solution, it is not necessary to recalculate all of the neighboring solutions to determine the incumbent solution. The candidate list with small size is used at this stage. The candidate list $C_L^1(\pi)$ of the current solution π is constructed as follows:

Construction of $C_L^1(\pi)$

Step1: $i \leftarrow 1$.

Step2: Job J_i is removed from the current position to the other arbitrary position in π, or is swapped with an arbitrary job. The probabilities of insertion and swapping are 50%, respectively. The new solution is represented as π^i. The move distance $D_i(\pi, \pi^i)$ is given by:

$$D_i(\pi, \pi^i) = |P_\pi^i - P_{\pi^i}^i| \quad \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

where P_π^i and $P_{\pi^i}^i$ are the position number of J_i in sequence π and π^i, respectively.

Step3: If $i = n$ stop; otherwise $i \leftarrow i + 1$ and go to Step 2.

By this way there are n neighboring solutions included in $C_L^1(\pi)$. At each iteration only n solutions are computed, and one solution π^l is selected as the incumbent solution for the next iteration. We define $\delta_i (i = 1, 2, \ldots, n)$ as the number of moves whose distance is i during the search process. If at an iteration π translates to π^l, and

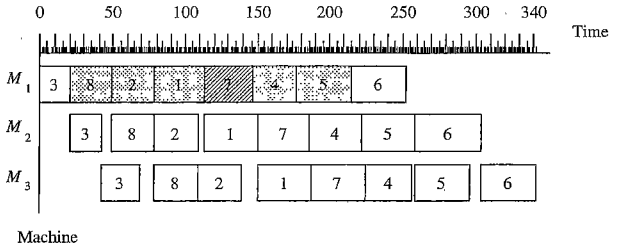


Fig. 1. Definition of neighboring jobs ($J_i = J_7$ and $I = 5$).

$D_l(\pi, \pi^l)$ is the move distance of J_l, then $\delta_{D_l(\pi,\pi^l)} \leftarrow \delta_{D_l(\pi,\pi^l)} + 1$.

5.2 **The Second Stage Process TS3S-II** In TS the best solution included in candidate list is selected as the incumbent solution. The EDD is used as the initial solution at the first stage. The solution generated with large $D_i(\pi, \pi^i)$ is difficult to be selected as an incumbent solution, when the due dates of job data distribute in the wide time range. Since we focused on the random selection in the first stage, the distribution of δ_i relates with the characteristic of the job data.

How can this information be used to predict which moves are likely to lead to improvements in the objective function value? The answer is that if we can restrict the move distance of job J_i in a suitable range,the evaluation of unnecessary non-improving moves is avoided and consequently search efficiency is improved. Therefore our purpose is to determine the suitable move range. The parameter K is introduced to restrict the moving of job in the sequence, and the value of K is determined as the minimal value obtained from the following equation.

$$\rho \leq \sum_{i=1}^{K} \delta_i \Big/ \sum_{i=1}^{n} \delta_i \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots (3)$$

We will give the value of ρ by analyzing the computation result later.

In our earlier paper [5], the set $C(\pi, J_i, I)$ is used to restrict the moving of each job when generating the neighboring solutions of the current solution. The definition of $C(\pi, J_i, I)$ is given as follows:

$$\sigma_{ik} = \begin{cases} s_{i1} - c_{k1} & \text{for } c_{k1} \leq s_{i1} \\ s_{k1} - c_{i1} & \text{for } s_{k1} > c_{i1} \end{cases} \quad \cdots\cdots\cdots (4)$$
$$k = 1, 2, \cdots, n, \ k \neq i$$

$$C(\pi, J_i, I) = \{J_k \mid k \neq i, k \in \{k_1, k_2, \cdot, k_I\} \ \sigma_{ik_l}(l = 1, 2, \cdots, I) \text{ are the least } I \text{ values of } \sigma_{\mathrm{ik}}\} \quad \cdots\cdots\cdots (5)$$

When $\sigma_{ij} = \sigma_{ik} (j \neq k)$, select one job from jobs J_j and J_k randomly and include it in set $C(\pi, J_i, I)$. Therefore the set $C(\pi, J_i, I)$ consists of the I nearest jobs from job J_i in sequence π. Fig. 1 shows an example of the Gantt chart for $(n, m) = (8, 3)$ and $I = 5$. For $J_i = J_7$ we have the set $C(\pi, J_7, 5) = \{J_1, J_4, J_2, J_5, J_8\}$.

