Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2015, Article ID 320140, 8 pages
http://dx.doi.org/10.1155/2015/320140

Research Article

A Local Search Algorithm for the Flow Shop Scheduling Problem with Release Dates

Tao Ren,[1] Meiting Guo,[1] Lin Lin,[1] and Yunhui Miao[2]

[1] *Software College, Northeastern University, Shenyang 110819, China*
[2] *School of Economics & Management, Shenyang University of Chemical Technology, Shenyang 110142, China*

Correspondence should be addressed to Yunhui Miao; miaoyunhui1227@163.com

Received 25 August 2014; Accepted 4 November 2014

Academic Editor: Baoqiang Fan

This paper discusses the flow shop scheduling problem to minimize the makespan with release dates. By resequencing the jobs, a modified heuristic algorithm is obtained for handling large-sized problems. Moreover, based on some properties, a local search scheme is provided to improve the heuristic to gain high-quality solution for moderate-sized problems. A sequence-independent lower bound is presented to evaluate the performance of the algorithms. A series of simulation results demonstrate the effectiveness of the proposed algorithms.

1. Introduction

In a flow shop scheduling model, each job must be processed on a set of machines in identical order. The goal is to determine the job sequence to optimize a certain predetermined objective function. At any given time, each machine can process at most one job, and each job can be handled by at most one machine. Meanwhile, each job cannot be preempted by the other jobs. Flow shop scheduling problems widely exist in industrial production and mechanical manufacturing. For example, in a steel-making process, molten steel is casted into semifinished slabs by a conticaster; after being heated by the heat furnace, the slabs are rolled into products in rolling mill. Obviously, it is a typical flow shop production model. As most of the problems are strongly NP-hard, it is impossible to obtain the global optimum solution in polynomial time. So the study of flow shop scheduling algorithms is very important for reducing running time and boosting productivity.

Since the first scheduling rule was presented by Johnson [1] for the two-machine flow shop problem with objective of makespan (i.e., the maximum completion time) minimization, many works have been conducted on this research area. A comprehensive survey of flow shop makespan problems by 2010 can be found in Potts and Strusevich [2] or Bai and Ren [3]. The up-to-date research works are mentioned as follows. A. Rudek and R. Rudek [4] proved the ordinary NP-hardness for two-machine flow shop makespan problem when job processing times are described by nondecreasing position dependent functions (aging effect) on at least one machine and indicated the strong NP-hardness if job processing times are varying on both machines. Aydilek and Allahverdi [5] presented a polynomial time heuristic algorithm for the two-machine flow shop makespan problem with release dates. For the minimizing makespan in an m-machine flow shop with learning considerations problem, Chung and Tong [6] proposed a dominance theorem and a lower bound to accelerate the branch-and-bound algorithm for seeking the optimal solution. For the criterion of makespan in flow shop model, a high-performing constructive heuristic with an effective tie-breaking strategy was proposed by Ying and Lin [7] to improve the quality of solutions. Similarly, Gupta et al. [8] proposed an alternative heuristic algorithm that is compared with the benchmark, Palmer's, CDS, and NEH algorithms, to solve n-job and m-machine flow shop scheduling problem with minimizing makespan. For the result of the job-related criterion, Bai [9] presented the asymptotic optimality of the shortest processing time-based algorithms for the flow shop problem to optimize total quadratic completion time with release dates. Bai and Zhang [10] extended the results to a general objective, total k-power completion time ($k \geq 3$).

In this paper, the flow shop scheduling problem for the minimization of makespan with release dates is addressed. Contrary to the static setting in which the jobs are simultaneously available, jobs arrive over time according to their release dates, which more closely approaches practical scheduling environments. Lenstra et al. [11] proved that the two-machine flow shop makespan problem with release dates is strongly NP-hard. It implies that the optimal solution of this problem cannot be found in polynomial time; heuristic algorithm may be more effective to obtain an approximate solution for large-sized problems. Therefore, a new modified GS algorithm (MGS) based on the algorithm of Gonzalez and Sahni [12] is presented for slow shop minimizing makespan with release dates. Then an improved scheme is provided to promote performance of the MGS algorithm. Moreover, a sequence-independent lower bound of the problem is presented. Computational experiments reveal the performances of the MGS algorithm, improved scheme, and lower bound in different size problems.

The remainder of this paper is organized as follows. The problem is formulated in Section 2, and the MGS algorithm and improved scheme are provided in Sections 3 and 4, respectively. The new lower bound and computational results are given in Section 5. This paper closes with the conclusion in Section 6.

