Scientia Iranica, Vol. 13, No. 1, pp 1–13
© Sharif University of Technology, January 2006

A Heuristic Approach Based on Tabu Search for Early/Tardy Flexible Job Shop Problems

N. Imanipour[1] and S.H. Zegordi[*]

This paper addresses minimizing Total Weighted Earliness/Tardiness (TWET) of jobs in a Flexible Job Shop (FJS) problem. The FJS problem is an extension of the classical Job Shop (JS) problem that implies each operation may be assigned to alternative available machines. So, a job may have alternative routing. The FJS problem with a TWET criterion is modeled as a mixed integer programming. The model is proven to be Np-complete. To solve the model, an algorithm, based on a Tabu Search approach (TS), is developed. The proposed algorithm employs TS to find the best routing of each job and a backward procedure to operations scheduling. Two neighboring functions are designed and their effect is investigated on the performance. The numerical experiments show the suggested algorithm efficiently solves the model in a reasonable CPU time.

INTRODUCTION

The FJS is an extension of the classical Job Shop (JS) problem, which is one of the most important and difficult problems in scheduling theory. In classical JS, each job has a known and fixed route. In the FJS problem, there is a type of flexibility called "routing flexibility", which means a job can be completed through alternative routes because each operation of a job can be processed by alternative machines. In fact, the FJS problem consists of two sub-problems: "Routing" and "Scheduling". These two sub-problems may be solved simultaneously or on a hierarchical basis. In other words, routing and scheduling decisions are made concurrently or sequentially. Some research has studied the FJS problem. The hierarchical approach of Brandimarte [1] that minimizes makespan, was the first research done in FJS. Chambers and Barnes [2,3] suggested a tabu search in order to minimize makespan in FJS. Chen et al. [4] developed a genetic algorithm for FJS to minimize makespan. Jansen et al. [5] designed a polynomial time approximation scheme for the non-preemptive version of an FJS problem with delivery times. They also studied a preemptive version of the FJS problem. The objective function in their research is minimizing makespan. Baykasohlu [6] proposed an approach in which linguistic terms are incorporated for the modeling of FJS scheduling with a process plan selection problem. The objective function is minimizing makespan or maximum tardiness. The FJS problem was studied by Kacem et al. [7] with a view to multiple objectives. They presented a heuristic, which is a hybrid evolutionary algorithm and fuzzy logic, for finding pareto optimal solutions. Recently, Kim et al. [8] presented a symbiotic evolutionary algorithm for minimizing makespan in a JS problem with three types of flexibility in the process planning.

All this research studied regular criteria as makespan. None of them have addressed JIT criteria such as TWET. The TWET is an on-time delivery measure, which controls both tardiness and earliness in compliance with JIT philosophy. In competitive markets, neither tardy nor early are desirable. Our investigation shows that most of the research on earliness/tardiness criteria has been focused on a single machine environment. In the present research, the FJS problem with TWET criterion is modeled. To solve the model, a heuristic approach is developed. The suggested algorithm is a combination of TS to solve the routing sub-problem and another heuristic, called the backward procedure, to solve the scheduling sub-problem.

The remainder of this paper is organized as follows. In the next section, the mathematical model is defined and the computational complexity is discussed. Then, the terminology and structure of the proposed

1. *Faculty of Management, University of Tehran, Tehran, I.R. Iran.*
*. *Corresponding Author, Department of Industrial Engineering, Tarbiat Modarres University, Tehran, I.R. Iran.*

algorithm are discussed. After that, the design of numerical experiments and the computational results are, respectively, reported and, finally, the conclusion is presented.

MATHEMATICAL MODEL

The FJS problem with TWET criterion is modeled as a mixed integer programming. The model is based on the model developed by Baker [9] for job shop scheduling. According to the Baker model, the route of each job is fixed and known. While, in the FJS problem, each job may have alternative routing, in the proposed model, a 0/1 variable is defined and added to solve the routing sub-problem. As a result, the model becomes non-linear.

Assumptions

1. All jobs and machines are available at time $t = 0$;
2. Each operation of a job can be performed on alternative machines;
3. The processing time of each operation of each job is known and deterministic;
4. Each job has a distinct and specific due date;
5. The earliness and tardiness weight of each job is known and deterministic;
6. Preemption is not allowed;
7. Transportation time between machines is negligible;
8. There is only one machine of each type.

Notations

Data

N	number of jobs;
n_j	number of operations of job j;
m	number of machines;
d_j	due date of job j;
α_j	earliness weight of job j;
β_j	tardiness weight of job j;
P_{ijk}	processing time required for kth operation of job j on machine i;
M_{jk}	set of alternative machines for kth operation of job j;
M	a very large positive value,
j	job index;
k	operation index;
i	machine index;
a_{jik}	$= \begin{cases} 1 & \text{if } m_i \in M_{jk} \\ 0 & \text{otherwise} \end{cases}$

Variables

f_{ijk}	completion time of kth operation of job j on machine i;
c_j	completion time of last operation of job j;
y_{ijk}	$\begin{cases} 1 & \text{if } k\text{th operation of job } j \text{ is performed on machine } i \\ 0 & \text{otherwise} \end{cases}$
X_{ijkph}	$\begin{cases} 1 & \text{if } k\text{th operation of job } j \text{ precedes } h\text{th operation of job } p \text{ on machine } i \\ 0 & \text{otherwise} \end{cases}$
S_j	appropriate change variable to job j.

Model

$$\min \sum_{j=1}^{N} [\max[\alpha_j(d_j \quad c_j), \beta_j(c_j \quad d_j)]] = \sum_{j=1}^{N} S_j. \quad (1)$$