Depending on $C(\pi, J_i, I)$ the candidate list $C_L^2(\pi)$ is

constructed as follows:

Construction of $C_L^2(\pi)$

Step1: $i \leftarrow 1$.

Step2: Remove J_i from the current position and place it at the position where an arbitrary job included in set $C(\pi, J_i, I)$ is placed, or swap J_i with an arbitrary job included in set $C(\pi, J_i, I)$. The probabilities of insertion and swapping are 50%, respectively.

Step3: If $i = n$ stop; otherwise $i \leftarrow i + 1$ and go to Step 2.

Because the I nearest jobs are found forward and backward from J_i in sequence π, when $I \cong 2 \times K$, the move whose distance is shorter than K can be obtained by Step 2.

The candidate list $C_L^2(\pi)$ is composed with n neighboring solutions. Because the moving of each job is restricted by the set $C(\pi, J_i, I)$, the search space in this stage is reduced.

In TS an unimproved solution can be selected as the incumbent solution in order to reduce the risk to trap in the local minimum. At the first stage the size of candidate list is small and the solutions included in candidate list are generated randomly. Therefore the solution in which a job is moved to an unreasonable position may be accepted at the first stage as an incumbent solution. At the second stage the jobs can be moved to only the I nearest positions. If a job is moved out of this range at the first stage, it is difficult to be put back at the second stage. Based on the initial solution π_s^1 and the best solution π_{best}^1 obtained by the first stage, we generate the initial solution π_s^2 for the second stage by the following algorithm.

Algorithm A

Step1: $\pi_s^2 \leftarrow \pi_{best}^1$

Step2: Calculate the value of $D_i(\pi_s^1, \pi_s^2), (i = 1, 2, \ldots, n)$ from Eq. (2). $D_i(\pi_s^1, \pi_s^2)$ is the difference of the positions of J_i in solutions π_s^1 and π_s^2.

Step3: $D_l \leftarrow \max\{D_i(\pi_s^1, \pi_s^2); i = 1, 2, \ldots, n\}$. If $D_l \leq K$, then stop.

Step4: Move job J_l in the solution π_s^2 from its current position to the position where J_l is placed in solution π_s^1. Save the new solution as π_s^2, and go to Step 2.

In TS the diversification strategy is used to drive the search into a new region of the solution space which has not been examined yet. The new feasible initial solution is needed to restart the search process. Two prominent methods are used to generate the new initial solution; multi-start approach and systematic approach [9]. In the multi-start approach the solution is generated randomly. In this case it represents a schedule which has been examined earlier, and the process is repeated. The systematic approach is to generate the solution based on how often each job has occurred at each position so far. This frequency is stored in a matrix which has to be updated at each iteration.

In TS3S-II a new initial solution is generated by algorithm A based on the solutions π_s^1 and π_{best}^1. The candidate list in TS3S-II is different from that used in TS3S-I, and then the search is not repeated. Because only two solutions are recorded during the search process, the computation time is reduced.

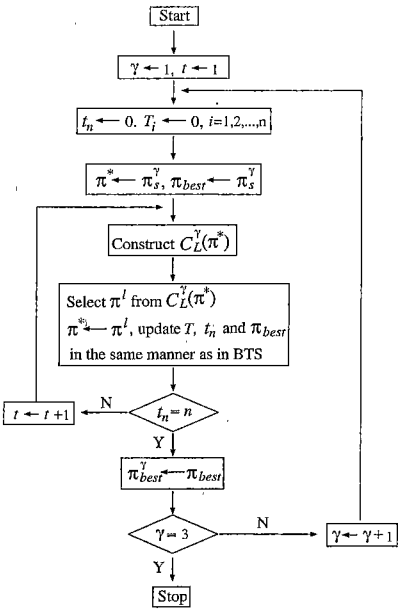


Fig. 2. The flowchart of TS3S.

5.3 **The Third Stage Process TS3S-III** In this stage the purpose is to improve the best solution obtained by two former stages. In the case of using the small neighborhood, the final solution may be of poor quality. Therefore the number of solution included in the candidate list $C_L^3(\pi)$ is increased. Because the value of I relates with the characteristic of job data, we also restrict the move of jobs when the candidate list $C_L^3(\pi)$ is constructed. The construction is as follows.