2. Problem Statement

In a flow shop problem, a set of n jobs has to be processed on m different machines in the same order. Job j, $j = 1, 2, \ldots, n$, is processed on machines i, $i = 1, 2, \ldots, m$, with a nonnegative processing time $p(i, j)$ and a release date r_j, which is the earliest time when the job is permitted to process. Each machine can process at most one job and each job can be handled by at most one machine at any given time. The machine processes the jobs in a first come, first served manner. The permutation schedule is considered in this paper, and the intermediate storage between successive machines is unlimited. The completion time of job j, $j = 1, 2, \ldots, n$, on machine i, $i = 1, 2, \ldots, m$, is denoted by $C(i, j)$. The goal is to determine a job sequence that minimizes the makespan, that is, $\min\{\max_{1\le j\le n} C(m, j)\}$.

3. The Modified GS Algorithm

Gonzalez and Sahni [12] presented the GS algorithm to solve the flow shop makespan problem. Based on its idea, a new heuristic named modified GS (MGS) algorithm is presented to deal with the flow shop makespan problem with release dates. A formal expression of the MGS algorithm is presented as follows.

3.1. The MGS Algorithm

Step 1. Divide the m machines into $m - 1$ groups.

Step 2. For each machine group $g = \{i - 1, i\}$, $i = 2, 3, \ldots, m$, whenever machine $i - 1$ becomes idle or new jobs arrive,

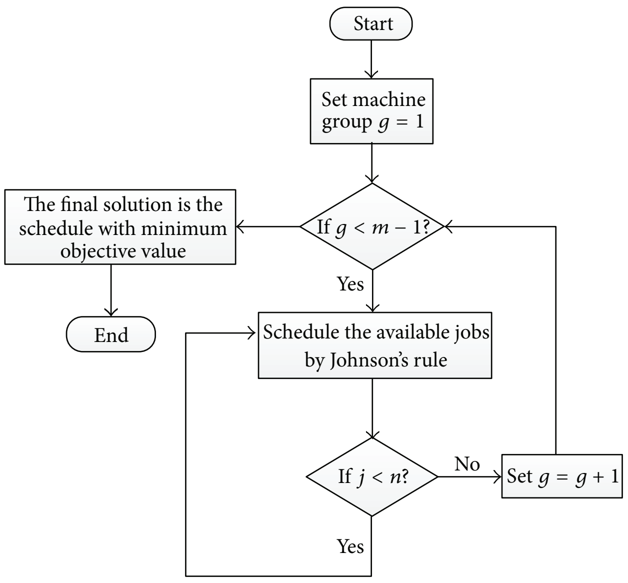
Start
Set machine group $g = 1$
If $g < m - 1$?
The final solution is the schedule with minimum objective value
End
Yes
Schedule the available jobs by Johnson's rule
If $j < n$?
No
Set $g = g + 1$
Yes

FIGURE 1: The flowchart of the MGS algorithm.

process the available jobs by Johnson's rule (i.e., first schedule the jobs with $p_g(1, j) < p_g(2, j)$ in order of nondecreasing $p_g(1, j)$ and then schedule the remaining jobs in order of nonincreasing $p_g(2, j)$, where $p_g(k, j)$ denotes the processing time of job j in group g on machine k, $k = 1, 2$); if no job is available, go to Step 3.

Step 3. Wait until a job arrives and go to Step 2. If all the jobs have been scheduled, go to Step 4.

Step 4. Terminate the program and calculate the objective values Z_g of the schedules. Select the minimum one as the final solution, $C_{\max} = \min_{1\le g\le m-1}\{Z_g\}$.

The flowchart of the algorithm is shown in Figure 1. An example is proposed to show the execution of the MGS algorithm.

Example 1. A flow shop scheduling problem involves three machines, M_1, M_2, and M_3, and four jobs, J_1, J_2, J_3, and J_4 with release dates. The release dates and processing times of the jobs are listed below. The objective function of the problem is $C_{\max}$. Consider

	J_1	J_2	J_3	J_4
r_j	0	9	2	7
M_1	2	1	1	3
M_2	5	5	4	4
M_3	6	7	1	7

(1)

The final sequence of the MGS algorithm is $\{J_1, J_3, J_4, J_2\}$. And the objective value is $C_{\max} = 29$. The scheduling process is shown in Figure 2.

