J. Ind. Eng. Int., 7(13), 74-83, Spring 2011
ISSN: 1735-5702

Technical note:

Flexible flowshop scheduling with equal number of unrelated parallel machines

I. Mahdavi [1]; V. Zarezadeh [2]; P. Shahnazari-Shahrezaei [3]*

[1]*Associate Professor, Dep. of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran*

[2]*M.Sc., Dep. of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran*

[3]*Ph.D. Student, Dep. of Industrial Engineering, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran*

Received: 18 July 2008; Revised: 31 December 2008; Accepted: 12 January 2009

Abstract: This article addresses a multi-stage flowshop scheduling problem with equal number of unrelated parallel machines. The objective is to minimize the makespan for a given set of jobs in the system. This problem class is NP-hard in the strong sense, so a hybrid heuristic method for sequencing and then allocating operations of jobs to machines is developed. A number of test problems are randomly generated and results obtained by proposed heuristic are compared with optimal solutions reported by the Lingo 8.0 package applying the branch & bound approach. The results show that the proposed hybrid method is more efficient when the problem sizes have been increased.

Keywords: *Flexible flowshop; Makespan; Unrelated machines*

1. Introduction

Production scheduling can be defined as the allocation of available production resources over time to perform a collection of tasks (Baker, 1974). In most manufacturing environments like process industries; e.g. the chemical and petrochemical, rubber, steel, textile and food, a set of tasks is sequentially performed by resources in several stages to complete a job. Such a system is referred to as the flowshop environment and belongs to the class of quantitative combinatorial optimization problems.

This paper considers a flexible flowshop scheduling problem, where each production stage is made up of equal number of unrelated parallel machines with the objective of minimizing makespan. The considered problem generalizes two other scheduling problems; namely, the flowshop problem and the single-stage parallel machines problem allowing considerable reduction in makespan and the delays caused by bottleneck stages.

The main characteristic of the considered problem is the differences among the machines. The processing time of the jobs on the different machines, correspond to the three classical parallel machines, may be identical, uniform or unrelated. The multiple machines are identical if they do not differ in speed. The multiple machines are purported uniformly in that they differ in speed, but they differ by some constant speed factors. Specifically, the machines are unrelated if there are no relationships between machines' speed but a hierarchy of the machines does exist. In better words, the machines are not necessarily uniform to a speed factor, but the machines can be ranked from the highest to the lowest speed.

On the other hand, the considered problem is primarily concerned with industrial scheduling, where jobs have to be assigned to scarce resources (machines) at each stage first and then sequenced on each resource (machine) over time to optimize the performance measure. Since the flowshop and the single-stage parallel machines problems are known to be NP-hard, our problem is strongly NP-hard (Kis and Pesch, 2005). Therefore there is no escape from applying simple dispatching rules, heuristics and improving meta-heuristics to solve it.

The flowshop scheduling problem and its generalizations is a very controversial issue and has been the target of researches since Johnson's seminal paper in 1954. A comprehensive review of flowshop scheduling problems over the last 50 years is provided by Gupta and Stafford Jr. (2006) and a makespan review by Hejazi and Saghafian (2005) and also by Framinan *et al.* (2005).

A detailed survey for the flexible flowshop problem has been given by Linn and Zhang (1999) and Wang (2005). Most of the aforementioned works explore three different

*Corresponding Author Email: irajarash@rediffmail.com
Tel.: +98 9111131380

issues: processing complexity, performance measures and solution methods. Although the flexible flowshop problem has been widely studied in the literature, most of the studies related to flexible flowshop problems are concentrated on problems with identical machines. While this assumption may be true in some benchmarks, in many real world cases the multiple machines are not identical because multiple machines at a processing stage typically have small differences in processing speed; the addition of a machine at a bottleneck stage to relieve it is usually with a newer (faster) processor and the multiple machines may be obtained from different vendors (and thus have different processing speed) (Roa and Santos, 2000). Yet a few number of researches considered the real life assumption that the multiple machines that may exist at a stage are uniform or unrelated in processing speed.