Construction of $C_L^3(\pi)$

Step1: $i \leftarrow 1$.

Step2: Remove J_i from the current position and place it at the position where every job included in set $C(\pi, J_i, I)$ is placed, to generate set $\{\pi^{iI_l}; l = 1, 2, \ldots, I\}$. π^{iI_l} is the l-th solution obtained by insertion of job J_i. Swap J_i with every job included in set $C(\pi, J_i, I)$ to generate set $\{\pi^{iS_l}; l = 1, 2, \ldots, I\}$. π^{iS_l} is the l-th solution obtained by swapping of job J_i.

Step3: If $i = n$ stop; otherwise $i \leftarrow i + 1$ and return to Step 2.

Then we have $C_L^3(\pi) = \{\pi^{iI_l} \cup \pi^{iS_l}, i = 1, 2, \ldots, n; l = 1, 2, \ldots, I\}$. In this stage an intensive search is performed in the reduced solution space, and the number of solutions included in candidate list increases with I.

5.4 **Whole Algorithm of TS3S** Fig. 2 shows the flowchart of the whole algorithm for TS3S. In the TS3S the same procedure as BTS is used repeatedly at the three search stages, but candidate lists $C_L^i(\pi); i = 1, 2, 3$ are used at the respective stages. $\pi_s^\gamma; \gamma = 1, 2, 3$ are the initial solutions for the respective stages. The solution π_s^1 generated by EDD is used as the initial solution of TS3S-I. The initial solution π_s^2 of TS3S-II is generated by algorithm A. The initial solution π_s^3 of TS3S-III is the better solution between π_{best}^1 and π_{best}^2. The suboptimal solution is obtained as the final solution π_{best}^3 of the third

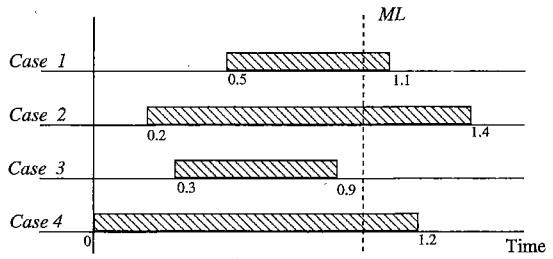


Fig. 3. Distribution of due date.

stage.

6. Computational Experiment

An experiment was carried out to examine the performance of TS3S. The algorithm is coded in C and the experiments are performed on Compaq ProLiant ML330. CPU is Pentium 866MHZ.

6.1 Test Problem We generated randomly a number of instances with various due dates and processing times. The processing times of operations p_{ij} are selected from discrete uniform distribution [1,100]. The due dates of job data are uniformly distributed between $ML \times (1 - \tau - \frac{R}{2})$ and $ML \times (1 - \tau + \frac{R}{2})$, where ML is a lower bound of the makespan, which is given by

$$ML = \max\{\max_{1\le j\le m}\{\sum_{i=1}^{n} p_{ij} + \min_{i}\sum_{l=1}^{j-1} p_{il} + \min_{i}\sum_{l=j+1}^{m} p_{il}\}, \max_{i}\sum_{j=i}^{m} p_{ij}\} \quad \cdots\cdots\cdots (6)$$

τ and R are the tardiness factor and the range of due date [10], respectively. Fig. 3 shows the following four cases of distribution of due date with different τ and R.

Case 1: low tardiness factor ($\tau = 0.2$) and small due date range ($R = 0.6$)
Case 2: low tardiness factor ($\tau = 0.2$) and wide due date range ($R = 1.2$)
Case 3: high tardiness factor ($\tau = 0.4$) and small due date range ($R = 0.6$)
Case 4: high tardiness factor ($\tau = 0.4$) and wide due date range ($R = 1.2$)

The release date of job J_i is uniformly distributed in the range $[0, d_i - \sum_{j=1}^{m} p_{ij}]$.

6.2 Computation Results by TS3S Ten test instances $\omega_i (i = 1, 2, \ldots, 10)$ with $m = 3$, $n = 200$ are generated for each case. Then 40 test instances are treated by TS3S, where the value ρ is changed from 65% to 95%. For each instance ω_i ten different runs are performed with various ρ by changing random numbers. At each run the objective values $Z(\pi^1_{best})$, $Z(\pi^2_{best})$, and $Z(\pi^3_{best})$ are recorded.