4. The Improved Scheme

To further promote the quality of the solution for medium-scale problems, some properties for two-machine flow shop

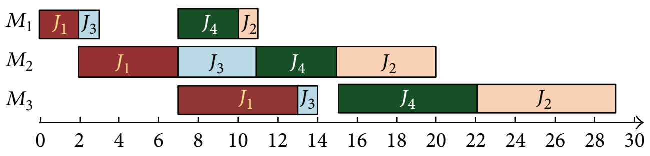

FIGURE 2: The schedule of the MGS algorithm in Example 1.

makespan problem with release dates are presented as follows.

Property 1. For problem $F2|r_j|C_{\max}$, if two adjacent jobs h and k satisfy

(i) $r_h \le r_k$,

(ii) $p(1,k) \le p(2,h)$ and $p(1,h) \le p(2,k)$,

(iii) $p(1,h) - p(1,k) \ge r_k - r_h$,

then the optimal sequence is that job k is scheduled before job h.

Proof. Define the completion times of the two jobs in Sequences $\{h,k\}$ and $\{k,h\}$ being $C_1(2,j)$ and $C_2(2,j)$, respectively, $j = h, k$. Therefore, for Sequence $\{h,k\}$,

$$C_1(2,k) = r_h + p(1,h) + p(2,h) + p(2,k) \tag{2}$$

and, for Sequence $\{k,h\}$,

$$C_2(2,h) = r_k + p(1,k) + p(2,k) + p(2,h). \tag{3}$$

By subtraction, it is obtained that

$$C_1(2,k) - C_2(2,h) = (p(1,h) - p(1,k)) - (r_k - r_h). \tag{4}$$

Note that

$$p(1,h) - p(1,k) \ge (r_k - r_h); \tag{5}$$

therefore,

$$C_1(2,k) - C_2(2,h) \ge 0. \tag{6}$$

That indicates the optimality of Sequence $\{k,h\}$. □

Property 2. For problem $F2|r_j|C_{\max}$, if two adjacent jobs h and k satisfy

(i) $r_h \le r_k$,

(ii) $p(1,k) \le p(2,h)$ and $p(1,h) \ge p(2,k)$,

(iii) $p(2,k) - p(1,k) \ge r_k - r_h$,

then the optimal sequence is that job k is scheduled before job h.

Proof. Define the completion times of the two jobs in Sequences $\{h,k\}$ and $\{k,h\}$ being $C_1(2,j)$ and $C_2(2,j)$, respectively, $j = h, k$. Therefore, for Sequence $\{h,k\}$,

$$C_1(2,k) = r_h + p(1,h) + p(2,h) + p(2,k) \tag{7}$$

and, for Sequence $\{k,h\}$,

$$C_2(2,h) = r_k + p(1,k) + p(1,h) + p(2,h). \tag{8}$$

By subtraction, it is obtained that

$$C_1(2,k) - C_2(2,h) = (p(2,k) - p(1,k)) - (r_k - r_h). \tag{9}$$

Note that

$$p(2,k) - p(1,k) \ge (r_k - r_h); \tag{10}$$

therefore,

$$C_1(2,k) - C_2(2,h) \ge 0. \tag{11}$$

That indicates the optimality of Sequence $\{k,h\}$. □

Property 3. For problem $F2|r_j|C_{\max}$, if two adjacent jobs h and k satisfy

(i) $r_h \le r_k$,

(ii) $p(1,k) \ge p(2,h)$ and $p(1,h) \le p(2,k)$,

(iii) $p(2,h) - p(1,h) \ge r_k - r_h$,

then the optimal sequence is that job k is scheduled before job h.

Proof. Define the completion times of the two jobs in Sequences $\{h,k\}$ and $\{k,h\}$ being $C_1(2,j)$ and $C_2(2,j)$, respectively, $j = h, k$. Therefore, for Sequence $\{h,k\}$,

$$C_1(2,k) = r_h + p(1,h) + p(1,k) + p(2,k) \tag{12}$$

and, for Sequence $\{k,h\}$,

$$C_2(2,h) = r_k + p(1,k) + p(2,k) + p(2,h). \tag{13}$$

By subtraction, it is obtained that

$$C_1(2,k) - C_2(2,h) = (p(2,h) - p(1,h)) - (r_k - r_h). \tag{14}$$

Note that

$$p(2,h) - p(1,h) \ge (r_k - r_h); \tag{15}$$

therefore,

$$C_1(2,k) - C_2(2,h) \ge 0. \tag{16}$$

That indicates the optimality of Sequence $\{k,h\}$. □

Property 4. For problem $F2|r_j|C_{\max}$, if two adjacent jobs h and k satisfy

(i) $r_h \le r_k$,

(ii) $p(1,k) \ge p(2,h)$ and $p(1,h) \ge p(2,k)$,

(iii) $p(2,k) - p(2,h) \geq r_k - r_h$,

then the optimal sequence is that job k is scheduled before job h.

