PERGAMON

Computers & Industrial Engineering 36 (1999) 549–564

computers & industrial engineering

www.elsevier.com/locate/dsw

Two-machine flowshop scheduling with bicriteria problem

Fuh-Der Chou[a], Ching-En Lee[b,*]

[a]*Department of Industrial Engineering and Management, Van Nun Institute of Technology, Taoyuan, Taiwan*
[b]*Department of Industrial Engineering and Management, National Chiao Tung University, Hsin Chu, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan*

Abstract

This paper attempts to solve a two-machine flowshop bicriteria scheduling problem with release dates for the jobs, in which the objective function is to minimize a weighed sum of total flow time and makespan. To tackle this scheduling problem, an integer programming model with N^2+3N variables and $5N$ constraints where N is the number of jobs, is formulated. Because of the lengthy computing time and high computing complexity of the integer programming model, a heuristic scheduling algorithm is presented. Experimental results show that the proposed heuristic algorithm can solve this problem rapidly and accurately. The average solution quality of the heuristic algorithm is above 99% and is much better than that of the SPT rule as a benchmark. A 15-job case requires only 0.018 s, on average, to obtain an ultimate or even optimal solution. The heuristic scheduling algorithm is a more practical approach to real world applications than the integer programming model. © 1999 Published by Elsevier Science Ltd. All rights reserved.

Keywords: Two-machine flowshop; Bicriteria scheduling; Flow time; Makespan

1. Introduction

Johnson's rule [8] of minimizing makespan ($C_{\max}$) in the two-machine flowshop scheduling environment within a polynomial time, presented in 1954, has received considerable interest in related fields [3,6,9–12,16]. Minimization of total flow time (F) or mean flow time ($\bar{F}$) in the two-machine flowshop scheduling environment have received increasing attention and have been proved to be a NP-complete problem [5]. Ignall and Schrage [7] first proposed a branch-

* Corresponding author. Tel.: +86-3-5712121-57311; fax: +86-3-5722392.
E-mail address: cclee@cc.nctu.edu.tw (C.E. Lee).

0360-8352/99/$ - see front matter © 1999 Published by Elsevier Science Ltd. All rights reserved.
PII: S0360-8352(99)00149-7

and-bound procedure to obtain an optimal solution for this problem. Van De Velde [16] and Hoogeveen and Van De Velde [6] developed different lower bound determination formulae to facilitate the efficiency of the branch-and-bound procedure. Kohler and Steiglitz [9] and Croce et al. [3] suggested different heuristic algorithms to obtain the approximate solution. Only one criterion ($C_{\max}$, F, or $\bar{F}$) was considered in these studies. However, a decision maker usually needs to consider two or more criteria at the same time. An optimal schedule under a specific criterion may be poor under another criterion. Therefore, a multi-criteria two-machine flowshop scheduling problem is important. Selen and Hott [14] and Wilson [17] developed integer programming models to minimize the weighted sum of total flow time and makespan. Rajendran [12] formulated the problem to minimize the total flow time where makespan was optimal. He proposed not only a branch-and-bound algorithm to obtain the optimal schedule but also two heuristics for improving the solution efficiency. Nagar et al. [10] presented a branch-and-bound algorithm to solve the two-machine flowshop scheduling problem, for which the objective function was to minimize a weighted sum of total flow time and makespan, and proposed a heuristic approach to find the upper bound for the branch-and-bound algorithm. In addition, Nagar et al. [11] developed a heuristic algorithm by integrating a branch-and-bound procedure and a genetic algorithm to find the approximate solution, under the objective function of the weighted sum of mean flow time and makespan.

The above mentioned studies assume that the job available times and machine ready times are all at time 0. In literature, dynamic scheduling has also been widely studied [1,4]. Church and Uzsoy [2] focused on the dynamic scheduling problem to minimize the maximal lateness ($L_{\max}$) of a single machine or parallel identical machines and proposed an event-driven periodic rescheduling policy to decompose the dynamic scheduling problem into a series of static ones. Sun and Lin [15] presented a dynamic scheduling framework which is carried out by solving a series of static backward scheduling problems. A rolling time window approach was adopted to decompose the job shop dynamic scheduling problem, whose objective function was to minimize the total weighted flow time and to satisfy all due-date constraints, into a series of static backward scheduling problems. Roy and Zhang [13] developed a fuzzy logic based dynamic scheduling algorithm for the job shop scheduling problem.

To increase system performance of a two-machine flowshop, the lowering of both throughput time and work in process (WIP) as much as possible is important. The scheduling criterion of makespan minimization can effectively reduce the throughput time while the scheduling criterion of total flow time minimization can effectively reduce work in process. This paper attempts to minimize the weighted sum of these two scheduling criteria in a static flowshop environment with release dates for the jobs. Assumptions made in this paper are:

1. setup time is known and is included in the processing time;
2. machine preemption is disallowed, each operation, once started, must be completed before another operation may be started on the same machine;
3. machines are stable and remain available throughout the scheduling period;
4. no job may be processed on more than one machine simultaneously;
5. a machine may only process one job at a time.

