2 단계 혼합흐름공정에서의 일정계획문제에 관한 연구

Two-Stage Hybrid Flow Shop Scheduling: Minimizing the Number of Tardy Jobs

Hyun-Seon Choi and **Dong-Ho Lee**

Department of Industrial Engineering
Hanyang University
Sungdong-gu, Seoul 133-791
KOREA

Abstract

This paper considers a hybrid flow shop scheduling problem for the objective of minimizing the number of tardy jobs. The hybrid flow shop consists of two stages in series, each of which has multiple identical parallel machines, and the problem is to determine the allocation and sequence of jobs at each stage. A branch and bound algorithm that gives the optimal solutions is suggested that incorporates the methods to obtain the lower and upper bounds. Dominance properties are also derived to reduce the search space. To show the performance of the algorithm, computational experiments are done on randomly generated problems, and the results are reported.

1. Introduction

A hybrid flow shop, alternatively called a flow shop with multiple processors, is an extended system of the classical flow shop. The overall system consists of two or more production stages in series, but there exist one or more parallel machines at each stage. The parallel machines are added to each stage of the flow shop for the objective of increasing productivity and/or flexibility. The hybrid flow shop can be found in the electronics industry such as printed circuit board (PCB) manufacturing, semiconductor manufacturing, and lead frame manufacturing [13, 14]. Also, a number of traditional industries, such as food, chemical and steel, have various types of hybrid flow shops [17].

In the hybrid flow shop, the flow of jobs is basically unidirectional through the serial production stages, and each job can be processed by one of the identical or nonidentical parallel machines at each stage. There may be finite buffers to decouple consecutive production stages. Also, a certain amount of setup time may be required when changing the product type at each machine.

Among various decision problems in hybrid flow shops, this paper considers the scheduling problem. In general, there are two types of decisions in hybrid flow shop scheduling: assigning jobs to machines at each stage and sequencing jobs at each machine.

There are a number of research articles on hybrid flow shop scheduling. (See Linn and Zhang [14] for a literature review.) Most of them deal with the measures without due-date, such as makespan or mean flow time. Gupta and Tunc [8] consider the objective of minimizing makespan, and suggest heuristic algorithms. Other heuristics are suggested by Chen [4] and Lee and Vairaktarakis [11] that consider two-stage hybrid flow shops. Fouad *et al*. [5] consider a three-stage hybrid flow shop scheduling problem in the woodworking industry and suggest heuristic algorithms. Brah and Hunsucker [3] suggest branch and bound algorithms that minimize makespan, and their lower bounds are improved by Moursli and Pochet [16]. Also, Azizoglu *et al.* [1] consider the objective of minimizing total flow time, and suggest another branch and bound algorithm.

Several research articles consider due-date based measures. Guinet and Solomon [7] develop several list scheduling algorithms for the multi-stage hybrid flow shop scheduling problem. The objectives considered are minimizing maximum tardiness or maximum completion time. Recently, Lee and Kim [12] considered a two-stage hybrid with parallel machines only at the first stage, and suggested a branch and bound algorithm that minimizes total tardiness, and Lee *et al*. [13] considered multi-stage hybrid flowshops and suggested a bottleneck-focused heuristic in which a schedule for a bottleneck stage is first constructed and then schedules for other stages are constructed based on the schedule for the bottleneck.

This paper focuses on a scheduling problem in two-stage hybrid flow shops with two or more identical parallel machines at each stage. The objective is to

minimize the number of tardy jobs. Here, a tardy job is defined as the job whose completion time is greater than its due date. This objective is important in many cases since the cost penalty incurred by a tardy job does not depend on how late it is, but the fact that it is late. For example, a late job may cause a customer to switch to another supplier, especially in the just-in-time production environment [10]. As noted in the previous research articles, the problem considered in this paper is an NP-hard problem. This can be easily seen from the fact that the parallel machine scheduling problem that minimizes the number of tardy jobs is NP-hard [6].