Proof. Define the completion times of the two jobs in Sequences $\{h,k\}$ and $\{k,h\}$ being $C_1(2,j)$ and $C_2(2,j)$, respectively, $j = h,k$. Therefore, for Sequence $\{h,k\}$,

$$C_1(2,k) = r_h + p(1,h) + p(1,k) + p(2,k) \tag{17}$$

and, for Sequence $\{k,h\}$,

$$C_2(2,h) = r_k + p(1,k) + p(1,h) + p(2,h). \tag{18}$$

By subtraction, it is obtained that

$$C_1(2,k) - C_2(2,h) = (p(2,k) - p(2,h)) - (r_k - r_h). \tag{19}$$

Note that

$$p(2,k) - p(2,h) \geq (r_k - r_h); \tag{20}$$

therefore,

$$C_1(2,k) - C_2(2,h) \geq 0. \tag{21}$$

That indicates the optimality of Sequence $\{k,h\}$. □

With these properties, an improved scheme is provided to promote the original solution obtained by the MGS algorithm. In a formal expression of the improved scheme, $\pi_{0,0}(x)$, $x = 1, 2, \ldots, n$, denotes the job found in the xth position of the original sequence $\pi_{0,0}$, $L(u)$ denotes the number of comparisons that job $\pi_{0,0}(x)$ sequentially compares forward with the uth job to a given job in a seed sequence, q denotes the number of comparisons from the current job to the end job, g denotes the number of groups, and $\Pi_{k,g}(x)$ denotes a sequence set that is generated by exchanging the xth job of $\pi_{0,0}$ with each different job in a seed sequence. The format expression can be summarized as follows.

4.1. Improved Scheme

Step 1. Generate the initial sequence $\pi_{0,0}$ with the MGS algorithm and calculate the objective value $Z_{0,0}$.

Step 2. Set $g = 1$, $x = 1$, and $k = 0$.

Step 3. Compare job $\pi_{0,0}(x)$ forward with the next $L(u) = h_0$ (if $h < h_0$, set $L(u) = h$) jobs in sequence $\pi_{k,g}$. If the two jobs a and b satisfy

(1) $r_a \leq r_b$,

(2) one of the following conditions:

(i) $r_b - r_a \leq p_g(1,a) - p_g(1,b)$,

(ii) $r_b - r_a \leq p_g(2,b) - p_g(1,b)$,

(iii) $r_b - r_a \leq p_g(1,a) - p_g(2,a)$,

(iv) $r_b - r_a \leq p_g(2,b) - p_g(2,a)$,

then exchange the jobs, calculate the objective value, and proceed to Step 4.

Step 4. If the objective value $Z_{y,g}$ obtained in Step 3 is smaller than $Z_{k,g}$, set $k := k+1$ and determine sequence $\pi_{k+1,g}$, such that $Z_{k+1,g} = \min\{Z_{y,g} \mid y \in \Pi_{k,g}(x)\}$. If $x \leq n$, set $x := x+1$ and return to Step 3; otherwise, proceed to Step 5.

Step 5. If $g \leq m-1$, set $g := g+1$ and return to Step 3; otherwise, proceed to Step 6.

Step 6. $\pi_{n,m-1}$ is the final schedule. Stop.

5. Computational Results

In this section, a series of computational experiments are designed to reveal the performances of the proposed algorithm and improved scheme in different size problems. Ten different random tests for each combination of the parameters were, respectively, performed, and the averages are reported. All the algorithms were coded in MATLAB R2012a and implemented on a PC with an Intel Core i7-2600 CPU (3.4 GHz × 4) and 4 GB RAM. The processing times of the jobs were randomly generated from a discrete uniform distribution on [1, 10] and a discrete normal distribution with expectation 5.5 and variance 1.7^2.

