International Journal of Industrial Engineering & Production Research (2010) pp. 97-104

September 2010, Volume 21, Number 2

International Journal of Industrial Engineering & Production Research

ISSN: 2008-4889

http://IJIEPR.iust.ac.ir/

# A Mathematical Programming Model for Flow Shop Scheduling Problems for Considering Just in Time Production

**R. Ramezanian, M.B. Aryanezhad & M. Heydari***

***R. Ramezanian***: *MSc in Department of Industrial Engineering, Iran University of Science and Technology.*
***M.B. Ariyanezhad***: *Professor in Department of Industrial Engineering, Iran University of Science and Technology.*
***M. Heydari***: *Assistant Professor in Department of Industrial Engineering, Iran University of Science and Technology.*

## KEYWORDS

Flow Shop Scheduling,
Bypass,
Mathematical Programming,
Meta-heuristic Algorithm

## ABSTRACT

*In this paper, we consider a flow shop scheduling problem with bypass consideration for minimizing the sum of earliness and tardiness costs. We propose a new mathematical modeling to formulate this problem. There are several constraints which are involved in our modeling such as the due date of jobs, the job ready times, the earliness and the tardiness cost of jobs, and so on. We apply adapted genetic algorithm based on bypass consideration to solve the problem. The basic parameters of this meta-heuristic are briefly discussed in this paper. Also a computational experiment is conducted to evaluate the performance of the implemented methods. The implemented algorithm can be used to solve large scale flow shop scheduling problem with bypass effectively.*



## 1. Introduction

Production scheduling is a decision-making process in the operation class. It can be defined as the allocation of available production resources to carry out certain tasks in an efficient way. Such a frequently occurring scheduling problem is difficult to solve due to its complex nature.

This paper is primarily concerned with industrial scheduling problems, where one has to sequence the jobs on each resource over time.

In a flow shop environment, a set of jobs must be processed on a number of sequential machines, processing routes of all jobs are the same, that is the operations of any job are processed in the same order, whereas a flow shop with bypass model, a generalization of the ordinary flow shop model, is more realistic, and it assumes that at least one job does not visit one machine. Moreover, a machine can process at most one job at a time and a job can be processed by at most one machine at a time. Preemption of processing is not allowed. The problem consists of sequencing the jobs to the each machine so that some optimality criteria are minimized.

Since flow shop as well as job shop problems with few exceptions have been proved to be NP-hard [1], heuristic procedures are the most suitable ones for their solution, especially for large-size instances. Several approaches and models are proposed to solve the scheduling problem, namely the discrete variable mathematical programming, simulation techniques and the network analysis. Johnson [2] was the first to propose a method to solve the scheduling problem in a flow shop production environment for a single criterion context. His algorithm has been utilized by other researchers, including, for instance, Palmer [3], Campbell et al. [4], Gupta [5], Gupta et al. [6] and Tadei et al. [7]. Flow shop scheduling with the makespan objective has been investigated for instance,

**Corresponding author. M. Heydari**
**Email**: *Mheydari@iust.ac.ir Reza_Ramezanian@ind.iust.ac.ir Mirarya@iust.ac.ir*
**Paper first received May. 17. 2009 ,and in revised form Oct. 27. 2010.**

Simons [8], Huq et al. [9], Zobolas et al. [10]. Flow shop scheduling with the sum of earliness and tardiness costs objective has been investigated for instance, bulbul et al. [11], Lauff and Werner [12].

Actually, we are not the first who observed the phenomenon of "bypass". The fact that the computational complexity of a more general problem (admitting missing operations) may be much harder than that of the corresponding problem without missing operations was the subject of some previous papers known in the literature. For instance, as observed by Leisten and Kolbe [13], Glass et al. [14]. Glass et al. [14] considers the no-wait scheduling of n jobs in a two-machine flow shop, where some jobs require processing on the first machine only. The objective is to minimize the maximum completion time, or makespan.

We consider the scheduling of *n* jobs in a *m* machine flow shop, where some jobs do not require processing on the some machines. The objective is to minimize the sum of earliness and tardiness costs. Just in time concept for the scheduling environment can be provided by considering minimization of the sum of earliness and tardiness costs as the objective function.

## 2. Problem Description

In the flow shop scheduling problem (FSSP) there are *m* machines in series. Every single job has to be processed on each machine.