An integer programming model with N^2+3N variable and $5N$ constraints for the static scheduling problem is formulated in this work. Although the integer programming model can

generate the optimal schedule, it is slow and does not always obtain a good schedule within an allowable time (a very large pivoting number, 900,000) for a large problem. Therefore, a heuristic scheduling algorithm is designed so that the two-machine flowshop scheduling with a bicriteria problem can be solved quickly and accurately.

In the following section, notations are defined. The following two sections describe the integer programming model and the heuristic scheduling algorithm. The next section exemplifies the heuristic works. Randomly generated experimental cases are tested and the results are analyzed. The effectiveness of the proposed heuristic algorithm is then demonstrated and presented in the following section. The last section describes Discussion and Conclusions.

2. Notation and definitions

2.1. Known variables

N number of jobs
α weight for the total flow time
β weight for the makespan
P_i processing time of job i on the first machine, $i = 1, 2, \ldots, N$
Q_i processing time of job i on the second machine, $i = 1, 2, \ldots, N$
T_i job i's start available time in the flowshop, $i = 1, 2, \ldots, N$
R_1 the ready available time of the first machine
R_2 the ready available time of the second machine.

2.2. Decision variable

Z_{ik} if job i is scheduled at the kth rank to be processed, $i = 1, 2, \ldots, N$; $k = 1, 2, \ldots, N$, $Z_{ik} = 1$, job i is scheduled at the kth rank to be processed
X_{ik} the ideal time on the second machine between the starting of the kth ranked job and the completion of the $k-1$th ranked job, $k = 1, 2, \ldots, N$
Y_k for the kth ranked job, the time between its completion at the first machine and its begin processing at the second machine, $k = 1, 2, \ldots, N$
S_k the starting time for the kth ranked job at the first machine, $k = 1, 2, \ldots, N$.

2.3. Auxiliary variables

A_k the kth ranked jobs processing time at the first machine

$$A_k = \sum_{i=1}^{N} Z_{ik} P_i; \quad k = 1, 2, \ldots, N$$

B_k the kth ranked jobs processing time at the second machine

$$B_k = \sum_{i=1}^{N} Z_{ik} Q_i; \quad k = 1, 2, \ldots, N$$

C_k the completion time for the kth ranked job at the second machine

$$C_k = R_2 + \sum_{j=1}^{k} X_j + \sum_{j=1}^{k} B_j; \quad k = 1, 2, \ldots, N$$

or

$$C_1 = R_2 + X_1 + B_1$$

and

$$C_k = C_{k-1} + X_k + B_k; \quad k = 2, 3, \ldots, N$$

D_k the kth ranked jobs start available time in the shop

$$D_k = \sum_{i=1}^{N} Z_{ik} T_i; \quad k = 1, 2, \ldots, N$$

$C_{\max}$ the completion time of all jobs in the shop; i.e. makespan

$$C_{\max} = C_N = R_2 + \sum_{j=1}^{N} X_j + \sum_{j=1}^{N} B_j$$

F_k the kth ranked jobs flow time in the shop

$$F_k = C_k - D_k; \quad k = 1, 2, \ldots, N$$

F the summation of all jobs flow times in the shop

$$F = \sum_{k=1}^{N} (C_k - D_k)$$

AS the set of scheduled jobs
NS the set of unscheduled jobs
θ the number of the scheduled jobs in AS
OV the objective function which is $(\alpha F + \beta C_{\max})$
OV^* the best solution obtained until now
NR_1 the first machines ready time to process the next job
NR_2 the second machines ready time to process the next job

RT_{1a} the completion time at the first machine if the next job to be processed on the first machine is job a

$$RT_{1a} = \max(NR_1, T_a) + P_a; \quad a \in NS$$

RT_{2a} the completion time at the second machine if the next job to be processed on the second machine is job a

$$RT_{2a} = \max(NR_2, RT_{1a}) + Q_a; \quad a \in NS$$

Index_a the decision index if job a is the next schedule job

$$\text{Index}_a = [\alpha(N-\theta)+\beta][X_{\theta+1} + Q_a] + [\alpha(N-1-\theta)+\beta]\min\{[X_{\theta+2} + Q_b]\}$$

$$a \in NS, \quad b \in \{NS - \{a\}\}$$

a^* the job with the smallest Index_a

3. Integer programming model

As discussed in Section 1, this study solves the two-machine flowshop bicriteria scheduling problem with release dates for the jobs. At first, an integer programming model is formulated as follows. The objective function is to minimize the weighted sum of the total flow time and makespan.