To evaluate the performance of an algorithm for problem $Fm|r_j|C_{\max}$, Bai et al. [13] presented a lower bound (LB1):

$$\mathrm{LB1} = \max_{\substack{1\leq i\leq m\\ 1\leq x\leq n}} \left\{ r_x + \sum_{j=x}^{n} p(i,j) + \sum_{k=1}^{i-1} p(k,x) + \sum_{k=i+1}^{m} p(k,n) \right\}. \tag{22}$$

However, the value of LB1 sometimes may be larger than the optimal value because LB1 is sequence-dependent. Consider the following example.

Example 2. A two-machine flow shop scheduling problem involves two jobs, J_1 and J_2, with release dates. The release dates and processing times of the jobs are listed below. The objective of the problem is $C_{\max}$:

	M_1	M_2	r_j
J_1	10	1	0
J_2	1	10	1

(23)

For sequence $\{J_1, J_2\}$, calculate the value of LB1:

$$\begin{aligned}\mathrm{LB1} &= r_1 + p(1,1) + p(2,1) + p(2,2)\\ &= 0 + 10 + 1 + 10 = 21.\end{aligned} \tag{24}$$

And the optimal schedule of the problem is $\{J_2, J_1\}$. The associated optimal value is

$$\begin{aligned}C_{\max}(\mathrm{OPT}) &= r_2 + p(1,2) + p(2,2) + p(2,1)\\ &= 1 + 1 + 10 + 1 = 13.\end{aligned} \tag{25}$$

Obviously, $\mathrm{LB1} > C_{\max}(\mathrm{OPT})$.

To guarantee that the lower bound value is strictly less than the optimal value, a new lower bound (LB2) is provided:

$$\text{LB2} = \max_{1\le i\le m}\left\{r_x + \sum_{j=x}^{n} p(i,j) + \min_{x\le j\le n}\sum_{k=1}^{i-1} p(k,j) + \min_{x\le j\le n}\sum_{k=i+1}^{m} p(k,j)\right\}. \quad (26)$$

Obviously, the last two items in LB2 guarantee that the lower bound is independent of the job sequence.

Theorem 3. *Let the processing times $p(i,j)$, $i = 1,2,\ldots,m$, $j = 1,2,\ldots,n$, be independent random variables having the same continuous distribution with bounded density defined on $(0,1]$. Then, for every j, $j = 1,2,\ldots,n$, with probability one,*

$$\lim_{n\to\infty}\frac{LB2}{n} = \lim_{n\to\infty}\frac{C_{\max}(OPT)}{n}. \quad (27)$$

Proof. Without loss of generality,

$$\text{LB1} - \text{LB2} \le \left(\sum_{k=1}^{i-1} p(k,x) - \min_{x\le j\le n}\sum_{k=1}^{i-1} p(k,j)\right) + \left(\sum_{k=i+1}^{m} p(k,n) - \min_{x\le j\le n}\sum_{k=i+1}^{m} p(k,j)\right). \quad (28)$$

Dividing n on both sides of (28) and taking limit,

$$\lim_{n\to\infty}\frac{\text{LB1}}{n} = \lim_{n\to\infty}\frac{\text{LB2}}{n}. \quad (29)$$

Bai et al. [13] proved that

$$\lim_{n\to\infty}\frac{\text{LB1}}{n} = \lim_{n\to\infty}\frac{C_{\max}(\text{OPT})}{n}. \quad (30)$$

Combining (29) and (30) yields the result of Theorem 3. □

Calculate Example 2 with LB2:

$$\text{LB2} = r_2 + p(1,2) + p(2,2) + p(2,1) = C_{\max}(\text{OPT}). \quad (31)$$

5.1. Tests for the MGS Algorithm. Several numerical tests are conducted to reveal the effectiveness of the MGS algorithm. Three, five, and ten machines and 50, 100, 200, 500, and 1000 jobs are tested, respectively. The release dates are drawn from a discrete uniform distribution on $[1, R_t n]$, where n is the number of jobs and R_t is a multiplier with the values of 1, 2, 5, and 8. The DSJF heuristic algorithm presented by Bai and Tang [14] is used as a reference for comparison. First, in Tables 1 and 2, we compare the performance of the MGS algorithm and DSJF heuristic by employing mean relative percentage $(Z^{\text{MGS}} - Z^{\text{DSJF}})/Z^{\text{DSJF}} \times 100\%$, where Z^{MGS} is the objective value of the MGS algorithm and Z^{DSJF} is the objective value of the DSJF heuristic.