All jobs have to follow the same route i.e., they have to be processed first on machine 1, then on machine 2, and so on. The flow shop scheduling problem with bypass consideration can be interpreted as a generalization of the classical flow shop model which is more realistic and assumes at least one job does not visit one machine.

The FSSP with bypass can be described as follows: Each of n jobs from set $J=\{1,2,..., n\}$ will be sequenced through m machines ($i=1, 2,...,m$). Job $j \in J$ has a sequence of $l_j$ operations through a subset m machines (jobs may have zero processing time on some machines) and a given due date $d_j$. Operation $O_{ij}$ corresponds to the processing of job $j$ on machine $i$ during an uninterrupted processing time $t_{ij}$ (processing time $t_{ij}$ can be zero). At any time, each machine can process at most one job and each job can be processed on at most one machine.

## 3. Mathematical Formulation

### 3-1. General Assumption

All *n* jobs to schedule are independent and are not available for processing at time zero.

A job has some operations that each of them is to be performed on a specified machine. Some jobs may not process on some machines so the processing time of them on that machines are zero (missing operations). (bypass).

Job descriptions are known in advance.

Jobs have no associated priority values.

One machine can process at most one job at a time.

Each job is processed on at most one machine at a time.

Setup times for the operations are sequence-independent and are included in processing times.

Machine is available at all times.

There is no travel time between stages; jobs are available for processing at a stage immediately after completing processing at the previous stage.

There is only one of each type of machine.

There is no precedence constraint among the jobs.

Preemption and splitting of any particular job is not allowed: a job, once started on a machine, continues in processing until it is completed.

Jobs are allowed to wait between two stages, and the storage is unlimited.

All programming parameters are deterministic and there is no randomness.

Any breakdowns and scheduled maintenance are not allowed.

No more than one operation of the same job can be executed at a time.

The processing times are independent of the sequence and are given.

### 3-2. Parameters

$n$: Number of jobs

$m$: Number of machines

$i$: Machine index

$j,h$: Job index

$t_{ij}$: Processing time of job $j$ on machine $i$

$R_j$: Release date of job $j$

$d_j$: Due date of job $j$

$H_j$: Holding (earliness) of job $j$ per time unit

$\beta_j$: Shortage (tardiness) cost of job $j$ per time unit

$\delta_{ij}$: A binary parameter that is equal to 1 if job $j$ is not processed on machine $i$, 0 otherwise.

$M$: A large constant ($M \to \infty$)

### 3.3. Decision Variables

$S_{ij}$ : Starting time of job $j$ on machine $i$

$C_{ij}$ : Completion time of job $j$ on machine $i$

$E_j$: Earliness of job $j$ $\quad E_j = \max \{d_j - C_{ij}, 0\}$

$T_j$: Tardiness of job $j$ $\quad T_j = \max \{C_{ij} - d_j, 0\}$

$Y_{ihj}$: A binary variable that is equal to 1 if job $j$ is processed immediately after job $h$ when processing on machine $i$, 0 otherwise.

The mathematical model for minimizing of the earliness cost, $H_j(E_j)$, and the tardiness cost, $\beta_j(T_j)$ is as follow. The earliness cost could represent the inventory cost for early finished stocks, and the tardiness cost could represent the penalty cost for the late delivery.

$$\min z = \sum_{j=1}^{n} (H_j E_j + \beta_j T_j) \quad (1)$$

$$C_{mj} + E_j - T_j = d_j; \ \forall j \quad (2)$$

$$C_{ij} = \delta_{ij} C_{(i-1)j} + (1 - \delta_{ij})(S_{ij} + t_{ij}); \quad i = 2,...,m, j = 1,..,n \quad (3)$$

$$S_{(i+1)j} \geq C_{ij}; \ i = 1,...,m-1, j = 1,..,n \quad (4)$$

$$S_{ij} \geq (C_{ih} - (1 - Y_{ihj})M)(1 - \delta_{ij}); \ \forall i, j, h \quad (5)$$

$$\sum_{j=1}^{n} Y_{ijj} = 0; \ \forall i \quad (6)$$

$$\sum_{j=1}^{n} Y_{ihj} \leq (1 - \delta_{ih})M; \ \forall i, h \quad (7)$$

$$\sum_{h=1}^{n} \sum_{j=1}^{n} Y_{ihj}(1 - \delta_{ij}) \leq \sum_{j=1}^{n} (1 - \delta_{ij}) - 1; \ \forall i \quad (8)$$