Objective function:

$$\min \ Z = \alpha\sum_{k=1}^{N}(C_k - D_k) + \beta C_N \tag{1}$$

Constraints:

$$\sum_{i=1}^{N} Z_{ik} = 1; \quad k = 1, 2, \ldots, N \tag{2}$$

$$\sum_{k=1}^{N} Z_{ik} = 1; \quad i = 1, 2, \ldots, N \tag{3}$$

$$S_k \geq D_k; \quad k = 1, 2, \ldots, N \tag{4}$$

$$S_1 \geq R_1; \quad S_k \geq (S_{k-1} + A_{k-1}); \quad k = 2, 3, \ldots, N \tag{5}$$

$$X_1 = S_1 + A_1 + Y_1 - R_2; \quad X_k = S_k + A_k + Y_k - C_{k-1}; \quad k = 2, 3, \ldots, N \tag{6}$$

Constraint (2) specifies that only one job be scheduled at the kth job priority. Constraint (3) defines that each job be scheduled only once. Constraint (4) stipulates that the begin processing time of the kth ranked job be greater than or equal to its arrival time. Constraint (5) represents that the begin processing time of the kth ranked job be greater than or equal to the previous jobs completion time at the first machine. Constraints (6) can be explained using Fig. 1 in which the idle time on the second machine to process the kth ranked job (X_k) equals the starting time for the kth ranked job on the first machine (S_k) plus its processing time on the first machine (A_k), plus the time between its completion on the first machine and begin processing time on the second machine (Y_k), minus the completion time for the $(k-1)$th ranked job at the second machine (C_{k-1}). All variables should be greater than or equal to zero and Z_{ik} is a binary integer.

4. Heuristic scheduling algorithm

Although the integer programming model provides the optimal solution, variables and constraints increase drastically when the number of jobs increases. Therefore, the optimal solution is not always attainable within the allowable time. A heuristic scheduling algorithm is therefore considered and employed. Fig. 2 shows the flow chart of the heuristic algorithm. The following is a step by step explanation of the algorithm.

1. Initialize. Let $AS=\{\emptyset\}$; $NS=\{1, 2, 3, \ldots, N\}$; $\theta=0$; $NR_1=R_1$; $NR_2=R_2$; $OV^*=$ a very large number $=10^{31}$.
2. Calculate Index_a for each job in NS. Take the job with the smallest Index_a as job a^* and go to (3). If there is more than one job having the smallest Index_a, then arbitrarily select one as a job a^* and push the rest of the jobs as well as the corresponding information (current values of AS, NS, θ, NR_1 and NR_2) into a stack. These jobs and the corresponding information in the stack are control points for further backtracking processes.
3. Append job a^* to AS, delete job a^* from NS, and let $\theta=\theta+1$. Recalculate $NR_1=\max(NR_1, T_{a^*})+P_{a^*}$ and $NR_2=\max(NR_2, NR_1)+Q_{a^*}$.
4. If $\theta < (N-1)$, then repeat (2) to (4); otherwise go to (5).

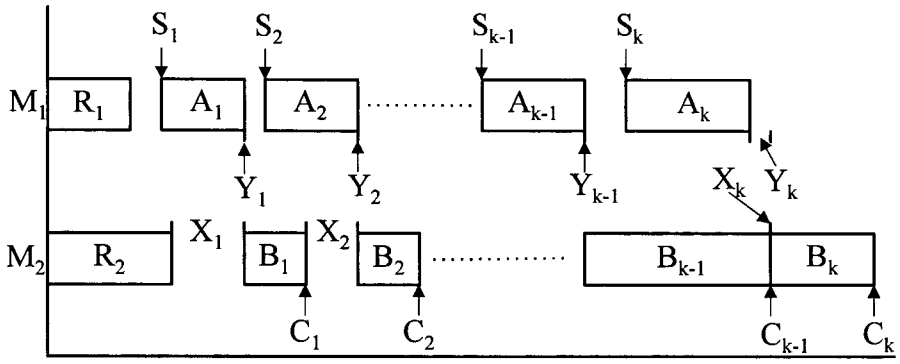

Fig. 1. The Gantt chart to illustrate constraints (6).