In Tables 1 and 2, the data show that the performance of the two algorithms is dependent on multiplier R_t. To further determine the dominance of the two algorithms, in Tables 3 and 4, we execute the following experiments with mean relative percentage $X/20 \times 100\%$, where X denotes the times when the DSJF heuristic dominates the MGS algorithm.

Table 1: Performance comparison of MGS and DSJF (uniform distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	7.42	3.52	1.80
	$R_t = 2$	6.14	2.17	−3.21
	$R_t = 5$	−15.78	−13.29	−20.74
	$R_t = 8$	−24.66	−19.83	−22.66
100 jobs	$R_t = 1$	9.68	8.68	2.34
	$R_t = 2$	4.61	5.07	1.49
	$R_t = 5$	−21.87	−23.73	−15.61
	$R_t = 8$	−20.01	−23.69	−24.37
200 jobs	$R_t = 1$	12.89	10.13	5.71
	$R_t = 2$	9.11	8.93	6.14
	$R_t = 5$	−17.62	−21.26	−17.40
	$R_t = 8$	−26.29	−23.18	−24.86
500 jobs	$R_t = 1$	14.82	11.71	10.39
	$R_t = 2$	10.32	11.55	9.10
	$R_t = 5$	−24.87	−19.07	−25.96
	$R_t = 8$	−18.53	−21.39	−28.83
1000 jobs	$R_t = 1$	15.31	15.31	11.68
	$R_t = 2$	13.03	10.64	10.77
	$R_t = 5$	−20.82	−22.72	−22.22
	$R_t = 8$	−20.07	−20.43	−20.66

Table 2: Performance comparison of MGS and DSJF (normal distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	4.29	1.08	−0.71
	$R_t = 2$	2.48	1.93	−0.55
	$R_t = 5$	−19.68	−19.24	−14.99
	$R_t = 8$	−18.13	−17.06	−18.33
100 jobs	$R_t = 1$	5.79	3.62	2.21
	$R_t = 2$	4.40	1.68	1.66
	$R_t = 5$	−20.74	−24.99	−25.53
	$R_t = 8$	−19.05	−23.04	−20.77
200 jobs	$R_t = 1$	6.73	6.13	3.08
	$R_t = 2$	5.53	4.66	3.61
	$R_t = 5$	−17.73	−16.65	−25.16
	$R_t = 8$	−13.17	−18.55	−19.62
500 jobs	$R_t = 1$	8.06	6.83	6.15
	$R_t = 2$	6.68	6.57	4.83
	$R_t = 5$	−21.99	−24.18	−19.46
	$R_t = 8$	−21.09	−24.58	−22.19
1000 jobs	$R_t = 1$	7.91	8.20	7.23
	$R_t = 2$	7.66	7.25	6.14
	$R_t = 5$	−23.72	−24.17	−24.65
	$R_t = 8$	−17.21	−23.45	−20.75

Table 3: Dominance tests of DSJF (uniform distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	95	75	70
	$R_t = 2$	85	80	25
	$R_t = 5$	15	5	0
	$R_t = 8$	0	0	0
100 jobs	$R_t = 1$	100	95	80
	$R_t = 2$	80	90	85
	$R_t = 5$	0	0	15
	$R_t = 8$	0	0	0
200 jobs	$R_t = 1$	100	95	95
	$R_t = 2$	90	95	95
	$R_t = 5$	25	5	0
	$R_t = 8$	0	0	0
500 jobs	$R_t = 1$	100	100	100
	$R_t = 2$	90	95	95
	$R_t = 5$	5	15	5
	$R_t = 8$	0	0	0
1000 jobs	$R_t = 1$	100	100	95
	$R_t = 2$	95	90	90
	$R_t = 5$	5	10	10
	$R_t = 8$	0	0	0

Table 4: Dominance tests of DSJF (normal distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	90	60	40
	$R_t = 2$	80	50	40
	$R_t = 5$	0	10	10
	$R_t = 8$	0	0	0
100 jobs	$R_t = 1$	100	95	95
	$R_t = 2$	100	70	95
	$R_t = 5$	5	5	5
	$R_t = 8$	0	0	0
200 jobs	$R_t = 1$	100	100	90
	$R_t = 2$	95	95	100
	$R_t = 5$	0	5	0
	$R_t = 8$	0	0	0
500 jobs	$R_t = 1$	100	100	100
	$R_t = 2$	90	100	95
	$R_t = 5$	10	10	0
	$R_t = 8$	0	0	0
1000 jobs	$R_t = 1$	95	100	100
	$R_t = 2$	100	95	95
	$R_t = 5$	0	10	5
	$R_t = 8$	0	0	0