$$\sum_{j=1}^{n} Y_{ihj} \leq 1; \ \forall i, h \quad (9)$$

$$\sum_{h=1}^{n} Y_{ihj} \leq 1; \ \forall i, j \quad (10)$$

$$Y_{ihj} \leq 1 - Y_{ijh}; \ \forall i, j, h \quad (11)$$

$$S_{1j} \geq R_j, \ \forall j \quad (12)$$

$$C_{1j} \geq R_j, \ \forall j \quad (13)$$

$$Y_{ijh} = \{0,1\}, \ \forall i, j, h \quad (14)$$

The objective function (1) considers the minimization of the earliness cost and the tardiness cost and the considered objective function provides just in time production in manufacturing systems. The constraint set (2) determines earliness and tardiness of each job. The constraint set (3) corresponds to the computation of the completion time of job (if job *j* is not processed on machine *i*, its completion time is the same as completion time on previous machine).

The constraint set (4) forces to start the processing of each job only when it has been completed on the precedent machine. The constraint set (5) forces to start the processing of each job only when its precedent job has been completed on the same machine. The constraint sets (6-11) determine sequence of jobs for any machine. The constraint set (12) bounds the job starting times to be after job release times in the system. The constraint set (13) insures that the job finishing times on the first machine to be after job release times (if job j does not require processing on the first machine, $C_{1j}=R_j$. (14) is logical constraint.

It must be noticed that when there is the bypass condition, the completion time of jobs do not necessarily determine on the last machine, but in our model the completion time of each job –that may occur on any machine – transferred on the last machine by considering a $\delta_{ij}$.

Consider a flow shop scheduling problem with bypass consideration with three jobs and three machines. Processing time of each job on each machine and other data is given in table 1. Job 2 and 3 do not require processing on the machine 2 and they can process on machine 3 right after their process completed on machine 1.

**Tab. 1 Processing times and other data**

| | | Machine | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | R | d | H | β |
| Job | 1 | 5 | 7 | 1 | 2 | 14 | 3 | 2 |
| | 2 | 3 | 0 | 1 | 4 | 8 | 2 | 3 |
| | 3 | 1 | 0 | 3 | 3 | 6 | 4 | 3 |

The optimum sequence of jobs on machines for minimizing the sum of earliness and tardiness costs is shown in Fig. 1.

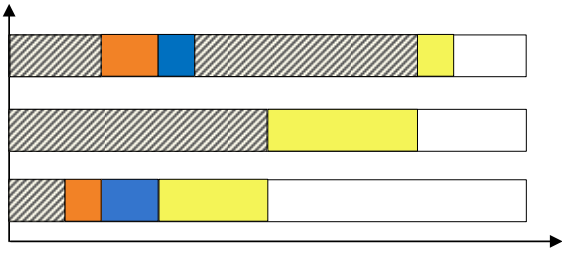

**Fig. 1 Gant chart of sequence vector on machines (objective function = 15)**

Although job 2 and 3 do not require processing on the machine 2 ($\delta_{22}=1$, $\delta_{23}=0$) but our model considers a virtual completion time for them on this machine (see the constraint set 3).

## 4. The Genetic Algorithm

Genetic algorithms have been proven to be powerful techniques for constrained optimization and combinatorial optimization problems.
The GA was proposed by Holland (1975) [15] to encode the factors of a problem by chromosomes, where each gene represents a feature of the problem. The GA's structure and parameter setting affect its performance.
The overall structure of our GA can be described as follows:

1. *Coding*: The genes of the chromosomes describe the jobs, and the order in which they appear in the chromosome describes the sequence of Jobs. Each chromosome represents a solution for the problem.

2. *Initial population*: The initial chromosomes are obtained by a Random dispatching rule for sequencing.

3. *Fitness evaluation*: The sum of earliness and tardiness cost is computed for each chromosome in the current generation.

4. *Selection*: In any iteration, chromosomes are chosen randomly for crossover and mutation.

5. *Offspring generation*: The new generation is obtained by changing the sequencing of operations (reproduction, enhanced order crossover and mutation). These rules preserve feasibility of new individuals. New individuals are generated until a fixed maximum number of individuals is reached.

6. *Stop criterion*: Fixed number of generations is reached. If the stop criterion is satisfied, the algorithm ends and the best chromosome, together with the corresponding schedule, are given as output. Otherwise, the algorithm iterates again steps 3–5.
Based on bypass consideration GA is adapted to consider operation with zero processing time on some machines. Following is the presented our proposed genetic algorithm.

