Journal of Southeast University (English Edition) Vol. 26, No. 1, pp. 26 – 30 Mar. 2010 ISSN 1003—7985

Tabu search for no-wait flowshop scheduling problem to minimize maximum lateness

Wang Chuyang Li Xiaoping Wang Qian

(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: In order to solve the no-wait flowshop scheduling problem to minimize the maximum lateness, three job-block-based neighborhoods are proposed, among which the block exchange neighborhood have a size of $O(n^4)$ while the block swap and the simplified block exchange neighborhoods have a size of $O(n^3)$. With larger sizes than the existing neighborhoods, the proposed neighborhoods can enhance the solution quality of local search algorithms. Speedup properties for the neighborhoods are developed, which can evaluate a neighbor in constant time and explore the neighborhoods in time proportional to their proposed sizes. Unlike the dominance-rule-based speedup method, the proposed speedups are applicable to any machine number. Three neighborhoods and the union of block swap and the simplified block exchange neighborhoods are compared in the tabu search. Computational results on benchmark instances show that three tabu search algorithms with $O(n^3)$ neighborhoods outperform the existing algorithms and the tabu search algorithm with the union has the best performance among all the tested algorithms.

Key words: tabu search; no-wait flowshop; scheduling; maximum lateness; neighborhood

The no-wait flowshop has many industrial applications[1]. Either the type of process or the lack of storage between intermediate machines leads to the no-wait constraint[2]. This paper considers the no-wait flowshop scheduling problem with the maximum lateness criterion, i.e., the $Fm \mid \text{nwt} \mid L_{\max}$ problem[3], which is NP-hard when $m \geqslant 2$[4]. Dileepan[5] presented a dominance rule for the 2-machine case. Wang and Cheng[6] proposed some heuristics for the same problem. For the general m-machine case with sequence independent setup times, Ruiz and Allahverdi[7] presented several heuristics. Allahverdi et al.[8-9] considered the $Fm \mid \text{nwt} \mid \alpha C_{\max} + (1-\alpha) L_{\max}$ problem, developed dominance rules for the 2- and 3-machine cases and proposed several metaheuristics. Obviously, $Fm \mid \text{nwt} \mid L_{\max}$ is a special case of $Fm \mid \text{nwt} \mid \alpha C_{\max} + (1-\alpha) L_{\max}$ with $\alpha = 0$.

Due to the NP-hardness of the $Fm \mid \text{nwt} \mid L_{\max}$ problem, local search is a practical choice. The efficiency of local search is mainly determined by the evaluation time of neighbors. Dileepan et al.[5,7] developed dominance rules for $m = 2$ and $m = 3$ to avoid computing objective values for some neighbors. However, dominance rules are not generally applicable.

This paper develops speedup properties for three new neighborhoods with large sizes. Large size neighborhoods increase the possibility of local search algorithms in finding high quality solutions, while speedup properties reduce the evaluation time of neighbors. The proposed speedup properties have an advantage over the dominance rule method; i.e., they are applicable to any m value. The tabu search (TS)[10-11] is chosen to employ the neighborhoods and their speedups, since the TS tries to find the best neighbor in each iteration and can greatly benefit from speedup properties.

1 Problem Formulation

In the m-machine no-wait flowshop scheduling problem, n jobs need to be processed in the same order on machines $1, 2, \ldots, m$, such that the operation of each job has to be processed without interruptions from the first to the last machines. To meet the no-wait constraint, the completion of a job on a given machine must coincide with the initiation of the operation on the succeeding machine. All jobs are released for processing at time zero. At any given time, a machine can process at most one job and a job can be processed on at most one machine. Each feasible schedule is a permutation of jobs $1, 2, \ldots, n$. The goal is to find a schedule minimizing the maximum lateness.