The objective of minimizing the number of tardy jobs is dealt with by Gupta and Tunc [9] that considers a two-stage hybrid flow shop with only one machine at the first stage. To solve the problem, they suggest several heuristic algorithms. Unlike this, we focus on general two-stage hybrid flow shops in which two or more machines may exist at each stage and suggest a branch and bound algorithm that gives optimal solutions. The methods to obtain lower and upper bounds are incorporated in the algorithm. Dominance properties are also derived to reduce the solution space. To show the performance of the algorithm, computational experiments are performed on randomly generated problems, and the results are reported.

In the next section, the problem considered here is described in more detail with a mathematical formulation. The branch and bound algorithm is presented in Section 3, and the results of computational test are presented in Section 4. Finally, Section 5 concludes the paper with a short summary and discussions on possible extensions.

2. Problem Description

Before describing the problem considered in this paper, we present a general structure of the multi-stage hybrid flow shop in Figure 1. In this figure, K and M_k denote the number of stages and the number of machines at stage k, respectively. Each job consists of K operations and the kth operations of the jobs are processed on stage k. Note that this paper focuses on the two-stage general hybrid flow shop, i.e., $K = 2$ and $M_k > 1$ for k=1, 2. Therefore, each job consists of

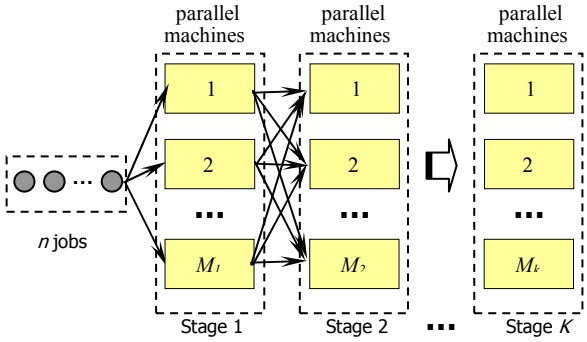

Figure 1. Hybrid flow shop: a schematic view

two operations. The first operations of all jobs are processed on one of the machines at the first stage, and each of the second operations of the jobs can be processed on one of those at the second stage. Here, the operations are processed sequentially, without overlapping between stages.

As stated earlier, there are two types of decision variables in the hybrid flow shop scheduling problem: (a) allocating jobs to machines at each stage; and (b) sequencing the jobs assigned to each machine. The objective is to minimize the number of tardy jobs, and can be represented as

$$\sum_{i=1}^{n} \delta(T_i),$$

Where $T_i = \max\{0, C_i - d_i\}$, i.e., tardiness of job i, and $\delta(a) = 1$ if $a > 0$, and 0 otherwise. Here, C_i and d_i denote the completion time and the due date of job i, respectively. Note that the completion times of jobs depend on the two decision variables, allocation and sequencing, and the problem considered here is to determine them for the objective of minimizing the number of tardy jobs in the two-stage hybrid flow shop.

In this paper, we consider the deterministic and static version of the problem. That is, all jobs are ready for processing at time zero, and job descriptors, processing times and due dates, are deterministic and given in advance. It is assumed that the parallel machines at each stage are identical. Other assumptions made in the problem considered here are: (a) there is a buffer of an infinite capacity between the two stages; (b) no job can be split or pre-emptied; (c) all machines are available at the beginning of the scheduling horizon; (d) each machine can process only one job at a time and each job can be processed on one machine; and (e) machine breakdowns are not considered.

The problem can be formulated as an integer programming model. The notations used are summarized below.