For each case the three values of instance ω_i obtained by the l-th run are denoted as $Z(\pi^{\gamma}_{best})^l_{i\rho} (\gamma = 1, 2, 3; i = 1, 2, \ldots, 10; l = 1, 2, \ldots, 10;$ and $\rho = 65\%, 70\%, \ldots, 95\%)$. Then their averages for ten runs

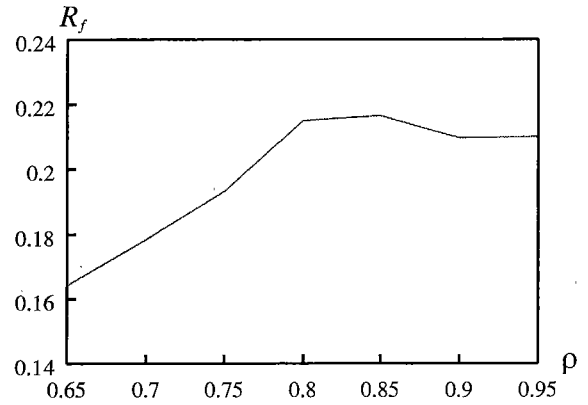


Fig. 4. Variation of R_f with ρ for problems in Case 4.

are given by

$$\bar{Z}(\pi^{\gamma}_{best})_{i\rho} = \frac{1}{10}\sum_{l=1}^{10} Z(\pi^{\gamma}_{best})^l_{i\rho} \quad \cdots\cdots\cdots (7)$$

The average relative improvement obtained by TS3S-II with respect to TS3S-I, and that obtained by TS3S-III with respect to TS-II are given respectively by

$$R_{21} = \frac{1}{10}\sum_{i=1}^{10} \frac{\bar{Z}(\pi^1_{best})_{i\rho} - \bar{Z}(\pi^2_{best})_{i\rho}}{\bar{Z}(\pi^1_{best})_{i\rho}} \quad \cdots\cdots\cdots (8)$$

$$R_{32} = \frac{1}{10}\sum_{i=1}^{10} \frac{\bar{Z}(\pi^2_{best})_{i\rho} - \bar{Z}(\pi^3_{best})_{i\rho}}{\bar{Z}(\pi^2_{best})_{i\rho}} \quad \cdots\cdots\cdots (9)$$

The total improvement rate is given by

$$R_f = \frac{1}{10}\sum_{i=1}^{10} \frac{\bar{Z}(\pi^1_{best})_{i\rho} - \bar{Z}(\pi^3_{best})_{i\rho}}{\bar{Z}(\pi^1_{best})_{i\rho}} \quad \cdots\cdots\cdots (10)$$

The values R_{21}, R_{32} and R_f depend on ρ, but its dependency is omitted in their expression for simplicity of notation.

Tables 1 and 2 show the values of R_{21} and R_{32} for various ρ. From the two tables the approaches of TS3S-II and TS3S-III are not effective for the instances with the low tardiness factor(Cases 1 and 2). For Case 3 a little improvement(R_{21}) is obtained by TS3S-II as shown in Table 1. For these three cases the value of ρ does not affect the values of R_{21} and R_{32}. On the other hand, Case 4 corresponds to the problems with high tardiness and wide due date range. Consequently it is considered that Case 4 treats difficult problems. For such case the significant improvements of R_{21} and R_{32} are obtained, and the approaches of TS3S-II and TS3S-III are effective.

The relationship between R_f and ρ for Case 4 is shown in Fig. 4. When $\rho < 80\%$ the value of R_f is improved as ρ increases, but when $\rho \ge 80\%$ this trend is saturated. Table 3 shows the variation of the average CPU time taken by TS3S with ρ. Because the number of solutions included in the candidate list is increased with ρ, the cost of CPU time also increases with ρ. Based on Fig. 4 and Table 3, and also because ρ does not affect the

Table 1. The average Improvement R_{21}.

ρ	65%	70%	75%	80%	85%	90%	95%
Case 1	0.006	0.005	0.006	0.004	0.006	0.006	0.005
Case 2	0.002	0.002	0.003	0.002	0.002	0.002	0.002
Case 3	0.012	0.011	0.011	0.012	0.011	0.011	0.011
Case 4	0.086	0.084	0.114	0.085	0.135	0.117	0.100

Table 2. The average Improvement R_{32}.