Let $t_{i,k}$ be the processing time of job i ($1 \leqslant i \leqslant n$) on machine k ($1 \leqslant k \leqslant m$) and d_i be its due date. To facilitate the computation, a dummy job 0 is used with zero processing time and a due date of positive infinity: $t_{0,k} = 0$ for $1 \leqslant k \leqslant m$ and $d_0 = +\infty$. A schedule is expressed as a solution $\boldsymbol{\pi} = \{\pi_{[0]}, \pi_{[1]}, \ldots, \pi_{[n]}, \pi_{[n+1]}\}$, where $\pi_{[i]}$ is the job at position i, $\pi_{[0]} = \pi_{[n+1]} = 0$ and $(\pi_{[1]}, \ldots, \pi_{[n]})$ is a permutation of jobs $1, 2, \ldots, n$. Let $C_{[i]}$ be the completion time and $d_{[i]}$ the due date of job $\pi_{[i]}$, and then the maximum lateness of $\boldsymbol{\pi}$ can be computed as

$$L_{\max}(\boldsymbol{\pi}) = \max_{i=1,2,\ldots,n} \{C_{[i]} - d_{[i]}\} \tag{1}$$

where $C_{[i]} = \sum_{j=1}^{i} D_{[j-1,j]}$ with $D_{[j-1,j]} = \max_{k=1,2,\ldots,m} \left\{ \sum_{h=k}^{m} (t_{[j],h} - t_{[j-1],h}) + t_{[j-1],k} \right\}$ being the difference in the completion time between $\pi_{[j]}$ and $\pi_{[j-1]}$ or the distance from $\pi_{[j-1]}$ to $\pi_{[j]}$[12].

Eq. (1) takes $O(nm)$ time to evaluate a solution. Due to the no-wait constraint, $D_{[j-1,j]}$ is fixed only if $\pi_{[j]}$ is processed immediately after $\pi_{[j-1]}$. The distance $D_{i,j}$ from job i to job j can be pre-computed and stored in a matrix $\boldsymbol{D} = (D_{i,j})$, $0 \leqslant i, j \leqslant n$, by the following expression[12]:

Received 2009-07-16.

Biographies: Wang Chuyang (1975—), male, graduate; Li Xiaoping (corresponding author), male, doctor, professor, xpli@seu.edu.cn.

Foundation items: The National Natural Science Foundation of China (No. 60672092, 60504029, 60873236), the National High Technology Research and Development Program of China (863 Program) (No. 2008AA04Z103).

Citation: Wang Chuyang, Li Xiaoping, Wang Qian. Tabu search for no-wait flowshop scheduling problem to minimize maximum lateness[J]. Journal of Southeast University (English Edition), 2010, 26(1): 26 – 30.

$$D_{i,j}=\begin{cases}\max\limits_{k=1,2,\cdots,m}\{\sum\limits_{h=k}^{m}(t_{j,h}-t_{i,h})+t_{i,k}\} & \\ \qquad 0\leqslant i\leqslant n;\ 1\leqslant j\leqslant n;\ i\neq j & \\ 0 \qquad\qquad 0\leqslant i\leqslant n;\ j=0 & \end{cases} \tag{2}$$

Since the lateness of the dummy job 0 is always negative infinity,

$$L_{\max}(\boldsymbol{\pi})=\max_{i=1,2,\cdots,n}\{C_{[i]}-d_{[i]}\}=\max_{i=0,1,\cdots,n+1}\{C_{[i]}-d_{[i]}\} \tag{3}$$

Eq. (3) can evaluate a solution in $O(n)$, but it is still time consuming for local search. The next section presents speedup properties for three new neighborhoods.

2 Speedup Properties for New Neighborhoods

The neighborhood is critical to the performance of local search algorithms. Three job-block-based neighborhoods are introduced and their speedup properties are developed. A job block is a subsequence in the current solution and the neighbor $\boldsymbol{\pi}'$ of the current solution $\boldsymbol{\pi}$ is generated by the moves of the neighborhoods as follows.

Definition 1 (block exchange, BE) By exchanging the nonadjacent job blocks $[\pi_{[h]},\ \cdots,\ \pi_{[i]}]$ and $[\pi_{[j]},\ \cdots,\ \pi_{[k]}]$, move BE($\boldsymbol{\pi}$, h, i, j, k) ($1\leqslant h\leqslant n-2$, $h\leqslant i\leqslant n-2$, $i+2\leqslant j\leqslant n$ and $j\leqslant k\leqslant n$) generates a neighbor $\boldsymbol{\pi}'=(\pi_{[0]},\cdots,\pi_{[h-1]},\pi_{[j]},\cdots,\pi_{[k]},\pi_{[i+1]},\cdots,\pi_{[j-1]},\pi_{[h]},\cdots,\pi_{[i]},\pi_{[k+1]},\cdots,\pi_{[n+1]})$.

