International Journal of Applied Operational Research
Vol. 4, No. 3, pp. 25-48, Summer 2014

Journal homepage: www.ijorlu.ir

A Hybrid Approach for Fuzzy Just-In-Time Flow Shop Scheduling with Limited Buffers and Deteriorating Jobs

M. Jannatipour*, B. Shirazi, I. Mahdavi

Received: 15 December 2013 **;** **Accepted:** 10 May 2014

Abstract This paper investigates the problem of just-in-time permutation flow shop scheduling with limited buffers and linear job deterioration in an uncertain environment. The fuzzy set theory is applied to describe this situation. A novel mixed-integer nonlinear program is presented to minimize the weighted sum of fuzzy earliness and tardiness penalties. Due to the computational complexities, the proposed mathematical model is NP-hard and therefore a hybrid meta-heuristic approach based on imperialist competitive algorithm and genetic algorithm (ICA-GA) is designed to tackle the considered flow shop scheduling problem. A set of random test problems with different structures are developed to evaluate the performance of this approach. The results illustrate the effectiveness of the presented model and hybrid algorithm for different problem sizes.

Keywords: Flow Shop Scheduling; Limited Buffer; Job Deterioration; Fuzzy Set Theory; Hybrid Algorithm.

1 Introduction

In classical flow shop scheduling problems, the capacity of intermediate buffer between any two consecutive machines are assumed to be infinite. In this situation, it is considered that jobs can be stored in the buffers for an unlimited time. This assumption obviously suggests a significant gap between theory and real-world manufacturing systems. Flow shop with limited intermediate buffers, is a generalization of classical flow shop environment. In this system, there are n jobs to be processed on m machines with first-in-first-out intermediate buffers. After finishing job process on a machine, it either has to be processed directly on the next machine or it has to be stored in the intermediate buffer. When buffer capacity completely occupied, the job has to remain on its current machine and this machine is blocked for other jobs. This blocking will be extended until the job, which momentarily is processed on the downstream machine, leaves that machine. Papadimitriou and Kanellakis [1] showed this problem is NP-hard in the strong sense even for two machines. For this reason, the most of

* **Corresponding Author.** (✉)
E-mail: Mjp_eng@ymail.com (M. Jannatipour)

M. Jannatipour
Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

B. Shirazi
Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

I. Mahdavi
Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

past studies focused on developing meta-heuristic algorithms for solving this problem. For instance, Norman [2] presented a tabu search method for flow shop scheduling problem with sequence-dependent setup times and finite intermediate buffers. Khosla [3] proposed a mixed integer linear model for a two-stage flow shop and developed two lower bounds and several heuristics to solve it. Nowicki [4],Wang et al. [5] and Liu et al. [6] presented a tabu search approach, a hybrid genetic algorithm and a hybrid particle swarm optimization to minimize makespan, respectively. Recently, Pan et al. [7,8] proposed a chaotic harmony search algorithm and a hybrid discrete differential evolution algorithm to solve the problem.

In classical flow shop systems, job processing times are also assumed to be constant. However, there are many manufacturing situations in which a job processed later needs more time than that same job processed earlier. First, Gupta and Gupta [9] investigated a scheduling problem in which the processing times depend on the jobs starting time with a polynomial function. This problem has been called scheduling with deteriorating jobs. They presented an example of steel rolling mills where the temperature of an ingot, while waiting to enter the rolling machine, drops below a certain level, requiring the ingot to be reheated before rolling. The flow shop scheduling problems with job deterioration is relatively unexplored. Mosheiov [10] addressed the makespan minimization problems under simple linear deterioration and proved that three-machine flow shop problem is NP-hard. Wang et al. [11] presented some polynomially solvable cases and developed several dominance properties and two lower bounds implemented in their proposed branch-and-bound approach. Wang and Xia [12] investigated no-wait flow shop scheduling problem with job deterioration. They showed that in this problem, polynomial-time algorithms exist to minimize the makespan and the weighted sum of completion times. Shiau et al. [13] addressed a simple linear deterioration model in a two-machine flow shop to minimize the mean flow time. Lee et al. [14] investigated a flow shop scheduling problem with job deterioration to minimize makespan. They developed an exact algorithm for solving the problems up to 32 jobs and a heuristic algorithm to solve large-sized cases. Ng et al. [15] considered a two-machine flow shop scheduling problem to minimize total completion time. They proposed lower bounds and dominance properties to speed up the proposed branch and bound algorithm. Yang and Wang [16] considered a two-machine flow shop scheduling problem to minimize total weighted completion time when processing times are a simple linear function of their execution start times. They presented several dominance properties and two lower bounds for the proposed branch-and-bound algorithm. Recently, Bank et al. [17] applied particle swarm optimization and simulated annealing algorithms to solve the problem. In another study, Bank et al. [18] developed a branch and bound algorithm to minimize total tardiness in a two-machine flow shop system with deteriorating jobs.

In addition to the above, in the real-world manufacturing systems, the time related parameters of jobs are often uncertain. There are basically two approaches to deal with this situation including the stochastic theory and fuzzy set theory. In the stochastic approach, uncertain data are modeled by specifying the probability distributions, for example inferred from past data. The fuzzy approach represents an alternative way to describe and model imprecision and uncertainty. In this work, fuzzy set theory is employed to handle the uncertainties in production scheduling. It provides a convenient alternative framework for modeling real-world systems mathematically and presents several advantages in the use of heuristic approaches namely [19]:

- The stochastic-probabilistic theory needs significant knowledge about the statistical distribution of the uncertain parameters. In contrast, fuzzy theory provides an efficient way to model imprecision even when no past data is available [20].

- The use of stochastic-probabilistic theory involves extensive computation and requires complete knowledge on the statistical distribution of the uncertain time parameters [21].
- Using fuzzy set theory decreases the computational efforts of the scheduling problem in comparison with the stochastic – probabilistic theory [22].
- Fuzzy theory is capable to make use of fuzzy rules in heuristic algorithms.
- Instead of optimizing the average behaviours, as in stochastic-probabilistic theory, fuzzy theory is looking for a solution to satisfy all constraints to some extent with a sufficient level of confidence.

Fuzzy scheduling has two major applications: scheduling under flexible constraints and scheduling under incomplete or imprecise information [23]. This study fits into the second class.

The concept of fuzzy sets was proposed by Zadeh [24]. Computing the fuzzy earliness and tardiness in scheduling problems were presented by Wu [25]. Recently Lai and Wu [26] have investigated the evaluation of the fuzzy completion times in the fuzzy flow shop scheduling problems using the virus-evolutionary genetic algorithms.

In this paper, a permutation flow shop system with limited intermediate buffers and deteriorating jobs is investigated under fuzzy environment. Here, a mixed integer nonlinear program is formulated to minimize the weighted sum of fuzzy earliness and tardiness penalties. This problem is in accordance with the concept of just-in-time (JIT) production. The earliness penalty is used for jobs which are produced earlier than their due dates and may cause inventory holding cost. In addition, the tardiness penalty is considered for missing a job's due date resulting dissatisfaction and lost sale. In this study, a hybrid meta-heuristic approach based on imperialist competitive algorithm (ICA) and genetic algorithm (GA) is proposed to find efficient solutions in reasonable computational times.

The paper is organized as follows. Section 2 gives brief basics about fuzzy set theory. In section 3, the proposed problem is described and modeled. Section 4 introduces the proposed solution approaches. Section 5 presents the experimental design which compares the achieved results of meta-heuristic approaches. Finally, Section 6 consists of conclusions and future work.

2 Fuzzy set theory

In this section, some related concepts of fuzzy set theory, which are necessary for the considered problem, are reviewed. Subsections 2.1 and 2.2 are assigned to the definition of fuzzy numbers and fuzzy earliness / tardiness calculations, respectively.

2.1 Fuzzy numbers

The fuzzy subset of real numbers R is defined by a function $\mu_{\tilde{a}} : R \to [0,1]$, called membership function of. The $\alpha - level$ set of $\tilde{a}$, denoted by $\tilde{a}_\alpha$, is defined by $\tilde{a}_\alpha = \{x \in R : \mu_{\tilde{a}} \geq \alpha\}$ for all $\alpha \in (0,1]$. It is seen that if is a fuzzy number, then the set of is a closed, bounded and convex subset of *R*, namely a closed interval in *R*. In this case, it is denoted by $\tilde{a}_\alpha = [\tilde{a}_\alpha^L, \tilde{a}_\alpha^U]$.