### 4.1. Design of Genes

In this paper, each gene is job and the chromosome is job sequence vector on machines. At first it is supposed that all jobs have a priority on each machine. It means that if a job does not be processed on a machine, a virtual priority is assigned that its processing time on the machine is zero. The priorities on machines are generated randomly.
Consider a flow shop scheduling with missing operation problem with 5 jobs and 3 machines (see table 2).

**Tab. 2 Processing time data**

| | | Machine | | |
|---|---|---|---|---|
| | | 1 | 2 | 3 |
| Job | 1 | 2 | 0 | 1 |
| | 2 | 0 | 1 | 2 |
| | 3 | 4 | 2 | 1 |
| | 4 | 0 | 0 | 5 |
| | 5 | 1 | 2 | 0 |

The job sequence for this example represented in fig. 2 can be translated into a list of ordered jobs below:

$$\textit{Machine I}: \; j_2 \succ j_1 \succ j_5 \succ j_3 \succ j_4$$
$$\textit{Machine II}: \; j_4 \succ j_1 \succ j_2 \succ j_5 \succ j_3$$
$$\textit{Machine III}: \; j_3 \succ j_2 \succ j_4 \succ j_5 \succ j_1$$

| Priority (k) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| job Sequence on machine 1: $V_1$(k) | 2 | 1 | 5 | 3 | 4 |
| job Sequence on machine 2: $V_2$(k) | 4 | 1 | 2 | 5 | 3 |
| job Sequence on machine 3: $V_3$(k) | 3 | 2 | 4 | 5 | 1 |

*Highlight jobs on a machine have zero processing time

**Fig. 2 Illustration of the job sequence vector on machines**

### 4.2. The Genetic Operators

#### 4.2.1. Reproduction

The best chromosomes which have a lower fitness function are chosen. This mechanism just copies the chosen chromosomes to the next generation.

#### 4.2.2. Crossover

Crossover operator recombines two chromosomes to generate a number of children. Offspring of crossover should represent solutions that combine substructures of their parental solutions. The enhanced order crossover expanded from the classical order crossover [16] works as follows:
Step1. Randomly choose two chromosomes, named parent 1 and parent 2.
Step 2. Do the following steps for the same machine in selected chromosomes (parent 1 and parent 2):
Step 1.2. Randomly select a subsection of job sequence for *i*th machine from parent 1.
Step 2.2. Produce a proto-child by copying the substring of job sequence into the corresponding positions.
Step 3.2. Starting with the first position from *i*th machine of parent 2, delete the jobs which are in the substring from *i*th machine of the second parent. The resulted sequence of jobs contains the jobs that the proto-child needs.
Step 4.2. Place the remaining jobs into the empty positions of the proto-child from left to right according

to the order of the sequence in the *i*th machine of second parent.

#### 4.2.3. Mutation

Our mutation mechanism works as follows:
Step 1. Randomly choose one chromosome.
Step 2. Do the Following steps for the same machine in selected chromosome:
Step 2.1. Randomly choose two priorities from *i*th machine of selected chromosome in step 1.
Step 2.2. Replace selected jobs with each other [17,18].

### 4.3. Fitness Function

The fitness function is the same as the objective function which defined in section 3. In the proposed genetic algorithm the lower fitness function is desired.

$E_j$: Earliness of job $j$ $E_j = \max\{d_j - C_{ij}, 0\}$

$T_j$: Tardiness of job $j$ $T_j = \max\{C_{ij} - d_j, 0\}$
$H_j$: Holding cost of job $j$ per time unit

$\beta_j$: Shortage cost of job $j$ per time unit

$$Fitness\ Function: \sum_{j=1}^{n} (H_j E_j + \beta_j T_j)$$

Note that the earliness cost could represent the inventory cost for early finished stocks, and the tardiness cost could represent the penalty specified in the contract for the late delivery.