The BE neighborhood has a size of $O(n^4)$. It can be simplified as follows.

Definition 2 (simplified block exchange, SBE) Move SBE ($\boldsymbol{\pi}$, i, j, k) generates a neighbor $\boldsymbol{\pi}'=(\pi_{[0]},\cdots,\pi_{[i-1]},\pi_{[j]},\cdots,\pi_{[k]},\pi_{[i+1]},\cdots,\pi_{[j-1]},\pi_{[i]},\pi_{[k+1]},\cdots,\pi_{[n+1]})$ when $1\leqslant i\leqslant n-2$ and $j\leqslant k\leqslant n$; $\boldsymbol{\pi}'=(\pi_{[0]},\cdots,\pi_{[j-1]},\pi_{[i]},\pi_{[k+1]},\cdots,\pi_{[i-1]},\pi_{[j]},\cdots,\pi_{[k]},\pi_{[i+1]},\cdots,\pi_{[n+1]})$ when $1\leqslant j\leqslant n-3$, $j+1\leqslant k\leqslant n-2$ and $k+2\leqslant i\leqslant n$.

The SBE neighborhood has a size of $O(n^3)$. Block swap moves exchange two adjacent job blocks as follows.

Definition 3 (block swap, BS) By swapping the adjacent blocks $[\pi_{[i]},\ \cdots,\ \pi_{[j]}]$ and $[\pi_{[j+1]},\ \cdots,\ \pi_{[k]}]$, move BS($\boldsymbol{\pi}$, i, j, k) ($1\leqslant i\leqslant n-1$, $i\leqslant j\leqslant n-1$ and $j+1\leqslant k\leqslant n$) generates a neighbor $\boldsymbol{\pi}'=(\pi_{[0]},\cdots,\pi_{[i-1]},\pi_{[j+1]},\cdots,\pi_{[k]},\pi_{[i]},\cdots,\pi_{[j]},\pi_{[k+1]},\cdots,\pi_{[n+1]})$.

The BS neighborhood also has a size of $O(n^3)$. For a job block $B_{[i,j]}=[\pi_{[i]},\ \pi_{[i+1]},\ \cdots,\ \pi_{[j]}]$ in $\boldsymbol{\pi}$, its maximum lateness $L_{[i,j]}=\max\limits_{k=i,\cdots,j}\{C_{[k]}-d_{[k]}\}$. Obviously $L_{\max}(\boldsymbol{\pi})=L_{[i,k]}$ for $i=0,\ 1$ and $k=n,\ n+1$. The maximum lateness for each job block in $\boldsymbol{\pi}$ is pre-computed and stored in an upper triangular matrix $\boldsymbol{L}=(L_{[i,j]})$, $0\leqslant i\leqslant n+1$, $i\leqslant j\leqslant n+1$, where

$$L_{[i,j]}=\begin{cases}C_{[i]}-d_{[i]} & 0\leqslant i\leqslant n+1;\ i=j\\ \max\{L_{[i,j-1]};\ L_{[j,j]}\} & 0\leqslant i<j\leqslant n+1\end{cases}$$

Let $\boldsymbol{C}=(C_{[i]})$ be the completion time vector of $\boldsymbol{\pi}$, then the distance sum in $B_{[i,j]}$ can be computed as $\sum\limits_{k=i+1}^{j}D_{[k-1,k]}=C_{[j]}-C_{[i]}$. Based on $\boldsymbol{C}$ and $\boldsymbol{L}$, the speedup properties for the moves are obtained.