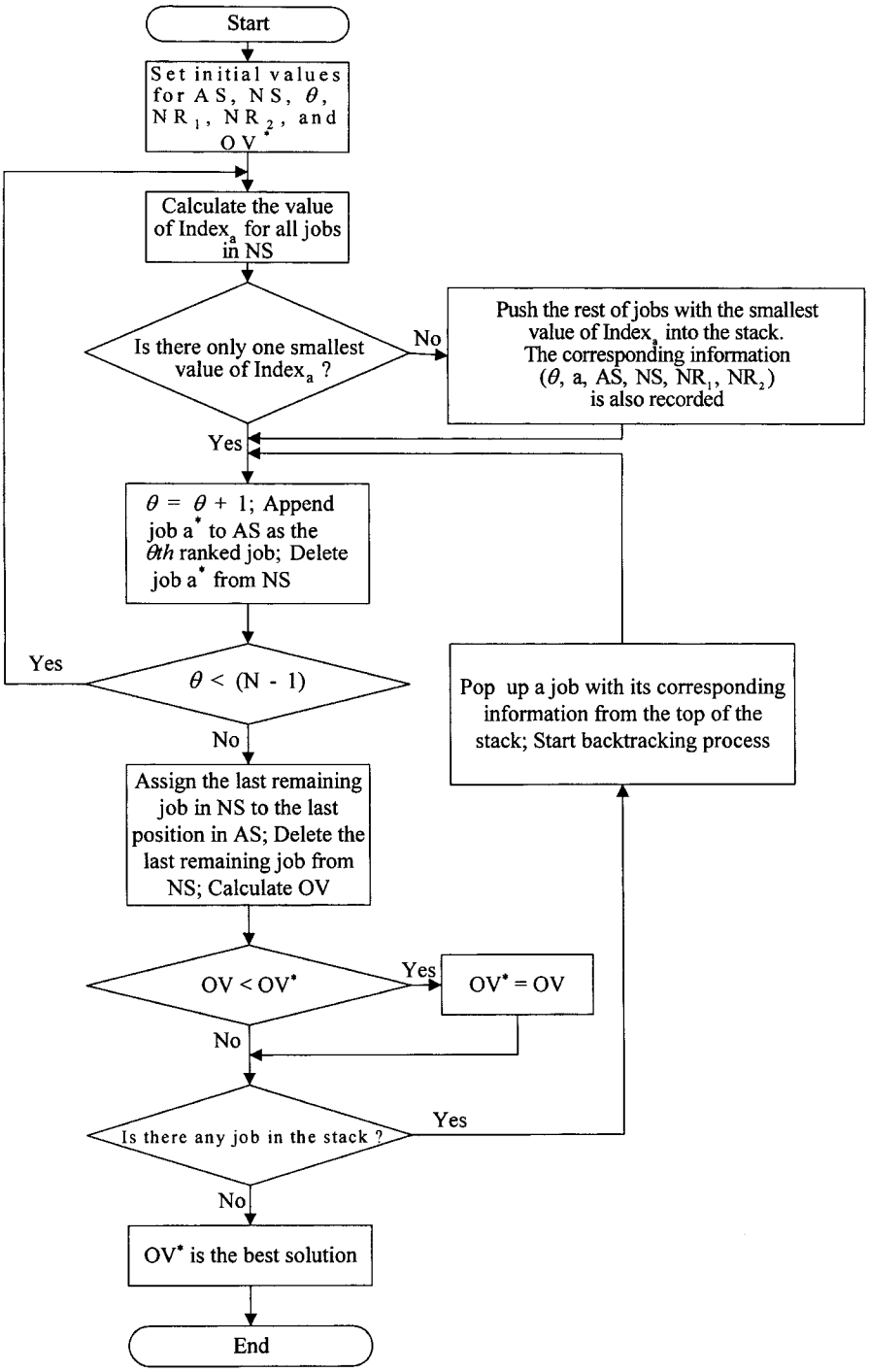

Fig. 2. The flowchart of the heuristic scheduling algorithm.

5. Assign the only remaining unscheduled job in NS to the last position of AS and delete it from NS. Recalculate NR_1 and NR_2 and calculate OV. If $OV < OV^*$, then $OV^* = OV$.
6. Check if there is any job in the stack. When there are jobs in the stack, they must be the jobs with the same smallest Index_a pushing into the stack at (2). These jobs need to be backtracked to see whether a better schedule, OV^*, can be obtained. Therefore, if there are jobs in the stack, then pop up the top one in the stack as job a^* to start the backtracking process and continue from (3).
7. The final result, OV^*, of the heuristic scheduling algorithm is obtained and stop.

The 'look-ahead' priority number, Index_a, plays the most important role in the above mentioned procedure. It determines the job sequence to be scheduled. Incorrect calculation of the value of Index_a yields a poor schedule. If the value of Index_a is not sensitive sufficiently, excessive, unnecessary backtrackings occur. Thus, an appropriate formula to calculate Index_a is essential to the heuristic scheduling algorithm. The objective function is rearranged as follows

$$\alpha\sum_{k=1}^{N}(C_k - D_k) + \beta C_N = \left\{\sum_{k=1}^{N}[\alpha(N+1-k)+\beta][X_k + B_k]\right\} + (\alpha N + \beta)R_2 - \alpha\sum_{k=1}^{N} D_k$$

$$= \left\{\sum_{k=1}^{N}[\alpha(N+1-k)+\beta][X_k + B_k]\right\} + \{\text{constant}\}$$

For minimization, the constant part is neglected. Therefore, the objective function can be represented as $\Sigma_{k=1}^{N}[\alpha(N+1-k)+\beta][X_k + B_k]$. To make the value of Index_a accurate and approach the global minima, Index_a considers the minimization of both the assignment of the current stage (if job a is scheduled) and that of one stage further.

$$\text{Index}_a = [\alpha(N-\theta)+\beta][X_{\theta+1} + Q_a] + [\alpha(N-1-\theta)+\beta]\min\{[X_{\theta+2} + Q_b]\}$$

$$X_{\theta+1} = \max[\max(NR_1, T_a) + P_a - NR_2, 0]$$

$$X_{\theta+2} = \max[\max(RT_{1a}, T_b) + P_b - RT_{2a}, 0]; \quad a \in NS, \quad b \in \{NS - \{a\}\}$$

The heuristic scheduling algorithm applies the greedy approach. In the kth stage

Table 1
Pertinent information for the example case

Job no.	P_i	Q_i	T_i
1 (J_1)	11	22	3
2 (J_2)	9	17	6
3 (J_3)	3	2	4
4 (J_4)	13	9	0

$R_1 = 4$; $R_2 = 7$

$(N-k+1)(N-k)$ calculations and comparisons are required. Since the Index_a is very effective in this study, the number of backtrackings is few and can be negligible. Therefore, the heuristic requires $\{N(N-1)+(N-1)(N-2)+\cdots+3\times2+2\times1=\frac{1}{3}N(N-1)(N+1)\}$ calculations and comparisons in total without counting the backtrackings.