## 5. Computational Results

In the scheduling literature, there is not a benchmark for the FSSP with bypass. To test the efficiency of the modified genetic algorithm for considering bypass assumption, a number of random instances were generated with the following characteristics:
1. Dimensions of the problem are between (m×n) = (3×3) and (m×n) = (30×30)
2. Holding (earliness) and shortage (tardiness) costs for each job at each stage are chosen randomly from U(1-5).
3. Release date for each job is chosen randomly from U(0-10).
4. Due date for each job is chosen randomly from U(15-30).
5. Processing time for each job on each machine is chosen randomly from U(0-5).
The proposed mathematical model for FSSP with bypass is solved by genetic algorithm as well as LINGO 8.0. The genetic algorithm was coded with MATLAB R2007(b) and all tests were conducted on a Pentium_IV PC at 3 GHz with 1.0GB of RAM.
The flow shop scheduling problem, when considered in the general case, gives $(n!)^m$ possible schedules. Even for problems as small as n = m = 5, the number of possible schedules is so large that a direct enumeration is economically impossible.
When the size of problem is small both genetic algorithm and LINGO can solve it in a short time. However, as the size of problem increases the computation time of LINGO increased exponentially. The comparison for small size problems between LINGO and genetic algorithm is shown in table 3.

**Tab. 3. Comparison results of GA and LINGO**

| Prob. | m | n | GA (Iteration=100, Population size=50) | | LINGO | |
|---|---|---|---|---|---|---|
| | | | FF * | Time (min) | FF | Time (min) |
| 1 | 2 | 5 | 6 | 0.022 | 6 | 0.417 |
| 2 | 3 | 3 | 2 | 0.017 | 2 | 0.017 |
| 3 | 3 | 4 | 3 | 0.023 | 3 | 0.083 |
| 4 | 3 | 5 | 7 | 0.027 | 2 | 0.800 |
| 5 | 4 | 3 | 8 | 0.022 | 8 | 0.067 |
| 6 | 4 | 4 | 18 | 0.032 | 18 | 1.510 |
| 7 | 5 | 2 | 12 | 0.022 | 12 | 0.017 |
| 8 | 5 | 3 | 43 | 0.028 | 43 | 0.083 |
| 9 | **5** | **5** | **40** | **0.046** | **69** | **12 (hours)**** |

* FF: Fitness function (Sum of the earliness and tardiness costs)

* * This problem is interrupted

As demonstrated in table 3, the adapted genetic algorithm has the ability to reach the optimal solution for small-sized problems. The implemented genetic algorithm can efficiently solve the problem in a considerably short time.
For large scale problems, the results by using the genetic algorithm is presented in table (4-6). According to fig. 3 and fig. 4, the adapted GA has the ability to reach stable solutions.
It is obvious while the problem size is increasing, the efficiency of the genetic algorithm decreases as demonstrated in table 4 and table 5.

**Tab. 4. Numerical example for 5 machines and 5 jobs**

**m=5, n=5, Iteration=100, Pop. size =50**
(Time: Second)

| Prob. | FF | Time | Prob. | FF | Time | Prob. | FF | Time |
|---|---|---|---|---|---|---|---|---|
| 1 | 46 | 2.67 | 34 | 41 | 2.36 | 67 | 45 | 2.50 |
| 2 | 41 | 2.67 | 35 | 49 | 2.44 | 68 | 45 | 2.53 |
| 3 | 41 | 2.49 | 36 | 50 | 2.54 | 69 | 46 | 2.51 |
| 4 | 46 | 2.46 | 37 | 41 | 2.59 | 70 | 49 | 2.62 |