Property 1 For $1\leqslant h\leqslant n-4$, $h+1\leqslant i\leqslant n-3$, $i+2\leqslant j\leqslant n-1$, $j+1\leqslant k\leqslant n$, the objective value of BE ($\boldsymbol{\pi}$, h, i, j, k) can be computed as

$$\begin{aligned}L_{\max}(\mathrm{BE}(\boldsymbol{\pi},h,i,j,k))=\max\{&L_{0,h-1};L_{j,k}+D_{[h-1,j]}-(C_{[j]}-C_{[h-1]});\\ &L_{i+1,j-1}+D_{[h-1,j]}+D_{[k,i+1]}+(C_{[k]}-C_{[j]})-(C_{[i+1]}-C_{[h-1]});\\ &L_{h,i}+D_{[h-1,j]}+D_{[k,i+1]}+D_{[j-1,h]}+(C_{[k]}-C_{[j]})+(C_{[j-1]}-C_{[i+1]})-D_{[h-1,h]};\\ &L_{k+1,n+1}+D_{[h-1,j]}+D_{[k,i+1]}+D_{[j-1,h]}+D_{[i,k+1]}-(D_{[h-1,h]}+D_{[i,i+1]}+D_{[j-1,j]}+D_{[k,k+1]})\}\end{aligned} \tag{4}$$

Proof When a job block is shifted in $\boldsymbol{\pi}$, its maximum lateness and the completion time in all jobs in the job block are changed by the same increment, which equals the change of the first job's completion time in the job block. BE($\boldsymbol{\pi}$, h, i, j, k) splits $\boldsymbol{\pi}$ into five job blocks, $B_{[0,h-1]}$, $B_{[h,i]}$, $B_{[i+1,j-1]}$, $B_{[j,k]}$ and $B_{[k+1,n+1]}$. Then it exchanges $B_{[h,i]}$ and $B_{[j,k]}$, resulting in four arcs $(h-1,\ h)$, $(i,\ i+1)$, $(j-1,\ j)$ and $(k,\ k+1)$ which are replaced by $(h-1,\ j)$, $(k,\ i+1)$, $(j-1,\ h)$ and $(i,\ k+1)$. When $B_{[h,i]}$ and $B_{[j,k]}$ are exchanged, $B_{[0,h-1]}$ is untouched. When $B_{[j,k]}$ is shifted backward, $L_{[j,k]}$ is decreased by $D_{[h-1,j]}-(C_{[j]}-C_{[h-1]})$, due to the fact that the completion time in the jobs in $B_{[j,k]}$ is reduced. Similarly, the changes of $L_{[h,i]}$, $L_{[i+1,j-1]}$ and $L_{[k+1,n+1]}$ can be analyzed and Eq. (4) can be obtained.

Similarly, the objective value of SBE($\boldsymbol{\pi}$, i, j, k) and BS($\boldsymbol{\pi}$, i, j, k) can be computed as follows.

Property 2 The objective value of SBE ($\boldsymbol{\pi}$, i, j, k) can be computed as

$$\begin{aligned}L_{\max}(\mathrm{SBE}(\boldsymbol{\pi},i,j,k))=\max\{&L_{0,i-1};L_{j,k}+D_{[i-1,j]}-(C_{[j]}-C_{[i-1]});\\ &L_{i+1,j-1}+D_{[i-1,j]}+D_{[k,i+1]}+(C_{[k]}-C_{[j]})-(C_{[i+1]}-C_{[i-1]});\\ &L_{i,i}+D_{[i-1,j]}+D_{[k,j+1]}+D_{[j-1,i]}+(C_{[k]}-C_{[j]})+(C_{[j-1]}-C_{[i+1]})-D_{[i-1,i]};\\ &L_{k+1,n+1}+D_{[i-1,j]}+D_{[k,j+1]}+D_{[j-1,i]}+D_{[i,k+1]}-(C_{[i+1]}-C_{[i-1]}+D_{[j-1,j]}+D_{[k,k+1]})\}\end{aligned} \tag{5}$$
$$1\leqslant i\leqslant n-3;\ i+2\leqslant j\leqslant n-1;\ j+1\leqslant k\leqslant n$$

$$\begin{aligned}L_{\max}(\mathrm{SBE}(\boldsymbol{\pi},i,j,k))=\max\{&L_{0,j-1};L_{i,i}+D_{[j-1,i]}-(C_{[i]}-C_{[j-1]});\\ &L_{k+1,i-1}+D_{[j-1,i]}+D_{[i,k+1]}-(C_{[k+1]}-C_{[j-1]});\\ &L_{j,k}+D_{[j-1,i]}+D_{[i,k+1]}+D_{[i-1,j]}+(C_{[i-1]}-C_{[k+1]})-D_{[j-1,j]};\\ &L_{i+1,n+1}+D_{[j-1,i]}+D_{[i,k+1]}+D_{[i-1,j]}+D_{[k,i+1]}-(D_{[j-1,j]}+D_{[k,k+1]}+C_{[i+1]}-C_{[i-1]})\}\end{aligned} \tag{6}$$
$$1\leqslant j\leqslant n-3;\ j+1\leqslant k\leqslant n-2;\ k+2\leqslant i\leqslant n$$