Brah *et al.* (1991) formulated a mixed integer linear programming for flexible flowshop models with the general case of unrelated parallel machines. Roa and Santos (2000) studied the two stage flowshops with multiple uniform machines with the objective of minimizing makespan. They applied dispatching procedures such as FIFO, LIFO, SPT, LPT etc. and also two heuristics based on modification of the famous Johnson's algorithm and based on a new application of a heuristic presented by Sule (1997) respectively. They evaluated their proposed dispatching rules and heuristics against a makespan lower bound developed by themselves and reported that the heuristic based on Johnson's algorithm outperforms other dispatching rules and heuristics in the two stage flexible flowshop. Soewandi and Elmaghraby (2003) also considered the same problem and developed a heuristic (S–E heuristic) and derived a machine speed-dependent worst-case ratio bound for it, but Kyparisis and Koulamas (2006) observed that the worst-case bound derived by Soewandi et al. for their heuristic is not indicative of the expected performance of S–E heuristic when the machine speeds vary significantly. They addressed this issue by deriving alternative tight speed-dependent bounds for the S–E heuristic and reported that this new bound facilitates the narrowing of the gap between average experimental performance and worst-case performance for the S–E heuristic. In another research, Koulamas and Kyparisis (2007) studied the problem of minimizing the makespan in an alike two-stage flowshop scheduling problem with uniform parallel machines which is a generalization of the assembly flowshop problem with concurrent operations in the first stage and a single assembly operation in the second stage. They proposed a heuristic with an absolute performance bound which became asymptotically optimal as the number of jobs became very large as they showed. The aforementioned team; namely Kyparisis and Koulamas (2006) also studied the multistage flexible flowshop scheduling problem with uniform parallel machines in each stage and the objective of minimizing makespan and developed a general class of heuristics which extend several well-known heuristics for the serial flowshop problem such as the slope index method developed by Palmer, the CDS heuristic of Campbell et al., and the Dannenbring heuristic (Gupta and Stafford, 2006). They obtained absolute performance guarantees for their heuristics based on a similar absolute performance guarantees for the corresponding serial flowshop heuristics.

Low (2005) has developed a mathematical model for the flowshop with unrelated parallel machines and independent setup and dependent removal times and proposed a simulated annealing heuristic for minimizing total flow time of jobs in the system. Then Low *et al.* (2008) reported a two-stage flowshop scheduling problem with unrelated alternative machines to minimize the makespan that focused on the functions of the alternative machines. In better words, their model has m unrelated alternative machines at the first machine center followed by a second machine center with a common processing machine in the system. For the processing of any job, it is assumed that the operation can be partially substituted by other machines in the first center, depending on its machining constraints. 16 combinations of heuristic algorithms and dispatching rules were applied by them and the associated computational experiments indicated that the performance of the modified Johnson's rule combined with the FF dispatching rule is the best heuristic among all proposed algorithms for the considered model.

In a new research, Jungwattanakit *et al.* (2009) has formulated a 0-1 mixed integer program for the flexible flowshop problem with unrelated parallel machines and sequence and machine dependent setup times to minimize a convex combination of makespan and the number of tardy jobs. They have investigated both constructive and iterative (SA, TS and GA-based algorithms) approaches based on developing a job sequence for the first stage by a constructive procedure and improving it later iteratively, by sequencing the jobs for the remaining stages by both the

permutation and FIFO rules and by assigning the jobs at the stages to a particular machine using a greedy algorithm. From their computational experiences found that among the constructive algorithms the insertion-based approach is superior to the others, whereas the proposed SA algorithms are better than TS and genetic algorithms among the iterative metaheuristic algorithms.

Ruiz and Marato (2006) proposed a heuristic based on a genetic algorithm to solve the flexible flowshop with sequence dependent setup times, unrelated parallel machines at each stage and machine eligibility constraints to bridge the existing gap between the theory of scheduling and its applications in real industrial settings. Recently Ruiz *et al.* (2008) have presented a complete formulation as well as a mixed integer programming mathematical model and some heuristics for a complex and realistic flowshop problem. In this problem several realistic characteristics such as release dates for machines, existence of unrelated parallel machines at each stage of the flowshop, machine eligibility, possibility for jobs to skip stages, sequence dependent setup times, possibility for setup times to be both anticipatory as well as non-anticipatory, positive and/or negative time lags between operations and generalized precedence relationships between jobs are jointly considered. They have solved a comprehensive benchmark and carried out statistical analysis by means of decision trees which have allowed identifying some counter-intuitive interactions among many different characteristics of the realistic problem considered. Furthermore, they have also proposed simple dispatching rules and an adaptation of the NEH algorithm.

This research focuses on the flexible flowshop problem with equal number of unrelated machines at each stage and development of good makespan schedules. In Section 2, the production model under study is described down to the last detail. In Section 3, a hybrid heuristic method is proposed and will be illustrated by an example in Section 4. In Section 5, experiments for verifying the performance of the proposed heuristic algorithm are described. Finally, conclusions are presented in Section 6.

2. Problem definition and notations

The flexible flowshop scheduling problem with unrelated parallel machines under consideration has the following characteristics:

1. A set of N jobs denoted by $I = \{i \mid i = 1,2,...,N\}$ is available at time zero and no job may be cancelled before completion.

2. The production model consists of L consecutive stages. The set of stages is denoted by $J = \{j \mid j = 1,2,...,L\}$.

