Journal of Industrial and Systems Engineering
Vol. 1, No. 1, pp 23-36
Spring 2007

JISE

Development of a Set of Algorithms for the Multi-Project Scheduling Problems

Farhad Ghassemi-Tari[1*], Laya Olfat[2]

[1]Department of Industrial Engineering, Sharif University of Technology, Iran
(ghasemi@sharif.edu)
[2] School of Management, Tabtabaei University, Iran
(l_olfat@yahoo.com)

ABSTRACT

In this paper, the problem of determining the best schedule for a set of projects has been modeled in the form of a generalized tardiness flowshop (GTF) problem. We develop a set of heuristic algorithms for minimizing the total tardiness of jobs in a GTF problem. In the generalized version of tardiness flowshop problems, a job is considered to be a collection of operations and there is a due date associated with the completion of each operation on each machine. Four algorithms based on the concept of "apparent tardiness cost" (ATC) are developed for solving the GTF problem. The relative effectiveness of the developed algorithms will then be evaluated through an extensive computational experiment.

Keywords: Generalized tardiness flowshop; Multi-project scheduling; Intermediate due date; Apparent tardiness cost

1. INTRODUCTION

In this paper we consider the problem of determining the best schedule for a set of projects to minimize the total tardiness of the project's deliverable tasks. It is assumed that, the tasks precedence structure in each project is in the form of a "linear precedence structure". The flowshop model is the most appropriate type of model for embodying this type of precedence structure. We therefore model this problem in the form of a tardiness flowshop problem. There is however, a major distinction between the traditional flowshop models with the model we are proposing for our multi-project scheduling problem. In traditional flowshop models with tardiness criterion, there is a single due date for each job or more precisely for the last operation of each job. In our project scheduling problem, however, there are several deliverable tasks for each project and a due date is assigned for each deliverable task. Therefore we are facing a tardiness flowshop model with intermediate due dates which we call the "generalized tardiness flowshop" (GTF) model. This nomination is due to the fact that, if we assign large values to the intermediate due dates, we obtain a traditional model as a special version of our proposed generalized tardiness flowshop model.

While considerable research has been devoted to the problem of minimizing the makespan, very little work is reported on minimizing the total tardiness of jobs on a permutation flowshop (Ghassemi and Olfat, 2004). Even in those research efforts considering different variants of the total

*Corresponding Author

tardiness measures none has been found to address the problem of scheduling jobs in a flowshop system with intermediate due dates.

Although a limited number of studies have addressed the tardiness criterion in flowshop scheduling problems, there are some valuable concepts in tardiness criterion of single machine, and job shop sequencing models which can be modified and used in flowshop problems (Baker and Bertrand, 1982). Among these studies is the earlier work of Carroll (1965) who presented a dynamic rule called cost over time (COVERT) for a single machine and job shop sequencing problems to minimize the total weighted tardiness of jobs. Under this rule the degree of criticality of jobs were used for determining the jobs sequence. Rachamaduagu and Morton (1982) introduced a rule called apparent tardiness cost (ATC) for the single machine sequencing with the total weighted tardiness criterion. Later Rachamaduagu (1984) conducted an experiment for comparing the ATC rule with COVERT. In this experiment he applied both rules to the single machine, parallel machine and flowshop sequencing problems, and concluded that ATC is a more efficient rule than COVERT in obtaining the solution. Vepsalainen and Morton (1987) reached the same conclusion when they used both rules, as well as some other rules in solving job shop sequencing problems. They considered job shop problems with tight due dates, loose due dates and different workloads and concluded that ATC performed better in all these cases and COVERT had the second best performance among all the rules the employed.

Baker (1984) conducted a research to evaluate different sequencing rules in job shop scheduling problems. He classified several rules in three groups, namely, slack based, critical ratio based, and the rule based on the time gap between completion of partially scheduled jobs and their associated due dates. He then evaluated different rules for each group. As the result he concluded the slack based rules had a lower performance, and among the rules in other two scheduling groups, the operation-oriented rules perform better than job-oriented rules. Based on the concepts of ATC, COVERT, and the critical ratio indices Olfat (1998) developed several heuristic rules for flowshop tardiness problems. In the same work, she also developed a set of rules, called ranking rules, by which a schedule is developed for each machine and jobs are ranked according to their positions in a sequence. The total rank of the jobs is then used to determine the final permutation schedules. She compared the developed rules through conducting an extensive computational experiment.

Later, Ghassemi-Tari and Olfat (2004) proposed two COVERT based algorithms for solving the generalized tardiness flowshop model. They conducted an extensive computational experiment for comparing the efficiency of their developed algorithms. Sapar and Henry (1996) evaluated six scheduling rules for two machine flowshop problems with the average tardiness criterion. Among the rules they considered, the modified due date rule (MDD) was reported to be superior under all variants of their experimental conditions. There have been some other related studies considering tardiness criterion, among which one can cite the works conducted by Lin and Zhang (1999) and Lee (2001).

There have also been some limited research efforts considering special versions of flow shop models. A hybrid flowshop scheduling model is one special type of the flowshop models. In a hybrid flowshop, one considers a shop in which there are serial stages, each with identical parallel machines. Lee and Kim (2004) suggested a branch and bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing the total tardiness. In their study a two-stage hybrid flowshop scheduling problem is considered with the objective of minimizing the total tardiness of jobs.