ρ	65%	70%	75%	80%	85%	90%	95%
Case 1	0.002	0.002	0.002	0.002	0.003	0.003	0.003
Case 2	0.002	0.004	0.003	0.003	0.003	0.003	0.004
Case 3	0.003	0.004	0.003	0.004	0.003	0.004	0.004
Case 4	0.072	0.087	0.071	0.120	0.072	0.083	0.100

Table 3. CPU time for various ρ.

ρ	65%	70%	75%	80%	85%	90%	95%
Case 1	227	260	280	333	383	399	467
Case 2	139	150	167	182	199	223	264
Case 3	158	187	239	249	315	381	432
Case 4	180	200	221	278	306	375	485

results in Cases 1∼3, we recommend $\rho = 80\%$ as the suitable value of an important parameter in TS3S.

Fig. 5 shows the convergence curve for one run of an instance in Case 4 with $\rho = 80\%$. In the first stage and the second stage, the considerable convergence speed is obtained though the size of the candidate list is small.

After continuing $n(=200)$ iterations with no improvement in the objective value in TS3S-I, the search of TS3S-I attains a local optimum. At this time the diversification is performed, which is to restart the search from a new initial solution. By using the new candidate list and the new initial solution in TS3S-II, the search escapes from the local optimum and converges on a solution better than the suboptimal solution obtained by TS3S-I. The suboptimal solution obtained by TS3S-II is improved by intensive search in TS3S-III.

6.3 Comparison with Other Algorithms BTS proposed by Armentano [3] and the genetic algorithm with search space reductions(RCGA) proposed by Zhao et al. [11] are employed to treat the instances in the four cases. The same procedure as presented in Section 3 is used by BTS. Instead of the candidate list $C_L(\pi)$ the insertion neighborhood $IN(\pi)$ is employed to determine the incumbent solution for the next iteration in BTS. It is shown in Zhaos' paper [11] that RCGA outperforms the standard genetic algorithm for solving large-scale flow shop problems. Then we adopt RCGA as an example of genetic algorithms. The population size of RCGA is set to 500 and the algorithm stops after 5000 generations. Crossover rate P_c is 1.0 and mutation rate P_m is 0.05.

We treat ten instances for each case. Each instance is treated ten times by BTS and RCGA with changing random numbers. Table 4 shows the average of the objective value obtained by each method for each case. For the instances in Cases 1 and 2, the difference of suboptimal solution among BTS, RCGA and TS3S is small. For such cases not so bad solutions can be also obtained by heuristic methods such as EDD or ERT(earliest release time). For Case 3 TS3S outperforms BTS and RCGA a little, and for Case 4 significant improvement is obtained by TS3S as compared with BTS and RCGA. For Case 3 and Case 4, the results obtained by heuristic methods are inferior.

Table 4. The average objective values obtained by various methods.

	EDD	ERT	BTS	RCGA	TS3S
Case 1	3582.5	3494.2	3402.4	3398.4	3399.3
Case 2	961.8	866.8	838.5	838	837.3
Case 3	41744.3	41565.3	27317.2	27472.5	26982.4
Case 4	33139.5	87549.5	4739.3	4753.9	4370.1

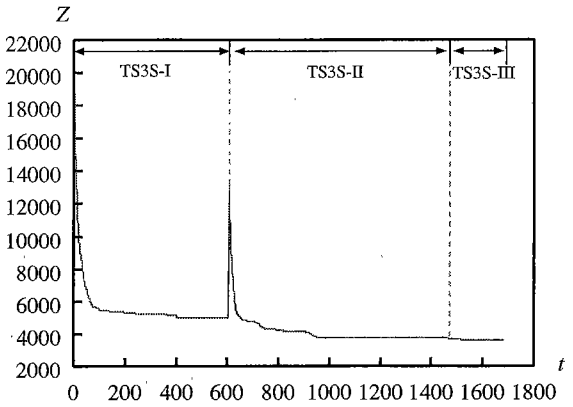


Fig. 5. Convergence curve of TS3S with $\rho = 80\%$.

Table 5. Comparison among BTS, RCGA and TS3S.