Parameters

N	number of jobs
M_k	number of identical machines in stage k
p_{ik}	processing time of job i at stage k, ($i = 1,\ldots, N$, $k = 1, 2$)
d_i	due date of job i
V	large number

Decision variables

x_{ijmk}	=1 if job j is processed directly after job i on machine m in stage k, and 0 otherwise
x_{0jmk}	=1 if job j is the first job to be processed on machine m in stage k, and 0 otherwise
x_{i0mk}	=1 if job i is the last job to be processed on machine m in stage k, and 0 otherwise
c_{ik}	completion time of job i at stage k,

Now, the integer programming model is given as follows. Note that the model is a modified one of that of Guinet and Solomon [7].

$$\text{Minimize} \sum_{i=1}^{N} \delta(T_i)$$

subject to

$$\sum_{m=1}^{M_k} \sum_{\substack{i=0 \\ i \neq j}}^{N} x_{ijmk} = 1 \qquad \text{for all } j \text{ and } k \qquad (1)$$

$$\sum_{j=1}^{N} x_{0jmk} \leq 1 \qquad \text{for all } m \text{ and } k \qquad (2)$$

$$\sum_{\substack{i=0 \\ i \neq h}}^{N} x_{ihmk} - \sum_{\substack{j=0 \\ j \neq h}}^{N} x_{hjmk} = 0 \qquad \text{for all } h, m \text{ and } k \qquad (3)$$

$$c_{jk} \geq c_{ik} + p_{jk} + \left(\sum_{m=1}^{M_k} x_{ijmk} - 1 \right) \cdot V \qquad \text{for all } j, k \text{ and } i = 0, \ldots N \qquad (4)$$

$$c_{jk} \geq c_{j,k-1} + p_{jk} \quad \text{for all } j \text{ and } k \qquad (5)$$

$$T_i = \max\{0, c_{i2} - d_i\} \qquad \text{for all } i \qquad (6)$$

$$\delta(T_i) \in \{0,1\} \qquad \text{for all } i \qquad (7)$$

$$x_{ijmk} \in \{0,1\} \qquad \text{for all } i, j, k \text{ and } m \qquad (8)$$

$$c_{jk} \geq 0 \quad \text{for all } j \text{ and } k \qquad (9)$$

$$c_{j0} = 0 \quad \text{for all } j \text{ and } c_{01} = 0 \qquad (10)$$

The objective function denotes minimizing the number of tardy jobs. Constraint (1) ensures that each job is processed once and once only at each stage. Constraint (2) specifies that each machine must be assigned to one job at most. Constraints (3) ensure that each job has a job predecessor and a job successor on its machine. The job completion time at each machine is represented by constraints (4) and (5). Constraint (6) specifies the tardiness of each job and is used to specify the number of tardy jobs. Finally, the other constraints (7), (8), (9) and (10) are the conditions on the decision variables.

3. Branch and Bound Algorithm

This section presents the branch and bound (B&B) algorithm suggested in this paper. First, we explain the branching scheme that generates all possible solutions. Then, the methods to obtain the lower and upper bounds are presented. As in the ordinary B&B algorithm, each node of the B&B tree can be deleted from further consideration (fathomed) if the lower bound at the node is greater than or equal to the incumbent solution value, i.e., the smallest upper bound of all nodes obtained so far. Dominance properties are also suggested to reduce the solution space.

3.1 Branching strategy

To generate all possible solutions in the two-stage hybrid flow shop scheduling, we adopt the idea suggested by Azizoglu *et al*. [1] that consider the problem of minimizing total flow time.

The entire B&B tree consists of two subtrees in series, each of which represents $N!$ orderings of jobs for each stage of the two-hybrid flow shop. In the first subtree, N nodes are branched at the first level, $N - 1$ nodes at the second level, and so on. Also, the second subtree starts from each of the leaf nodes of the first subtree. That is, N nodes are branched at level $N + 1$, $N - 1$ nodes at the level $N + 2$, and so on. In this way, we can generate $(N!)^2$ orderings of jobs.

Each node of the subtree corresponds to a partial schedule in the corresponding stage. More specifically, at each node, a set of jobs can be specified by going back on the path from that node toward the root node, and each of these jobs are allocated and sequenced to the earlier available machine in sequence. In this way, we can generate all possible allocation and sequence at each stage since we consider the regular measure of performance. (See Azizoglu *et al*. [1] for more details.) For node selection (or branching), the depth-first rule is used in this paper. In this rule, if the current node is not fathomed, the next node to be considered is its child node with the smallest index.