Property 3 For $1\leqslant i\leqslant n-2$, $i+1\leqslant j\leqslant n-1$, $j+1\leqslant k\leqslant n$, the objective value for BS($\boldsymbol{\pi}$, i, j, k) can be computed as

$$\begin{aligned}L_{\max}(\mathrm{BS}(\boldsymbol{\pi},i,j,k))=\max\{&L_{0,i-1};L_{j+1,k}+D_{[i-1,j+1]}-(C_{[j+1]}-C_{[i-1]});\\ &L_{i,j}+D_{[i-1,j+1]}+D_{[k,i]}+(C_{[k]}-C_{[j+1]})-D_{[i-1,i]};\\ &L_{k+1,n+1}+D_{[i-1,j+1]}+D_{[k,i]}+D_{[j,k+1]}-(D_{[i-1,i]}+D_{[j,j+1]}+D_{[k,k+1]})\}\end{aligned} \tag{7}$$

Due to the matrix $\boldsymbol{L}$, a neighbor can be evaluated in constant time and each neighborhood can be searched in time proportional to its size. The large size of the neighborhoods can make full use of $\boldsymbol{L}$ and extend the regions that local search can explore.

3 Tabu Search

The TS is one of the most successful metaheuristics for NP-hard combinatorial optimization problems[11]. It starts from an initial current solution and looks through its neighborhood for a neighbor with the best objective value, then continues from the best neighbor as a new current solution. One of the main ideas of the TS is to use a tabu list as a short-term memory to avoid cycling, to overcome local optima or to guide the search process to the unexplored regions of the solution space.

Trying to find the best neighbor in each iteration, the TS can greatly benefit from the speedup properties and it is therefore chosen for the considered problem. The TS in this paper is adapted from the robust TS[13] and works as follows.

Algorithm 1 Tabu search

Compute the distance matrix $\boldsymbol{D}$;

Generate an initial solution $\boldsymbol{\pi}$ by the NEH heuristic;

Initialize the tabu list, $\boldsymbol{\pi}^* \leftarrow \boldsymbol{\pi}$, count$\leftarrow$0;

While (the termination condition is not met) do

Compute the completion time vector $\boldsymbol{C}$ and the matrix $\boldsymbol{L}$ for $\boldsymbol{\pi}$;

Select the best among moves that are not tabu or aspired;

Update the tabu list according to the selected move;

Transform $\boldsymbol{\pi}$ according to the selected move;

If ($L_{\max}(\boldsymbol{\pi}) < L_{\max}(\boldsymbol{\pi}^*)$) $\boldsymbol{\pi}^* \leftarrow \boldsymbol{\pi}$;

count$\leftarrow$count + 1;

Endwhile

Return $\boldsymbol{\pi}^*$.

The initial solution is generated by the NEH heuristic[14] adapted for the considered problem[7, 9]. Here a fast implementation of the NEH is provided.

Algorithm 2 Fast NEH

Sort n jobs by their due dates in a non-decreasing order and let $\boldsymbol{\pi}'$ be the resultant permutation;

$\boldsymbol{\pi} \leftarrow (0, \boldsymbol{\pi}'_{[1]}, 0)$;

For $k = 2, \ldots, n$ do

Compute the completion time vector $\boldsymbol{C}$ for $\boldsymbol{\pi}$;

Compute $L_{[0,i-1]}$ and $L_{[i,k+1]}$ for $i = 0, 1, \ldots, k+1$;

Let $p = \arg \min_{i=1,2,\ldots,k} \{L_{[0,i-1]};\ C_{[i-1]} + D_{\pi_{[i-1]},k} - d_k;\ D_{\pi_{[i-1]},k} + D_{k,\pi_{[i]}} - D_{\pi_{[i-1]},\pi_{[i]}} + L_{[i,\mathrm{len}(\pi)-1]}\}$

Insert $\pi'_{[k]}$ at position p in $\boldsymbol{\pi}$;

Endfor

Return $\boldsymbol{\pi}$.