	CPU time (sec.)			Improvement rate	
	BTS	RCGA	TS3S	$R_{BTS/RCGA}$	$R_{BTS/TS3S}$
Case 1	440	358	333	0.002	0.001
Case 2	434	358	182	0	0.001
Case 3	778	358	249	-0.006	0.01
Case 4	1065	358	278	-0.025	0.094

Let $\bar{Z}_i^B$ and $\bar{Z}_i^G (i = 1, 2, \ldots, 10)$ be the average objective values obtained by BTS and RCGA for each instance, respectively. The average improvement rate of RCGA for BTS and that of TS3S with $\rho = 80\%$ for BTS are given, respectively, by

$$R_{BTS/RCGA} = \frac{1}{10}\sum_{i=1}^{10}\frac{\bar{Z}_i^B - \bar{Z}_i^G}{\bar{Z}_i^B} \quad \cdots\cdots\cdots (11)$$

$$R_{BTS/TS3S} = \frac{1}{10}\sum_{i=1}^{10}\frac{\bar{Z}_i^B - \bar{Z}(\pi_{best}^3)_{i80\%}}{\bar{Z}_i^B} \quad \cdots (12)$$

Table 5 shows the CPU times taken by three algorithms and the values of $R_{BTS/RCGA}$ and $R_{BTS/TS3S}$ for four cases of test problems. Since the search space is reduced in TS3S, the computation time taken by TS3S is shorter than that taken by BTS and RCGA.

From the CPU time taken by BTS for computing the problems for Case 4, we also consider the instances in Case 4 are more difficult problems, because more CPU time is needed.

It is observed from Table 5 that RCGA does not outperform BTS for solving large-scale flow shop scheduling problems. When $\rho = 80\%$, the solutions obtained by TS3S are better than those obtained by RCGA, and less CPU time is taken by TS3S.

Table 6. The statistical value of the differences in the suboptimal objective value and CPU time.

	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Data8	Data9	Data10
$Z_{BTS/TS3T}$	24.9	13.0	9.5	8.8	12.1	3.7	35.7	4.0	69.0	43.0
$Z_{RCGA/TS3T}$	13.2	11.2	17.7	8.5	12.8	2.6	6.9	10.9	10.31	17.3
$T_{BTS/TS3T}$	31.3	23.8	25.8	28.6	39.5	23.0	78.4	26.2	35.6	19.3
$T_{RCGA/TS3T}$	6.7	8.3	9.7	4.9	3.1	18.3	27.7	73.3	19.3	29.7

Armentano also proposed a neighborhood restriction strategy in TS, by which 40% computation time is reduced for BTS, but the suboptimal solution is not better than that obtained by BTS [3]. On the other hand, Table 5 shows that, as compared with BTS, TS3S attains $25 \sim 68\%$ reduction in CPU time and almost same accuracy for Case 1 to 3. In addition for Case 4 TS3S outperforms BTS significantly from the viewpoints of CPU time and accuracy.

For further investigation, each of ten instances in Case 4, called Data 1 to Data 10, is treated 100 times by BTS, RCGA and TS3S with changing random numbers, and the result of statistics for each instance is obtained. The level of significance α is set to 0.05. In order to apply the 95% statistical test to these data, the following statistical values are defined for the differences in the suboptimal objective value:

$$Z_{BTS/TS3S} = \frac{\bar{Z}_B - \bar{Z}(\pi^3_{best})}{\sqrt{\frac{S_B^2}{99} + \frac{S_T^2}{99}}} \quad \cdots\cdots\cdots\cdots (13)$$

$$Z_{RCGA/TS3S} = \frac{\bar{Z}_G - \bar{Z}(\pi^3_{best})}{\sqrt{\frac{S_G^2}{99} + \frac{S_T^2}{99}}} \quad \cdots\cdots\cdots\cdots (14)$$

Where $\bar{Z}_B$, $\bar{Z}_G$ and $\bar{Z}(\pi^3_{best})$ are, respectively, the average objective values obtained by BTS, RCGA and TS3S for each data. The values S_B^2, S_G^2 and S_T^2 are the sampling variances of $\bar{Z}_B$, $\bar{Z}_G$ and $\bar{Z}(\pi^3_{best})$, respectively. In the same way, the statistical values $T_{BTS/TS3S}$ and $T_{RCGA/TS3S}$ are defined for the differences in the CPU time.

Table 6 shows the result of the 95% statistical testing for ten data. All the values in this table are larger than 1.96, which means that significant differences in both the suboptimal objective value and the CPU time exist between TS3S and BTS(or RCGA) under the assumption of normal distribution because of large sampling size. This shows that TS3S outperforms other two methods.