5. Example illustration

A four-job, two-machine flowshop example is used to illustrate the decision process of the proposed heuristic algorithm. The objective function is assumed to be $0.5F + 0.5C_{\max}$. Table 1 shows the pertinent information.

First of all, system status must be initialized: $AS=\{\emptyset\}$, $NS=\{1, 2, 3, 4\}$, $\theta=0$, $NR_1=R_1=4$, $NR_2=R_2=7$, and $OV^*=10^{31}$. The Index_a for each job in NS is calculated. Since job 3 has the smallest Index_a which is 45.0, a^* is 3.

Stage 1

$$AS=\{\oslash\};\quad NS=\{1, 2, 3, 4\};\quad \theta=0;\quad NR_1=R_1=4;\quad NR_2=R_2=7;\quad OV^*=10^{31}$$

$$\begin{aligned}\text{Index}_1 &= (2.5)\{\max[\max(4, 3)+11-7, 0]+22\}+(2.0)\{\min[(0+17), (0+2), (0+9)]\}\\ &= 2.5\times30+2.0\times2=71.0\end{aligned}$$

$$\begin{aligned}\text{Index}_2 &= (2.5)\{\max[\max(4, 6)+9-7, 0]+17\}+(2.0)\{\min[(0+22), (0+2), (0+9)]\}\\ &= 2.5\times25+2.0\times2=66.5\end{aligned}$$

$$\begin{aligned}\text{Index}_3 &= (2.5)\{\max[\max(4, 4)+3-7, 0]+2\}+(2.0)\{\min[(9+22), (7+17), (11+9)]\}\\ &= 2.5\times2+2.0\times20=45.0\end{aligned}$$

$$\begin{aligned}\text{Index}_4 &= (2.5)\{\max[\max(4, 0)+13-7, 0]+9\}+(2.0)\{\min[(2+22), (0+17), (0+2)]\}\\ &= 2.5\times19+2.0\times2=51.5\end{aligned}$$

After completion of Stage 1, AS, NS, θ, NR_1, and NR_2 need to be updated. The Index_a for each unscheduled job in NS is then recalculated. a^* is 2 because job 2 has the smallest value of Index_a.

Stage 2

$$AS=\{3\};\quad NS=\{1, 2, 4\};\quad \theta=1;\quad NR_1=7;\quad NR_2=9;\quad OV^*=10^{31}$$

$$\text{Index}_1 = (2.0)\{\max[\max(7, 3) + 11 - 9, 0] + 22\} + (1.5)\{\min[(0 + 17), (0 + 9)]\}$$
$$= 2.0 \times 31 + 1.5 \times 9 = 75.5$$

$$\text{Index}_2 = (2.0)\{\max[\max(7, 6) + 9 - 9, 0] + 17\} + (1.5)\{\min[(0 + 22), (0 + 9)]\}$$
$$= 2.0 \times 24 + 1.5 \times 9 = 61.5$$

$$\text{Index}_4 = (2.0)\{\max[\max(7, 0) + 13 - 9, 0] + 9\} + (1.5)\{\min[(2 + 22), (0 + 17)]\}$$
$$= 2.0 \times 20 + 1.5 \times 17 = 65.5$$

The process continues and it is shown in the following two stages, Stage 3 and 4.
Stage 3

$$AS = \{3, 2\}; \quad NS = \{1, 4\}; \quad \theta = 2; \quad NR_1 = 16; \quad NR_2 = 33; \quad OV^* = 10^{31}$$

$$\text{Index}_1 = (1.5)\{\max[\max(16, 3) + 11 - 33, 0] + 22\} + (1.0)\{(0 + 9)\} = 1.5 \times 22 + 1.0 \times 9$$
$$= 42.0$$

$$\text{Index}_4 = (1.5)\{\max[\max(16, 0) + 13 - 33, 0] + 9\} + (1.0)\{(0 + 22)\} = 1.5 \times 9 + 1.0 \times 22$$
$$= 35.5$$

Stage 4

$$AS = \{3, 2, 4\}; \quad NS = \{1\}; \quad \theta = 3; \quad NR_1 = 29; \quad NR_2 = 42; \quad OV^* = 10^{31}$$

$$C_{\max} = 64; \quad F = 135; \quad OV = 99.5; \quad OV^* = OV = 99.5$$

In Stage 4, the values of $C_{\max}$, F, and OV are calculated. Since no backtracking is necessary, the heuristic algorithm stops here. The final schedule is J_3–J_2–J_4–J_1, while $C_{\max} = 64$, $F = 135$, and $OV^* = 99.5$. The solution is the same as the optimal solution obtained from the integer programming model.