3. Each stage $j \in J$ is equipped with $M > 1$ nonidentical machines. The set of machines at stage j is denoted by $E_j = \{e_{jk} \mid k = 1,2,...,M\}$, $j \in J$.

4. All jobs have to be processed serially through all stages. Thus job $i \in I$ consists of a sequence of L operations, each of them corresponding to the processing of job i at stage j on machine e_{jk} during an uninterrupted processing time $P_{ijk} > 0$.

5. The machines are continuously available from time zero onwards and may remain idle.

6. Each machine can process one job at a time. Furthermore, a job can be processed by any of the machines and will be processed by a single machine at each stage.

7. Setup and removal times are assumed to be a part of the processing time and are independent of the job sequence.

8. The jobs can wait in between stages and the intermediate storage is unlimited.

To maximizing system utilization, the objective is to develop a schedule that minimizes the makespan. In a flowshop based model, a schedule that minimizes the makespan also minimizes the sum of job waiting times and the sum of machine idle times.

For the sake of completeness, a mixed integer linear programming formulation for the considered production model is included in this section. The framework of mathematical model initially developed by Brah *et al.* (1991).

2.1. Input parameters

N	The number of jobs.
M	The number of parallel machines at each stage.
J	The index number of stages; $j=1,\ldots,L$.
P_{ijk}	The processing time of job i at stage j on machine e_{jk}.

L The number of stages.

i The index number of jobs; $i=1,\ldots,N$.

k The index number of machines at each stage; $k=1,\ldots,M$.

2.2. Decision variables

C_{ij} The completion time of job i at stage j

$C_{\max}$ The makesapan

$$X_{irj}\begin{cases}1, & \text{if job } i \text{ precedes job } r \text{ at stage } j \\ & \text{on machine } e_{jk} \\ 0, & \text{otherwise}\end{cases}$$

$$Y_{ijk}\begin{cases}1, & \text{if job } i \text{ at stage } j \text{ is assigned to machine } e_{ik} \\ 0, & \text{otherwise}\end{cases}$$

2.3. Mathematical formulation

Min C_{max}

Subject to:

$$C_{max} \geq C_{iL} \quad \forall i \tag{1}$$

$$\sum_{k=1}^{M} Y_{ijk} = 1 \quad \forall i, j \tag{2}$$

$$\begin{cases} C_{i1} \geq \sum_{k=1}^{M} Y_{i1k} P_{i1k} & \forall i \\ C_{ij} - C_{i,j-1} \geq \sum_{k=1}^{M} Y_{ijk} P_{ijk} & \forall\ i \text{ and } j \geq 2 \end{cases} \tag{3}$$

$$\begin{cases} 10^{31}\left(2 - Y_{ijk} - Y_{rjk} + X_{irj}\right) + C_{ij} - C_{rj} \geq P_{ijk} \\ 10^{31}\left(3 - Y_{ijk} - Y_{rjk} - X_{irj}\right) + C_{rj} - C_{ij} \geq P_{rjk} \\ \forall r, i, j, k \ \ such\ that\ i < r \end{cases} \tag{4}$$

$$Y_{ijk} \in [0,1] \quad \forall i, j, k \tag{5}$$

$$X_{irj} \in [0,1] \quad \forall i,\ r,\ j\ such\ that\ i < r \tag{6}$$

$$C_{ij} \geq 0 \quad \forall i, j \tag{7}$$

$$C_{\max} \geq 0 \tag{8}$$

Constraints (2) guarantee the assignment of each job to one and only one machine at each stage. Constraints (3) ensure that it is not allowed to start processing the jobs at next stage unless they have completed processing at previous stage. In other words, they ensure that no job may be cancelled before completion. Constraints (4) are designed to deal with noninterference among the jobs using a common machine at any stage. Therefore, the difference between the processing times of any two jobs assigned to the same machine must be such that they do not overlap.

Although the flexible flowshop scheduling problem under study has a simple formulation, its NP-hardness essentially restricts the use of classical optimization methods based on mixed integer linear programming and branch & bound. In the next section, a heuristic algorithm is proposed to obtain a near optimal solution within a reasonable amount of time.

3. Proposed heuristic algorithm

In essence, the flexible flowshop problem consists of two sub-problems: assigning operations of each job to machines and sequencing operations on each machine. Gupta (1971) proposed a functional heuristic for the flowshop scheduling problem minimizing makespan. On the other hand, Sule (1997) addresses a method for scheduling jobs on the single-stage parallel processing problem with non-identical machines.

Although their procedures are designed for two basically different models but have been reviewed and integrated to obtain a newly modified / applied heuristic for sequencing first and then the assignment of operations in the flexible flowshop under study which is primarily a generalization of the classical flowshop and parallel shop. The significant side of the proposed heuristic is the collocation and reformation of the structure of the model under study which is the key to adapt it. The following is a step-by-step explanation of the algorithm.