In another research attempt, Choi et al, (2005) considered the total tardiness scheduling problem for the hybrid flowshop in which a set of orders with reentrant lots were scheduled. In their model, each order is composed of multiple lots with the same due date, and each lot can be processed on any one of parallel machines at each stage. In addition, they considered a situation in which lots of certain orders have to visit some stages twice. They proposed a set of heuristic algorithms with some supporting computational experiments, to reveal that the suggested algorithms perform better than the well-known dispatching rules for various scheduling problems. There are some other works dealing with the tardiness hybrid flowshop, such as the works conducted by Janiak et al, (2005), Lin and Liao (2003), and Gupta et al, (2002).

The bi-criteria flowshop model is another type of special flowshop models. Allahverdi (2004) proposed a heuristic algorithm for the flowshop scheduling problem with bi-criteria of makespan and tardiness. A weighted sum of makespan and maximum tardiness are used as the objective to be minimized. In his paper, two types of the problem were addressed. In one a limit was assigned to the maximum tardiness while in the second type this constraint was relaxed. A new heuristic was proposed and compared to two existing heuristics. A computational experiment was performed and the results indicated that the proposed heuristic was much better than the existing ones.

In a similar study, Parthasarathy and Rajendran (1998) proposed a heuristic for the problem of job scheduling in flowshop and flowline-based manufacturing cells with the bi-criteria of minimizing mean makespan and mean tardiness of jobs. They developed a heuristic algorithm, based on the simulated annealing technique. They then evaluated their proposed algorithm against the existing heuristics and showed that the proposed algorithm performed better than the existing heuristics.
Later Chakravarthy and Rajendran (1999) developed the bi-criteria of makespan and maximum tardiness minimization heuristic for a flowshop model. Their developed heuristic was also based on the simulated annealing technique. They performed a computational experiment for evaluating the proposed heuristic against the existing heuristic and the computational evaluation revealed that the proposed heuristic performed better than the existing ones.

Some other tardiness scheduling problems have considered batch processing of jobs; the related works have been presented by (Yeh and Allahverdi, 2004; Etiler et al, 2004; Bilge et al, 2004).

Mosheiov (2003) considered the problem of batch processing in which two elements of “job identical processing times” and “job-dependent weights” were combined in an *m*-machine flow shop scheduling model. A common due date was considered for the jobs. He considered the (just in time) objective of minimizing the maximum earliness/tardiness cost. He then introduced a polynomial time solution approach for the proposed problem.

Thorough review of literature has shown that no single research effort, considering the flowshop tardiness problem with intermediate due dates, has been reported to date.

In this paper four heuristic algorithms based on the concept of Apparent Tardiness Cost (ATC) are developed for tardiness minimization of flowshop scheduling problems with intermediate due dates. We then evaluate the effectiveness of the four proposed algorithms using an extensive experimental study.

2. NOMENCLATURE

The following notations are used throughout this manuscript:
n: Number of jobs

m: Number of operations or machines
p_{ij}: Processing time for operation j of job i.
d_{ij}; Due date for operation j of job i.
r_{ij}: The total processing time of job i on machines $1,2,\ldots$, and $j-1$.
t_{ij}: The completion time of the j^{th} operation of the job which is sequenced directly before job i on machine j.
C_{ij}: Completion time of job i on machine j.
TT: Total tardiness of a schedule
$TT(S_j)$: Total tardiness of schedule S_j
TT_j: Total tardiness of a schedule of all jobs on machine j.
$T_k(S)$: Tardiness of job k in schedule S
A_j: An ordered set of partially scheduled jobs on machine j.
B: A set of unscheduled jobs (complement of A_j).
S_j: A permutation schedule, in which jobs are sequenced according to the order of jobs determined on machine j.

3. THE MODEL

Considering n projects as n jobs, m activities in each project as m operations, and m research teams as m machines we can build a flow shop model for our multi-project scheduling problem. We assume that activities in each project have a special precedence structure. In particular, each activity after the first has exactly one direct predecessor and each activity before the last has exactly one direct successor. Figure 1 illustrates a pure flow- shop model for a multi-project scheduling problem when the precedence of the project tasks is structured according to a linear precedence structure or equivalently as a flowshop model. The pure flowshop consists of exactly m different machines and each job consists of exactly m operations, each of which requires a different machine.

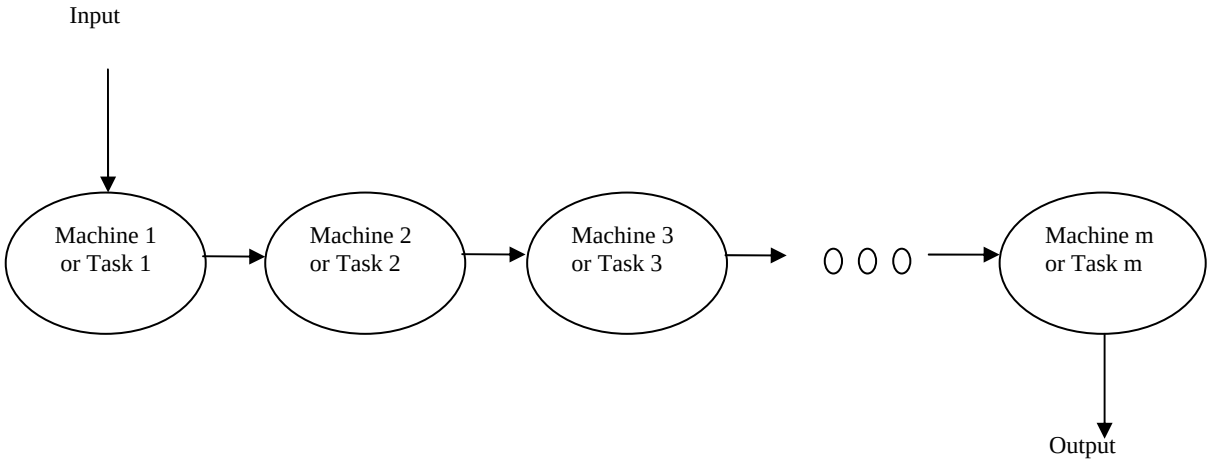

Figure 1. Project tasks flow in a pure flowshop

In the general case, jobs may require fewer than m operations, or their operations may not always require adjacent machines, or the initial and/or final operations may not always occur at machine 1 and m. Nevertheless, the flow of work is still unidirectional and we can represent the general case as a pure flowshop in which some of the operation times are zero. Figure 2 represents the flow of the project tasks in a more general flowshop model.