Subject to:

$$f_{ij(k+1)} \quad P_{ij(k+1)} + M(1 \quad a_{ij(k+1)} y_{ij(k+1)}) \geq f_{ljk}$$

$$j = 1, \cdots, N; \qquad k = 1, \cdots, n_j \quad 1;$$

$$i, l = 1, \cdots, m, \qquad (2)$$

$$f_{ijk} \quad f_{iph} + M X_{ijkph} \geq P_{ijk} y_{ijk}$$

$$i = 1, \cdots, m; \qquad j = 1, \cdots, N \quad 1;$$

$$p = j+1, \cdots, N, \qquad K = 1, \cdots, n_j;$$

$$h = 1, \cdots, n_p, \qquad (3)$$

$$f_{iph} \quad f_{ijk} + M(1 \quad X_{ijkph}) \geq P_{iph} y_{iph} \qquad (4)$$

$$f_{ij1} \geq P_{ij1} y_{ij1};$$

$$j = 1, \cdots, N; \qquad i = 1, \cdots, m, \qquad (5)$$

$$f_{ijk} \leq M y_{ijk},$$

$$j = 1, \cdots, N; \quad k = 1, \cdots, n_j; \quad i = 1, \cdots, m, \qquad (6)$$

$$\sum_{i=1}^{m} a_{ijk} y_{ijk} = 1$$

$$j = 1, \cdots, N; \qquad k = 1, \cdots, n_j, \qquad (7)$$

$$c_j = \sum_{i=1}^{m} a_{ijn_j} f_{ijn_j}; \quad j = 1, \cdots, N, \qquad (8)$$

$$\alpha_j(d_j \quad c_j) \leq S_j; \quad j = 1, \cdots, N, \qquad (9)$$

$$\beta_j(c_j \quad d_j) \leq S_j; \quad j = 1, \cdots, N. \qquad (10)$$

Relation 1 indicates the objective function that focuses on minimizing the total weighted earliness/tardiness of jobs. As remarked, the objective function (Relation 1) is nonlinear; however, by defining the change variables, S_j, and considering the constraints set of Relations 9 and 10, the objective function becomes linear (second part of Relation 1). The constraint set of Relation 2 demonstrates that the $(k+1)$th operation of job j on machine i can only start after the kth operation of the same job is completed. The constraints sets of Relations 3 and 4 ensure that any two operations are not processed by the same machine in a single time. The constraint set of Relation 5 guarantees that if the 1st operation of job j is assigned to machine i, its completion time must be equal to, or greater than, its processing time. The constraint set of Relation 6 states that if machine i is not assigned to the kth operation of job j, the completion time of it on machine i must be considered 0. The constraint set of Equation 7 ensures that each operation of each job is assigned to only one machine among the alternative machines. Finally, the constraint set of Equation 8 determines the completion time of the jobs.

Computational Complexity

As mentioned above, the FJS problem could be divided to two sub-problems: "Routing" and "Scheduling". The number of the job's routing combinations is determined by the following relation:

$$\prod_{j=1}^{N}\prod_{k=1}^{n_j}|M_{jk}|. \tag{11}$$

Each of the above combinations converts the FJS to a JS problem. Considering the JS problem is known as Np-complete [10], the FJS problem is Np-complete as well. However, the FJS problem with a TWET criterion was formulated in the previous section but is very difficult to solve and takes a long time because of using nonlinear mixed integer programming, even in small instances. (See the computational results in [11].)

ALGORITHM STRUCTURE

The Tabu Search (TS), introduced by Glover [12], is a local search method that can provide optimal or near optimal solutions for combinatorial optimization problems. It has the ability to intensify the search via short-term memory and diversify the search via long-term memory into new regions. The principles of TS have been documented in [13-15]. As a successful method, TS rapidly gained a high position among scheduling problem solving methods. Barnes et al. [16] have done a comprehensive research on applying TS in scheduling problem solving.

The proposed algorithm is a heuristic approach based on the tabu search. Figure 1 indicates its general structure. The algorithm seeks the best routing of jobs in order to improve the objective function. At the beginning, an initial route is randomly generated for each job and this route is assumed as the current route. Then, the schedule maker is recalled and the operations are scheduled. In each iteration, a neighborhood of the current route is generated. A move is defined as a strategy for generating neighborhood. When a move is accepted, it becomes tabu for the next specific iterations. It means that this move is forbidden unless an aspiration level value is satisfied. The aspiration level is a measure for accepting tabu moves. After evaluating all neighbors, the best neighbor (based on the objective function) is selected from among the neighbors that are not tabu, or that satisfy the aspiration level. The selected neighbor (route) replaces the current route. If a lower bound value is reached or other termination criteria are met, the algorithm will be stopped, otherwise the tabu list is updated and this process repeats. Table 1 shows the pseudo code of the proposed algorithm.

To clarify the issue, the basic elements of the algorithm are described by a simple example, which includes 4 jobs and 3 machines. Minimum and maximum operations per job are 3 and 4, respectively. Maximum flexibility (max-flex) is set to 3. This means there are maximum 3 alternative machines for each operation. The data of this problem is stated in Table 2.

Initial Solution (Initial Routing)

A route consists of N sub-strings, any one of which is related to a job and has max-opr length. Initial routing (solution) is generated by allocating each operation to one machine of its possible alternative machines, randomly. An initial routing for the given example

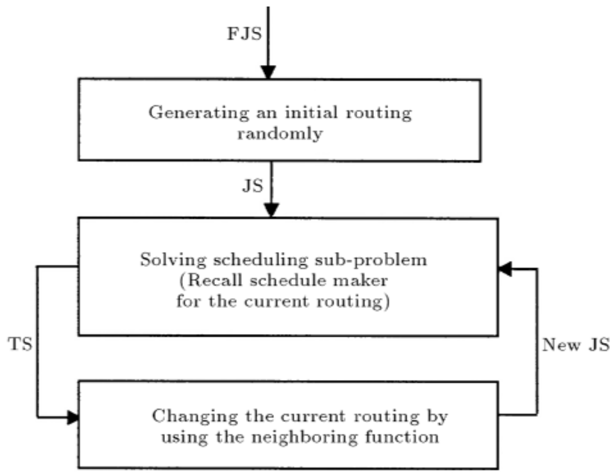
FJS
Generating an initial routing randomly
JS
Solving scheduling sub-problem (Recall schedule maker for the current routing)
TS
New JS
Changing the current routing by using the neighboring function

Figure 1. General structure of the proposed algorithm.

Table 1. Pseudo code of the proposed algorithm.

```
Initialize:
        Randomly generate an initial route: ($R^i$);
        Recall schedule maker for $R^i$ and calculate the objective function for it: ($f(R^i)$);
        Set $R^o = R^c = R^i, f(R^o) = f(R^c) = f(R^i), AL = f(R^i), n$   iter = 0 $TL = \{\}$;
        Set max-iter and $LB$.
      Do{
        Generate neighboring solutions of $R^c$ by defined neighboring function: ($NR^c$).
        For (all neighboring solutions)
            Recall schedule maker.
        Select the best neighboring solution, which is not tabu or satisfies $AL$ and replace $R^c$.
        If (there is an improvement in the objective function) {
          $n$-iter=0
          Update the best solution ($R^o$ and $f(R^o)$).
          }
        Else $n$ - iter = $n$ - iter + 1
        Update $TL$ and $AL$.
      }While ($n$ - iter < max - iter or $f(R^o)$ >LB)
Report the best solution found, which includes the best known routing and the best known schedule.
```

Table 2. Data of a simple example with 4 jobs and 3 machines.