Proposition 2.1.1. Let $\tilde{a}$ and $\tilde{b}$ be two fuzzy numbers. Then $\tilde{a} \oplus \tilde{b}$ and $\tilde{a}\ \Theta\ \tilde{b}$ are also fuzzy numbers:

$$(\tilde{a} \oplus \tilde{b})_\alpha = [\tilde{a}_\alpha^l + \tilde{b}_\alpha^l, \tilde{a}_\alpha^u + \tilde{b}_\alpha^u] \quad (1)$$

$$(\tilde{a} \Theta \tilde{b})_\alpha = [\tilde{a}_\alpha^l - \tilde{b}_\alpha^u, \tilde{a}_\alpha^u - \tilde{b}_\alpha^l] \quad (2)$$

Furthermore,

$$(\max\{\tilde{a}, \tilde{b}\})_\alpha = [\max\{\tilde{a}_\alpha^l, \tilde{b}_\alpha^l\}, \max\{\tilde{a}_\alpha^u, \tilde{b}_\alpha^u\}] \quad (3)$$

In the application of fuzzy theory, the triangular and trapezoidal fuzzy numbers are utilized, frequently. In this study, processing times and due dates are considered as triangular and trapezoidal fuzzy numbers, respectively.

The triangular fuzzy number $\tilde{a}$ is denoted by $\tilde{a} = (a^l, a, a^u)$ and its membership function is defined by:

$$\mu_{\tilde{a}}(r) = \begin{cases} (x - a^l)/(a - a^l) & if\ a^l \le x \le a \\ (a^u - x)/(a^u - a) & if\ a < x < a^u \\ 0 & otherwise \end{cases} \quad (4)$$

The $\alpha - level$ set of $\tilde{a}$ is: $\tilde{a}_\alpha = [(1-\alpha)a^l + \alpha a, (1-\alpha)a^u + \alpha a]$ That is, $\tilde{a}_\alpha^l = (1-\alpha)a^l + \alpha a$ *and* $\tilde{a}_\alpha^u = (1-\alpha)a^u + \alpha a$.

It can be shown that $\tilde{a} \oplus \tilde{b}$ calculated through Eq. (5) is also a triangular fuzzy number. $\tilde{a} \oplus \tilde{b} = (a^l, a, a^u) \oplus (b^l, b, b^u) = (a^l + b^l, a + b, a^u + b^u)$ (5)

The trapezoidal fuzzy number is also introduced to describe the fuzzy due dates. For a trapezoidal fuzzy number denoted as $\tilde{a} = (a^l, a_1, a_2, a^u)$, the membership function is given by:

$$\mu_{\tilde{a}}(x) = \begin{cases} (x - a^l)/(a_1 - a^l) & if\ a^l \le x \le a_1 \\ 1 & if\ a_1 < x \le a_2 \\ (a^u - x)/(a^u - a_2) & if\ a_2 < x \le a^u \\ 0 & otherwise \end{cases} \quad (6)$$

It can be seen that: $\tilde{a}_\alpha^l = (1-\alpha)a^l + \alpha a_1$ *and* $\tilde{a}_\alpha^u = (1-\alpha)a^u + \alpha a_2$ where $\tilde{a}_\alpha = [\tilde{a}_\alpha^l, \tilde{a}_\alpha^u]$.

In this study, processing times and due dates are considered as $\tilde{p}_{ij} = (p_{ij}^L, p_{ij}, p_{ij}^U)$ and $d_j = (d_j^L, d_{j1}, d_{j2}, d_j^U)$ respectively.

2.2 Fuzzy Earliness and Tardiness

From Proposition 2.1.1, the values of earliness $\tilde{E}_j$ and tardiness $\tilde{T}_j$ of each job j are fuzzy numbers and is defined as follows:

$$\tilde{E}_j = \tilde{\max}\{0, \tilde{d}_j \Theta \tilde{C}_j\} \tag{7}$$

and $$\tilde{T}_j = \tilde{\max}\{0, \tilde{C}_j \Theta \tilde{d}_j\}. \tag{8}$$

Therefore, the $\alpha - level$ closed intervals of $\tilde{E}_j$ and $\tilde{T}_j$ can be transacted by:

$$\tilde{E}_{j\alpha}^{L} = \max\{0, \tilde{d}_{j\alpha}^{L} \Theta \tilde{C}_{j\alpha}^{U}\},\ \tilde{E}_{j\alpha}^{U} = \max\{0, \tilde{d}_{j\alpha}^{U} \Theta \tilde{C}_{j\alpha}^{L}\}, \tag{9}$$

$$\tilde{T}_{j\alpha}^{L} = \max\{0, \tilde{C}_{j\alpha}^{L} \Theta \tilde{d}_{j\alpha}^{U}\},\ \tilde{T}_{j\alpha}^{U} = \max\{0, \tilde{C}_{j\alpha}^{U} \Theta \tilde{d}_{j\alpha}^{L}\}. \tag{10}$$

As mentioned above, the proposed objective function is the weighted sum of fuzzy earliness and tardiness penalties:

$$f(\pi) = \sum_{j=1}^{n} (e_j \tilde{E}_j \oplus t_j \tilde{T}_j) \tag{11}$$

For minimizing this fuzzy-valued objective function, the ranking method proposed by Fortemps and Roubens [27] is applied. In this method, for any two fuzzy numbers $\tilde{a}$ and $\tilde{b}$, $\tilde{a} \le \tilde{b}$ if and only if $\eta(\tilde{a}) \le \eta(\tilde{b})$, where $\eta(\tilde{a})$ is calculated through Eq. (12).

$$\eta(\tilde{a}) = \frac{1}{2}\int_0^1 (\tilde{a}_\alpha^L + \tilde{a}_\alpha^U) d\alpha. \tag{12}$$

Given that:

$$\tilde{f}_\alpha(\pi) = [\tilde{f}_\alpha^L(\pi), \tilde{f}_\alpha^U(\pi)] = [\sum_{j=1}^{n} (e_j \tilde{E}_{j\alpha}^L + t_j \tilde{T}_{j\alpha}^L), \sum_{j=1}^{n} (e_j \tilde{E}_{j\alpha}^U + t_j \tilde{T}_{j\alpha}^U)], \tag{13}$$

and substituting Eq. (13) in Eq. (12), it is seen that $\eta(\tilde{f}(\pi)) = \frac{1}{2}\sum_{j=1}^{n} (e_j h_j + t_j u)$ where:

$$h_j = \int_0^1 \max\{0, \tilde{d}_{j\alpha}^L - \tilde{C}_{j\alpha}^U\} d\alpha + \int_0^1 \max\{0, \tilde{d}_{j\alpha}^U - \tilde{C}_{j\alpha}^L\} d\alpha \tag{14}$$

and

$$u_j = \int_0^1 \max\{0, \tilde{C}_{j\alpha}^L - \tilde{d}_{j\alpha}^U\} d\alpha + \int_0^1 \max\{0, \tilde{C}_{j\alpha}^U - \tilde{d}_{j\alpha}^L\} d\alpha. \tag{15}$$

The above $\eta(\tilde{f}(\pi))$ is calculated considering the fuzzy processing times $\tilde{p}_{ij} = (p_{ij}^L, p_{ij}, p_{ij}^U)$ as triangular fuzzy numbers and the fuzzy due dates $d_j = (d_j^L, d_{j1}, d_{j2}, d_j^U)$ as trapezoidal fuzzy numbers for $j = 1, ..., n$ and $i = 1, ..., m$. Therefore, it is seen that:

$(\tilde{d}_j)_\alpha^L = (1-\alpha)d_j{}^L + \alpha d_{j1}$ and $(\tilde{d}_j)_\alpha^U = (1-\alpha)d_j{}^U + \alpha d_{j2}$.