Step 1. Without the loss of generality, rank the M machines at each stage from the highest to the lowest speed. So the most efficient machine (the one taking the least amount of time to process) at stage j is machine e_{j1}, the next efficient machine is e_{j2}, and so on.

Step 2. Form M simple L stage flowshops F_k

(k=1,2,…,M), each of which contains machine e_{jk} at stage *j (j=1,2,…,L)*. Denote O_k as the list of jobs and C_k as the makespan of jobs that will be assigned to the k^{th} flowshop.

Step 3. Dealing with each flowshop F_k *(k=1, 2,…, M)*:

Step 3.1. For each job $i \in I$, find π_{ik} as follows:

$$\pi_{ik} = \min_{j=1}^{L-1}\left(P_{ijk} + P_{i(j+1)k}\right) \tag{9}$$

Step 3.2. Form the subgroup of jobs U_k that take less time on the first machine than on the last in flowshop F_k, such that $U_k = \left\{i \in I \middle| P_{i1k} < P_{iLk}\right\}$.

Step 3.3. Form the subgroup of jobs V_k that take less time (or equal) on the last machine than on the first in flowshop F_k, such that:

$$V_k = \left\{i \in I \middle| P_{i1k} \geq P_{iLk}\right\} \tag{10}$$

Step 3.4. Sort the jobs in U_k in an ascending order of $\pi_{ik}'s$; if two or more jobs have the same value of π_{ik}, sort them in an arbitrary order.

Step 3.5. Sort the jobs in V_k in a descending order of $\pi_{ik}'s$; if two or more jobs have the same value of π_{ik}, sort them in an arbitrary order.

Step 3.6. Rank the jobs in the sorted order of U_k, then in the sorted order of V_k; Call this the job list of flowshop *k* denoted by Q_k.

Step 4. Assign all the jobs to the set of unscheduled jobs denoted by *NS*.

Step 5. Assign all the jobs to the first flowshop F_1 in order of the established priorities of jobs in the Q_1(Set O_1=Q_1) and then calculate C_1. C_k for *k=2,…,M* is 0, because no jobs are yet assigned to flowshops 2 through M ($O_k = \phi; k = 2,...,M$).

Step 6. Set $C_{\max} = \max\left\{C_k \middle| k = 1,...,M\right\}$.

Step 7. Take the job in *NS* which its elimination from the set O_1 has the most effect on decreasing C_1 as J_c. Temporarily remove J_c from flowshop F_1. Calculate C_1^T as the corresponding makespan of jobs assigned to F_1 after temporary deletion of J_c.

Step 8. Temporarily assign J_c to all flowshops F_k *(k=2,…,M)*. Note that the relative position of jobs assigned to each set O_k *(k=2,…,M)*must be the same as in Q_k. Calculate C_k^T as the corresponding makespan of jobs assigned to F_k after temporary insertion of J_c.

Step 9. Set $k_c = \arg\min_{k=2}^{M}\left\{C_k^T\right\}$, If

$$\max\left\{C_1^T, C_{k_c}^T\right\} < C_{\max}, \tag{11}$$

make the elimination of J_c from F_1 and insertion it to the k_c^{th} flowshop permanent, Update C_k *(k=1,…,M)* else the reassignment of J_c is rejected.

Step 10. Delete job J_c from *NS*. If $NS = \phi$, stop else go to Step 6.

After Step 10, scheduling is finished and the sequence of jobs assigned to each machine at each stage and the corresponding makespan of the jobs (C_{max}) has been found.

It is important to consider that the same sequence of jobs is maintained throughout each flowshop F_k and the relative position of jobs in each sequence is the same as in job list Q_k.

4. An illustrative example

Consider a job shop with four stages and three nonidentical machines at each stage. There are six jobs to be processed, and the time estimates for each job on each processor at each stage are given in Table 1. So *N*=6, *L*=4, *M*=3 and $I = \left\{i \middle| i = 1,...,6\right\}$, $J = \left\{j \middle| j = 1,...,4\right\}$ and $E_j = \left\{e_{jk} \middle| k = 1,2,3\right\}, j \in J$.

The algorithm proceeds as follows:

(1) According to step 1, the machines at each stage are already numbered so that the machine numbers are in the increasing order of a single job times on each machine. So

$$E_j = \left\{e_{jk} \middle| k = 1,2,3\right\}, j \in J \tag{12}$$

(2) According to step 2, three machine groups, F_1, F_2, F_3 are formed, each of which is thought of as a three-machine flowshop. Without loss of generality, we may assume the flowshops are constructed as follows:

$$F_1 \to \left\{e_{11} + e_{21} + e_{31}\right\} \quad F_2 \to \left\{e_{12} + e_{22} + e_{32}\right\}$$

$$F_3 \to \left\{e_{13} + e_{23} + e_{33}\right\}$$

where e_{jk} is the kth machine at the jth stage.