Operation / Job	$k = 1$	$k = 2$	$k = 3$	$k = 4$	d_j	α_j	β_j
$j = 1$	20	21	24	—	103	5	21
$j = 2$	9	13	24	21	97	4	16
$j = 3$	23	21	14	—	83	9	38
$j = 4$	17	18	16	15	123	16	66

could be:

$$R^0 = [(2\ 1\ 1\ 0)(2\ 1\ 2\ 1)(2\ 2\ 3\ 0)(1\ 2\ 1\ 2)].$$

Neighboring Function

Each neighbor of a route is generated by a move. A move is made by changing the machine assigned to a selected operation while others are fixed. As an example, two neighbors of R^0 are:

$$NR_{11} = [(\underline{3}\ 1\ 1\ 0)(2\ 1\ 2\ 1)(2\ 2\ 3\ 0)(1\ 2\ 1\ 2)],$$

$$NR_{21} = [(2\ 1\ 1\ 0)(\underline{1}\ 1\ 2\ 1)(2\ 2\ 3\ 0)(1\ 2\ 1\ 2)].$$

For generating the neighboring set of each route, two methods are used:

$N1$: For each operation whose number of alternative machines is greater than 1 all possible moves are evaluated.

$N2$: For each operation, only one move is done randomly.

Tabu List

Tabu list is an array with fixed and defined length. In the problem solving process, whenever a neighbor solution is selected, its appropriate reverse move will be forbidden and added to the tabu list. For example, if, in a selected move, the 3rd operation of job 4 is allocated from machine M_1 to machine M_3, then, $M_{43} = 1$ will be forbidden and added to the list. This means that in the next iterations, allocating operation 3rd of job 4 to machine M_1 is tabu unless this allocation improves the objective function.

Aspiration Level

At the onset, the aspiration level is set as the initial value of the objective function. At the end of each iteration, whenever an improvement in the objective function is achieved, the aspiration level will be updated and will be considered as the best value obtained for the objective function.

Termination Criteria

The number of sequential iterations without improvement in the objective function is used as a termination criterion. Another criterion is to achieve the optimal solution or a lower bound. The optimal solution of the test problems is unknown. Considering the test problems are randomly generated, then, in some cases, the total process time of a job may be larger than its due date, hence, this job will definitely face unavoidable tardiness that is determined as:

$$\mathrm{MT}_j = \begin{cases} \sum_{k=1}^{n_j} P_{jk} \quad d_j & \text{if } \sum_{k=1}^{n_j} P_{jk} > d_j \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

Based on Equation 12, the lower bound of TWET could be determined by the following equation:

$$\mathrm{LB} = \sum_{j=1}^{n} \beta_j \times \mathrm{MT}_j. \tag{13}$$

It is obvious that this lower bound just covers the tardiness part of the objective function. Since the objective function has two parts, achieving this lower bound may be impossible.

Schedule Maker

The schedule maker is a constructive approach, based on the Giffler and Thompson algorithm [17]. It schedules operations in a "forward" procedure. This means that the operations of each job are scheduled in main order. For example, if job j has 3 operations, these are scheduled 1st, 2nd and 3rd, respectively. Whenever some available operations need the same machine, the schedule maker uses the priority rule in order to select an operation with higher priority.

There are many priority rules in the literature but most of them are generally designed for regular criteria. The TWET is a non-regular criterion. Ow and Morton [18] suggested a priority rule for minimizing TWET criterion in the single machine problem. Their rule was developed, in order to be used in FJS problems. However, the numerical experiments showed the better efficiency of this rule among other selected priority rules, but results were not satisfactory because earliness had the lions share of the objective function. (See computational results in [11].) In this phase, a heuristic procedure called the "backward procedure" was developed . The backward procedure is similar to the forward, but, first, the sequence of operations of each job becomes reverse, which are later scheduled. In order to specify the priority of operations, a priority rule, based on the Shortest Processing Time (SPT), is developed as follows:

$$\text{priority index for operation } O_{jk} : I_{jk} = k \times \frac{\alpha_j + \beta_j}{p_{jk}}. \tag{14}$$

When some operations are ready to process on the same machine, an operation with a higher priority is selected. In the next section, the backward procedure is described in detail.

Backward Procedure

When the objective function is TWET, the ideal schedule is a feasible schedule in which each job is fulfilled on its due date; not early, not tardy. In other words, the completion time of each job is the same as its due date. According to Figure 2, if the largest due date is specified with $d_{\max}$ and job j is completed exactly on its due date, then, the derivation completion time of job j from $d_{\max}$ would be equal to $(d_{\max} \quad d_j)$. The idea of a backward procedure is based on this point. For each job, a virtual ready time is defined by Equation 15, which is:

$$rt_j = \begin{cases} d_{\max} \quad d_j & \text{if } d_j \geq \sum_{k=1}^{n_j} p_{jk} \\ d_{\max} \quad \sum_{k=1}^{n_j} p_{jk} & \text{otherwise} \end{cases}. \tag{15}$$

Then, the operations of each job are scheduled in reverse order. For example, if job j has 3 operations, these operations are scheduled 3rd, 2nd and 1st, respectively. In other words, the kth operation of each job could be scheduled only if the $(k+1)$th operation is scheduled and completed and causes the obtained schedule to become feasible. It should be taken into account that the last operation of each job could not be started earlier than its virtual ready time. After all operations are scheduled, actual scheduling time, interval (T), is calculated as follows:

$$T = \begin{cases} d_{\max} & \text{if } c_{\max} \leq d_{\max} \\ c_{\max} & \text{otherwise} \end{cases}. \tag{16}$$

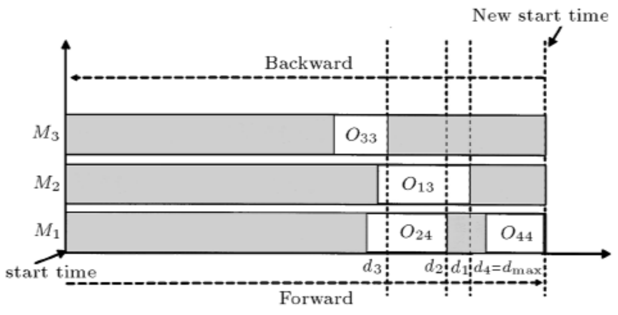

Figure 2. Ideal schedule for a simple example with 4 jobs and 3 machines.