Figure 2 shows an example of the B&B tree for a problem with 3 jobs. It can be seen from the figure that this method determines the job schedule from the first to the second stage.

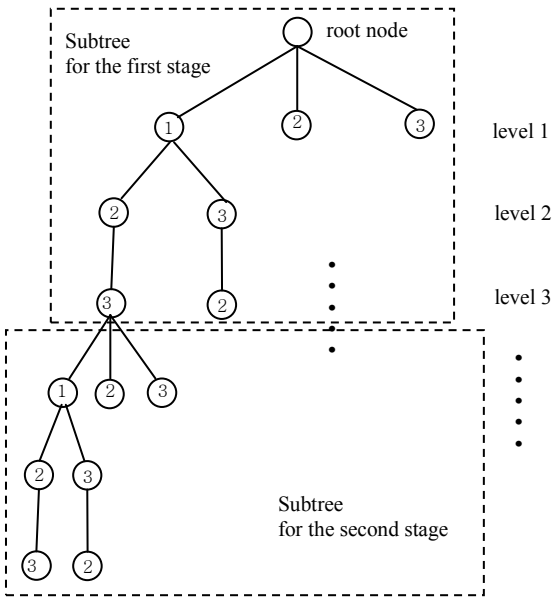

Figure 2. Branch and bound tree: example

3.2 Bounding strategy

This subsection presents the methods to obtain the lower and upper bounds.

Obtaining lower bound

The lower bound suggested in this paper is computed at each node of the B&B tree. As stated earlier, each node of the B&B tree corresponds to a partial schedule and the lower bound is computed by estimating the smallest (and also infeasible) completion time of each job at the second stage. To do this, we use the partial schedule and the jobs not included in the partial schedule.

Before describing the method, let PS_l denote the set of jobs included in the partial schedule at node l. Two cases are considered in the computation of the lower bound.

Case 1: Current node l in the first-stage subtree

In this case, the lower bound at node l is obtained as

$$NT_1 + NT_2,$$

where NT_1 is the number of tardy jobs for those in PS_l and NT_2 is the number of tardy jobs for those not in PS_l.

First, NT_1 is obtained by estimating the smallest (and also infeasible) completion time of each job at the second stage. That is, it can be set as

$$c_{i2} = \begin{cases} c_{i1} + p_{i2} & \text{if } c_{i1} < \varphi_T \\ \varphi_T + p_{i2} & \text{otherwise,} \end{cases}$$

where φ_T is ready time of the earliest available machine at second stage. Note that the smallest completion time given above is always smaller than that of the optimal schedule since the delay times are ignored. Then, the number of tardy jobs NT_1 can be calculated by comparing the smallest completion time and due date of each job in PS_l.

Second, NT_2 is obtained by relaxing the hybrid shop problem to the single machine problem that minimizes the number of tardy jobs. First, we modify the processing time of unscheduled job $i \notin PS_l$ at the second stage as

$$p_{i2}' = p_{i2} / M_2,$$

i.e., the job splitting is allowed, and the corresponding single machine problem is solved using the optimal algorithm of Moore. (See Moore [15] for more details.) Then, the sequence of unscheduled jobs (for $i \notin PS_l$) is obtained. Finally, NT_2 is calculated by comparing the completion time and the due date of each job not in PS_l.

Note that in this case, $NT_1 + NT_2$ can be the lower bound because it is computed in such a way that each unscheduled job is completed at the first stage before its release time at the second stage. Also the single machine problem is solved using the processing time divided by the number of parallel machines at the second stage.

Case 2: Current node l in the second-stage subtree

In this case, the lower bound is also obtained as

$$NT_1 + NT_2.$$

Here, NT_1 is obtained using the method suggested in the first case, i.e., estimating the smallest (and also infeasible) completion time of each job at the second stage. On the other hand, NT_2 is obtained by estimating the completion time as

$$c_{i2} = c_{i1} + p_{i2} \quad \text{for } i \notin PS_l.$$

Note that in this case, the waiting time of unscheduled jobs at the second stage, i.e. $\max\{0, c_{i1} - \varphi_T\}$, is ignored. Therefore, $NT_1 + NT_2$ can be the lower bound. That is, it is always less than the optimal number of tardy jobs.