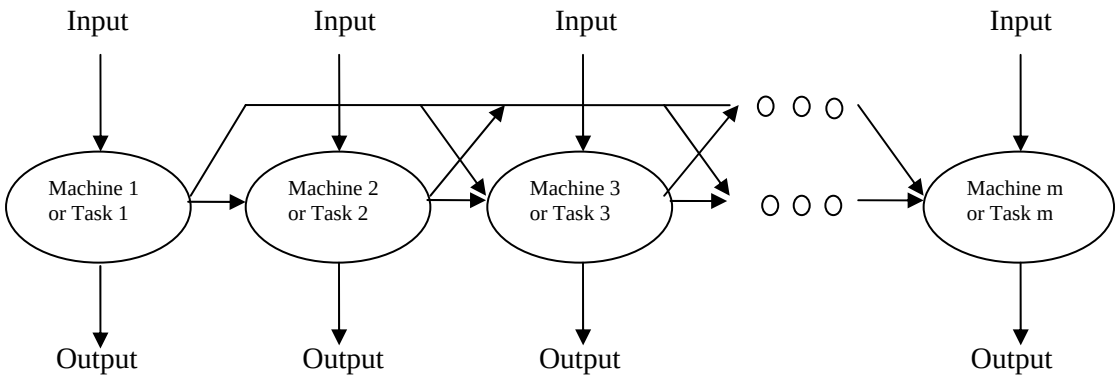

Figure 2. Project tasks flow in a general flowshop

Thus each project requires a specific sequence of activities to be carried out for the project to be completed, and there is a due date associated with the completion of each activity. From now on we refer to each project as a job each activity as an operation, and each research team or department as a machine. Therefore we have a flowshop model with *n* jobs, each having *m* operations and *m* machines for performing *m* operations of each job. The flowshop is characterized by a flow of work that is unidirectional. In other words, a flowshop model consists of a natural machine order and hence the operation *j* of job *i* is performed on machine *j*. It should be noted that some of jobs may require fewer than *m* operations, that their operations may not always require adjacent machines in the same order. For these jobs we assume there are also *m* operations where the corresponding processing times for the missing operations can be assumed to have a zero value. We propose our algorithm for the generalized tardiness flow shop model under the following conditions:

- A set of *m*-operations, *n*-jobs is available for processing at the beginning of scheduling time.
- Setup times for the operations are sequence-independent and are included in processing times.
- Operation processing times (p_{ij}) are known and are deterministic.
- There is a corresponding due date for completion of each operation (d_{ij}).
- As long as there is a job, *m*-machines are available for processing the jobs.
- Individual job operations can not be preempted.

As it is common in sequencing we use the bracket symbol for specifying the position of a job in a sequence. Therefore *[k]=i* indicates that job *i* has the k^{th} position in a sequence. We now present the developed algorithm for minimizing the total tardiness of flowshop scheduling problem in which the total tardiness is defined as the summation of the tardiness of the jobs on the last machine as well as the tardiness of the jobs on the intermediate machines.

To present a mathematical programming formulation for the proposed problem, we consider a set of *n* jobs, each having *m* operations to be performed on *m* different machines with a unidirectional precedence structure. In this case we can number machines according to the operation numbers of each job. Then we will have a one to one correspondence in operation numbers and machine numbers. Therefore as in the case of the "pure' flowshop model, the j^{th} operation of each job is performed on the j^{th} machine for all values of *j=1, 2…m.*

We intend to find a permutation schedule to minimize the total tardiness. To do so let x_i^j denote the variable indicating the starting time of the j^{th} operation of job *i*. This problem then can be formulated as a mathematical programming model as follows:

$$\min TT = \sum_{i=1}^{n}\sum_{j=1}^{m}\max(x_i^j + p_{ij} - d_{ij}, 0)$$

$$\text{Subject to } x_l^j - x_k^j \geq p_{kj} \quad \text{for } k << l,\ j = 1,2,..n$$

The difficulty in this formulation arises from the fact that the direct precedence relation *(k<<l)* requires that all permutations of the jobs to be considered in solving this mathematical programming model. Solving this mathematical programming model, for large values of *n* and/or *m* (in a practical size problem), leads to computational complexity due to the exponentially growth of solution space. Therefore the only solution approach for practical cases is the use of a heuristic approach. In the following section we describe the development of the heuristic algorithms for this problem.

4. DEVELOPMENT OF ALGORITHMS

In this section we present the heuristic algorithms we developed for solving flowshop scheduling problems with the total tardiness criterion, while considering a tardiness for each operation of each job completed after its associated due date. The developed algorithms are dynamic procedures by their nature of producing the final schedule. In contrast to the static procedures by which a schedule is produced by *m*-jobs-at-a-time sequencing rules, a dynamic procedure produces a schedule using one job at a time sequencing rules. In the latter case information regarding completion times of the scheduled jobs is used for selecting the next job to be sequenced.