Table 5: Asymptotic performance tests for MGS (uniform distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	5.37	11.74	22.26
	$R_t = 2$	5.55	13.86	20.74
	$R_t = 5$	7.20	13.53	19.77
	$R_t = 8$	1.87	1.76	3.59
100 jobs	$R_t = 1$	3.61	10.59	17.01
	$R_t = 2$	3.93	9.52	15.40
	$R_t = 5$	3.55	7.01	14.35
	$R_t = 8$	0.90	1.47	2.54
200 jobs	$R_t = 1$	3.04	5.08	11.27
	$R_t = 2$	4.29	7.94	12.73
	$R_t = 5$	2.75	6.18	10.98
	$R_t = 8$	0.36	0.82	0.88
500 jobs	$R_t = 1$	1.71	4.33	8.81
	$R_t = 2$	1.26	2.43	10.01
	$R_t = 5$	1.58	3.48	7.39
	$R_t = 8$	0.12	0.36	0.63
1000 jobs	$R_t = 1$	0.53	2.94	5.59
	$R_t = 2$	3.25	4.69	7.72
	$R_t = 5$	1.12	2.62	5.12
	$R_t = 8$	0.059	0.23	0.30

Table 6: Asymptotic performance tests for MGS (normal distribution %).

		$m = 3$	$m = 5$	$m = 10$
50 jobs	$R_t = 1$	2.91	7.29	12.67
	$R_t = 2$	3.17	7.23	15.00
	$R_t = 5$	2.85	5.70	9.97
	$R_t = 8$	1.00	1.43	2.98
100 jobs	$R_t = 1$	2.84	5.22	11.18
	$R_t = 2$	1.91	6.52	10.23
	$R_t = 5$	2.29	4.80	8.20
	$R_t = 8$	0.33	0.55	1.51
200 jobs	$R_t = 1$	1.52	3.76	7.37
	$R_t = 2$	1.18	4.78	7.59
	$R_t = 5$	1.30	3.01	6.11
	$R_t = 8$	0.27	0.27	1.13
500 jobs	$R_t = 1$	0.96	2.23	4.57
	$R_t = 2$	1.30	3.67	5.08
	$R_t = 5$	0.89	2.44	4.72
	$R_t = 8$	0.097	0.18	0.48
1000 jobs	$R_t = 1$	0.69	1.67	3.32
	$R_t = 2$	1.30	1.63	3.26
	$R_t = 5$	0.59	1.65	3.31
	$R_t = 8$	0.032	0.049	0.18

The results of Tables 3 and 4 indicate that the DSJF heuristic completely dominates the MGS algorithm as $R_t = 1$ and the opposite case as $R_t = 8$. Therefore, in the two cases, obtaining the near-optimal solution with the proper one of the algorithms directly is more practical. To demonstrate the asymptotic optimality of the MGS algorithm, we compare its objective value with the associated value of LB2. The mean relative percentage $(Z^{\mathrm{A1}} - Z^{\mathrm{LB2}})/Z^{\mathrm{LB2}} \times 100\%$ is employed, where Z^{A1} is the objective value of the MGS algorithm and Z^{LB2} is the objective value of LB2.

Table 7: Comparing IS with the DSJF %.

$n = 20$				$n = 50$			
	$m = 3$	$m = 5$	$m = 10$		$m = 3$	$m = 5$	$m = 10$
$R_t = 0.1$	3.79	7.08	6.96	$R_t = 0.1$	0.98	3.64	4.68
$R_t = 0.05$	5.95	7.66	4.40	$R_t = 0.04$	2.35	3.97	3.42

Table 8: Comparing IS2 with the MGS %.

$n = 20$				$n = 50$			
	$m = 3$	$m = 5$	$m = 10$		$m = 3$	$m = 5$	$m = 10$
$R_t = 0.1$	5.95	4.54	3.98	$R_t = 0.1$	2.01	2.80	3.61
$R_t = 0.05$	5.47	7.10	4.56	$R_t = 0.04$	2.73	3.63	4.34

The data in Tables 5 and 6 indicate the asymptotic optimality of the MGS algorithm. Contrarily, for the fixed number of jobs, ratios enlarge as the number of machines increases from 3 to 10. The cause may be that the larger the number of machines is, the larger the quantity of idle times is, which enlarges the gap between the value of objective and its lower bound.