The correctness of the implementation can be verified in a way similar to the proof of property 1. The time complexity of the NEH is $O(n^2)$.

The tabu list is based on pairs of adjacent jobs or arcs, and is implemented as a matrix $\boldsymbol{T}$ indexed by jobs. Its element $T_{[i,j]} = c$ means job i cannot be the immediate predecessor of job j until iteration c (included) and initially $\boldsymbol{T}_{[i,j]} = 0$. When a move is applied to the current solution, the arcs deleted from the current solution are put into $\boldsymbol{T}$. A move is tabu if any arc it adds into the current solution is tabu-active. A tabu move is aspired if its maximum lateness is less than the one of $\boldsymbol{\pi}^*$. If there is no available move in the neighborhood, the TS enters the next iteration and does nothing.

A random dynamic tabu tenure method is adopted[13]. If a job pair (i, j) is added to the tabu list at iteration i, then $\boldsymbol{T}_{[i,j]} = i + r$ with $r \in U[0.4n, 0.6n]$[15]. When several job pairs are added to the tabu list at the iteration, their tabu tenures are possibly different.

TS algorithms corresponding to three neighborhoods are denoted as TS_BE, TS_SBE and TS_BS. The union of SBE and BS is also tested and its tabu search algorithm is denoted as TS_SBE + BS. To fairly compare the neighborhoods, a fixed CPU time is used as the termination condition.

4 Computational Experiments

Four proposed TS algorithms are compared with SGALS and IGLS[7, 9]. The 800 instances in Ref. [9] are adopted, which are generated by controlling problem characteristics: $n = \{20, 40, 60, 80, 100\}$ and $m = \{5, 10, 15, 20\}$. Processing times are generated from the uniform range $U[1, 100]$, while due dates are from $U[P(1 - T - R/2), P(1 - T + R/2)]$, where $P = \mathrm{LB}(C_{\max}) = \min_{i=1,2,\ldots,n}(\sum_{k=1}^{m-1} t_{i,k}) + \sum_{i=1}^{n} t_{i,m}$[16] is a lower bound on the makespan; T is the tardiness factor and R is the relative range of due dates. Four different combinations of $T = \{0.0, 0.6\}$ and $R = \{0.2, 0.6\}$ are tested.

Since there is no zero objective function value in our experiments, the following transformed ARPD (TARPD), $\mathrm{TARPD} = \frac{100}{R} \sum \frac{L_{\max}^{\mathrm{H}} - L_{\max}^{\mathrm{Best}}}{|L_{\max}^{\mathrm{Best}}|}$, is used as the response variable, where $L_{\max}^{\mathrm{Best}}$ is the best maximum lateness found during our experiments and $|L_{\max}^{\mathrm{Best}}|$ is the absolute value of $L_{\max}^{\mathrm{Best}}$. All the algorithms are programmed in Java and run on a PC with an Intel Pentium 4 CPU(2.93 GHz) and 1 GB of main memory for the same CPU time set to $30nm$ ms[9]. Each algorithm runs five replications on each instance, i.e., $R = 5$. The computational results are given in Tab. 1.

As shown in Tab. 1, among three neighborhoods when used in the TS, BS is the best and BE is the worst. The union of SBE and BS is superior to any single neighborhood. The poor performance of BE indicates that the neighborhood type is more important than its size. For example, BS is averagely better than SBE by 1.88% in solution quality. This observation is also in accordance with the conclusion that the insertion neighborhood is better than the exchange neighborhood[17-18], since BS is an extension of the insertion neighborhood while SBE is an extension of the exchange neighborhood. Moreover, all the TS algorithms using $O(n^3)$ neighborhoods outperform SGALS and IGLS. Especially, TS_SBE + BS averagely outperforms IGLS by 2.87%.