The developed algorithms are based on the concept of the apparent tardiness cost (ATC).
The ATC is an index that has been used for the dynamic sequencing rules in single machine scheduling problems. Based on this index, we develop four sequencing algorithms for solving our flow shop scheduling problem, which we name as *AT1* through *AT4* algorithms.

Algorithm AT1

Step 1. Let *j=1*.
Step 2. Let $t_{ij}=0$, $TT_j=0$, $C_{ij}=0$, $A_j=\phi$, and $B=\{1,2,...,n\}$.
Step 3. Calculate $ATCI(t_{ij})$ as follows:

$$ATCI(t_{ij}) = \frac{1}{p_{ij}}\exp\left(-\frac{\max[0, d_{ij}-(p_{ij}+t_{ij})]}{\frac{1}{n}\sum_{i=1}^{n} p_{ij}}\right)$$

Step 4. Calculate $AT1_i$ as follow:

$$AT1_i = \max_{i\in B}\{ATCI(t_{ij})\}.$$

Step 5. Let *i* be the associated job of $AT1_i$, remove *i* form *B* and place it in the last position of A_j.
Step6. Let $C_{ij} = t_{ij} + p_{ij}$, and $TT_j = TT_j + \max\{0, C_{ij} - d_{ij}\}$. If *B* is empty, go to step 7, otherwise let $t_{ij} = C_{ij}$, and go to step 3.
Step 7. Define the permutation schedule S_j by sequencing jobs according to the order of A_j.

Step 8. Let $TT(S_j)$ be defined as the total tardiness of schedule S_j and calculate the value of $TT(S_j)$. If $j=m$, let $TT(S_k) = \min_j \{TT(S_j)\}$, select schedule S_k as the final schedule and stop. Otherwise let $j=j+1$, and go to step 2.

Algorithm AT2

Step 1. Let $j=1$.
Step 2. Let $t_{ij}=0$, $TT_j=0$, $C_{ij}=0$, $A_j=\phi$, and $B=\{1,2,...,n\}$.
Step 3. Calculate $ATCII(t_{ij})$ as follows:

$$ATCII(t_{ij}) = \frac{1}{d_{ij} p_{ij}} \exp\left(-\frac{\max[0, d_{ij} - (p_{ij} + t_{ij})]}{\frac{1}{n}\sum_{i=1}^{n} p_{ij}}\right)$$

Step 4. Calculate $AT2_i$ as follow:

$$AT2_i = \max_{i \in B} \{ATCII(t_{ij})\}.$$

Step 5. Let i be the associated job of $AT2_i$, remove i form B and place it in the last position of A_j.
Step 6. Let $C_{ij} = t_{ij} + p_{ij}$, and $TT_j = TT_j + \max\{0, C_{ij} - d_{ij}\}$. If B is empty, go to step 7, otherwise let $t_{ij} = C_{ij}$, and go to step 3.
Step 7. Define the permutation schedule S_j by sequencing jobs according to the order of A_j.
Step 8. Let $TT(S_j)$ be defined as the total tardiness of schedule S_j and calculate the value of $TT(S_j)$. If $j=m$, let $TT(S_k) = \min_j \{TT(S_j)\}$, select schedule S_k as the final schedule, then stop. Otherwise let $j=j+1$, and go to step 2.

Algorithm AT3

Step 1. Let $j=1$, and $r_{ij} = 0$ for $i = 1,2,\cdots,n$ and for $j = 1,2,\cdots,m$
Step 2. Let $TT_j=0$, $t_{ij}=0$, $C_{ij}=0$, $A_j=\phi$, and $B=\{1,2,...,n\}$.
Step 3. Calculate $ATCIII(t_{ij})$ as follows:

$$ATCIII(t_{ij}) = \frac{1}{p_{ij}} \exp\left(-\frac{\max[0, d_{ij} - (p_{ij} + \max(t_{ij}, r_{ij}))]}{\frac{1}{n}\sum_{i=1}^{n} p_{ij}}\right)$$

Step 4. Calculate $AT3_i$ as follow:

$$AT3_i = \max_{i \in B} \{ATCIII(t_{ij})\}.$$

Step 5. Let i be the associated job of $AT3_i$, remove i form B and place it in the last position of A_j.
Step 6 Let $C_{ij} = \max\{t_{ij}, r_{ij}\} + p_{ij}$ and $TT_j = TT_j + \max\{0, C_{ij} - d_{ij}\}$. If B is empty, go to step 7, otherwise let $t_{ij} = C_{ij}$, and go to step 3.
Step 7 Define the permutation schedule S_j by sequencing jobs according to the order of A_j.
Step 8. Let $TT(S_j)$ be defined as the total tardiness of schedule S_j and calculate the value of $TT(S_j)$. If $j=m$, let $TT(S_k) = \min_j \{TT(S_j)\}$, select schedule S_k as the final schedule, then stop. Otherwise let $j=j+1$, and $r_{ij} = \sum_{k=1}^{j-1} p_{ik}$, then go to step 2.

Algorithm AT4

Step 1. Let *j=1,* and $r_{ij} = 0$ for $i = 1,2,\cdots,n$ and for $j = 1,2,\cdots,m$

Step 2. Let $TT_j=0$, $t_{ij}=0$, $C_{ij}=0$, $A_j=\phi$, and $B=\{1,2,...,n\}$.