Table 3. Pseudo code of schedule maker with backward procedure.

```
**Initialize:**
    $S = \{O_{jk}/j = 1, \cdots, N;\ k$ is equal to the last operation of each job$\}$;
    Calculate ready time of each job $(rt_j)$,
    Set the earliest start time for all members of $S$: $et_{jk} = rt_j$
    Set the idle time of each machine: $It_i = 0$
    **While** ($S$ is not empty)
    {
        $Q_{jk} = et_{jk} + P_{jk}\ \forall\ O_{jk} \in S$
        Determine $Q^*$ (minimum $Q_{jk}$) and call the assigned machine to it $m^*$;
        Define the set $C$ as: $C = \{O_{jk}/O_{jk} \in S\ \&\ et_{jk} \leq Q^*\ \&\ m_{jk} = m^*\}$
        Calculate the priority index for all member of the set $C$ via relation: $I_{jk} = k \times (\frac{\alpha_j + \beta_j}{P_{jk}})$
        Select the operation with the maximum priority index and call it $O^*_{jk}$.
        Schedule $O^*_{jk}$ at possible earliest time on $m^*$.
        Update the idle time (It) of $m^*$.
        Delete $O^*_{jk}$ from $S$.
        **If** $(K > 1)$
          Add $O_{j(k\quad 1)}$ to $S$;
        Update the earliest start time for the members of $S$: $et_{jk} = \max(Q_{j(k+1)}, It_{m_{jk}})$
    }
    Calculate $C_{\max}$ for the obtained schedule.
    **If** $(C_{\max} <= d_{\max})$ $T = d_{\max}$;
    **Else**                $T = C_{\max}$;
    Calculate the real time interval of each operation: $rst_{jk} = T \quad ft_{jk}$,
                                                         $rft_{jk} = T \quad st_{jk}$.
    Calculate the completion time of each job: $c_j = rtf_{jn_j}$
    Calculate the objective function.
**Report** the obtained schedule and its objective function.
```

et_{jk}: possible earliest time of O_{jk}	rft_{jk}: real finish time of O_{jk}
st_{jk}: start time of O$_{jk}$	c_j: completion time of job j
ft_{jk}: finish time of O_{jk} : $ft_{jk} = st_{jk} + P_{jk}$	It_m: idle time of machine m
rst_{jk}: real start time of O_{jk}	T: scheduling time interval

In this phase, the schedule shifted back as much as T unit. Finally, by calculating the earliness or tardiness of each job, the objective function could be determined. Table 3 describes the pseudo code of the schedule maker with a backward procedure.

To clarify the backward procedure, consider the above-mentioned example. The current route is: [(1, 1, 2, 0) (3, 3, 2, 1) (2, 3, 2, 0) (1, 3, 3, 1)]. According to the pseudo code of Table 3 , at the beginning, the set S is determined as: $S = \{O_{13}, O_{24}, O_{33}, O_{44}\}$ with $\{et_{jk}\} = \{20, 26, 40, 0\}$ and $\{Q_{jk}\} = \{20, 26, 40, 0\}$, thus, $Q^* = 15$ and $m^* = 1$. The set C is equal to $C = \{O_{44}\}$; because the set C has only 1 operation, there is no need for calculating the priority index for the members of set C. Consequently, the O_{44} is selected and scheduled in time interval $[0_{15}]$ on machine M_1. Then, the set S is updated as: $S = \{O_{13}, O_{24}, O_{33}, O_{43}\}$ with $\{et_{jk}\} = \{20, 26, 40, 15\}$. The WHILE loop is repeated until

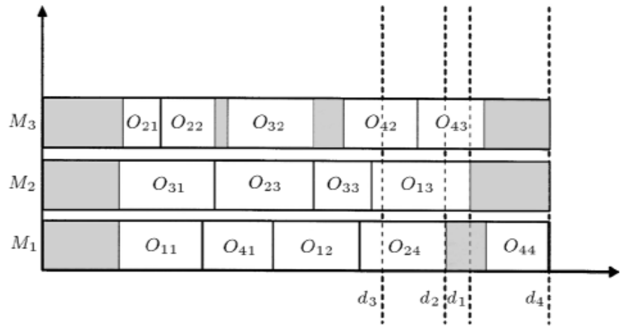

Figure 3. The obtained schedule by backward procedure for a simple example with 4 jobs and 3 machines.

all operations are scheduled. Figure 3 indicates the obtained schedule for this example. In this schedule, job 1, 2 and 4 are completed on time and job 3 has 4/units of earliness. The TWET is equal to 36.

NUMERICAL EXPERIMENTS

Design of Test Problems

The design of the test problems is similar to previous studies for TWET [18,19]. Ow and Morton presented a method, which controls due dates by two main factors: τ and R. τ is the tardiness factor to be a coarse measure of the proportion of jobs that might be expected to be tardy in a given sequence. The other is the due date range, R, by which the range of due date distribution is controlled. If $\overline{P}$ is the average processing time and $\overline{d}$ is the average due date, τ can be determined by the following equation:

$$\tau = 1 \quad \frac{\overline{d}}{N \times \overline{P}}. \tag{17}$$

The process of generating test data is as follows:

- Processing time, $P_{jk} \sim U[5, 25]$;
- Due date, $d_j \sim U[\overline{d} \;\; 1 \quad \frac{R}{2}), \overline{d}(1 + \frac{R}{2})]$; where $\overline{d} = \sum_{j=1}^{N} \sum_{k=1}^{n_j} P_{jk} \times (1 \quad \tau)$;
- Tardiness weight, $\beta_j = \text{WIP} \times (\sum_{k=1}^{n_j} P_{jk}/n_j)$ and earliness weight, $\alpha_j = 0.25\beta_j$, where WIP determines the cost per processing time and is generated from $U \sim [0, 5]$;
- The size of test problems is determined based on the different combinations of five factors: The number of jobs (N), the number of machines (m), minimum (Min-Opr) and maximum (Max-Opr) operations per job and maximum flexibility (Max-Flex). The test problems are generated in seven sizes, according to Table 4. These combinations are taken from [1].