The problem parameters can also affect the performance of tested algorithms. As shown in Tab. 1, the group of instances with $T = 0$ and $R = 0.6$ seems harder than other groups, on which all the tested algorithms generate larger TARPD. Fig. 1 depicts the impact of the job number on the performance of the tested algorithms, where the TAPRD is averaged on all the instances with the same job numbers. As

shown in Fig. 1, the performance of all the tested algorithms deteriorates as the job number increases. However, TS_SBE + BS and TS_BS are less sensitive to the job number than the others.

Tab. 1 TARPD of tested algorithms when the CPU time is set to 30*nm* ms

T	*R*	*n*	TS_BE	TS_SBE	TS_BS	TS_SBE + BS	SGALS	IGLS
0	0.2	20	0.01	0.01	0	0	0.08	0
0	0.2	40	1.72	0.85	0.36	0.07	1.32	1.36
0	0.2	60	4.95	2.43	1.43	0.6	3.07	4.36
0	0.2	80	6.65	2.56	1.84	0.58	5.2	7.35
0	0.2	100	8.84	3.08	1.76	0.57	5.82	8.17
	Average		4.43	1.79	1.08	0.36	3.1	4.25
0	0.6	20	0.25	0.21	0	0	0.74	0
0	0.6	40	16.5	8.04	1.93	0.75	5.86	4.16
0	0.6	60	18.8	9.06	6.58	2.33	11	12.8
0	0.6	80	33.1	15.8	10.2	4.53	16.8	23
0	0.6	100	55.1	21.1	12.6	8.52	23.6	35.2
	Average		24.7	10.8	6.25	3.23	11.6	15
0.6	0.2	20	0	0	0	0	0.02	0
0.6	0.2	40	0.75	0.45	0.15	0.06	0.6	0.47
0.6	0.2	60	1.24	0.67	0.35	0.31	0.86	1.2
0.6	0.2	80	1.75	0.73	0.39	0.57	1.18	1.77
0.6	0.2	100	2.45	1.55	0.4	0.88	1.55	2.22
	Average		1.24	0.68	0.26	0.37	0.84	1.13
0.6	0.6	20	0.01	0.21	0	0	0.04	0
0.6	0.6	40	1.59	1.52	0.25	0.35	0.7	0.36
0.6	0.6	60	2.7	2.57	0.57	0.94	0.99	1.09
0.6	0.6	80	3.47	2.87	0.54	1.41	1.19	1.51
0.6	0.6	100	11.7	3.71	0.58	2.09	1.45	1.79
	Average		3.9	2.17	0.39	0.96	0.87	0.95
Total average			8.58	3.87	1.99	1.23	4.1	5.34

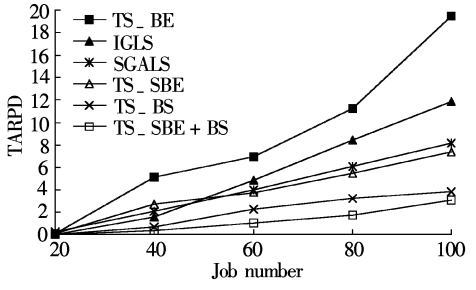

Fig. 1 Impact of job number on the average performance of tested algorithms

5 Conclusion

For the no-wait flowshop scheduling problem with maximum lateness criterion, three job-block-based neighborhoods and their speedup properties are presented. The speedups can evaluate a neighbor in constant time and they are applicable to any number of machines. The neighborhoods are compared with each other in the tabu search and the resultant algorithms are compared with the existing algorithms.

Among the proposed neighborhoods when used in the tabu search, the block swap is the best. The union of the block swap and the simplified block exchange is more effective than any single one of them. All the tabu search algorithms with $O(n^3)$ neighborhoods outperform the existing algorithms and are also less sensitive to the increase in the job number. Especially, the tabu search with the union of the simplified block exchange and the block swap averagely outperforms the best existing algorithms by 2.87%. The instance group with $T=0$ and $R=0.6$ seems more difficult than other groups.

References

[1] Brown S I, McGarvey R, Ventura J A. Total flowtime and makespan for a no-wait *m*-machine flowshop with set-up times separated [J]. *Journal of the Operational Research Society*, 2004, **55**(6): 614 − 621.