Step 3. Calculate $ATCIV(t_{ij})$ as follows:

$$ATCIV(t_{ij}) = \frac{1}{d_{ij} p_{ij}} \exp\left(-\frac{\max[0, d_{ij} - (p_{ij} + \max(t_{ij}, r_{ij}))]}{\frac{1}{n}\sum_{i=1}^{n} p_{ij}}\right)$$

Step 4. Calculate $AT4_i$ as follow:

$$AT4_i = \max_{i \in RB}\{ATCIV(t_{ij})\}.$$

Step 5. Let *i* be the associated job of $AT4_i$, remove *i* form *B* and place it in the last position of A_j.

Step 6 Let $C_{ij} = \max\{t_{ij}, r_{ij}\} + p_{ij}$ and $TT_j = TT_j + \max\{0, C_{ij} - d_{ij}\}$. If *B* is empty, go to step 7, otherwise let $t_{ij} = C_{ij}$, and go to step 3.

Step 7. Define the permutation schedule S_j by sequencing jobs according to the order of A_j.

Step 8. Let $TT(S_j)$ be defined as the total tardiness of schedule S_j and calculate the value of $TT(S_j)$. If *j=m,* let $TT(S_k) = \min_j\{TT(S_j)\}$, select schedule S_k as the final schedule, then stop. Otherwise let *j=j+1,* and $r_{ij} = \sum_{k=1}^{j-1} p_{ik}$, then go to step 2.

5. COMPUTATIONAL EXPERIMENTS

In this section we implement extensive numerical experiments to evaluate the effectiveness of the proposed heuristic algorithms. To show the effectiveness of the proposed algorithms, computational experiments are conducted with a variety of randomly generated test problems. We first describe the mechanism of generating the test problems, and then we will present the experimental results.

5.1. Generation of the Test Problems

We employed the same concepts of generating the test problems which have been commonly used in the similar experiments. For each test problem with the size of *n* jobs and *m* operations, we used a uniform distribution with the rage [1-10] to generate p_{ij}'s. The operations due dates are generated in two steps. In the fist step the due date of the last operation (d_{im}) is generated using the following uniform distribution:

$d_{im} \sim U\,[(1\text{-}TF\text{-}R/2)\times C,\ (1\text{-}TF\text{+}R/2)\times C]$

where *TF* is the due date tightness parameter, *R* is the range adjusting parameter, and *C* is defined as

$$C = \min_i\{\sum_{j=1}^{m-1} p_{ij}\} + \sum_{i=1}^{n} p_{im}.$$

follows:

Then we use d_{im} to generate all other d_{ij}*'s* as follows:

$$d_{ij} = d_{i,(j+1)} - \alpha_i p_{i,(j+1)} \; ; \; j = m-1, m-2, \ldots, 1$$

where, $\alpha_i = \dfrac{d_{im}}{\sum_{j=1}^{m} p_{ij}}$.

Using the concept of pseudo random number generation and by employing the above mentioned formulas we generated the experimental test problems. By assigning different values to *m*, *n*, and *TF*, we defined several different test problems scenarios. To reduce any bias of the generated data, we defined a sample size (*k*) for the test problems and for each scenario we employed the average value of the sample's solutions for any further experimental analysis. In other word, for each scenario, *k* test problems are generated using different seeds for the random number generation process and by using the average value of the solutions of the test problems we compare the relative effectiveness of the proposed algorithms. The following section summarizes the results.

5.2. Computational Results

We let *R*=0.02, and generated 96 scenarios through the combinations of four different values of *m*, eight different values of *n*, and three different values of *TF*. For each scenario 40 test problems were randomly generated to obtain the p_{ij} and the d_{ij} values. For each scenario, all 40 test-problems were solved by the proposed algorithms, to determine the average of total tardiness. We then selected the algorithm with the minimum average of the total tardiness as the reference algorithm, and then we determined the relative measure of effectiveness of each algorithm, by the deviations of its average solution from the average solution of the reference algorithm.

To define the relative efficiency, we calculated the relative deviation as:

$$RAD_i = \frac{ATT_i - \min_k \{ATT_k\}}{\min_k \{ATT_k\}} \times 100$$

where RAD_i is the relative average deviation of the i^{th} algorithm, and ATT_i is the average total tardiness of the 40 test problems solved by the i^{th} algorithm. We then defined 96 different scenarios by assigning 8 different values to *n (n=5, n=10, n=15, n=20, n=25, n=30, n=35, n=40),* 4 different values to *m (m=5, m=10, m=15, m=20),* and 3 different values to *TF (TF=0.1, TF=0.2, TF=0.4)*. For each scenario 40 test problems were generated and solved by the developed algorithms and their relative average deviations were calculated. Table (1) demonstrates the relative average deviation values of the 40 test problems obtained by each algorithm according to the different values of *n* and *TF*, when *m=5*. Similarly Tables (2) through (4) illustrate the relative average deviation values of 40 test problems for different scenarios when *m=10, m=15,* and *m=20* respectively. Through these tables it can be observed that in most cases, algorithm *AT4* has the best effectiveness followed by *AT3* which has the second best effectiveness ranking among the four algorithms.

We further analyzed the relative effectiveness of the developed algorithms by comparing them in the absence of the toughness factors (*TF*). To conduct this analysis, we first defined a new index by calculating the average of the deviations over three different *TF* values. We then plotted a curve for each algorithm which illustrates the variations of this index, according to the different values of *n*. Figures 1 through 4 demonstrate these curves for *m=5* , *m=10, m=15,* and *m=20* respectively. These curves reveal that, except for *m=5,* algorithm *AT4* demonstrates a better effectiveness. It can also be concluded that algorithm *AT3* performs superbly when *m=5*.