6. Experimental results

Experimental results are divided into three parts. The first part gives the optimal solution or the lower bound value, if the allowable pivoting limit is reached, as the basis to prove the effectiveness of the heuristic scheduling algorithm. Another part provides the solution obtained using the heuristic scheduling algorithm. The third part shows the result of the SPT rule as a benchmark. All experimental tests have been run on a personal computer with Pentium

Table 2
Execution time comparison and solution quality of the heuristic scheduling algorithm

Number of jobs	Number of optimal solutions	Average execution CPU time (s)		Experiments when optimal solutions are obtained		Experiments when optimal solution cannot be obtained with pivot number = 900,000		The overall average solution quality	
		Integer programming model	Heuristic scheduling algorithm	Average solution quality	The worst case solution quality	Average solution quality	The worst case solution quality	Heuristic scheduling algorithm	SPT rule
$\alpha = 1.0$, $\beta = 0.0$									
3	30	0.059	0.00033	100.00000	100.00000	–	–	100.00000	97.70168
6	30	0.621	0.00133	99.41877	94.75138	–	–	99.41877	91.10435
9	30	6.426	0.00200	99.25884	93.65854	–	–	99.25884	90.97655
12	30	240.714	0.00533	98.34627	93.96985	–	–	98.34627	88.46551
15	17	1522.998	0.00933	99.08953	97.18085	96.62405	94.65704	98.02116	85.17721
$\alpha = 0.7$, $\beta = 0.3$									
3	30	0.051	0.00067	100.00000	100.00000	–	–	100.00000	98.26873
6	30	6.761	0.00633	99.06406	94.30866	–	–	99.06406	94.19573
9	30	6.761	0.00633	99.19763	94.15297	–	–	99.19763	90.80471
12	30	178.531	0.01167	98.39312	94.16156	–	–	98.39312	89.24156
15	17	1543.937	0.01600	98.96973	96.75560	96.55758	94.53589	97.92447	86.02620
$\alpha = 0.5$, $\beta = 0.5$									
3	30	0.057	0.00100	100.00000	100.00000	–	–	100.00000	98.18479
6	30	0.766	0.00200	99.33925	95.00000	–	–	99.33925	94.14605
9	30	7.525	0.00400	99.17590	94.68504	–	–	99.17590	91.26523
12	29	233.664	0.00700	98.54180	95.86288	90.84507	90.84507	98.28525	89.90443
15	16	1636.844	0.01467	99.33404	97.66573	96.35359	94.22955	97.94317	86.91586
$\alpha = 0.3$, $\beta = 0.7$									
3	30	0.053	0.00167	100.00000	100.00000	–	–	100.00000	97.46342
6	30	0.956	0.00233	99.68452	98.14077	–	–	99.68452	94.52946
9	30	8.699	0.00567	99.06549	90.59917	–	–	99.06549	92.88967
12	28	304.222	0.01167	98.68871	95.95469	95.49057	94.84694	98.47550	91.01287
15	15	1502.332	0.01967	98.98651	97.39848	96.80478	94.45757	97.89565	85.49032

(continued on next page)

Table 2 (*continued*)

Number of jobs	Number of optimal solutions	Average execution CPU time (s)		Experiments when optimal solutions are obtained		Experiments when optimal solution cannot be obtained with pivot number = 900,000		The overall average solution quality	
		Integer programming model	Heuristic scheduling algorithm	Average solution quality	The worst case solution quality	Average solution quality	The worst case solution quality	Heuristic scheduling algorithm	SPT rule
$\alpha = 1.0$, $\beta = 0.0$									
$\alpha = 0.0$, $\beta = 1.0$									
3	30	0.068	0.00167	100.00000	100.00000	–	–	100.00000	96.26376
6	30	0.470	0.00333	99.44362	94.68085	–	–	99.44362	94.95523
9	30	14.050	0.00967	99.28043	94.24460	–	–	99.28043	91.58406
12	28	169.641	0.01567	99.71183	97.15909	97.83128	96.24060	99.58646	89.73439
15	16	743.747	0.02933	99.82535	99.00498	99.52324	98.61111	99.78507	86.57606

133 MHz CPU and 24 Mb RAM. The machine ready available time and the job start available time are both uniformly distributed between 0 and 10 [$U(0, 10)$]. The job processing time is uniformly distributed between 1 and 25 [$U(1, 25)$]. The number of jobs ranges from 2 to 15. Thirty sets of testing examples are experimented for each job number. Five sets of (α, β) values (1.0, 00), (0.7, 0.3) (0.5, 0.5), (0.3, 0.7), and (0.0, 1.0) for the objective function of each testing example are calculated. The following formula is applied to determine the solution quality of the heuristic scheduling algorithm.

$$\text{solution quality} = \left(1 - \frac{\text{heuristic } OV - \text{optimal } OV \text{ or lower bound}}{\text{optimal } OV \text{ or lower bound}}\right) \times 100\%$$

The integer programming model does not provide optimal solutions for some testing examples within the allowable pivoting limit, so the solutions are substituted with lower bound values. Therefore, the actual values of solution quality of the proposed heuristic algorithm are slightly higher than those shown in Table 2 with the problem size of $N = 3, 6, 9, 12$ and 15.