Based on the triangular shape of the processing times, the fuzzy completion time of each job j ($\tilde{C}_j$) will turn out to be a triangular fuzzy number $\tilde{C}_j = (C_j^L, C_j, C_j^U)$. Therefore:

$$(\tilde{C}_j)_\alpha^L = (1-\alpha)C_j{}^L + \alpha C_j \text{ and } (\tilde{C}_j)_\alpha^U = (1-\alpha)C_j{}^U + \alpha C_j. \quad (16)$$

Now, regarding the graph of $\tilde{C}_j$ as triangle and the graph of $\tilde{d}_j$ as trapezoid, there will be five cases describing their positional relations [25]. In other words, the following five cases are supposed to be discussed in order to calculate h_j and u_j in Eqs. (14) and (15):

Case (I): If $C_j^U <= d_j^L$ then

$$e_j h_j + t_j u_j = \frac{1}{2} e_j (d_j^L + d_{j1} + d_{j2} + d_j^U - C_j^L - 2C_j - C_j^U). \quad (17)$$

Case (II): If $C_j <= d_{j1}$ and $C_j^U >= d_j^L$ then

$$e_j h_j + t_j u_j = \frac{1}{2} e_j (d_j^L + d_{j1} + d_{j2} + d_j^U - C_j^L - 2C_j - C_j^U) + \frac{1}{2}(e_j + t_j)\frac{(C_j^U - d_j^L)^2}{C_j^U - C_j + d_{j1} - d_j^L}. \quad (18)$$

Case (III): If $d_{j1} <= C_j <= d_{j2}$, then

$$e_j h_j + t_j u_j = \frac{1}{2} e_j (d_j^U + d_{j2} - C_j^L - C_j) + \frac{1}{2} t_j (C_j^U + C_j - d_j^L - d_{j1}). \quad (19)$$

Case (IV): If $C_j >= d_{j2}$ and $C_j^L <= d_j^U$ then

$$e_j h_j + t_j u_j = \frac{1}{2} t_j (C_j^L + 2C_j + C_j^U - d_j^L - d_{j1} - d_{j2} - d_j^U) + \frac{1}{2}(e_j + t_j)\frac{(d_j^U - C_j^L)^2}{C_j - C_j^L + d_j^U - d_{j2}}. \quad (20)$$

Case (V): If $d_j^U <= C_j^L$ then the graph of $\tilde{d}_j$ is completely on the left of the graph of $\tilde{C}_j$, which gives:

$$e_j h_j + t_j u_j = \frac{1}{2} t_j (C_j^L + 2C_j + C_j^U - d_j^L - d_{j1} - d_{j2} - d_j^U). \quad (21)$$

For each job j, one of the above five cases (I)–(V) will be observed, and $e_j h_j + t_j u_j$ can be obtained. In this work, ICA, GA and hybrid ICA-GA methods are designed to find a near optimal schedule.

3 Problem Definition

In this section, the considered problem is defined, and a novel mixed integer non-linear mathematical model is formulated. The problem can be stated as follows. *n* jobs from a set *j= {1,2,3,...,n}* will be sequentially processed on machine 1, machine 2, and so on until last machine *m*. Preemption and machine idle time is not allowed. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same on all machines. Between each successive pairs of machines *i* and *i+1*, there exists a limited buffer with the capacity size equal to $B_i \geq 0$. Each job must go through the buffer B_i on its route from machine *i* to *i+1*, and jobs obey the First-In-First-Out rule in the buffer. If the buffer B_i is completely occupied, the job has to remain on the current machine *i* until a free place is available in the buffer. The release time of all jobs is zero and the setup time on each machine is included in the processing time. The jobs are also assumed to be deteriorating. The processing times of jobs are considered as linear function of their starting process times on machines. Processing times and due dates are also assumed to be triangular and trapezoidal fuzzy numbers, respectively. Transportation times between machines are negligible and processors are available with no breakdowns. The aim is to find a sequence for processing all jobs on all machines so that the weighted sum of fuzzy earliness and tardiness penalties is minimized.

3.1 Notations

Indices	
i	index for machines
j	index for jobs
k	index for job positions in a sequence
parameters	
n	number of jobs
m	number of machines
λ_j	Deterioration rate of job *j*
e_j	Earliness penalty of job *j*
t_j	Tardiness penalty of job *j*
$\tilde{p}_{i,j}$	Normal fuzzy processing time of job *j* on machine *i*
$\tilde{d}_j$	Fuzzy due date of job *j*
B_i	buffer capacity between two successive machines *i*, *i+1*
Decision variables	
$\tilde{C}_{i,k}$	Fuzzy completion time of *k*th job on machine *i*
$\tilde{C}_j$	Final fuzzy completion time of job *j*
$\tilde{E}_j$	Fuzzy earliness of job *j*
$\tilde{T}_j$	Fuzzy tardiness of job *j*
x_{jk}	1 If job j is selected for sequence position *k*,
	0 otherwise

3.2 The Mathematical Model

In this section, a Mixed Integer nonlinear Programming (MINP) mathematical model is presented. The objective function and constraints can be formulated as follows:

$$\min \quad z = \sum_{j=1}^{n} (e_j \tilde{E}_j \oplus t_j \tilde{T}_j) \tag{22}$$

s.t.

$$\sum_{k=1}^{n} x_{jk} = 1 \qquad \forall j \tag{23}$$

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad \forall k \tag{24}$$

$$\tilde{C}_{1,1} = \sum_{j=1}^{n} (\tilde{p}_{1,j} . x_{j1}) \tag{25}$$

$$\tilde{C}_{i,1} = \sum_{j=1}^{n} ((\tilde{C}_{i-1,1} \oplus \tilde{p}_{i,j}) . x_{j1}) \qquad i = 2,3,...,m \tag{26}$$

$$\tilde{C}_{1,k} = \sum_{j=1}^{n} ((\tilde{C}_{1,k-1} \oplus \tilde{p}_{1,j}) . x_{jk}) \quad k = 2,3,...,B_1 + 1 \tag{27}$$

$$\tilde{C}_{1,k} = \sum_{j=1}^{n} ((\max(\tilde{C}_{1,k-1} \oplus \tilde{p}_{1,j}, \tilde{C}_{2,k-B_1-1})) . x_{jk}) \quad k > B_1 + 1 \tag{28}$$

$$\tilde{C}_{i,k} = \sum_{j=1}^{n} (((\max(\tilde{C}_{i,k-1}, \tilde{C}_{i-1,k}))(1+\lambda_j) \oplus \tilde{p}_{i,j}) . x_{jk}) \qquad i = 2,3,...,m-1$$
$$k = 2,3,...,B_i + 1 \tag{29}$$

$$\tilde{C}_{i,k} = \sum_{j=1}^{n} (\max((\max(\tilde{C}_{i,k-1}, \tilde{C}_{i-1,k}))(1+\lambda_j) \oplus \tilde{p}_{i,j}, \tilde{C}_{i+1,k-B_i-1}) . x_{jk}) \quad i = 2,3,...,m-1 \tag{30}$$
$$k > B_i + 1$$

$$\tilde{C}_{m,k} = \sum_{j=1}^{n} (((\max(\tilde{C}_{m,k-1}, \tilde{C}_{m-1,k}))(1+\lambda_j) \oplus \tilde{p}_{m,j}) . x_{jk}) \quad k = 2,3,...,n \tag{31}$$

$$\tilde{C}_j = \sum_{k=1}^{n} (\tilde{C}_{m,k} . x_{jk}) \quad j = 1,2,...,n \tag{32}$$

$$\tilde{E}_j = \max(0, \tilde{d}_j \Theta \tilde{C}_j) \quad j = 1,2,...,n \tag{33}$$

$$\tilde{T}_j = \max(0, \tilde{C}_j \Theta \tilde{d}_j) \quad j = 1,2,...,n \tag{34}$$

$$x_{jk} \in \{0,1\} \; j = 1,2,...,n \qquad k = 1,2,...,n \tag{35}$$

Eq. (22) shows the objective function which is the minimization of the weighted sum of fuzzy earliness and tardiness penalties of all jobs. Constraint (23) ensures that each job is assigned to exactly one sequence position and constraint (24) ensures that in each sequence position, one and only one job is processed. Constraints (25) determine the fuzzy completion time of the first job on machine 1. Constraints (26) illustrate the fuzzy completion time of the first job on machine *i*. Constraint (27) and (28) are related to the fuzzy completion time of the *k*th job on the first machine regarding the job sequence, buffer capacity and blocking possibility. Constraint (29) and (30) give the completion time of the *k*th job on machine *i* taking into account buffer capacity, blocking possibility and deterioration rate.

The actual fuzzy processing times of the jobs that have to wait before being processed on a machine *i* will be calculated in accordance with the fuzzy starting time and the deterioration rate. In other words, as shown in Eq. (36), the processing time of each job on each machine is a linear function of its starting time [28]:

$$\tilde{p}_{[i,j]} = \tilde{p}_{i,j} \oplus \lambda_j (\tilde{s}_{i,j}) \tag{36}$$

where is the fuzzy actual processing time of job *j* on machine *i* and is the fuzzy starting time of job *j* on machine *i*.