Table (1). Average deviation of the *TT* from the best solution *(*for *m=5)*

TF Value	Algorithms	*n=5*	*n=10*	*n=15*	*n=20*	*n=25*	*n=30*	*n=35*	*n=40*
0.1	*AT1*	*2.79*	*1.78*	*0.96*	*0.42*	*0.13*	*0.13*	*0.10*	*0.27*
	AT2	*2.85*	*2.48*	*2.07*	*2.05*	*2.35*	*1.52*	*1.87*	*2.1*
	AT3	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
	AT4	*0.38*	*1.02*	*0.77*	*1.30*	*2.06*	*1.42*	*1.77*	*1.76*
0.2	*AT1*	*2.18*	*0.90*	*0.66*	*0.76*	*0.28*	*0.18*	*0.28*	*0.31*
	AT2	*2.74*	*2.04*	*2.05*	*1.68*	*1.78*	*2.37*	*2.23*	*2.02*
	AT3	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
	AT4	*0.42*	*1.29*	*0.99*	*0.89*	*1.32*	*1.83*	*1.87*	*1.66*
0.4	*AT1*	*1.73*	*1.42*	*0.89*	*0.63*	*0.34*	*0.17*	*0.12*	*0.16*
	AT2	*2.28*	*2.43*	*3.04*	*1.57*	*2.09*	*1.85*	*1.60*	*2.45*
	AT3	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
	AT4	*0.96*	*0.52*	*1.65*	*1.28*	*1.60*	*1.51*	*1.17*	*1.92*

Table (2). Average deviation of the *TT* from the best solution *(*for *m=10)*

TF Value	Algorithms	*n=5*	*n=10*	*n=15*	*n=20*	*n=25*	*n=30*	*n=35*	*n=40*
0.1	*AT1*	*2.92*	*3.47*	*2.61*	*2.59*	*2.04*	*1.32*	*1.49*	*1.02*
	AT2	*1.73*	*2.50*	*2.40*	*0.88*	*1.07*	*0.53*	*0.92*	*0.52*
	AT3	*0*	*0.61*	*0.52*	*1.07*	*0.34*	*0*	*0.50*	*0.31*
	AT4	*0.14*	*0*	*0*	*0*	*0*	*0.41*	*0*	*0*
0.2	*AT1*	*1.89*	*2.38*	*2.71*	*1.94*	*1.77*	*1.28*	*1.52*	*1.00*
	AT2	*2.11*	*2.66*	*2.26*	*1.36*	*1.13*	*1.26*	*1.25*	*0.59*
	AT3	*0*	*0*	*0.13*	*0.17*	*0.25*	*0*	*0.57*	*0.41*
	AT4	*0.28*	*0.75*	*0*	*0*	*0*	*0.21*	*0*	*0*
0.4	*AT1*	*2.92*	*4.69*	*2.46*	*1.60*	*1.30*	*0.65*	*1.05*	*0.85*
	AT2	*2.79*	*4.40*	*2.71*	*1.43*	*0.89*	*1.08*	*0.79*	*0.70*
	AT3	*0.22*	*0*	*0*	*0.11*	*0.56*	*0*	*0.25*	*0*
	AT4	*0*	*0.93*	*0.11*	*0*	*0*	*0.35*	*0*	*0.08*

Table (3). Average deviation of the *TT* from the best solution *(*for *m=15)*

TF Value	Algorithms	*n=5*	*n=10*	*n=15*	*n=20*	*n=25*	*n=30*	*n=35*	*n=40*
0.1	*AT1*	*0.84*	*2.76*	*3.57*	*2.69*	*1.93*	*1.36*	*1.76*	*1.94*
	AT2	*0.45*	*2.36*	*3.08*	*1.82*	*1.08*	*0.39*	*0.88*	*0.93*
	AT3	*0.24*	*0.63*	*0.17*	*0.85*	*0.58*	*0.57*	*0.58*	*0.79*
	AT4	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0.2	*AT1*	*1.82*	*2.50*	*2.46*	*2.75*	*1.82*	*1.49*	*1.95*	*2.39*
	AT2	*1.68*	*2.05*	*1.44*	*2.24*	*1.29*	*0.91*	*0.97*	*1.32*
	AT3	*0*	*0*	*0.09*	*0.48*	*0.63*	*0.02*	*0.72*	*1.28*
	AT4	*0.24*	*0.33*	*0*	*0*	*0*	*0*	*0*	*0*
0.4	*AT1*	*1.09*	*3.42*	*3.06*	*2.37*	*2.11*	*2.30*	*1.93*	*1.80*
	AT2	*0.83*	*3.19*	*2.09*	*2.16*	*1.46*	*1.68*	*1.33*	*1.15*
	AT3	*0*	*0.02*	*0.02*	*0.14*	*0.77*	*0*	*0.13*	*0.35*
	AT4	*0.14*	*0*	*0*	*0*	*0*	*0.06*	*0*	*0*

Table (4). Average deviation of the *TT* from the best solution *(for m=20)*

TF Value	Algorithms	*n=5*	*n=10*	*n=15*	*n=20*	*n=25*	*n=30*	*n=35*	*n=40*
0.1	*AT1*	*1.04*	*2.68*	*2.94*	*3.53*	*2.02*	*1.99*	*1.29*	*1.96*
	AT2	*1.84*	*2.32*	*2.50*	*2.36*	*0.93*	*1.08*	*0.61*	*1.08*
	AT3	*0.17*	*0.18*	*0.37*	*0.64*	*0.22*	*0.39*	*0.41*	*0.69*
	AT4	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
0.2	*AT1*	*1.53*	*2.88*	*3.18*	*3.11*	*2.24*	*1.97*	*1.43*	*1.94*
	AT2	*0.69*	*3.33*	*2.97*	*1.69*	*1.97*	*0.87*	*0.56*	*1.10*
	AT3	*0*	*0*	*0*	*0.30*	*0.82*	*0.58*	*0.23*	*0.44*
	AT4	*0.06*	*0.52*	*0.04*	*0*	*0*	*0*	*0*	*0*
0.4	*AT1*	*1.47*	*2.94*	*3.17*	*3.50*	*2.21*	*2.42*	*2.30*	*1.75*
	AT2	*1.14*	*3.36*	*2.65*	*2.94*	*1.61*	*1.20*	*1.85*	*0.81*
	AT3	*0.01*	*0*	*0*	*0.61*	*0.24*	*0.22*	*0.35*	*0.74*
	AT4	*0*	*0.44*	*0.12*	*0*	*0*	*0*	*0*	*0*