The test data generated for each problem size are classified into four types of combinations between the tardiness factor ($\tau = 0.2, 0.6$) and the due date range ($R = 0.6, 1.6$).

Design of Test Method and Parameter Setting

The algorithm is coded in Borland C language and is implemented on a 486 PC, 100 MHZ. For each size, 5 test problems are randomly generated and for each test problem, 4 due date sets are determined based on 4 combinations of τ and R. Thus, 140 test problems ($7 \times 4 \times 5$) are developed. To avoid the effect of initial solution on results, each test problem is solved using 10 random initial solutions. Based on preliminary experiments, for each test problem, the length of the tabu list and the max-iter are considered equal to $N/2$ and max-flex, respectively. The max-iter is the maximum sequential iteration without improvement in the objective function.

Compared Methods

Studying the FJS problem with TWET has no background in the literature. The effect of forward and backward procedures and the two neighboring functions, $N1$ and $N2$, is investigated on the performance of the proposed algorithm. The performance is measured based on quality and time. The proposed lower bound is used as a base for evaluating the quality of the

Table 4. Characteristics of test problems.

Problem Code	N	M	Min-Opr	Max-Opr	Max-Flex
FJS1	5	5	10	10	3
FJS2	10	5	5	5	3
FJS3	15	4	5	10	2
FJS4	15	8	3	10	3
FJS5	15	8	10	10	5
FJS6	20	5	5	5	3
FJS7	20	10	10	15	3

obtained solutions. Also, the best and the mean value of the objective function and the frequency of the best solution found are selected as quality indexes. The mean CPU time is selected as an index for time.

COMPUTATIONAL RESULTS

In the first phase, the effect of two forward and backward procedures is studied on the performance. The proposed algorithm, with two procedures, is employed to solve a midsize problem (FJS2). According to the results reported in Table 5, the backward procedure is completely superior to the forward. Hence, the next experiments focus on the backward procedure.

In the next phase, the algorithm with a backward procedure is employed to solve all test problems while the neighboring function is set as $N1$ or $N2$. The obtained results are shown in Tables 6 to 12. Considering different combinations between τ and R, these tables compare the best and the mean values of TWET, CPU times and the frequency of the best known solution found. Analysis of these computational results point to the following:

- The proposed algorithm is able to find the good solutions (optimal in some cases) for test problems with different sizes. 1n 70% of trials, the LB value (minimum unavoidable tardiness) is achieved;

Table 5. Comparison of results for FJS2 test problem.

	Prob.	Opt. or	TS with Forward Procedure			TS with Backward Procedure		
		Best Known	Cost (TWET)		Frq.	Cost (TWET)		Frq.
	No.	Cost	Best	Mean	Opt.	Best	Mean	Opt.
	1	0	728	1141	0	0	11	8
	2	0	715	1363	0	0	44	7
$\pi = 0.2$	3	0	964	1264	0	0	141	6
$R = 0.6$	4	0	2157	3884	0	0	76	3
	5	0	1153	2034	0	0	69	6
Total					0			30
	1	0	1296	1556	0	0	9	5
	2	0	946	1003	0	0	0	10
$\pi = 0.2$	3	0	888	924	0	0	1	9
$R = 1.6$	4	0	1075	1252	0	0	0	10
	5	0	3187	3288	0	0	0	10
Total					0			44
	1	0	839	1840	0	0	32	2
	2	0	1799	2555	0	0	120	4
$\pi = 0.6$	3	0	1323	2113	0	0	40	4
$R = 0.6$	4	135	880	1508	0	135	164	4
	5	184	2195	4317	0	184	251	3
Total					0			17
	1	0	1210	1506	0	0	0	10
	2	1140	2396	2712	0	1140	1140	10
$\pi = 0.6$	3	1390	3995	4329	0	1390	1531	2
$R = 1.6$	4	448	2089	2317	0	448	848	6
	5	2336	3104	3309	0	2336	2350	2
Total					0			30

Table 6. Comparison of results for FJS1 test problem (CPU time is in second).

	Prob.	**LB**	**Mean**	**Opt. or**	**TS with $N1$**				**TS with $N2$**			
			Initial	**Best Known**	**Cost (TWET)**		**Frq.**	**CPU**	**Cost (TWET)**		**No.**	**CPU**
	No.		**Cost**	**Cost**	**Best**	**Mean**	**Opt.**	**Time**	**Best**	**Mean**	**Opt.**	**Time**
$\pi = 0.2$ $R = 0.6$	1	0	122	0	0	0	10	0.2	0	0	10	0.2
	2	0	70	0	0	1	9	0.3	0	1	9	0.2
	3	0	69	0	0	7	7	0.9	0	7	7	0.6
	4	0	93	0	0	0	10	0.3	0	0	10	0.2
	5	0	76	0	0	0	10	0.5	0	1	8	0.6
Total, Average							46	0.4			44	0.4
$\pi = 0.2$ $R = 1.6$	1	0	37	0	0	0	10	0.2	0	0	10	0.1
	2	0	81	0	0	0	10	0.2	0	0	10	0.2
	3	0	74	0	0	0	10	0.4	0	0	10	0.3
	4	0	114	0	0	0	10	0.1	0	0	10	0.1
	5	0	100	9	9	9	10	1.5	9	9	10	1.0
Total, Average							50	0.5			50	0.3
$\pi = 0.6$ $R = 0.6$	1	0	109	0	0	15	8	0.5	0	15	8	0.4
	2	0	200	0	0	18	9	0.6	0	18	9	0.5
	3	0	72	0	0	0	10	0.1	0	0	10	0.1
	4	0	207	22	22	59	2	2.2	22	54	3	1.3
	5	0	32	0	0	0	10	0.3	0	0	10	0.4
Total, Average							39	0.7			40	0.5
$\pi = 0.6$ $R = 1.6$	1	0	141	0	0	0	10	0.2	0	4	9	0.3
	2	0	27	0	0	0	10	0.2	0	0	10	0.1
	3	5416	10352	6556	6556	6572	6	2.5	6556	6580	3	1.6
	4	572	1394	572	572	579	9	0.4	572	579	8	0.3
	5	616	10370	1690	1690	1690	2	2.4	1690	3404	3	1.7
Total, Average							37	1.1			33	0.8

Table 7. Comparison of results for FJS2 test problem (CPU time is in second).