Obviously, the longer a job has to wait for being processed, the longer is its actual processing time. In constraints (25) to (28), disregarding the deterioration of jobs, the normal processing times have been applied because the mentioned jobs will not wait before being processed.

The fuzzy completion time of the *k*th job on the last machine is calculated in constraints (31) considering deterioration rates. Constraint (32) determines the fuzzy completion time of job *j* after all the stages of the flow shop system. For each job, constraints (33) and (34) give the fuzzy earliness and tardiness values, respectively. Finally, constraint (35) indicates that the variable x_{jk} is binary.

4 Proposed algorithms

In this section, the proposed genetic, imperialist competitive and hybrid algorithms will be introduced.

4.1 A genetic algorithm solution approach

In general, the both exact and heuristic algorithms can be used to solve scheduling problems. Exact methods are only practical for small instances and even in that case, the computational time tends to be very high. Therefore, researchers have focused to develop heuristics. Genetic algorithm (GA) is a well-known meta-heuristic approach inspired by the natural evolution of the living organisms. GA is applied to tackle both discrete and continuous optimization problems. Generally, the input of the GA is a set of solutions called population of individuals that will be evaluated. A fitness value is assigned to each solution (chromosome) according to its performance. The population evolves by a set of operators until some stopping criterion is visited. The chromosome representation and the designed genetic operators are two important challenges in the design of GA. A detailed description of main factors for the proposed GA is reported below.

4.1.1 Initialization

The initial population consists of *Pop_size* chromosomes of solutions that each chromosome is related to a candidate solution of the problem. The most frequently used encoding scheme for the flow shop scheduling problem, is a simple permutation of jobs [29]. The relative order of jobs in the permutation illustrates the processing order of jobs on all machines in the shop. To qualify encoding scheme, the permutation of jobs is shown through random keys (RK). Each job has a random number between 0 and 1, and these RKs show the relative order of the jobs. In this paper, the largest RK value is firstly handled and assigned the smallest rank value 1, the second largest RK value is addressed and assigned a rank value 2, and so on. For example, the encoded solution {0.47, 0.83, 0.51, 0.12, 0.26} represents the permutation {2, 3, 1, 5, 4} (Fig 1).

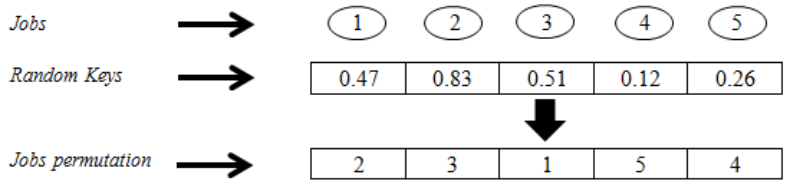

Fig. 1 Jobs permutation generation

4.1.2 Evaluation

The fitness value of each chromosome is evaluated by Eq. (37).

$$f(k) = 1/(1+\sum_{j=1}^{n}(e_j h_j + t_j u_j)) \qquad k = 1,...,pop\_size \tag{37}$$

where, $f(k)$ is the fitness value of kth chromosome in a population and $\sum_{j=1}^{n}(e_j h_j + t_j u_j)$ is the defuzzified objective function of the problem.

4.1.3 Parent selection strategy

The parent selection strategy means how to choose chromosomes in the current population that will create off-spring for the next generation [30]. The most common method for the selection mechanism is the "*roulette wheel*" sampling. In this method each chromosome is selected based on probability proportionated to its fitness value (Eq. 38). Solutions with higher fitness value have more chance to be in the pool of parents for creation of off-springs. A chromosome can be selected as a parent one more time.

$$PV(k) = f(k)/(\sum_{j=1}^{pop\_size} f(j)) \qquad k = 1,...,pop\_size \tag{38}$$

4.1.4 Design of genetic operators

4.1.4.1 Crossover operator

In this paper, uniform crossover namely position-based operator [31], is applied. The steps of this method are introduced as follows:

1. Randomly choose two sequences from the population as two parents.

Parent 1	3	2	5	1	4

Parent 2	4	1	3	5	2

2. Create binary string (BS) and assign a randomly generated binary (0,1) to each cell.

BS	0	1	0	0	1

3. Copy the genes from the parent 1 to the locations of the ‘‘1’’s in the binary string to the same positions in the offspring.

Offspring		2			4

4. The genes that have already been selected from the parent 1 are deleted from the parent 2, so that the repetition of a gene in the new offspring is avoided.

Parent 2	-	1	3	5	-

5. Complete the remaining empty gene locations with the undeleted genes that remain in the parent by preserving their gene sequence in parent 2.

Offspring	1	2	3	5	4

4.1.4.2 Mutation operator

The main purpose of applying mutation is to maintain the diversity of the population in the successive generations and to avoid convergence to a local optimum. In this study, a mutation operator, called single point mutation (SPO) is used. The procedure of SPO can be defined as follows: the RK of a randomly selected job is randomly regenerated and then, the permutation of jobs is rewritten. An illustrative example is shown in Fig 2.

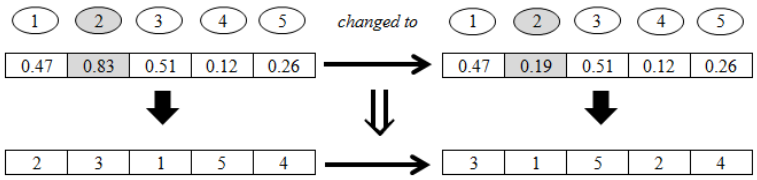

Fig. 2 Single point mutation

4.1.4.3 Reproduction operator

In this paper, an elitism strategy is applied as reproduction operator. In this strategy, the best chromosomes are automatically copied to the next generation.

4.1.5 Stopping criteria

In this study, two stopping criteria are applied: (1) maximum number of elapsed generation (G_{max}) and (2) number of successive generations when the algorithm doesn't have any improvement. This number is considered as $0.25*G_{max}$.

Therefore, the algorithm runs until one of these stopping criteria is visited.

The values of the control parameters of the proposed GA are determined using Taguchi experimental design method as follows: pop_size =70, P_c = 0.6, P_m= 0.12 and G_{max}=150.

4.2 Imperialist competitive algorithm

According to the world history, imperialism is a policy for expanding the authority of a powerful government. By acquiring territory or dominating the economic and political systems of other countries, an imperialist try to attempt to master other feebler countries. Also in some cases the reason of control another country was just preventing the opponent imperialist from taking possession of it. This competition resulted in a growth of the great empires and fall down of weaker ones [32]. Imperialist competitive algorithm, proposed by Atashpaz-Gargari and Lucas [33] is a novel global search heuristic that uses imperialistic competition process as a source of inspiration.

4.3 Hybrid ICA-GA

The performance of the proposed hybrid imperialist competitive algorithm- genetic algorithm (ICA-GA) can be divided into two sections. The first section is producing an appropriate solution. In fact, the first part is an input for the second one. The task of the second part is improving the population into a near optimal sequence. Since ICA make more searches on different sequences of JIT flow shop due to its high convergence rate and execution speed, it is used as an appropriate tool for producing the appropriate and good first population for GA. On the other hand, GA can be a good tool for improving the population due to its accuracy in finding near global optimum solutions. The main steps of the designed hybrid algorithm are summarized in the pseudo code shown in Fig. 3.

5 Computational results

5.1 Data Generation

To solve the presented mathematical model, and for the purpose of evaluating the effectiveness of the proposed hybrid algorithm, a number of test problems are randomly generated with different structures. Input data, such as number of jobs, number of machines, processing times, due dates, deterioration rates and buffer capacities, are generated as shown in Table 1.

As can be seen in Table 1, the values of d_{j2} were generated between $(1-\tau-R/2)M$ and, where and R are two parameters calling tardiness factor and due date range. In this study, it is considered that $\tau = 0.2, 0.6$ and $R = 0.6, 1.6$. These values typically cover various problems and hence, they are appropriate for the earliness/tardiness objective function [34]. M is the maximum completion times of all jobs that are obtained from Johnson's order.