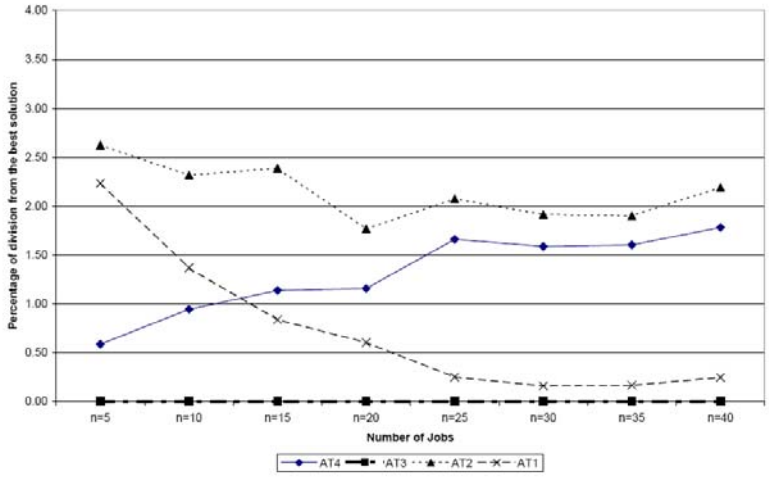

Figure 1. Relative effectiveness of the developed algorithms, for average TF and for m=5.

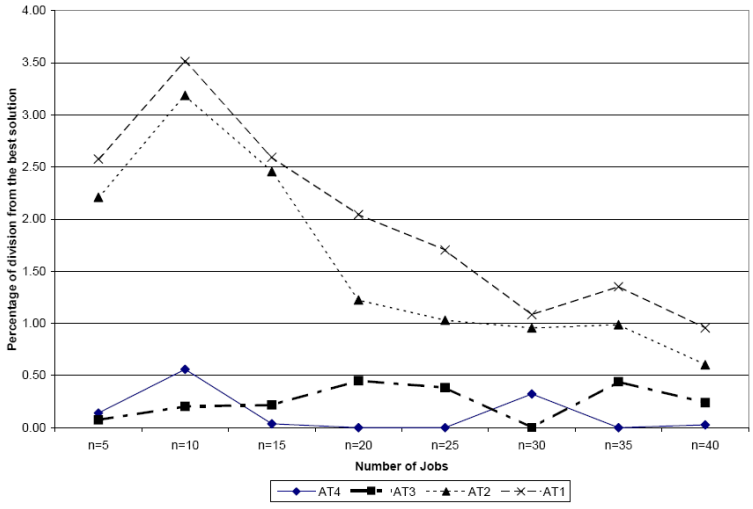

Figure 2. Relative effectiveness of the developed algorithms, for average TF and for m=10.

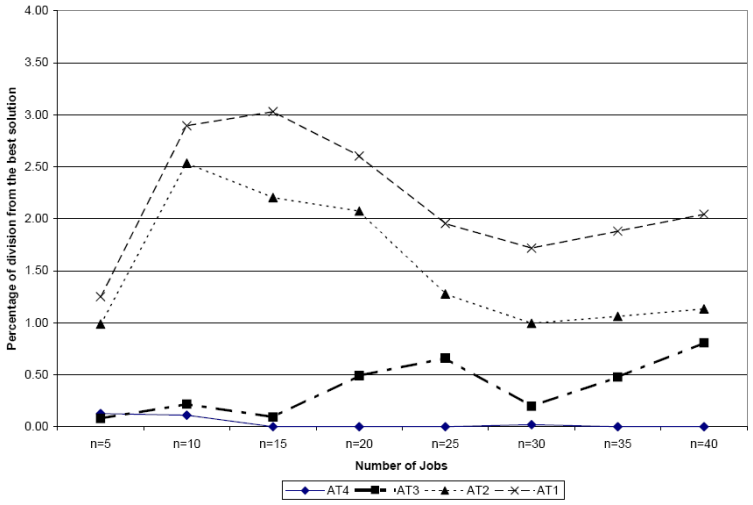

Figure 3. Relative effectiveness of the developed algorithms, for average TF and for m=15

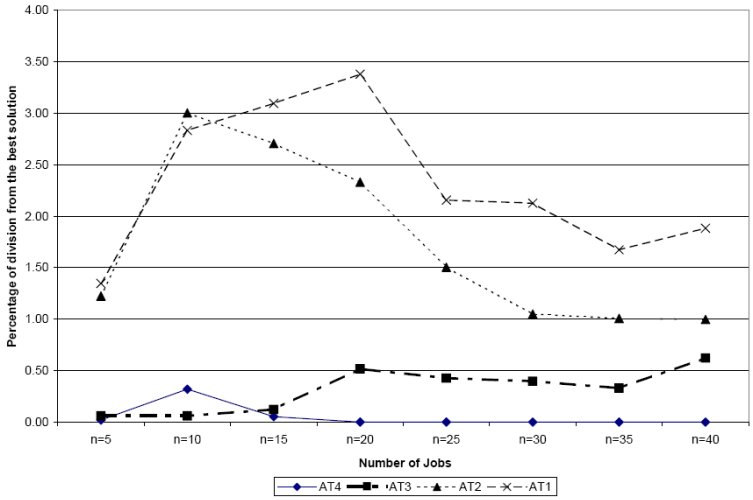

Figure 4. Relative effectiveness of the developed algorithms, for average TF and for m=20.