	Prob.	**LB**	**Mean**	**Opt. or**	**TS with $N1$**				**TS with $N2$**			
			Initial	**Best Known**	**Cost (TWET)**		**Frq.**	**CPU**	**Cost (TWET)**		**Frq.**	**CPU**
	No.		**Cost**	**Cost**	**Best**	**Mean**	**Opt.**	**Time**	**Best**	**Mean**	**Opt.**	**Time**
$\pi = 0.2$ $R = 0.6$	1	0	188	0	0	11	8	1.7	0	11	8	1.3
	2	0	323	0	0	18	9	2.2	0	18	9	1.7
	3	0	288	0	0	33	7	1.5	0	33	7	1.1
	4	0	288	0	0	10	6	3.1	0	14	5	2.3
	5	0	509	0	0	69	6	2.6	0	69	6	1.7
Total, Average							36	2.2			31	1.6
$\pi = 0.2$ $R = 1.6$	1	0	108	0	0	9	5	2.5	0	10	4	1.8
	2	0	108	0	0	0	10	0.4	0	0	10	0.3
	3	0	36	0	0	11	9	0.5	0	11	9	0.3
	4	0	320	0	0	0	10	1.0	0	0	10	0.7
	5	0	124	0	0	0	10	0.8	0	0	10	0.6
Total, Average							44	1.0			43	0.7
$\pi = 0.6$ $R = 0.6$	1	0	153	0	0	30	2	4.5	0	27	2	3.4
	2	0	546	0	0	122	5	3.2	0	133	4	2.4
	3	0	511	0	0	33	5	3.5	0	30	5	2.6
	4	0	348	0	135	166	4	4.4	135	176	5	3.2
	5	0	632	0	184	215	4	4.7	184	226	4	3.3
Total, Average							20	4.1			20	3.0
$\pi = 0.6$ $R = 1.6$	1	0	549	0	0	0	10	1.5	0	0	10	1.1
	2	1140	1292	1140	1140	1140	10	0.5	1140	1140	10	0.5
	3	1344	1950	1390	1390	1489	3	3.4	1390	1531	2	2.6
	4	448	2181	448	448	743	7	1.7	448	848	6	1.3
	5	2336	2436	2336	2336	2351	1	3.9	2336	2350	2	2.7
Total, Average							31	2.2			30	1.6

Table 8. Comparison of results for FJS3 test problem (CPU time is in second).

	Prob.	LB	Mean	Opt. or	TS with $N1$				TS with $N2$			
	No.		Initial	Best Known	Cost (TWET)		Frq.	CPU	Cost (TWET)		Frq.	CPU
			Cost	Cost	Best	Mean	Opt.	Time	Best	Mean	Opt.	Time
	1	0	334	0	0	12	8	7.4	0	12	8	7.4
	2	0	461	32	32	99	1	18.2	32	99	1	18.2
$\pi = \mathbf{0.2}$	3	0	304	0	0	70	1	13.9	0	70	1	13.9
$R = \mathbf{0.6}$	4	0	455	36	36	80	2	22.0	36	80	2	22.0
	5	0	915	182	182	355	1	20.2	182	355	1	20.2
Total, Average							13	16.3			13	16.3
	1	0	280	0	0	26	7	6.9	0	26	7	6.9
	2	0	44	15	15	15	10	4.3	15	15	10	4.3
$\pi = \mathbf{0.2}$	3	0	27	0	0	0	10	1.9	0	0	10	1.9
$R = \mathbf{1.6}$	4	0	365	0	0	55	2	10.2	0	55	2	10.2
	5	0	243	52	52	52	10	10.7	52	52	10	10.7
Total, Average							39	6.8			39	6.8
	1	0	1302	312	312	418	1	28.0	312	418	1	28.0
	2	0	1236	247	247	321	1	29.0	247	321	1	29.0
$\pi = \mathbf{0.6}$	3	0	2200	588	588	707	1	21.5	588	707	1	21.5
$R = \mathbf{0.6}$	4	0	886	129	129	259	1	21.5	129	259	1	21.5
	5	0	1109	173	173	285	1	33.4	173	285	1	33.4
Total, Average							5	26.7			5	26.7
	1	0	437	225	225	227	8	15.6	225	227	8	15.6
	2	0	192	0	0	5	9	5.0	0	5	9	5.0
$\pi = \mathbf{0.6}$	3	0	262	0	0	0	10	6.5	0	0	10	6.5
$R = \mathbf{1.6}$	4	0	1007	81	81	286	1	18.9	81	286	1	18.9
	5	0	559	194	194	256	4	11.3	194	256	4	11.3
Total, Average							32	11.5			32	11.5

Table 9. Comparison of results for FJS4 test problem (CPU time is in second).

	Prob.	LB	Mean	Opt. or	TS with $N1$				TS with $N2$			
	No.		Initial	Best Known	Cost (TWET)		Frq.	CPU	Cost (TWET)		Frq.	CPU
			Cost	Cost	Best	Mean	Opt.	Time	Best	Mean	Opt.	Time
	1	0	286	0	0	6	4	16.7	0	3	6	11.2
	2	0	383	0	0	30	9	13.3	0	30	9	9.8
$\pi = \mathbf{0.2}$	3	0	290	0	0	0	10	5.2	0	0	10	3.6
$R = \mathbf{0.6}$	4	0	343	91	91	97	9	18.5	91	97	9	13.9
	5	0	164	0	0	0	10	3.5	0	0	10	2.4
Total, Average							42	11.4			44	8.2
	1	0	126	0	0	0	10	1.1	0	0	10	0.8
	2	0	100	0	0	0	10	0.9	0	0	10	0.6
$\pi = \mathbf{0.2}$	3	0	64	0	0	0	10	2.4	0	0	10	1.7
$R = \mathbf{1.6}$	4	0	224	0	0	0	10	6.3	0	0	10	4.4
	5	0	44	0	0	0	10	1.6	0	0	10	1.3
Total, Average							50	2.5			50	1.8
	1	0	192	0	0	9	7	11.4	0	3	8	8.2
	2	0	685	0	0	79	1	21.1	0	64	1	15.5
$\pi = \mathbf{0.6}$	3	0	666	0	0	28	3	29.7	0	16	5	20.5
$R = \mathbf{0.6}$	4	0	422	21	21	84	1	24.9	21	86	1	16.0
	5	0	689	0	0	73	2	30.4	0	73	3	24.4
Total, Average							14	23.5			18	16.9
	1	0	165	14	14	17	9	14.5	14	17	9	9.7
	2	0	213	0	0	1	9	5.9	0	1	9	3.9
$\pi = \mathbf{0.6}$	3	0	142	0	0	7	9	7.2	0	7	9	5.1
$R = \mathbf{1.6}$	4	0	237	0	0	0	10	4.1	0	0	10	3.3
	5	0	316	78	78	78	10	12.0	78	78	10	7.9
Total, Average							47	8.7			47	6.0

Table 10. Comparison of results for FJS5 test problem (CPU time is in second).