**m=5, n=5, Iteration=100, Pop. size =50**
(Time: Second)

| Prob. | FF | Time | Prob. | FF | Time | Prob. | FF | Time |
|---|---|---|---|---|---|---|---|---|
| 5 | 45 | 2.43 | 38 | 46 | 2.47 | 71 | 41 | 2.58 |
| 6 | 45 | 2.60 | 39 | 41 | 2.51 | 72 | 45 | 2.46 |
| 7 | 45 | 2.28 | 40 | 46 | 2.60 | 73 | 45 | 2.40 |
| 8 | 50 | 2.43 | 41 | 45 | 2.50 | 74 | 50 | 2.52 |
| 9 | 41 | 2.36 | 42 | 41 | 2.64 | 75 | 41 | 2.42 |
| 10 | 45 | 2.55 | 43 | 45 | 2.59 | 76 | 41 | 2.43 |
| 11 | 41 | 2.35 | 44 | 41 | 2.47 | 77 | 49 | 2.63 |
| 12 | 41 | 2.48 | 45 | 41 | 2.55 | 78 | 46 | 2.56 |
| 13 | 49 | 2.52 | 46 | 49 | 2.53 | 79 | 45 | 2.68 |
| 14 | 41 | 2.60 | 47 | 46 | 2.58 | 80 | 49 | 2.23 |
| 15 | 46 | 2.44 | 48 | 49 | 2.43 | 81 | 45 | 2.43 |
| 16 | 46 | 2.49 | 49 | 41 | 2.61 | 82 | 49 | 2.53 |
| 17 | 45 | 2.47 | 50 | 50 | 2.59 | 83 | 45 | 2.50 |
| 18 | 41 | 2.56 | 51 | 49 | 2.36 | 84 | 45 | 2.50 |
| 19 | 41 | 2.58 | 52 | 45 | 2.30 | 85 | 41 | 2.59 |
| 20 | 50 | 2.38 | 53 | 49 | 2.43 | 86 | 41 | 2.49 |
| 21 | 49 | 2.55 | 54 | 45 | 2.32 | 87 | 41 | 2.29 |
| 22 | 45 | 2.53 | 55 | 41 | 2.61 | 88 | 41 | 2.55 |
| 23 | 41 | 2.47 | 56 | 50 | 2.39 | 89 | 41 | 2.62 |
| 24 | 41 | 2.59 | 57 | 50 | 2.40 | 90 | 49 | 2.50 |
| 25 | 49 | 2.46 | 58 | 45 | 2.32 | 91 | 41 | 2.48 |
| 26 | 50 | 2.38 | 59 | 49 | 2.36 | 92 | 41 | 2.47 |
| 27 | 46 | 2.35 | 60 | 41 | 2.40 | 93 | 46 | 2.35 |
| 28 | 41 | 2.52 | 61 | 41 | 2.49 | 94 | 49 | 2.49 |
| 29 | 41 | 2.69 | 62 | 49 | 2.56 | 95 | 49 | 2.61 |
| 30 | 41 | 2.67 | 63 | 41 | 2.44 | 96 | 50 | 2.45 |
| 31 | 45 | 2.40 | 64 | 46 | 2.53 | 97 | 50 | 2.62 |
| 32 | 45 | 2.44 | 65 | 46 | 2.25 | 98 | 50 | 2.53 |
| 33 | 45 | 2.40 | 66 | 49 | 2.56 | 99 | 49 | 2.47 |

| Min (FF) | Max (FF) | Mean (FF) | StDev (FF) | C.V. (FF) | Range (FF) | Mean (Time) |
|---|---|---|---|---|---|---|
| 41 | 50 | 45.04 | 3.411 | 0.076 | 9 | 2.49 |

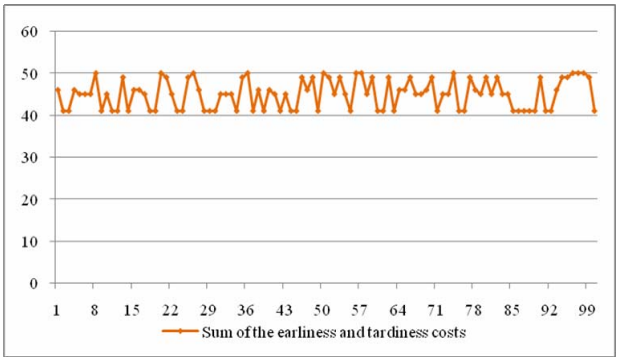


**Fig. 3. Line chart of table 4 (Sum of earliness and tardiness costs)**

**Tab. 5. Numerical example for 10 machines and 10 jobs**

**m=10, n=10, Iteration=100, Population size =100**
(Time: Second)

| Prob. | FF | Time | Prob. | FF | Time | Prob. | FF | Time |
|---|---|---|---|---|---|---|---|---|
| 1 | 1338 | 24.44 | 11 | 1443 | 24.77 | 21 | 1187 | 24.66 |
| 2 | 1452 | 24.53 | 12 | 1666 | 24.44 | 22 | 1262 | 24.60 |
| 3 | 1332 | 24.93 | 13 | 1256 | 24.77 | 23 | 1517 | 24.56 |
| 4 | 1625 | 22.32 | 14 | 1501 | 24.50 | 24 | 1417 | 24.62 |
| 5 | 1341 | 25.09 | 15 | 1212 | 22.46 | 25 | 1400 | 22.16 |
| 6 | 1287 | 25.02 | 16 | 1210 | 24.90 | 26 | 1307 | 22.33 |
| 7 | 1241 | 22.72 | 17 | 1232 | 24.83 | 27 | 1357 | 24.92 |
| 8 | 1592 | 25.25 | 18 | 1403 | 24.72 | 28 | 1507 | 24.95 |
| 9 | 1587 | 24.98 | 19 | 1645 | 25.20 | 29 | 1584 | 24.74 |
| 10 | 1231 | 24.81 | 20 | 1380 | 24.87 | 30 | 1374 | 24.70 |

| Min (FF) | Max (FF) | Mean (FF) | StDev (FF) | C.V. (FF) | Range (FF) | Mean (Time) |
|---|---|---|---|---|---|---|
| 1187 | 1666 | 1396.2 | 144.1 | 0.103 | 479 | 24.393 |