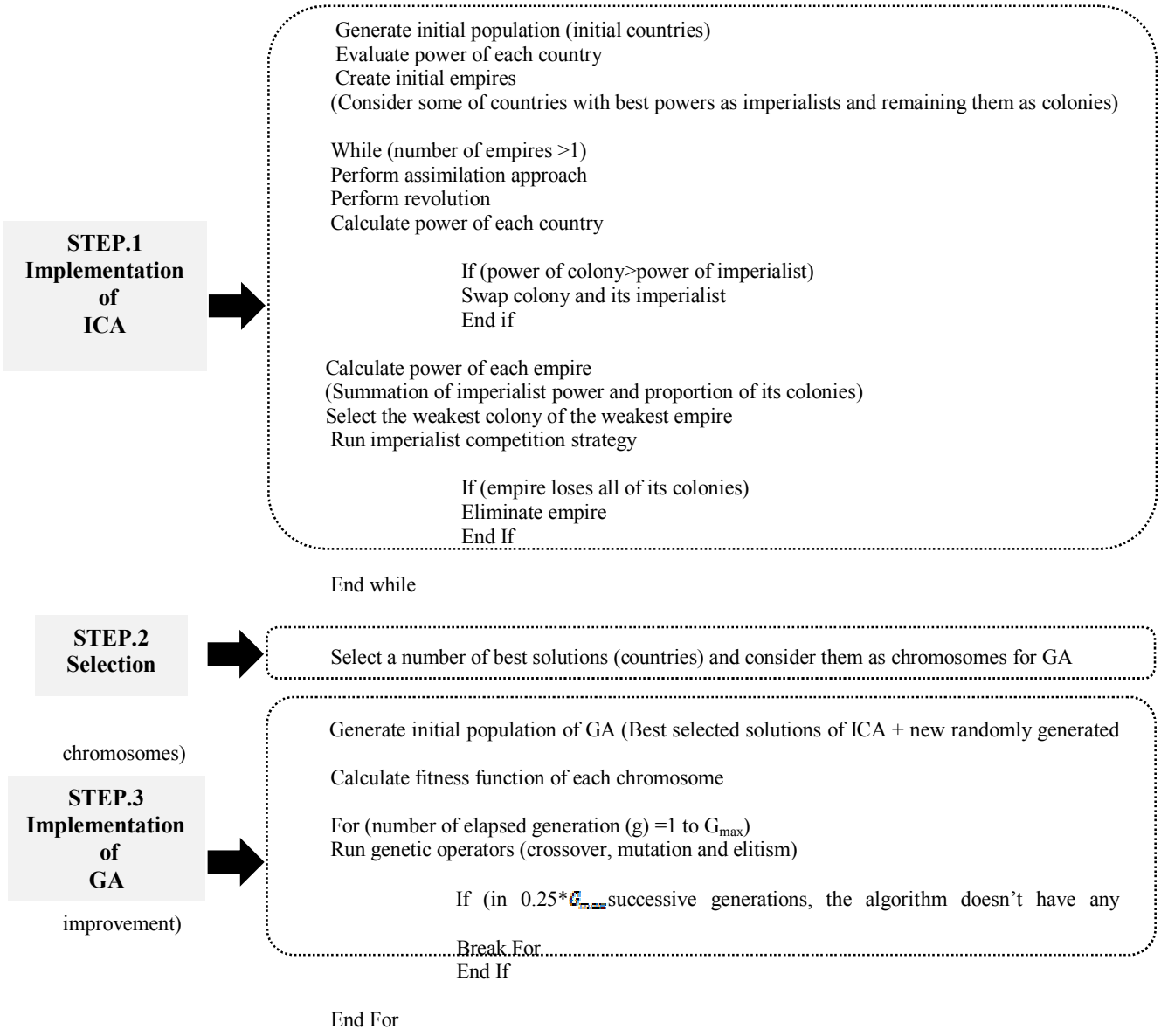

Fig.3 The main steps of the designed hybrid algorithm

To produce trapezoidal fuzzy numbers $\tilde{d}_j$, and triangular fuzzy numbers $\tilde{p}_{ij}$ the following methods are used:

$\tilde{d}_j = (d_{j2} - w_j - w'_j, d_{j2} - w_j, d_{j2}, d_{j2} + w_j)$, $\tilde{p}_{ij} = (p_{ij} - w_{ij}, p_{ij}, p_{ij} + w_{ij})$.

The values of controllable parameters for each type of numerical instance are presented in Table 2. Table 3 shows an example in small size, for a given problem of *type a* with five jobs and three machines. The computational results of the given test problem is shown in Table 4. This table contains fuzzy completion time (), sum of the fuzzy earliness and tardiness and the assigned position to each job (optimal jobs sequence) and finally, the obtained optimal value of objective function. As can be seen in Table 4, the optimal sequence of the given problem is (3, 5, 1, 4, 2) and the optimal value of objective function is 25.0738.

Table 1 Information for the data generation

Parameter	Values
No. of jobs (n)	4, 5, 6, 8, 10, 20, 30, 50,80
No. of machines (m)	3, 4, 5, 10, 15
Processing time (p_{ij})	U[10,100]
Due date ()	U[$(1-\tau-R/2)M$, $(1-\tau+R/2)M$]
Tardiness factor (τ)	0.2, 0.6
due date range (R)	0.6, 1.6
w_j, w'_j, w_{ij}	U[1, 5]
deterioration rate ()	U[0, 0.01]
earliness and tardiness penalties ()	U[0, 0.1]

Table 2 The values of controllable parameters for each type of numerical instance.

Type	*T*	*R*	Range of d_{j2}
a	0.2	0.6	[0.5M, 1.1M]
b	0.2	1.6	[0, 1.6M]
c	0.6	0.6	[0.1M, 0.7M]
d	0.6	1.6	[0, 1.2M]

Table 3 sample problem data.

Job	$\tilde{p}_{1,j}$	$\tilde{p}_{2,j}$	$\tilde{p}_{3,j}$	d_j	λ_j	e_j	t_j
1	(16.84,18.78,19.72)	(33.31,35.06,39.82)	(53.26,59.22,62.18)	(277.11,283.06,289.01,297.96)	0.002624	0.057907	0.021736
2	(65.57,69.02,78.47)	(12.55,13.21,14.87)	(81.10,86.42,91.74)	(461.40,469.31,471.42,477.11)	0.001351	0.095112	0.09428
3	(81.91,84.11,88.32)	(68.91,72.53,73.16)	(36.61,38.54,42.47)	(348.15,355.38,364.61,372.84)	0.008552	0.033409	0.074658
4	(32.10,34.84,36.58)	(68.61,71.17,75.73)	(61.51,68.96,72.41)	(303.40,311.31,318.22, 327.13)	0.003751	0.004783	0.003977
5	(90.52,96.34,101.15)	(56.29,59.25,64.21)	(18.35,22.48,23.60)	(516.06,524.34,532.62,541.90)	0.002859	0.056954	0.068037

Table 4 Results of the given problem.

Job	C_j	Positional relation	$0.5*(e_j h_j + t_j u_j$	$x_{jk}=1$
1	(317.07,335.85,359.67)	V	2.1878	x_{13}
2	(477.11,505.36,547.72)	II	5.6487	x_{25}
3	(187.43,195.18,207.95)	I	10.9454	x_{31}
4	(395.48,418.38,451.37)	I	0.4678	x_{44}
5	(248.22,263.38,280.55)	I	5.8242	x_{52}
Total			**25.0738**	

5.2 Experimental Results

Three algorithms GA, ICA and hybrid ICA-GA are applied to solve the considered scheduling problem. These algorithms have been coded using MATLAB 7.4 and executed in the Intel Core (TM) 2 Due 2.53 GHz and 4GB RAM personal computer. Before that, a number of random small-sized problems are solved using the optimal solution approach B&B employing the LINGO 9.0 software, and their computational times are evaluated. The information of the small-sized problems has been shown in Table 5.

Table 5 The information of the random type small-sized problems

Problem	Problem information		type
	No. of jobs	No. of machines	
1	4	3	c
2	4	4	b
3	5	3	c
4	5	4	d
5	6	3	a
6	6	4	a
7	8	3	b
8	8	4	c
9	10	3	d
10	10	4	a

Obtained results of these small-sized problems have been shown in Table 6. As can be seen in Table 6, computational time of B&B method, even for small-sized problem is very long. For instance, for a given test problem with 10 jobs, the optimal solution has not been achieved after 12 hours. This can justify the use of meta-heuristic approaches to solve the considered fuzzy flow shop scheduling problem.