Table 1: Processing times for the six jobs.

Jobs	*Stages*											
	1			2			3			4		
	Machines			Machines			Machines			Machines		
	1	2	3	1	2	3	1	2	3	1	2	3
1	25	30	32	45	54	58	52	62	68	40	48	52
2	7	9	10	41	51	56	22	28	30	66	82	90
3	41	47	52	55	63	70	33	38	42	21	24	27
4	74	88	100	12	14	16	24	28	33	48	57	67
5	7	9	11	15	20	25	72	96	120	52	69	86
6	12	15	16	14	16	20	22	25	28	32	35	38

(3) Dealing with each flowshop $F_k(k=1,2,3)$:

(3-1) For each job $i \in I$, π_{ik} is calculated as shown in Table 2.

(3-2) The subgroups of jobs $U_k(k=1,2,3)$ are formed such that $U_k = \{i \in I \mid P_{i1k} < P_{iLk}\}$ as:

$$k=1 \rightarrow U_1 = \{1,2,5,6\}$$
$$k=2 \rightarrow U_2 = \{1,2,5,6\}$$
$$k=3 \rightarrow U_3 = \{1,2,5,6\}$$

(3-3) The subgroups of jobs $V_k(k=1,2,3)$ are formed such that $V_k = \{i \in I \mid P_{i1k} \geq P_{iLk}\}$ as:

$$k=1 \rightarrow V_1 = \{3,4\}$$
$$k=2 \rightarrow V_2 = \{3,4\}$$
$$k=3 \rightarrow V_3 = \{3,4\}$$

(3-4) The jobs in U_k are sorted in ascending order of π_{ik}'s as follows:

$$k=1 \rightarrow U_1 = \{5,6,2,1\}$$
$$k=2 \rightarrow U_2 = \{5,6,2,1\}$$
$$k=3 \rightarrow U_3 = \{5,6,2,1\}$$

(3-5) The jobs in V_k are sorted in descending order of π_{ik}'s as follow:

$$k=1 \rightarrow V_1 = \{3,4\}$$
$$k=2 \rightarrow V_2 = \{3,4\}$$
$$k=3 \rightarrow V_3 = \{3,4\}$$

(3-6) The jobs are ranked in the sorted order of U_k, then in the sorted order of V_k ; called the job list of flowshop k denoted by Q_k. So:

$$k=1 \rightarrow Q_1 = \{5,6,2,1,3,4\}$$
$$k=2 \rightarrow Q_2 = \{5,6,2,1,3,4\}$$
$$k=3 \rightarrow Q_3 = \{5,6,2,1,3,4\}$$

(4) According to step 4, Let *NS*={1,2,3,4,5,6}.

(5) According to step 5, all the jobs are assigned to the first flowshop F_1 in order of the established priorities of jobs in the Q_1. So:

$$O_1 = \{5,6,2,1,3,4\} \rightarrow C_1 = 353$$
$$O_2 = \phi \rightarrow C_2 = 0$$
$$O_3 = \phi \rightarrow C_3 = 0$$

(6) According to step 6 and followed by (5), $C_{\max} = \max\{353,0,0\} = 353$.

(7) According to step 7 and Table 3 below, elimination of J_5 from the set O_1 has the most effect on decreasing C_1. $J_c=J_5$ is temporarily removed from flowshop F_1, so $C_1^T = 273$.

(8) According to step 8, $J_c=J_5$is temporarily assigned to flowshops F_k $(k=2,3)$. The values of C_k^T as the corresponding makespan of jobs assigned to F_k after temporary insertion of J_c are as follows:

$$k=2 \rightarrow C_2^T = 194$$
$$k=3 \rightarrow C_3^T = 242$$

(9) According to Step 9,

$$\begin{cases} \min\{C_2^T, C_3^T\} = \min\{194,242\} \\ \qquad = 194 \rightarrow k_c = 2 \rightarrow C_{k_c}^T = 194 \\ \max\{C_{k_c}^T, C_1\} = \max\{194,273\} = 273 < C = 353 \end{cases}$$

so $J_c=J_5$ is removed from F_1 and assigned to F_2.