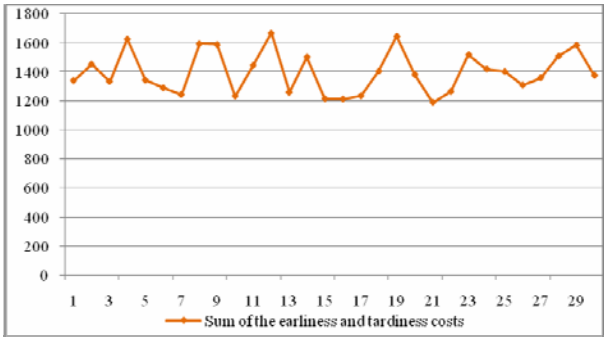


**Fig. 4. Line chart of table 4 (sum of earliness and tardiness costs)**

Elapsed time to solve several large-sized problems by proposed GA is given in table 5.

**Tab. 5. Numerical examples of large-sized problem**

| | GA (Iteration=100, Population size=100) | | |
|---|---|---|---|
| **Prob.** | **m** | **n** | **Time** (min) |
| 1 | 4 | 10 | 0.327 |
| 2 | 4 | 15 | 0.480 |
| 3 | 4 | 20 | 0.658 |
| 4 | 5 | 10 | 0.411 |
| 5 | 5 | 15 | 0.634 |
| 6 | 5 | 20 | 0.765 |
| 7 | 6 | 10 | 0.513 |
| 8 | 6 | 15 | 0.757 |
| 9 | 8 | 10 | 0.681 |
| 10 | 8 | 15 | 1.033 |
| 11 | 10 | 10 | 0.762 |
| 12 | 10 | 15 | 1.082 |

| GA (Iteration=100, Population size=100) | | | |
|---|---|---|---|
| **Prob.** | **m** | **n** | **Time** (min) |
| 13 | 10 | 20 | 1.242 |
| 14 | 10 | 30 | 1.409 |
| 15 | 20 | 10 | 0.799 |
| 16 | 20 | 20 | 1.801 |
| 17 | 20 | 30 | 2.877 |
| 18 | 30 | 30 | 3.489 |

If we increase the number of jobs whilst the number of machine is fixed, solving time according to proposed GA increases piecewise linearly (see fig. 5).

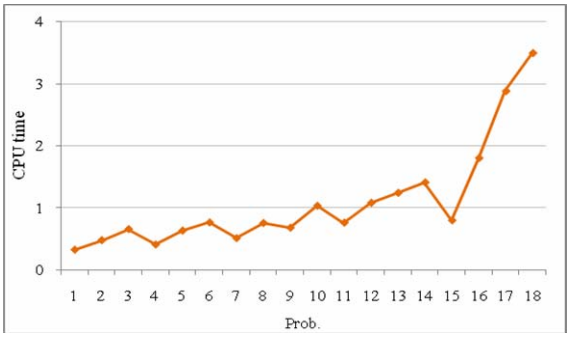


**Fig. 5. Line chart of elapsed time (Table 5)**

If we increase the number of machines whilst the number of jobs is fixed, solving time according to proposed GA increases piecewise linearly (see fig. 6).

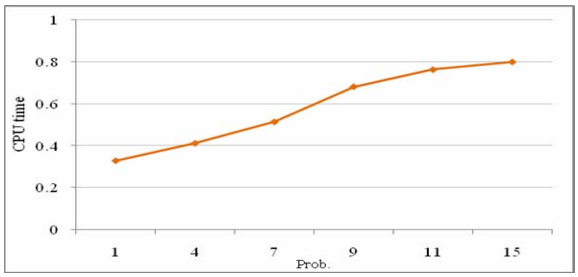


**Fig. 6. Line chart of elapsed time for fix 10 jobs (Table 5)**

## 6. Conclusions

In this paper, we presented a mathematical formulation model for minimizing sum of the earliness and tardiness costs in flow shop scheduling problem with bypass consideration (some jobs may not process on some machines) which is often occurring in shop environment of real world. We proposed genetic algorithm to solve this problem with medium and large size. Computational experiments have been performed to demonstrate that the proposed GA is efficient and flexible.