7. Conclusion

In this paper a three-stage tabu search has been proposed for solving large-scale flow shop scheduling problems. Because in TS3S different candidate lists are used during the search process, the risk of trapping into local minimum is reduced. Consequently the quality of suboptimal solution is improved. Small size candidate lists are employed by TS3S-I and TS3S-II, and then the computation time is saved.

The recommendation of an important parameter is given as $\rho = 80\%$ after analyzing the computation result. Based on ρ and the move distance recorded in TS3S-I, the parameter I can be determined, from which the search space is reduced in TS3S-II and TS3S-III.

Based on the computational experiences for four cases of test problems, TS3S is proved to be effective, especially for the difficult problems(Case 4), as compared with the genetic algorithm and the basic tabu search.

(Manuscript received March 22, 2002, revised October 9, 2002)

References

(1) T. Sen and S. Gupta: "A state-of-art survey of static scheduling research involving due dates", *OMEGA*, Vol.12, pp.63-76 (1994)

(2) F. Glover: Tabu search-part I, *ORSA J. Computing*, Vol.1, pp.190-206 (1986)

(3) V. A. Armentano and D. P. Ronconi: "Tabu search for total tardiness minimization in flowshop scheduling problems", *Computers &Operations Research*, Vol.26, pp.219-235 (1999)

(4) A. B. Diaz: "Restricted neighborhood in the tabu search for flow shop problem", *European Journal of Operational Research*, Vol.62, pp.27-37 (1992)

(5) Y. Xu, Y. Tian, and N. Sannomiya: "A tabu search based on simulation of adaptive fish behavior and its application to flow shop scheduling problems", *Proc. of 2001 IEEE Intern. Conf. on System, Man and Cybernetics*, pp.2980-2985 (2001)

(6) Y. Xu, Y. Tian, and N. Sannomiya: "Two-stage tabu search for solving parallel machine problems", *Trans. of the Society of Instrument and Control Engineers*, Vol.38, No.3, pp.330–332 (2002)

(7) M. Dell'Amico and M.Trubian: "Applying tabu search to the job-shop scheduling problem", *Annals of Operations Research*, Vol.41, pp.231-252 (1993)

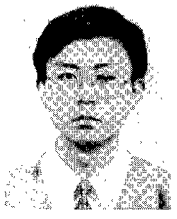
(8) E. Arts and J.K. Lenstra: "Local search in Combinatorial Optimization", *Wiley Interscience Publication* (1993)

(9) R. Klein: "Scheduling of resource-constrained projects", *Kluwer Academic Publishers*, pp.196-208 (1999)

(10) Ow. PS: "Focused scheduling in proportional flowshops", *Management Science*, Vol.31, pp.852-869 (1985)

(11) Y. Zhao and N. Sannomiya: "An improvement of genetic algorithms by search space reduction in solving large-scale flowshop problems", *Trans. IEE of Japan*, Vol.121-C, No.6, pp.1010-1015 (2001)

Yuedong Xu (Non-member) received the B.E. degree in Electronics and Information Department from Qingdao University, China in 1995 and received the M.E. degree in Electronics and Information Science, Kyoto Institute of Technology. Since 2001, he has been a doctor-course student at Division of Information and Production Science, Graduate School of Kyoto Institute of Technology. His research interests include combinatorial problems, metaheuristics and production schedulings.



Yajie Tian (Non-member) received the Ph.D at Division of Information and Production Science, Graduate School of Kyoto Institute of Technology in 1996. She was with the Graduate School of Informatics of Kyoto University as a research associate from 1997 to 2002. Then she joined the Human Information Science Laboratories of ATR where she is currently a Senior Researcher. Her present research interests are the modeling and optimization techniques combined with real world problems and multi-agent systems acquiring intelligent skills of experts. She is a member of SICE and IEEE.

Nobuo Sannomiya (Member) received the B.E., M.E. and D.E. degrees in Electrical Engineering all from Kyoto University in 1962, 1964 and 1969, respectively. Since 1986 he has been a Professor at Department of Electronics and Information Science, Kyoto Institute of Technology. His present research interests include modeling and optimization techniques, and their applications. Especially his research works are directed toward metaheuristic approaches to the optimal scheduling of manufacturing systems.