Obtaining upper bound

The initial upper bound, i.e. feasible solution value, is obtained using two priority rules, EDD (earliest due date) and minimum slack time. Here, the EDD rule is used at the first stage after modifying the due date of each job as

$$d_i' = d_i - p_{i2},$$

and the slack time is defined as.

$$d_i - (p_{i1} + p_{i2})$$

Note that the initial upper bound is set to the minimum of those obtained by the two rules. Also, the upper bound is updated if it is improved at the leaf nodes of the B&B tree.

3.3. Dominance properties

As stated earlier, the dominance properties are used to reduce the number of partial schedules that need to be examined in the search for the optimal schedule. Two properties are suggested in this paper. Note that the two dominance properties are checked at each node of the B&B tree.

The first property, which is given below, specifies the condition that a job should be positioned last at the first stage. The proof is omitted here since it is similar to that of Azizoglu and Kirca [2].

Proposition 1. There exists an optimal schedule in which job w is processed at final position on any one of the machines at first stage if

$$d_w' \geq \frac{1}{M_1}\left\{ \sum_{i=1}^{n} p_{i1} + (M_1 - 1) max_i \{p_{i1}\} \right\}$$

where $d_w' = d_w - p_{w2}$.

The second property specifies the condition that partial schedule $\sigma \bullet i$ is dominated by $\sigma \bullet j$ and job i is tardy job in partial schedule $\sigma \bullet i$, Therefore job i should be positioned last at the first stage. Where $\sigma \bullet i$ is a partial schedule obtained by appending job i to the end of partial schedule σ.

Proposition 2. A partial schedule $\sigma \bullet i$ is dominated by $\sigma \bullet j$ for any partial schedule σ, i.e., job i should be positioned last at the first stage, if $p_{i1} > p_{j1}$ and $\varphi_T + p_{i1} > d_i - p_{i2}$. Here, φ_T is the ready time of the earliest available machine at the first stage in the partial schedule σ.

Proof. Let i and j denote unscheduled jobs with respect to a partial schedule σ at a machine and $c(\sigma \bullet j)_1$ denote the completion time of job j at the first stage in the partial schedule $\sigma \bullet j$. Then, two cases should be considered.

Case1: $c(\sigma \bullet j)_1 > d_j - p_{j2}$

In this case, jobs i and j are both tardy jobs at the first stage since $\varphi_T + p_{i1} > d_i$`. Hence, $NT(\sigma \bullet i) = NT(\sigma \bullet j)$, where $NT(\bullet)$ denotes the number of tardy jobs in schedule $\bullet$.

Case2: $c(\sigma \bullet j)_1 \le d_j - p_{j2}$

In this case, job j may not be a tardy job, i.e., $NT(\sigma \bullet i) \ne NT(\sigma \bullet j)$. Hence, it is better to position job i to the last position since $p_{i1} > p_{j1}$. In other words, the unscheduled jobs can be moved earlier, which results in $NT(\sigma \bullet j) \le NT(\sigma \bullet i)$.

Therefore, the number of tardy jobs for partial schedule $\sigma \bullet j$ is less than that of $\sigma \bullet i$. This completes the proof. ■

4. Computational Experiments

To show the performance of the B&B algorithm suggested in this paper, computational tests were done on randomly generated test problems, and the results are reported in this section. All algorithms were coded in C++ and the test was performed on a workstation with an Intel Xeon processor operating at 3.20 GHz 120 MHz clock speed.