			Mean	**Opt. or**	**TS with $N1$**				**TS with $N2$**			
	Prob.	**LB**	**Initial**	**Best Known**	**Cost (TWET)**		**Frq.**	**CPU**	**Cost (TWET)**		**Frq.**	**CPU**
	No.		**Cost**	**Cost**	**Best**	**Mean**	**Opt.**	**Time**	**Best**	**Mean**	**Opt.**	**Time**
	1	0	413	0	0	24	3	127.0	0	26	3	49.3
	2	0	310	0	0	37	6	66.7	0	26	7	28.5
$\boldsymbol{\pi = 0.2}$	3	0	301	0	0	9	8	55.6	0	9	8	27.1
$\boldsymbol{R = 0.6}$	4	0	351	0	0	50	4	82.3	0	53	4	36.8
	5	0	174	0	0	0	10	23.9	0	0	10	10.3
Total, Average							31	71.1			29	30.4
	1	0	43	0	0	0	10	3.4	0	0	10	1.4
	2	0	83	0	0	0	10	6.7	0	0	10	2.7
$\boldsymbol{\pi = 0.2}$	3	0	187	0	0	1	9	38.5	0	1	9	16.7
$\boldsymbol{R = 1.6}$	4	0	133	0	0	0	10	5.9	0	0	10	3.1
	5	0	87	0	0	0	10	9.7	0	0	10	4.1
Total, Average							49	12.8			49	5.6
	1	0	596	0	0	15	8	84.5	0	11	8	35.8
	2	0	570	0	0	59	3	169.8	0	53	3	67.9
$\boldsymbol{\pi = 0.6}$	3	0	578	0	0	39	5	136.8	0	64	5	71.5
$\boldsymbol{R = 0.6}$	4	0	705	0	0	22	8	102.1	0	39	5	55.3
	5	0	519	0	0	51	3	109.0	0	48	4	49.1
Total, Average							27	120.0			25	56.0
	1	0	107	0	0	0	10	28.2	0	0	10	13.3
	2	0	172	0	0	0	10	27.5	0	0	9	16.4
$\boldsymbol{\pi = 0.6}$	3	0	150	0	0	7	8	45.6	0	7	8	18.9
$\boldsymbol{R = 1.6}$	4	0	168	0	0	1	9	42.1	0	0	10	12.3
	5	0	156	0	0	0	10	17.7	0	0	10	7.0
Total, Average							47	32.2			47	13.6

Table 11. Comparison of results for FJS6 test problem (CPU time is in second).

			Mean	**Opt. or**	**TS with $N1$**				**TS with $N2$**			
	Prob.	**LB**	**Initial**	**Best Known**	**Cost (TWET)**		**Frq.**	**CPU**	**Cost (TWET)**		**Frq.**	**CPU**
	No.		**Cost**	**Cost**	**Best**	**Mean**	**Opt.**	**Time**	**Best**	**Mean**	**Opt.**	**Time**
	1	0	596	215	215	230	4	47.4	215	240	2	31.0
	2	0	753	0	0	58	6	40.0	0	58	6	26.1
$\boldsymbol{\pi = 0.2}$	3	0	404	0	0	13	6	32.4	0	18	8	19.2
$\boldsymbol{R = 0.6}$	4	0	386	0	0	40	3	32.8	0	38	8	22.5
	5	0	606	112	112	116	8	43.8	112	143	5	28.7
Total, Average							27	39.3			29	25.5
	1	0	227	0	0	0	10	13.6	0	0	10	9.2
	2	0	193	0	0	0	10	8.0	0	0	10	5.7
$\boldsymbol{\pi = 0.2}$	3	0	300	0	0	21	8	11.4	0	21	8	9.5
$\boldsymbol{R = 1.6}$	4	0	235	26	26	26	10	24.7	26	26	10	16.5
	5	0	374	126	126	167	6	18.5	126	167	6	14.1
Total, Average							44	15.2			44	11.0
	1	0	1570	28	28	303	1	76.7	71	280	0	53.1
	2	0	985	68	68	239	1	75.9	99	298	0	46.0
$\boldsymbol{\pi = 0.6}$	3	0	1099	47	47	162	1	70.5	47	214	1	51.9
$\boldsymbol{R = 0.6}$	4	0	1426	286	286	415	1	55.7	351	430	0	42.5
	5	0	1689	501	505	727	0	67.8	501	741	1	49.3
Total, Average							5	69.3			5	48.6
	1	0	313	20	20	32	5	35.4	20	30	6	24.4
	2	0	678	0	0	13	8	32.7	0	15	8	24.9
$\boldsymbol{\pi = 0.6}$	3	0	680	0	0	36	7	36.7	0	36	7	24.8
$\boldsymbol{R = 0.6}$	4	0	709	9	9	41	4	52.1	9	33	5	34.8
	5	0	332	18	18	19	9	36.3	18	19	9	24.9
Total, Average							33	38.6			35	24.9

Table 12. Comparison of results for FJS7 test problem (CPU time is in second).