6. CONCLUSIONS

In this paper we proposed the "generalized tardiness flowshop scheduling model", as a new version of the flowshop scheduling model with the total tardiness as its measure of performance for solving the multi-project scheduling problems. In doing so, we considered a traditional tardiness flowshop model in which for every operation of a job there is an associated due date, and pursued the aim of finding a permutation schedule for a set of projects which minimizes the total tardiness.

Based on the modification of the ACT index, we developed four new indices by which the sequence of the tasks in each project can be determined. We then employed these indices to develop four algorithmic procedures for solving the generalized version of the tardiness flowshop problem.

We then used the concept of pseudo random number generation to generate a set of test problems and conducted extensive numerical experiments to evaluate the effectiveness of the proposed heuristic algorithms. For different values of the input parameters, such as the number of jobs, the

number of machines, and the values of due date tightness parameter, we defined a large set of scenario test problems. For each scenario, 40 randomly generated test problems were generated and solved to determine the relative effectiveness of the proposed algorithms. As the result it was shown that both *AT4* and *AT3* algorithms perform better than others among the four proposed algorithms.

REFERENCES

[1] Allahverdi A. (2004), A new heuristic for m-machine flow shop scheduling problem with bicriteria of makespan and maximum tardiness; *Computers & Operations Research* 31(2); 157-180.

[2] Baker K.R., Bertrand J.W.M. (1982), A Dynamic priority rule for sequencing against due dates; *Journal of Operational Management* 3(1); 37-42.

[3] Baker K.R. (1984), Sequencing rules & due date assignments in a job shop; *Management Science* 30(9); 1093-1104.

[4] Bilge U., Kirac F., Kurtulan M., Pekgun P. (2004), A tabu search algorithm for parallel machine total tardiness problem; *Computers & Operations Research* 31(3); 397-414.

[5] Carroll D.C. (1965), *Heuristic sequencing of jobs with single & multiple components*, Ph.D. dissertation, Sloan School of Management, MIT, Mass.

[6] Chakravarthy K., Rajendran C. (1999), A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization; *Production planning and control*, 10(7); 701-714.

[7] Chio S.W., Lee G.C., Kim Y.D. (2005), Minimizing total tardiness of orders with reentrant lots in a hybrid flow shop; *International Journal of Production Research*, 43(11); 2149-2167.

[8] Etiler O., Toklu B., Atak M., Wilson J. (2004), A generic algorithm for flow shop scheduling problems; *Journal of Operations Research Society*, 55(8); 830-835.

[9] Ghessmi-Tari F., Olfat L. (2004), Two COVERT based algorithms for solving the generalized flow-shop problems; *Proceedings of the 34th International Conference on Computers and Industrial Engineering*, 29-37.

[10] Gupta N.D., Kruger K., Lauff V., Werner F., Sotskov Y.N. (2002), Heuristics for hybrid flow shops with controllable processing times and assignable due dates; *Computers & Operations Research*, 29(10); 1417-1439.

[11] Lee G.C., Kim Y.D. (2004), A branch and bound for a two stage hybrid flow shop scheduling problem minimizing total tardiness; *International Journal of Production Research* 42(22); 4731-4743.

[12] Lee L. (2001), Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs; *Computers & Operations Research* 28(9); 835-852.

[13] Lin H.T., Liao C.J. (2003), A case study in a two-stage hybrid flow shop with setup time and dedicated machines; *International Journal of Production Economics* 86(2); 133-143.

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

[15] Mosheiov G. (2003), Scheduling unit processing time jobs on an m-machine flowshop; *Journal of the Operations Research Society* 54(4); 437-441.

[16] Olfat L. (1998), *Development of a set of algorithms for flowshop tardiness problems*, Ph.D. dissertation, School of Management, University of Tehran, Tehran, Iran.

[17] Parthasarathy S., Rajendran C. (1998), Scheduling to minimize mean tardiness and weighted tardiness in flowshop and flowline-based manufacturing cell; *Computers & Industrial Engineering* 34(2); 431-546.

[18] Rachamaduagu R.V., Morton T.E. (1982), *Myopic heuristic for the single machine weighted tardiness problem*, working paper #38-82-83, GsIA, Carnegie Mellon University.

[19] Rachamaduagu R.V. (1984), Myopic heuristic in open shop scheduling, modeling & simulation, *Proceedings of the 14 Annual Pittsburgh Conf.,* 1245-1250.

[20] Sapar H., Henry M.C. (1996), Combinatorial Evaluation of six dispatching rules in dynamic two machine flowshop; *International Journal of Management Science* 24,(1); 73-81.

[21] Vepsalainen A.P.J., Morton T.E. (1987), Priority rules for job shops with weighted tardiness costs; *Management Science* 23(8); 1035-1047.

[22] Yeh W., Allahverdi A. (2004), A branch and Bound algorithm for the three machine flow shop scheduling problem with bi-criteria of make-span and total flow time; *International Transaction in Operations Research* 11(3); 323-332.