At first, the genetic and imperialist competitive algorithms are applied to solve the problem, separately. For small-sized problems, these two algorithms reach to the optimal solution.

Table 6 Computational mean times to obtain optimal solutions by B&B, GA and ICA.

Problem	B&B			GA		ICA	
	Best solution	Optimal solution	Mean CPU time[a]	Best solution	Mean CPU time[a]	Best solution	Mean CPU time[a]
1	8.03	8.03	00:00:03	8.03	00:00:01	8.03	00:00:01
2	25.22	25.22	00:00:12	25.22	00:00:01	25.22	00:00:02
3	14.26	14.26	00:02:11	14.26	00:00:03	14.26	00:00:03
4	139.47	139.47	00:03:28	139.47	00:00:04	139.47	00:00:04
5	159.19	159.19	00:06:54	159.19	00:00:05	159.19	00:00:06
6	181.65	181.65	01:21:31	181.65	00:00:06	181.65	00:00:06
7	231.11	231.11	04:02:21	231.11	00:00:06	231.11	00:00:07
8	70.31	70.31	08:52:37	70.31	00:00:09	70.31	00:00:08
9	373.41	-	12:00:00	233.41	00:00:12	233.41	00:00:12
10	1021.37	-	12:00:00	839.37	00:00:15	839.37	00:00:14

[a] Computational time (hour: minute: second).

However, the final solution of the GA is better than ICA in medium and large-sized instances, but the convergence rate and execution speed of ICA is higher than GA. A typical comparative performance of these two algorithms has been shown in Fig. 4. (As stated, ICA can be stopped in 2 ways, continuing until only one empire exists and continuing until a specified iteration is achieved.)

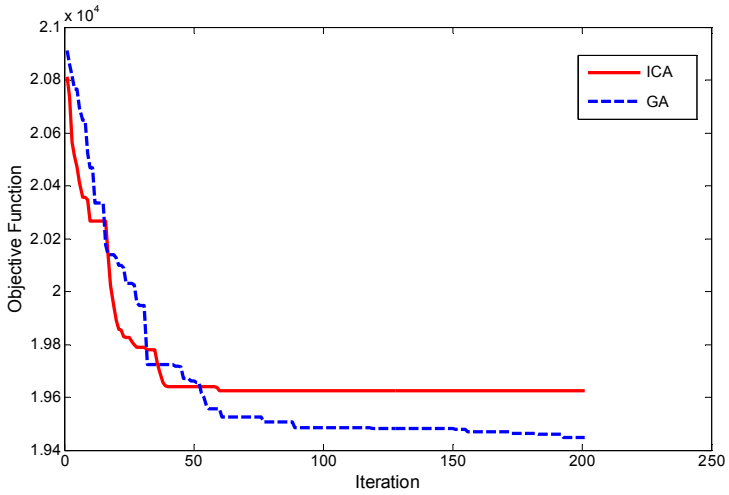

Fig. 4 Comparison between objective function values and convergence rates of GA and ICA for a given problem with 30 jobs and 10 machines.

The hybrid algorithm has been proposed to achieve good final solutions with reasonable computational time especially in medium and large-sized cases. The performance of this algorithm is compared with GA and ICA for a large number of instances with different structures.

In order to provide a better computational experience, comparing the results with other works is forehead. By reviewing the literature, we found a similar work, of course with different structure, for our comparison purpose. The literature review specifies that there is no study on scheduling problems on permutation flow shop with limited intermediate buffers, deteriorating jobs, earliness/tardiness and fuzzy parameters simultaneously. So, we compare the proposed ICA-GA with a particle swarm optimization with local search (PSO_{LS}) method proposed in [17].

Table 7 shows the obtained results of four algorithms for medium and large-sized problems. Each case is solved 10 times and finally, the best solution is reported.
As shown in Table 7, the solutions obtained by the hybrid meta-heuristic approach are more efficient than the final solutions of medium and large-sized problems reported by other algorithms. In fact, the final solution of hybrid algorithm is better than other three algorithms.
The most common performance measure used in the literature to compare four algorithms is the relative percentage deviation (RPD) which is computed as follows.

$$RPD = \frac{sol_{a\lg} - sol_{\min}}{sol_{\min}} * 100 \tag{39}$$

where $sol_{a\lg}$ is the average solution value obtained for a given algorithm after 10 runs, and $sol_{\min}$ is the best solution obtained for each instance by any of the four algorithms. Tables 8 and 9, show the average RPD of four algorithms for medium and large-sized cases, respectively.
Obviously, the hybrid algorithm provides better results than GA, ICA and PSO_{LS}. In order to verify the statistical validity of the results shown in Tables 8 and 9, and to determine the best algorithm, design of experiments and analysis of variance (ANOVA) is performed and different algorithms are considered as a factor and the response variable RPD. The means plot and LSD intervals (at the 90% confidence level) of four algorithms are shown in Fig. 5 and Fig. 6 for medium and large-sized instances, respectively. The results demonstrate that there is a clear statistical difference among the performances of the algorithms.

Table 7 The quality of results obtained for GA, ICA, ICA-GA and PSO_{LS} in medium and large-sized problems.

Instance	Problem size (job*machine)	Type	GA	ICA	ICA-GA	PSO $_{LS}$
1	20*5	a	4243.8	4274.1	4243.8	4243.8
2	20*5	b	4656.6	4685.7	4652.7	4652.7
3	20*5	c	1352.5	1353.2	1352.5	1352.5
4	20*5	d	1757.1	1769.6	1759.5	1757.1
5	20*10	a	8627.1	8678.8	8627.1	8630.3
6	20*10	b	9276.7	9305.3	9276.7	9276.7
7	20*10	c	3732.0	3770.3	3732.0	3732.0
8	20*10	d	7510.5	7556.3	7510.5	7510.5
9	20*15	a	14813.0	14870.1	14809.2	14809.2
10	20*15	b	10205.1	10283.2	10205.1	10205.1
11	20*15	c	4473.4	4533.1	4439.2	4461.8
12	20*15	d	11025.2	11088.4	11011.3	11022.4
13	30*5	a	8071.8	8129.8	8071.8	8071.8
14	30*5	b	7185.8	7238.7	7111.3	7160.0
15	30*5	c	1354.8	1416.7	1343.2	1343.2
16	30*5	d	5630.6	5687.5	5611.9	5641.7
17	30*10	a	19479.3	19588.1	19479.3	19479.3
18	30*10	b	16435.7	16506.7	16418.5	16431.7
19	30*10	c	7530.3	7588.9	7510.0	7530.3
20	30*10	d	10859.7	11178.8	10664.3	10774.7
21	30*15	a	31214.0	31311.1	31119.6	31183.4
22	30*15	b	31557.5	31617.8	31445.2	31541.1
23	30*15	c	11418.3	11516.1	11403.2	11427.5
24	30*15	d	25179.5	25331.1	24872.9	24907.5
25	50*5	a	21706.4	22109.8	21302.0	21302.0
26	50*5	b	19249.7	19355.4	19198.1	19240.6
27	50*5	c	5634.5	5876.9	5602.0	5621.6

Instance	Problem size (job*machine)	Type	GA	ICA	ICA-GA	PSO $_{LS}$
28	50*5	d	14124.0	14334.2	14009.3	14060.4
29	50*10	a	60304.2	60543.1	60243.8	60265.1
30	50*10	b	47861.5	48344.7	47544.1	47903.2
31	50*10	c	14867.7	15134.8	14654.2	14811.8
32	50*10	d	47024.1	47187.2	46654.8	46971.7
33	50*15	a	87769.1	88230.0	87545.5	87747.3
34	50*15	b	99714.9	100233	99233.4	99274.3
35	50*15	c	35218.7	35740.9	34980.6	35022.4
36	50*15	d	73967.7	74560.8	73778.5	73810.3
37	80*5	a	60068.2	60323.5	60008.4	60019.9
38	80*5	b	68438.3	69130.7	68020.1	68098.8
39	80*5	c	13450.0	13889.0	13129.8	13309.5
40	80*5	d	43608.7	44321.6	43121.9	43222.5
41	80*10	a	148917	151334	146556	146173.4
42	80*10	b	150617	153217	149870	150105.3
43	80*10	c	58565.0	59200.6	58565.0	58572.4
44	80*10	d	109663.3	111232.2	108776.2	109572.6
45	80*15	a	274895.0	277654.1	272114.3	272186.2
46	80*15	b	281266.6	287655.1	278906.9	280326.6
47	80*15	c	106896.4	107877.5	105998.0	106343.1
48	80*15	d	177444.2	180556.3	175667.3	175919.3