For the test, 960 problems were generated randomly, i.e., 10 problems for each of 96 combinations of the number of machines (1, 2, 3 and 4 at the first stage and 2, 3 and 4 at the second stage), four levels of the number of jobs (10, 12, 14 and 15), and two levels of the due date tightness (loose, tight). The processing times were generated from $DU(10, 40)$, where $DU(a, b)$ is the discrete uniform distribution with range $[a, b]$. Due dates were generated using the method of Gupta (1998). That is, they were generated from $DU(P\alpha, P\beta)$, where $\alpha = \{0.2, 0.4, 0.6, 0.8\}$ and $\beta = \{0.2, 0.4, 0.6, 0.8\}$ with $\beta > \alpha$, and

$$p = \left\{ \begin{array}{l} \left(\sum_{i=1}^{n} p_{i1} / M_1 \right) + \left(\sum_{i=1}^{n} p_{i2} / M_2 \right) \\ + (n-1) \max \left[\sum_{i=1}^{n} p_{i1} / M_1 ; \sum_{i=1}^{n} p_{i2} / M_2 \right] \end{array} \right\} / n$$

Note that the parameters α and β ($\beta > \alpha$) were set to {0.6, 0.8} {0.6, 0.8} for the case of loose due dates and {0.2, 0.4} and {0.2, 0.4} for the case of tight due dates.

Test results on different problem sizes are summarized in Table 1 that shows the number of problems that the B&B algorithm gave the optimal solutions within 5000 seconds and average CPU seconds (in parenthesis). It can be seen from the table that the B&B algorithm gives the optimal solutions for most test problems. However, the computation times increase significantly when the number of jobs increases. Also, the number of machines at each stage plays an important role in problem difficulties. That is, the test problems having relatively large number of machines at the first stage were easier to solve. This is because our dominance properties consider the parallel machines at the first stage.

Table1. Performance of the algorithm

(a) Cases of loose due dates

Number of machines at each stage		Number of jobs			
		10	12	14	15
M_1=1	M_2=2	10(0.5)*	10(8.4)	10(105.3)	9(2269.4)
	M_2=3	10(0.3)	10(10.6)	10(253.1)	10(1006.9)
	M_2=4	10(0.4)	10(14.1)	10(193.7)	9(3014.8)
M_1=2	M_2=2	10(0.3)	10(5.3)	10(85.2)	10(956.4)
	M_2=3	10(0.2)	10(16.2)	10(140.9)	10(1009.1)
	M_2=4	10(0.4)	10(7.4)	10(78.6)	10(2983.4)
M_1=3	M_2=2	10(0.2)	10(6.6)	10(133.5)	10(1096.4)
	M_2=3	10(0.5)	10(8.4)	10(91.0)	10(3089.3)
	M_2=4	10(0.3)	10(18.3)	10(156.2)	10(2040.6)
M_1=4	M_2=2	10(0.2)	10(8.2)	10(163.1)	10(563.4)
	M_2=3	10(0.3)	10(3.2)	10(289.4)	10(1902.3)
	M_2=4	10(0.2)	10(5.1)	10(170.4)	10(2634.5)

* number of problems that the B&B gave the optimal solutions out of 10 problems and CPU seconds (in parenthesis)

(b) Cases of tight due dates

Number of machines at each stage		Number of jobs			
		10	12	14	15
M_1=1	M_2=2	10(2.8)	10(12.3)	10(585.3)	8(2625.3)
	M_2=3	10(1.2)	10(10.6)	10(725.3)	8(3025.1)
	M_2=4	10(1.1)	10(15.7)	9(663.4)	7(1005.6)
M_1=2	M_2=2	10(0.5)	10(6.3)	10(383.3)	9(2006.4)
	M_2=3	10(0.9)	10(15.8)	10(425.6)	10(3523.2)
	M_2=4	10(0.7)	10(13.8)	10(528.9)	9(4019.3)
M_1=3	M_2=2	10(0.6)	10(9.5)	10(631.5)	10(1263.4)
	M_2=3	10(2.0)	10(3.1)	10(226.4)	10(1991.6)
	M_2=4	10(1.5)	10(15.3)	10(341.6)	9(2536.1)
M_1=4	M_2=2	10(0.9)	10(6.9)	10(163.8)	10(10949.2)
	M_2=3	10(1.6)	10(17.5)	10(512.3)	10(1697.2)
	M_2=4	10(1.3)	10(8.6)	10(226.7)	10(2463.8)