Table 2: Calculation of π_{ik} for each job at each flowshop.

i	Flowshop F_1 $\pi_{i1}=\min_{j=1}^{3-1}\left(P_{ij1}+P_{i(j+1)1}\right)$	Flowshop F_2 $\pi_{i2}=\min_{j=1}^{3-1}\left(P_{ij2}+P_{i(j+1)2}\right)$	Flowshop F_3 $\pi_{i3}=\min_{j=1}^{3-1}\left(P_{ij3}+P_{i(j+1)3}\right)$
1	min(25+45, 45+52, 52+40)=70	min(30+54, 54+62, 62+48)=84	min(32+58, 58+68, 68+52)=90
2	min(7+41, 41+22, 22+66)=48	min(9+51, 51+28, 28+82)=60	min(10+56, 56+30, 30+90)=66
3	min(41+55, 55+33, 33+21)=54	min(47+63, 63+38, 38+24)=62	min(52+70, 70+42, 42+27)=69
4	min(74+12, 12+24, 24+48)=36	min(88+14, 14+28, 28+57)=42	min(100+16, 16+33, 33+67)=49
5	min(7+15, 15+72, 72+52)=22	min(9+20, 20+96, 96+69)=29	min(11+25, 25+120, 120+86)=36
6	min(12+14, 14+22, 22+32)=26	min(15+16, 16+25, 25+35)=31	min(16+20, 20+28, 28+38)=36

Table 3: The effects of elimination of the jobs in *NS* from F_1 on decreasing C_1 in the first iteration.

$i \in NS$	C_1-C_1^T
1	40
2	66
3	21
4	48
5	80
6	32

(10) According to step 10, J_c=$J_5$is deleted from *NS*. Because $NS \neq \phi$, steps 6 to 10 of the proposed algorithm is repeated until $NS = \phi$. The iterations are shown in Table 4.

The final solution includes the jobs assigned to each flowshop and sequence of them in order of the established priorities of jobs at each flowshop in step 3. The results are shown in Table 5.

At each flowshop, the sequence of jobs do not change, so as a matter of fact, the assignment of operations of each job to machines at each stage and sequencing operations on each machine are determined and the final schedule in the considered flexible flowshop is obtained.

5. Computational experiments and results

To determine the quality of the proposed heuristic, a spreadsheet model of the considered problem was developed and a number of test problems were randomly generated and solved by proposed heuristic coded in Microsoft Visual Basic for Applications. The results were compared to the optimal solutions obtained via the implementation of the mixed integer linear programming formulation (presented in section 2) by the Lingo 8.0. All experimental tests were implemented on a personal computer with an Intel Pentium III 633 GHz CPU and 256 MB of RAM.

Four sets of problems were tested, respectively for 3 to 6 jobs. Each job has three operations and each stage has two nonidentical machines. The processing time of each operation was randomly generated and each set of problems was executed for 15 tests. The makespans for problems of three to six jobs by proposed heuristic and by the branch & bound approach are shown respectively in Figures 1 to 4.

The mean relative error between the optimal makespans and those obtained by proposed heuristic for each set of test problems is shown in Table 6. Over the entire collection of instances, the average relative error of the proposed heuristic is 6.08%. The average execution times for solving problems of three to six jobs by the Lingo 8.0 is shown in Figure 5. From the figures, it is easily seen that the proposed heuristic got a little larger makespan than the branch & bound approach did, but the computational time needed by the Lingo 8.0 was however much larger than that needed by proposed algorithm.

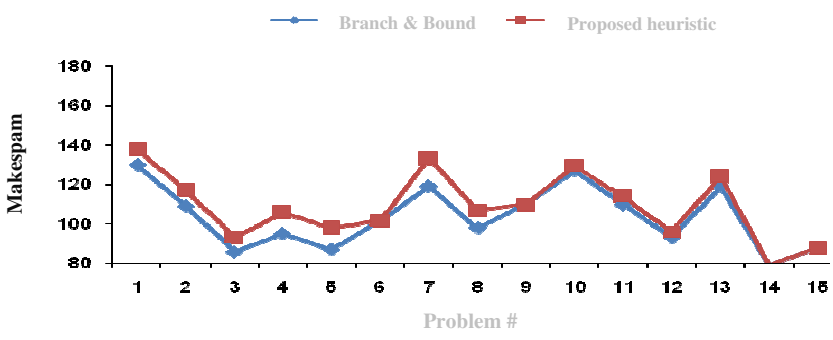

Figure 1: Makespans of 15 tests for three jobs.

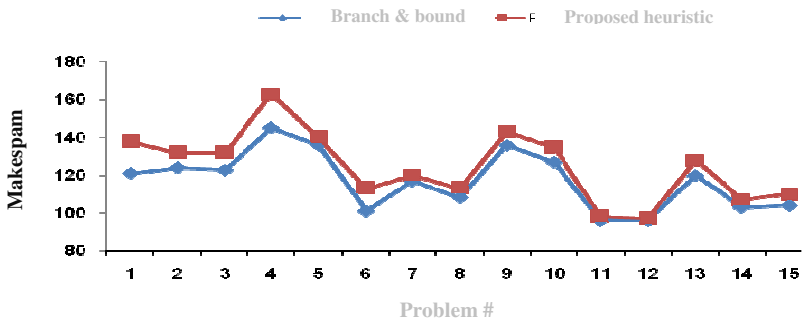

Figure 2: Makespans of 15 tests for four jobs.