Table 8 Comparison between relative percentage deviation (RPD) of GA, ICA, ICA-GA and PSO_{LS} for medium-sized problems.

instance	Problem size (job*machine)	Type	GA	ICA	ICA-GA	PSO $_{LS}$
1	20*5	a	0.54	1.68	0.44	0.52
2	20*5	b	0.74	1.43	0.70	0.72
3	20*5	c	1.60	3.98	1.07	1.43
4	20*5	d	1.18	3.02	0.93	1.14
5	20*10	a	0.29	1.10	0.51	0.38
6	20*10	b	0.27	1.02	0.50	0.41
7	20*10	c	1.12	2.77	1.53	1.42
8	20*10	d	0.14	1.01	0.02	0.09
9	20*15	a	0.22	0.57	0.13	0.18
10	20*15	b	0.53	0.95	0.32	0.39
11	20*15	c	1.47	2.63	1.22	1.35
12	20*15	d	0.47	0.87	0.42	0.44
13	30*5	a	1.27	1.72	0.83	1.15
14	30*5	b	2.40	2.96	1.44	1.82
15	30*5	c	7.49	11.57	6.08	6.58
16	30*5	d	1.02	1.86	0.39	0.73
17	30*10	a	0.22	0.50	0.15	0.21
18	30*10	b	0.46	0.96	0.38	0.41
19	30*10	c	1.42	1.62	1.21	1.33
20	30*10	d	3.17	5.55	2.31	2.54
21	30*15	a	0.66	0.93	0.72	0.69
22	30*15	b	0.59	1.71	0.21	0.38
23	30*15	c	1.40	1.98	1.53	1.44
24	30*15	d	1.76	4.05	1.24	1.55
Average			**1.27**	**2.35**	**1.01**	**1.14**

Table 9 Comparison between relative percentage deviation (RPD) of GA, ICA, ICA-GA and PSO_{LS} for large-sized problems.

instance	Problem size (job*machine)	Type	GA	ICA	ICA-GA	PSO_{LS}
1	50*5	a	2.68	5.94	1.18	1.73
2	50*5	b	0.93	1.07	0.26	0.53
3	50*5	c	3.81	6.92	2.08	2.84
4	50*5	d	2.47	3.68	1.05	1.46
5	50*10	a	0.64	0.92	0.37	0.40
6	50*10	b	1.50	2.57	1.02	1.06
7	50*10	c	2.87	6.83	3.62	3.04
8	50*10	d	1.00	1.69	0.69	0.54
9	50*15	a	0.57	1.16	0.48	0.52
10	50*15	b	0.82	1.32	0.66	0.71
11	50*15	c	1.64	3.26	1.04	1.24
12	50*15	d	0.73	1.79	0.45	0.53
13	80*5	a	0.34	1.29	0.17	0.25
14	80*5	b	1.26	2.51	0.46	0.74
15	80*5	c	6.48	9.68	4.80	5.39
16	80*5	d	2.71	4.07	1.50	1.60
17	80*10	a	2.43	5.49	2.27	2.35
18	80*10	b	2.36	5.34	0.90	1.43
19	80*10	c	0.91	1.77	0.87	0.89
20	80*10	d	4.38	5.30	3.57	3.97
21	80*15	a	2.41	3.67	1.76	2.15
22	80*15	b	2.65	3.62	1.94	2.11
23	80*15	c	2.75	3.67	1.59	2.27
24	80*15	d	3.15	4.43	4.31	3.51
Average			**2.15**	**3.67**	**1.54**	**1.72**

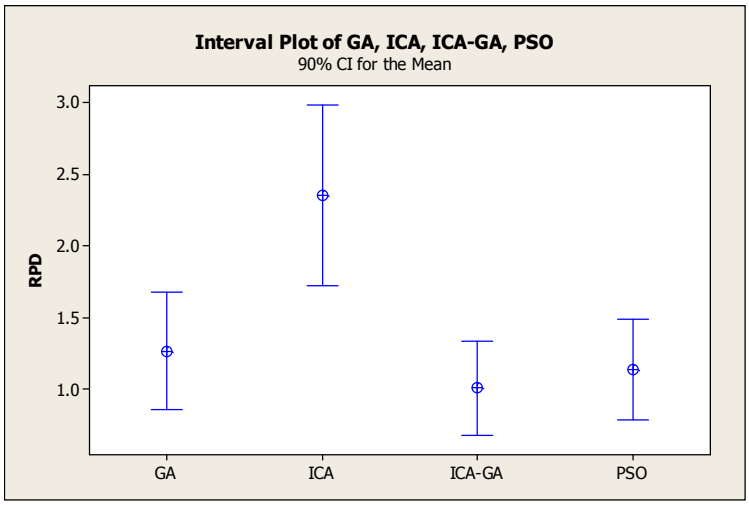

Fig. 5 The means plot and LSD intervals for RPD of four algorithms (medium-sized problems).

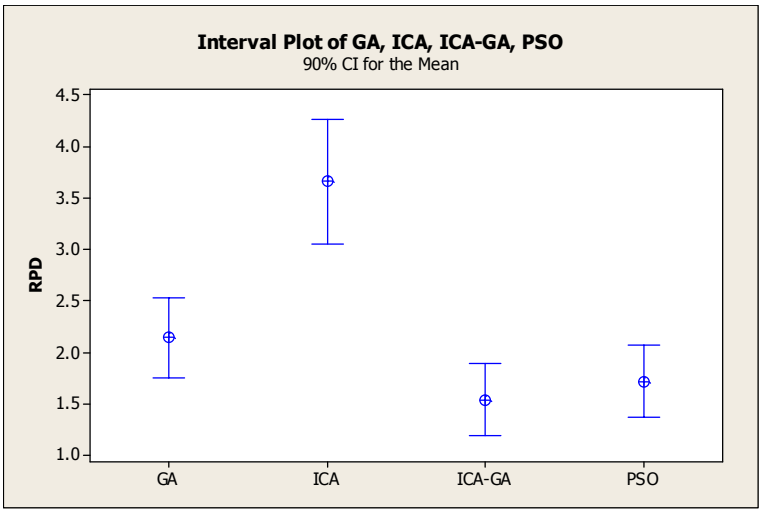

Fig. 6 The means plot and LSD intervals for RPD of four algorithms (large-sized problems).

In order to consider the effects of number of jobs on four algorithms, a two ways ANOVA is applied. Plot of RPD for the interaction between the type of algorithm and number of jobs is illustrated in Fig. 7. As depicted, in all cases (n=20, n=30, n=50 and n=80), the ICA-GA has better performance than other algorithms.

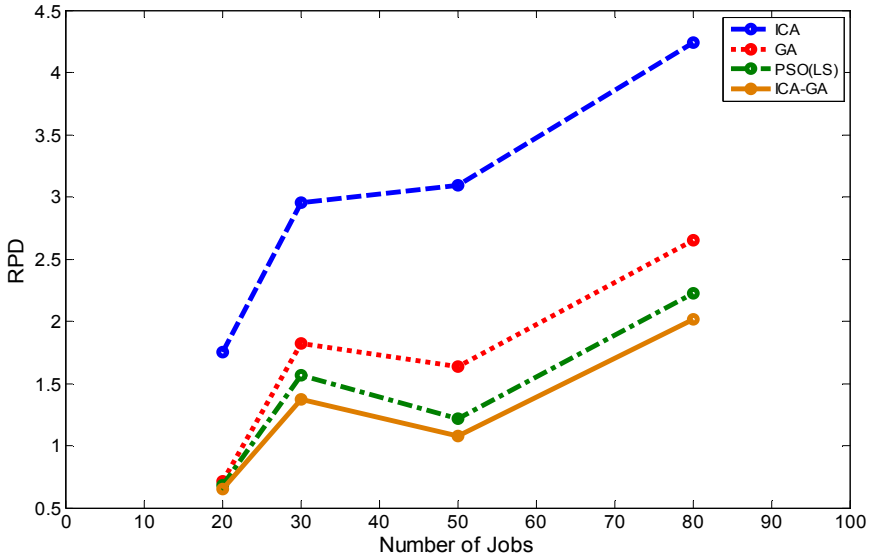

Fig. 7 RPD for the interaction between the type of algorithm and the number of jobs.