Further research can be done to use other meta-heuristics algorithms such as simulated annealing (SA), tabu search (TS), ant colony optimization (ACO). Hybrid algorithms should be developed by using a local search algorithm within a GA. This means that, after generating an offspring, the solution should be improved by applying for instance TS or SA before applying the selection criterion of GA.

## References

[1] Gonzalez, T., Sahni, S., *"Flow Shop and Job Shop Scheduling: Complexity and Approximation*," Operations Research, Vol. 26, 1978, pp. 36–52.

[2] Johnson, S., "*Optimal Two and Three Sage Production Schedules with Set-up Time Included*," Naval Research Logistics Quarterly, Vol. 1, 1954, pp. 61–68.

[3] Palmer, D.S., "*Sequencing Jobs Through a Multistage Process in the Minimum Total Time: A Quick Method of Obtaining a Near Optimum*," Operation Research Quarterly, Vol. 16, 1965, pp. 101–107.

[4] Campbell, H., Dudek, R., Smith, M., *"A Heuristic Algorithm for the n-Job, m-Machine Sequencing Problem,"* Management Science, Vol. 16, 1970, pp. 630–637.

[5] Gupta, J.N.D, *"Optimal Scheduling in a Multistage Flowshop*," AIIE Transactions, Vol. 4, 1972, pp. 238–243.

[6] Gupta, J.N.D., Neppall, V.R., Werner, F., *"Minimizing Total Flow Time in a Two-Machine Flow-Shop Problem with Minimum Makespan,"* International Journal Of Production Economics, Vol. 69, 2001, pp. 323–338.

[7] Tadei, R., Gupta, J.N.D., Croce, F.D., Cortesi, M., "*Minimising Makespan in the Two-Machine Flow-Shop with Release Times*," Journal of the Operational Research Society, Vol. 49, 1998, pp. 77–85.

[8] Simons, J.V., *"Heuristics in Flow Shop Scheduling with Sequence Dependent Setup Times*," Omega, Vol. 20, 1992, pp. 215–225.

[9] Huq, F., Cutright, K., Martin, C., "*Employee Scheduling and Makespan Minimization in a Flow Shop with Multiprocessor work Stations: a Case Study*," Omega, Vol. 32, 2004, pp. 121–129.

[10] Zobolas, G.I., Tarantilis, C.D., Ioannou, G., *"Minimizing Makespan in Permutation Flow Shop Scheduling Problems using a Hybrid Metaheuristic Algorithm*," Computers & Operations Research, vol. 36, 2009, pp. 1249 – 1267.

[11] Bulbul, K., Kaminsky, P., Yano, C., "*Flow Shop Scheduling with Earliness, Tardiness, and Intermediate Inventory Holding Costs,"* Naval Research Logistics, Vol. 51(3), 2004, pp. 407-445.

[12] Lauff, V., Werner, F., "*On the Complexity and Some Properties of Multi-Stage Scheduling Problems with Earliness and Tardiness Penalties*," Computers and Operations Research, Vol. 31(3), 2004, pp. 317-345.

[13] Leisten, R., Kolbe, M., "*A Note on Scheduling Jobs with Missing Operations in Permutation flow Shops*," International Journal of Production Research, Vol. 36(9), 1998, pp. 2627-2630.

[14] Glass, C.A., Gupta, J.N.D., Potts, C.N., "*Two-Machine No-Wait Flow Shop Scheduling with Missing Operations*," Mathematics of Operations Research, Vol. 24(4), 1999, pp. 911-924.

[15] Holland, J.H., Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press, 1975.

[16] Gen, M., Cheng, R., Genetic algorithms & engineering design, New York: Wiley, 1997.

[17] Gao, J., Gen, M., Sun, L., Zhao, X., *"A Hybrid of Genetic Algorithm and Bottleneck Shifting for Multiobjective Flexible Job Shop Scheduling Problems*," Computers & Industrial Engineering, Vol. 53, 2007, pp. 149–162.

[18] Gen, M., Tsujimura, Y., Kubota, E., *"Solving Job-Shop Scheduling Problem Using Genetic Algorithms*," In Proceeding of the 16th international conference on computer and industrial engineering, Ashikaga, Japan, 1994, pp. 576–579.