[2] Hall N G, Sriskandarayah C. A survey of machine scheduling problems with blocking and no-wait in process [J]. *Operations Research*, 1996, **44**(3): 510 − 525.

[3] Graham R L, Lawler E L, Lenstra J K, et al. Optimization and approximation in deterministic sequencing and scheduling: a survey [J]. *Annals of Discrete Mathematics*, 1979, **5**: 287 − 326.

[4] Röck H. Some new results in flow shop scheduling [J]. *Mathematical Methods of Operations Research*, 1984, **28**(1): 1 − 16.

[5] Dileepan P. A note on minimizing maximum lateness in a two-machine no-wait flowshop [J]. *Computers and Operations Research*, 2004, **31**(12): 2111 − 2115.

[6] Wang X L, Cheng T C E. A heuristic approach for two-machine no-wait flowshop scheduling with due dates and class setups [J]. *Computers and Operations Research*, 2006, **33**(5): 1326 − 1344.

[7] Ruiz R, Allahverdi A. No-wait flowshop with separate set-up times to minimize maximum lateness [J]. *The International Journal of Advanced Manufacturing Technology*, 2007, **35**(5/6): 551 − 565.

[8] Allahverdi A, Aldowaisan T. No-wait flowshops with bicriteria of makespan and maximum lateness [J]. *European*

Journal of Operational Research, 2004, **152**(1): 132 - 147.
[9] Ruiz R, Allahverdi A. New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness [J]. *International Journal of Production Research*, 2009, **47**(20): 5717 - 5738.
[10] Glover F. Tabu search—Part Ⅰ [J]. *ORSA Journal on Computing*, 1989, **1**(3): 190 - 206.
[11] Glover F, Laguna M. *Tabu search*[M]. Dordrecht: Kluwer Academic Publishers, 1997.
[12] Li Xiaoping, Wang Qian, Wu Cheng. Heuristic for no-wait flow shops with makespan minimization [J]. *International Journal of Production Research*, 2008, **46**(9): 2519 - 2530.
[13] Taillard E. Robust taboo search for the quadratic assignment problem [J]. *Parallel Computing*, 1991, **17**(4/5): 443 - 455.
[14] Nawaz M, Enscore Jr E E, Ham I. A heuristic algorithm for the *m*-machine, *n*-job flow shop sequencing problem [J]. *Omega, International Journal of Management Science*, 1983, **11**(1): 91 - 95.
[15] Misevicius A. A tabu search algorithm for the quadratic assignment problem [J]. *Computational Optimization and Applications*, 2005, **30**(1): 95 - 111.
[16] Potts C N, Van Wassenhove L N. A decomposition algorithm for the single machine total tardiness problem [J]. *Operations Research Letters*, 1982, **1**(5): 177 - 181.
[17] Reeves C R. Landscapes, operators and heuristic search [J]. *Annals of Operations Research*, 1999, **86**(1): 473 - 490.
[18] Fink A, Voß S. Solving the continuous flow-shop scheduling problem by metaheuristics [J]. *European Journal of Operational Research*, 2003, **151**(2): 400 - 414.

最小化最大延误无等待流水车间调度问题的禁忌搜索算法

王初阳　李小平　王　茜

(东南大学计算机科学与工程学院, 南京 210096)

摘要: 为求解最小化最大延误无等待流水车间调度问题,提出了 3 个基于任务块交换的邻域,其中块交换邻域的规模为 $O(n^4)$,块对换和简化块交换邻域的规模为 $O(n^3)$. 所提邻域的规模均大于现有邻域,因此可提高局部搜索算法的解质量. 给出了 3 个邻域的加速性质,使一个相邻解的评估时间为常量,邻域的评估时间与其规模成正比. 同基于支配规则的加速方法相比,所提出的加速性质适用于任何机器数. 在禁忌搜索中比较了 3 个邻域,以及块对换和简化块交换邻域的并集. 标准实例集上的计算结果表明:3 个基于 $O(n^3)$ 邻域的禁忌搜索算法均好于现有算法;在所有的测试算法中,采用邻域并集的禁忌搜索算法的性能最好.

关键词: 禁忌搜索;无等待流水车间;调度;最大延误;邻域
中图分类号: TP278