The integer programming model solves the problem using LINDO V5.01 software package. Since the allowable pivoting limit is set to 900,000, the computing time falls within about 2400 CPU (s). The lower bound value instead of the optimal solution will be used when the pivoting limit is reached. The heuristic scheduling algorithm is programmed in a C^{++} language. Initial data is stored in a file. The *IO* processing times produce some discrepancies. To estimate the execution time accurately, each testing example is processed continuously 20 times and the average CPU time is calculated.

According to Table 2, the average computing time of the integer programming model will drastically increase as the number of jobs increases. When the number of jobs is greater than 12, some of the testing examples cannot get the optimal solutions within the allowable pivoting

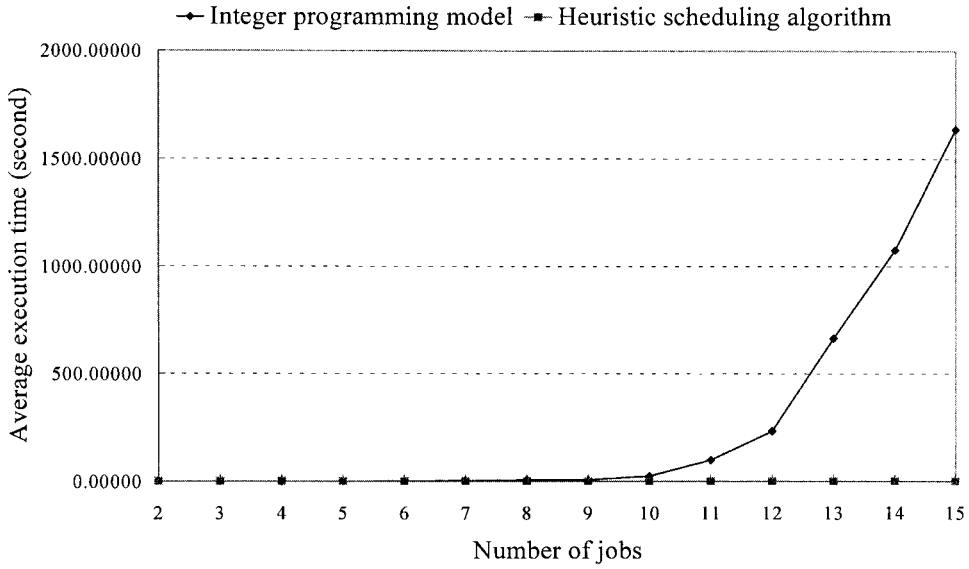

Fig. 3. Average execution times at $\alpha = 0.5$ and $\beta = 0.5$.

limit. Therefore, the average execution time of the integer programming model is somewhat underestimated and the actual execution time is higher than the numbers shown in Table 2. The average execution time of the heuristic scheduling algorithm also increases when the number of jobs increase. Fig. 3 shows the curves of execution times of the proposed heuristic and the IP-solver, respectively, at $\alpha = 0.5$ and $\beta = 0.5$. When the number of jobs increases to 15, the integer programming model requires at least 1389.977 s to produce a solution. However, for 59 out of 150 sets of the experimental examples, lower bound solutions are used within the allowable pivoting limit. The average execution time of the heuristic scheduling algorithm, however, is only 0.018 s when $N = 15$. The corresponding solution quality of the heuristic algorithm is fairly good which can be seen from Table 2. It can also be found that the solution quality of the heuristic algorithm for any number of jobs (when $N \leq 15$) is much better than that of the SPT rule in Table 2. However, the heuristic algorithm takes a little bit longer (no more than 0.03 s) to yield a solution than the SPT rule does. The complexity of the heuristic scheduling algorithm is fairly low, but it can quickly obtain an ultimate or optimal schedule.

Table 2 summaries the solution quality of the heuristic scheduling algorithm. When optimal solution is obtained using the integer programming model, the average solution quality for each job number is above 98.1565%. When the lower bound value is used because the pivoting limit is reached, the average solution quality for each job number is slightly lower, but above 90.84507% where it is underestimated in this situation. In summary, the average solution quality decreases as the number of jobs increases, because some testing examples cannot yield optimal solutions under the allowable pivoting limit. Although the underestimation of the solution quality dilutes the overall solution quality and lowers the average solution quality for the heuristic algorithm, the overall average solution quality for each job number is above 97.73083% (Table 2). The overall average solution quality for 2100 (5*30*(15−2+1)) sets of testing examples is above 99%. Therefore, we can assure that the result of the heuristic scheduling algorithm is fairly effective and pragmatic in this bicriteria scheduling environment.