	Prob.	LB	Mean	Opt. or	TS with $N1$				TS with $N2$			
			Initial	Best Known	Cost (TWET)		Frq.	CPU	Cost (TWET)		Frq.	CPU
	No.		Cost	Cost	Best	Mean	Opt.	Time	Best	Mean	Opt.	Time
	1	0	286	0	0	0	10	51.7	0	0	10	34.7
	2	0	152	0	0	15	8	177.4	0	15	8	51.1
$\pi = 0.2$	3	0	355	0	0	84	3	173.9	0	82	2	131.3
$R = 0.6$	4	0	264	66	66	66	10	193	66	66	10	132.5
	5	0	402	0	0	29	8	103.3	0	29	8	72.9
Total, Average							39	139.9			38	84.5
	1	0	435	96	96	126	8	130.4	96	188	5	91.8
	2	0	161	15	15	15	10	123.0	15	15	10	83.5
$\pi = 0.2$	3	0	214	0	0	0	10	33.9	0	0	10	22.4
$R = 1.6$	4	0	58	0	0	0	10	24.9	0	0	10	17.0
	5	0	64	0	0	0	10	15.0	0	0	10	10.7
Total, Average							48	65.4			45	45.1
	1	0	771	0	0	79	1	261.6	0	79	1	178.9
	2	0	457	0	0	26	7	166.9	0	7	8	115.1
$\pi = 0.6$	3	0	746	15	15	76	3	307.2	15	89	3	218.0
$R = 0.6$	4	0	620	0	0	64	4	191.1	0	72	2	157.1
	5	0	685	0	0	33	5	248.0	0	23	7	145.3
Total, Average							20	235.0			21	162.9
	1	0	168	0	0	0	10	48.1	0	1	9	36.3
	2	0	76	0	0	0	10	29.9	0	0	10	20.0
$\pi = 0.6$	3	0	152	0	0	3	7	88.9	0	0	8	52.5
$R = 1.6$	4	0	240	0	0	4	9	66.5	0	4	9	44.6
	5	0	83	0	0	0	10	57.1	0	0	10	36.9
Total, Average							46	58.1			46	30.1

- When neighboring function $N2$ is applied, CPU times are significantly decreased. In 90% of the experiments, the algorithm with $N2$ achieved the solutions with the same efficiency as by $N1$, but, in less CPU time. Based on the results, in some cases, $N2$ decreases CPU time up to 50%. The better performance of $N2$ (in terms of quality solution and CPU time) can be explained as follows: when $N2$ is applied, the neighboring set of the current solution is randomly created, thus, the TS rarely fall into local optima;

- For each test problem, comparing the average of the initial value of TWET with that of the best found value indicates that routing flexibility has a significant effect on the objective function. Comparison of the obtained results for FJS4 and FJS5 problems verifies that when more the flexibility increases, the possibility of developing better routing and scheduling becomes higher.

CONCLUSION

In this paper, a generalized job shop problem, called Flexible Job Shop (FJS), is presented. The FJS problem is considered as one of the most difficult combinatorial optimization problems. The objective function is Total Weighted Earliness and Tardiness (TWET). The problem is modeled as a mixed (binary) integer programming. Since the model is ineffective, due to the large number of zero/one variables and nonlinear constraints, a heuristic algorithm is designed, which combines TS with a schedule maker based on priority rules to solve routing and scheduling sub-problems. Two neighboring functions are designed and their effects are investigated on the performance of the proposed algorithm. Also, a backward procedure is newly applied to the FJS problem. The obtained results show the efficiency of the backward procedure, in terms of quality solution and CPU time.

Due date assignment in a FJS problem should be a topic for future research. Also, studying the multi-objective FJS is an interesting area.

REFERENCES

1. Brandimarte, P. "Routing and scheduling a flexible job shop by tabu search", *Annals of Operations Research*, **41**, pp 157-183 (1993).
2. Chambers, J.B. and Barnes, J.W. "Flexible job shop scheduling by tabu search", *Graduate Program in Operations Research and Industrial Engineering*, the University of Texas at Austin, USA, Technical Report, Available from http://www.cs.utexas.edu/users/jbc/ (1996).
3. Chambers, J.B. and Barnes, J.W. "Reactive search for flexible job shop scheduling", *Graduate Program in Operations Research and Industrial Engineering*, the University of Texas at Austin, Technical Report, Available from http://www.cs.utexas.edu/users/jbc/ (1998).
4. Chen, H., Hlow, J., and Lehman, C. "A genetic algorithm for flexible job shop scheduling", *0-7803-5180-0-5/99 IEEE*, pp 1120-1125 (1999).
5. Jansen, K., Mastrolilli, M., and Solis-Oba, R. "Approximation algorithms for flexible job shop problems", *Proceedings of Latin American Theoretical Informatics (LATIN 2000)*, **LNCS 1776**, pp 68-77, (2000).
6. Bakasoghlu, A. "A linguistic-based meta heuristic optimization model for flexible job shop scheduling", *Int. J. of Production Research*, **40**, pp 4523-4543 (2002).
7. Kacem, I., Hammadi, S. and Borne, P. "Pareto-optimality approach for flexible job shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic", *Mathematics and Computers in Simulation*, **60**(3-5), pp 245-276 (2002).
8. Kim, Y.K., Park, K. and Ko, J. "A Symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling", *Computers & Operations Research*, **30**, pp 1151-1171,(2003).
9. Baker, K.R., *Introduction to Sequencing and Scheduling*, John Wiley & Sons (1974).
10. Garey, M.R., Johnson, D.S. and Sethi, R. "The complexity of the flow shop and job shop scheduling", *Mathematics of Operations Research*, **1**, pp 117-129 (1976).
11. Imanipour, N. "Flexible job shop scheduling with earliness and tardiness penalties", MS Project, Modarres University, Tehran, Iran (1999).
12. Glover, F. "Future paths for integer programming and links to artificial intelligence", *Computer and Operations Research*, **13**, pp 533-549 (1986).
13. Glover, F. "Tabu search part I", *ORSA Journal on Computing*, **1**(13), pp 190-206 (1989).
14. Glover, F. "Tabu search part II", *ORSA Journal on Computing*, **2**(1), pp 4-32 (1990).
15. Glover, F. "Tabu search: A tutorial", *Interfaces*, **20**(4), pp 74-94 (1990).
16. Barnes, J.W., Langunan, M. and Glover, F. "Overview of tabu search approach to production scheduling", *Intelligent Scheduling Systems*, Kluwer Academic Publisher (1995).
17. Giffler, B. and Thopmson, G.L. "Algorithms for solving production scheduling problems", *Operations Research*, **8**(4) (1960).
18. Ow, P.S. and Morton, T.E. "The single machine early/tardy problem", *Management Science*, **32**(2) pp 177-191 (1989).
19. Zegordi, S.H., Itoh, K. and Enkawa, T. "A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling", *Int. Journal of Production Research*, **33**(5), pp 1449-1466 (1995).