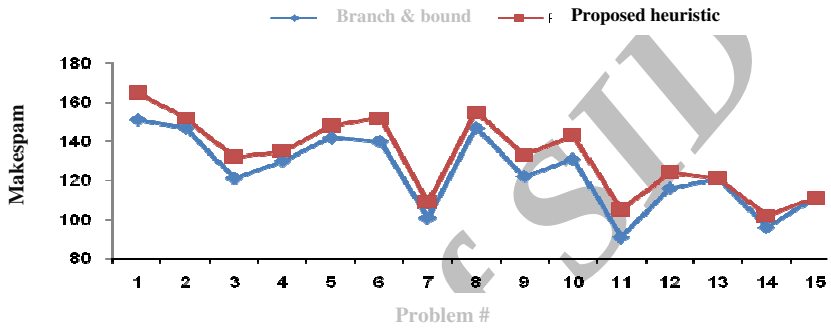

Figure 3: Makespans of 15 tests for five jobs.

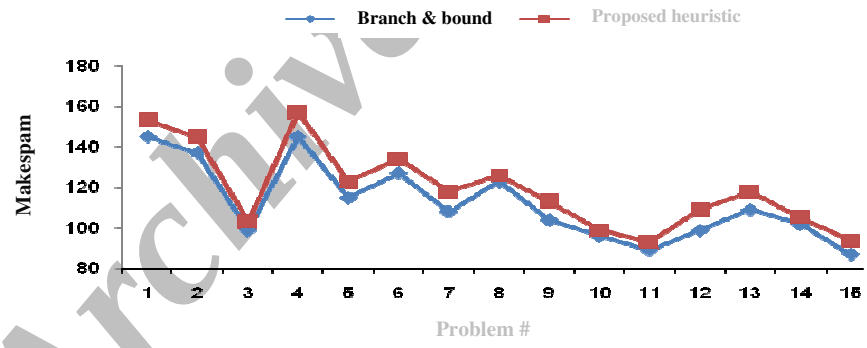

Figure 4: Makespans of 15 tests for six jobs.

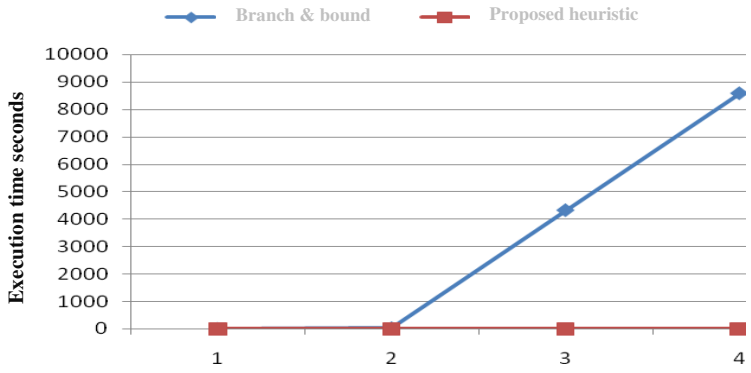

Figure 5: The average CPU times for processing different numbers of jobs.

Table 4: The iterations of steps 6 to 10 of the proposed heuristic solving the illustrative example.

Iteration 1

C_{max}=353
NS={J_6, J_2, J_1, J_3, J_4}
J_c={J_5}

F_k	O_k	C_k^T
F_1	{J_6, J_2, J_1, J_3, J_4}	273
F_2	{J_5}	194
F_3	{J_5}	242
	$C_{k_c}^T$	194
	$\max\{C_{k_c}^T, C_1^T\}$	273
Decision	273<353	
	Assigning J_5 to F_2	

Iteration 2

C_{max}=273
NS={J_6, J_2, J_1, J_3}
J_c={J_4}

F_k	O_k	C_k^T
F_1	{J_6, J_2, J_1, J_3}	225
F_2	{J_5,J_4}	251
F_3	{J_4}	216
	$C_{k_c}^T$	216
	$\max\{C_{k_c}^T, C_1^T\}$	225
Decision	225<273	
	Assigning J_4 to F_3	

Iteration 3

C_{max}=225
NS={J_6,J_2,J_3}
J_c={J_1}

F_k	O_k	C_k^T
F_1	{J_6,J_2,J_3}	176
F_2	{J_5,J_1}	242
F_3	{J_1,J_4}	277
	$C_{k_c}^T$	242
	$\max\{C_{k_c}^T, C_1^T\}$	242
Decision	242>225	
	Remaining J_1 to F_1	