5. Concluding Remarks

The paper considered a two-stage hybrid flow shop scheduling problem for the objective of minimizing the number of tardy jobs, and suggested a branch and bound algorithm that can give the optimal solutions. The methods to calculate lower and upper bounds are suggested, and two properties that characterize the

optimal solutions were also suggested to reduce the search space. Test results of computational experiments showed that the B&B algorithm suggested in this paper gave the optimal solutions for moderate-sized problems within a reasonable amount of computation time.

This research can be extended in several directions. First, it is needed to develop more efficient algorithms to solve practical-sized problems. To do this, it may be necessary to develop heuristic algorithms, rather than the optimal algorithm Second, to make the research more practical, the problem should be extended to the case of general hybrid flowshops with more than two stages. In this case, the simulation study may be more applicable. Finally, the systems with uniform or unrelated parallel machines at each stage can be a practical extension.

Acknowledgements

This research was supported by Korea Research Foundation Grant funded by Korean Government (MOEHRD) (KRF-2005-041-D00893). This grant is gratefully acknowledged.

References

1. Azizoglu, M., Cakmak, E. and Kondakci, S., 2001, A flexible flow shop problem with total flow time minimization. *European Journal of Operational Research* 132, 528-538.
2. Azizoglu, M. and Kirca, O., 1998, Tardiness minimization on parallel machines. *International Journal of Production Economics* 55, 163-168.
3. Brah, S. A. and Hunsucker, J. L., 1991, Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research* 51, 88-99.
4. Chen, B., 1995, Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at on stage. *Journal of the Operational Research Society* 46, 231-244.
5. Fouad, R., Abdelhakim, A. and Salah, E. E., 1998, A hybrid three-stage flowshop problem Efficient heuristics to minimize makespan. *European Journal of Operational Research* 109, 321-329.
6. Garey, M. R. and Johnson, D. S., 1979, A Guide to the Theory of NP-Completeness. *Computers and Intractability.*
7. Guinet, A. G. P. and Solomon M. M., 1996, Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research* 34, 1643-1654.
8. Gupta, J. N. D. and Tunc, E. A., 1991, Scheduling for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research* 29, 1480-1502.
9. Gupta, J. N. D. and Tunc, E. A., 1998, Minimizing tardy jobs in a two-stage hybrid flowshop. *International Journal of Production Research* 36, 2397-2417.
10. Ho, J. C. and Chang Y-L., 1995, Minimizing the number of tardy jobs for m parallel machines. *European Journal of Operational Research* 84, 343-355.
11. Lee, C. Y. and Vairaktarakis, G. L., 1994, Minimizing makespan in hybrid flow shops. *Operations research letters* 16, 149-158.
12. Lee, G. C. and Kim, Y. D., 2004, A branch-and-bound algorithm for a two-stage hybrid flow shop scheduling problem minimizing total tardiness. *International Journal of Production Research* 42, 4731-4743.
13. Lee, G. C., Kim, Y. D. and Choi, S. W., 2004, Bottleneck-focused scheduling for a hybrid flow shop. *International Journal of Production Research*42, 165-181.
14. Linn, R. and Zhang, W., 1999, Hybrid flow shop scheduling, *Computers ad Industrial Engineering*, 37, 57-61.
15. Moore, J. M., 1968, An n-job, one-machine sequencing algorithm for minimizing the number of late jobs. *Management Science* 15, 102-109.
16. Mourisli, O. and Pochet, Y., 2000, A branch-and-bound algorithm for the hybrid flow shop. *International Journal of Production Economics* 64, 113-125.
17. Tsubone, H., Ohba, M. and Uetake, T., 1996, The impact of lot sizing and sequencing on manufacturing performance in a two-stage hybrid flow shop. *International Journal of Production Research* 34, 3037-3053.