5.2. Tests for the Improved Scheme. We compare the effectiveness of the improved scheme (IS) with that of the DSJF heuristic/MGS algorithm. In Tables 7 and 8, mean relative percentage $(Z^{A2} - Z^{IS})/Z^{IS} \times 100\%$ was employed, where Z^{A2} is the objective value of the DSJF heuristic/MGS algorithm and Z^{IS} is the objective value of IS. Three, five, and ten machines with 20 and 50 jobs are tested, respectively. The value of multiplier R_t is 0.1, 0.04 (for 50 jobs) and 0.05 (for 20 jobs). The processing times of the jobs were randomly generated from a discrete uniform distribution on $[1, 10]$.

The results indicate that the improved scheme promotes the performance of the algorithms effectively for moderate-sized problems. As the problem scale and the range of release dates continue to enlarge, the improvement is weakened and the running time lengthens. Therefore, obtaining the near-optimal solution with the MGS algorithm directly for large-scale problems is more practical.

6. Conclusions

This paper presented a modified heuristic algorithm, that is, the MGS algorithm, for the flow shop makespan problem with release dates. And an improved scheme was introduced to improve the quality of the original solution. To evaluate the algorithms numerically, a new lower bound that is sequence independent was provided for the problem. The computational results demonstrate the dominance of the MGS algorithm and the effectiveness of the improved scheme.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (61473073, 61104074, and 61203329), Fundamental Research Funds for the Central Universities (N130417006, N110417005), Science and Technology Plan Project of Shenyang City (F11-264-1-63), and Program for Liaoning Excellent Talents in University (LJQ2014028).

References

[1] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[2] C. N. Potts and V. A. Strusevich, "Fifty years of scheduling: a survey of milestones," *Journal of the Operational Research Society*, vol. 60, no. 1, pp. S41–S68, 2009.

[3] D. Bai and T. Ren, "New approximation algorithms for flow shop total completion time problem," *Engineering Optimization*, vol. 45, no. 9, pp. 1091–1105, 2013.

[4] A. Rudek and R. Rudek, "Makespan minimization flowshop with position dependent job processing times—computational complexity and solution algorithms," *Computers & Operations Research*, vol. 40, no. 8, pp. 2071–2082, 2013.

[5] H. Aydilek and A. Allahverdi, "A polynomial time heuristic for the two-machine flowshop scheduling problem with setup times and random processing times," *Applied Mathematical Modelling*, vol. 37, no. 12-13, pp. 7164–7173, 2013.

[6] Y.-H. Chung and L.-I. Tong, "Makespan minimization for m-machine permutation flowshop scheduling problem with learning considerations," *International Journal of Advanced Manufacturing Technology*, vol. 56, no. 1–4, pp. 355–367, 2011.

[7] K.-C. Ying and S.-W. Lin, "A high-performing constructive heuristic for minimizing makespan in permutation flowshops," *Journal of Industrial and Production Engineering*, vol. 30, no. 6, pp. 355–362, 2013.

[8] D. Gupta, K. K. Nailwal, and S. Sharma, "A heuristic for permutation flowshop scheduling to minimize makespan," in *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, vol. 259 of *Advances in Intelligent Systems and Computing*, pp. 423–432, Springer, New Delhi, India, 2014.

[9] D. Bai, "Asymptotic analysis of online algorithms and improved scheme for the flow shop scheduling problem with release dates," *International Journal of Systems Science*, 2013.

[10] D. Bai and Z. Zhang, "Asymptotic optimality of shortest processing time-based algorithms for flow shop and open shop problems with nonlinear objective functions," *Engineering Optimization*, vol. 46, no. 12, pp. 1709–1728, 2014.

[11] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Operations Research*, vol. 1, pp. 343–362, 1977.

[12] T. Gonzalez and S. Sahni, "Flowshop and jobshop schedules: complexity and approximation," *Operations Research*, vol. 26, no. 1, pp. 36–52, 1978.

[13] D. Bai, M. Huo, and L. Tang, "A new lower bound for flow shop makespan with release dates," in *Proceedings of the IEEE International Conference on Service Operations and Logistics, and*

Informatics (SOLI '08), pp. 276–280, Beijing, China, October 2008.

[14] D. Bai and L. Tang, "New heuristics for flow shop problem to minimize makespan," *Journal of the Operational Research Society*, vol. 61, no. 6, pp. 1032–1040, 2010.