Iteration 4

C_{max}=225
NS={J_6,J_3}
J_c={J_2}

F_k	O_k	C_k^T
F_1	{J_6, J_1,J_3}	195
F_2	{J_5,J_2}	276
F_3	{J_2,J_4}	253
	$C_{k_c}^T$	253
	$\max\{C_{k_c}^T, C_1^T\}$	253
Decision	242>225	
	Remaining J_2 to F_1	

Iteration 5

C_{max}=225
NS={J_6}
J_c={J_3}

F_k	O_k	C_k^T
F_1	{J_6, J_2, J_1}	204
F_2	{J_5,J_3}	218
F_3	{J_3,J_4}	268
	$C_{k_c}^T$	218
	$\max\{C_{k_c}^T, C_1^T\}$	218
Decision	218<225	
	Assigning J_3 to F_2	

Iteration 6

C_{max}=218
NS={ }
J_c={J_6}

F_k	O_k	C_k^T
F_1	{J_2, J_1}	185
F_2	{J_5, J_6, J_3}	253
F_3	{J_6,J_4}	232
	$C_{k_c}^T$	232
	$\max\{C_{k_c}^T, C_1^T\}$	232
Decision	232<218	
	Remaining J_6 to F_1	

Table 5: The final schedule in the considered flexible flowshop.

F_k	O_k	C_k	
F_1	J_6	80	204
	J_2	155	
	J_1	204	
F_2	J_5	194	218
	J_3	218	
F_3	J_4	216	216
		C_{max}	218

Table 6: The performance of the proposed heuristic.

Set of test problems	Mean relative error
Three jobs	5.30%
Four jobs	6.21%
Five jobs	6.57%
Six jobs	6.23%
Average	6.08%

References

Baker, K. R., (1974), *Introduction to sequencing and scheduling*. John Wiley & Sons, New York.

Brah, S. A.; Hunsucker, J. L.; Shah, J., (1991), Mathematical modeling of scheduling problems. *Journal of Information & Optimization Sciences,* 12(1), 113-137.

Framinan, J. M.; Leisten, R.; Ruiz-Usano, R., (2005), Comparison of heuristics for flow time minimisation in permutation flowshops. *Computers & Operations Research*, 32(5), 1237-1254.

Gupta, J. N. D.; Stafford Jr., E. F., (2006), Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3), 699-711.

Gupta, J. N., (1971), A functional heuristic algorithm for the flowshop scheduling problem. *Operational Research Quarterly*, 22(1), 39-47.

Hejazi, S. R.; Saghafian, S., (2005), Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research*, 43(14), 2895-2929.

Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P.; Werner, F., (2009), A comparison of scheduling algorithms for flexible flowshop problems with unrelated parallel machines, setup times and dual criteria. *Computers & Operations Research*, 36(2), 358-378.

Kis, T.; Pesch, E., (2005), A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research*, 164(3), 592-608.

Koulamas, C.; Kyparisis, G. J., (2007), A note on the two-stage assembly flowshop scheduling problem with uniform parallel machines. *European Journal of Operational Research,* 182(2), 945-951.

Kyparisis, G. J.; Koulamas, C., (2006), A note on makespan minimization in two-stage flexible flowshops with uniform machines. *European Journal of Operational Research,* 175(2), 1321-1327.

Kyparisis, G. J.; Koulamas, C., (2006), Flexible flowshop scheduling with uniform parallel machines. *European Journal of Operational Research,* 168(2), 985-997.

Linn, R.; Zhang, W., (1999), Hybrid flowshop scheduling: A survey. *Computers & Industrial Engineering,* 37(1-2), 57-61.

Low, C., (2005), Simulated annealing heuristic for flowshop scheduling problems with unrelated parallel machines. *Computers & Operations Research,* 32(8), 2013-2025.

Low, C.; Hsu, C.-J.; Su, C.-T., (2008), A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines. *Computers & Operations Research,* 35(3), 845-853.

Roa, I.; Santos, D. L., (2000), *Flowshops with Non-Identical Multiple Processors: A Study on Makespans*. Proceedings of The 5th Annual International Conference on Industrial Engineering -Theory, Applications and Practice, Hsinchu, Taiwan.

Ruiz, R.; Maroto, C., (2006), A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3), 781-800.

Ruiz, R.; Serifoglu, F. S.; Urlings, T., (2008), Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4), 1151-1175.

Soewandi, H.; Elmaghraby, S. E., (2003), Sequencing on two-stage hybrid flowshops with uniform machines to minimize makespan. *IIE Transactions*, 35(5), 467-477.

Sule, D. R., (1997), *Industrial scheduling*. 1st edition, PWS Publishing, Boston, 93-96.

Wang, H., (2005), Flexible flowshop scheduling: optimum, heuristics, and artificial intelligence solutions. *Expert Systems*, 22(2), 78-85.