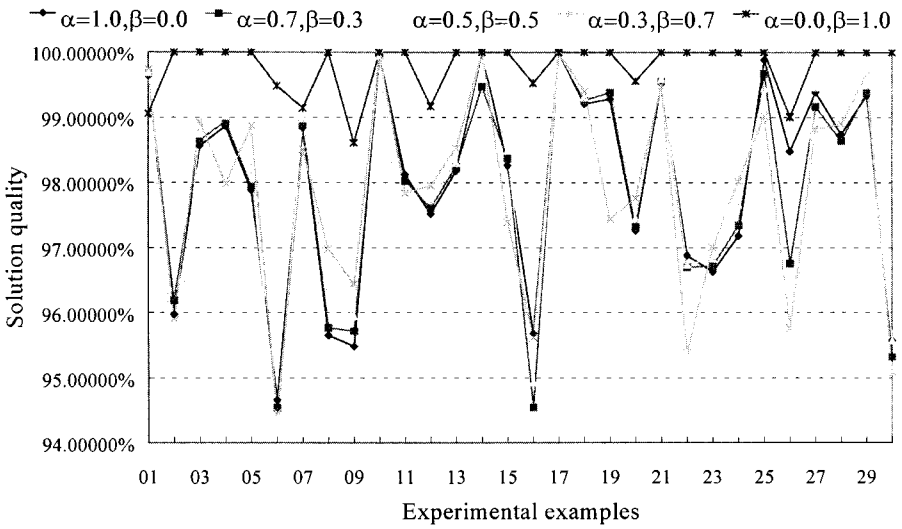

Fig. 4. The solution quality for 15 jobs.

Fig. 4 shows the solution quality for a 15-job case. In these 150 sets of testing examples, 91 sets can obtain optimal solutions within the allowable pivoting limit, while the remaining 59 sets cannot generate the optimal schedules through the integer programming model. Although 29 out of 150 optimal schedules are generated using the heuristic algorithm, the overall average solution quality is above 98.31% for th 15-job case. In contrast to the slow integer programming model, the heuristic scheduling algorithm is successful at producing ultimate schedules and is quick.

7. Conclusion

To schedule a two-machine flowshop environment, most studies developed optimization procedures for single criterion static scheduling problems. This paper attempts to solve a two-machine flowshop bicriteria scheduling problem with release dates of jobs, in which the objective function is to minimize a weighted sum of total flow time and makespan. An integer programming model and a heuristic scheduling algorithm are presented. Experimental results show that the proposed heuristic algorithm solves this problem quickly and accurately. In comparison to the SPT rule, the proposed heuristic algorithm yields much better solution quality. The overall average solution quality of the heuristic algorithm is above 99%. Processing of the 15-job case requires only 0.018 s on average to obtain an ultimate or even optimal solution. The heuristic scheduling algorithm is more practical to solve real world applications than the integer programming model.

References

[1] Baker KR. Introduction to sequencing and scheduling. New York: Wiley, 1974.
[2] Church LK, Uzsoy R. Analysis of periodic and event-driven rescheduling policies in dynamic shops. International Journal of Computer Integrated Manufacturing 1992;5(3):153–63.
[3] Croce FD, Narayan V, Tadei R. The two-machine total completion time flow shop problem. European Journal of Operational Research 1996;90:227–37.
[4] French S. Sequencing and scheduling: an introduction to the mathematics of the job-shop. Chichester: Ellis Horwood, 1982.
[5] Gonzalez T, Sahni S. Flowshop and jobshop schedules: complexity and approximation. Operations Research 1978;26(1):36–52.
[6] Hoogeveen JA, Van De Velde SL. Stronger Lagrangian bounds by use of slack variables: application to machine scheduling problem. In: Proceedings of the Third IPCO Conference, Erice, Italy, 1993. p. 195–208.
[7] Ignall E, Schrage LE. Application of the branch and bound technique to some flowshop scheduling problems. Operations Research 1965;13:400–12.
[8] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1:61–8.
[9] Kohler WH, Steiglitz K. Exact, approximate and guaranteed accuracy algorithms for the flowshop problem $n-2-F-\bar{F}$. Journal of the Association for Computing Machinery 1975;22:106–14.
[10] Nagar A, Heragu SS, Haddock J. A branch and bound approach for a two-machine flowshop scheduling problem. Journal of the Operational Research Society 1995;46:721–34.
[11] Nagar A, Heragu SS, Haddock J. A combined branch and bound and genetic algorithm based approach for a flowshop scheduling problem. Annals of Operations Research 1996;63:397–414.

[12] Rajendran C. Two-stage flowshop scheduling problem with bicriteria. Journal of the Operational Research Society 1992;43:871–84.
[13] Roy U, Zhang X. A heuristic approach to n/m job shop scheduling: fuzzy dynamic scheduling algorithms. Production Planning and Control 1996;7(3):299–311.
[14] Selen WJ, Hott D. A mixed integer goal programming formulation of the standard flowshop scheduling problem. Journal of the Operational Research Society 1986;37:1121–8.
[15] Sun D, Lin L. A dynamic job shop scheduling framework: a backward approach. International Journal of Production Research 1994;32(4):967–85.
[16] Van De Velde SL. Minimizing the sum of job completion times in the two-machine flow shop by Lagrangian relaxation. Annals of Operations Research 1990;26:257–68.
[17] Wilson JM. Alternative formulations of a flowshop scheduling problem. Journal of the Operational Research Society 1989;40:395–9.