Another two ways ANOVA test is applied to see the effect of number of machines on performance of the four presented algorithms. The results are demonstrated in Fig. 8. In all cases (m=5, m=10 and m=15) the ICA-GA performs better than GA, ICA and PSO_{LS}.

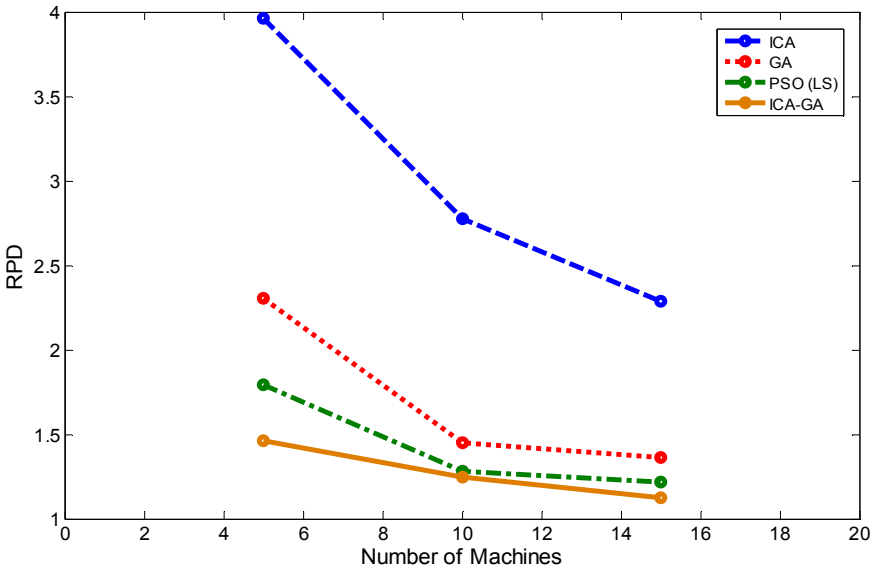

Fig. 8 RPD for the interaction between the type of algorithm and the number of machines.

As shown in Fig. 4 and Fig. 5 and according to the computational results the overlap between GA, hybrid algorithm and PSO_{LS} is considerable. Therefore, another criterion is required to judge about the capability of algorithms. We ran 48 problem instances (each instance has been performed in 10 replications) and the mean value of computational times is reported. The mean time of the hybrid algorithm was 14.06 % better than the GA and 21.51% better than PSO_{LS} ones (see Table 10).

Table 10 Superiority of ICA-GA in computational times (S) rather than GA and PSO_{LS} for medium and large-sized problems

Instance	Problem size (job*machine)	Type	GA	PSO $_{LS}$	ICA-GA	Improvement (%)	
						GA	PSO_{LS}
1	20*5	a	24.22	26.08	22.61	7.17	15.39
2	20*5	b	21.47	23.03	20.12	6.82	14.58
3	20*5	c	20.14	21.03	18.73	7.70	12.47
4	20*5	d	18.11	18.05	16.11	12.48	12.08
5	20*10	a	25.32	26.03	22.75	11.54	14.68
6	20*10	b	27.11	28.02	24.83	9.31	12.98
7	20*10	c	24.18	26.00	22.64	6.99	15.05
8	20*10	d	25.36	26.03	22.72	11.72	14.69
9	20*15	a	22.65	23.09	20.35	11.58	13.74
10	20*15	b	24.39	26.07	22.32	9.37	16.91
11	20*15	c	22.17	23.09	20.71	7.10	11.53
12	20*15	d	23.79	24.08	20.90	13.83	15.21
13	30*5	a	25.31	27.04	22.59	12.49	20.18
14	30*5	b	22.19	24.07	20.57	8.24	17.44
15	30*5	c	26.11	28.10	23.77	10.17	18.56
16	30*5	d	25.08	27.02	21.83	15.05	23.96
17	30*10	a	27.35	29.04	24.72	10.73	17.55
18	30*10	b	30.62	33.04	27.23	12.57	21.47
19	30*10	c	32.21	34.06	29.31	9.93	16.24
20	30*10	d	28.56	30.05	25.65	11.56	17.37
21	30*15	a	34.90	35.06	30.75	13.68	14.20
22	30*15	b	30.26	32.07	27.44	10.44	17.04

Instance	Problem size (job*machine)	Type	GA	PSO_{LS}	ICA-GA	Improvement (%)	
						GA	PSO_{LS}
23	30*15	c	34.61	34.01	29.15	18.93	16.88
24	30*15	d	35.85	37.03	31.01	15.61	19.43
25	50*5	a	43.01	46.02	37.36	15.31	23.37
26	50*5	b	42.04	49.02	36.70	14.55	33.56
27	50*5	c	40.75	44.00	35.41	15.11	24.30
28	50*5	d	4513	45.07	38.41	17.53	17.36
29	50*10	a	48.37	49.10	41.14	17.69	19.46
30	50*10	b	47.91	52.03	42.85	11.94	21.56
31	50*10	c	48.52	52.02	42.15	15.25	23.57
32	50*10	d	48.08	51.06	40.10	19.90	27.34
33	50*15	a	61.10	67.05	54.58	12.11	23.03
34	50*15	b	59.00	66.03	50.67	16.60	30.50
35	50*15	c	59.93	65.08	52.46	14.37	24.19
36	50*15	d	61.77	64.02	52.36	18.11	22.42
37	80*5	a	73.23	83.07	60.44	21.24	37.52
38	80*5	b	72.89	86.03	62.83	16.07	37.00
39	80*5	c	71.24	79.09	60.44	17.95	30.94
40	80*5	d	70.17	78.04	60.50	15.98	29.00
41	80*10	a	86.54	92.04	73.71	17.42	24.88
42	80*10	b	86.09	91.06	70.31	22.46	29.53
43	80*10	c	86.43	93.06	74.21	16.48	25.41
44	80*10	d	85.12	97.08	73.75	15.50	31.73
45	80*15	a	104.41	112.06	90.81	14.99	23.41
46	80*15	b	104.34	115.05	89.30	16.84	28.83
47	80*15	c	110.15	118.05	90.98	21.18	29.86
48	80*15	d	109.17	108.10	87.27	25.19	23.97
Average						**14.06**	**21.51**

The results show that, the proposed hybrid algorithm is still much better than PSO_{LS} and thus is a reliable algorithm for treating such a problem.

6 Conclusions and future work

In this paper, a just-in-time flow shop scheduling problem with limited buffers and deteriorating jobs was investigated in a fuzzy environment. For this problem, a mixed integer non-linear program has been formulated. The aim of this model was to minimize the weighted sum of fuzzy earliness and tardiness penalties considering a set of jobs that have non-identical due dates. Due to NP-hardness of the problem, an efficient hybrid meta-heuristic approach based on imperialist competitive algorithm and genetic algorithm (ICA-GA) was proposed to solve the mathematical model. The objectives of proposing a hybrid algorithm were to improve the solution quality and to reduce the run time. Since, the initial solution is produced by ICA, having more rapid convergence rate, provide a better solution to be optimized by GA in fewer replications for finding the best solution. The performance of the proposed hybrid algorithm has been verified by a number of random numerical examples. Computational results demonstrated the superiority of the proposed approach in the jobs sequencing as compared with GA and ICA methods. The proposed algorithm was also compared with a similar work in the literature, PSO_{LS}, and provided a much better solution quality in more reasonable time span. Future studies can focus on the other features of deterioration such as non-linear functions. In addition, designing other meta-heuristic approaches may be devised for the further works. Also, to improve the quality of solution in hybrid approach, local search methods can be employed.

References

1. Papadimitriou, C.H., Kanellakis, P.C., (1980). Flow shop scheduling with limited temporary storage, Journal of Association Computing Machine, 27,533–549.
2. Norman, B.A., (1996). Scheduling flow shops with finite buffers and sequence dependent setup times, Computers & Industrial Engineering, 36 (1), 163–177.
3. Khosla, I., (1995). The scheduling problem where multiple machines compete for a common local buffer, European Journal of Operational Research, 84,330–42.
4. Nowicki, E., (1999). The permutation flow shop with buffers: a tabu search approach, European Journal of Operational Research, 116,205–219.
5. Wang, L., Zhang, L., Zheng, D.Z., (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, Computers & Operations Research, 33, 2960–2971.
6. Liu, B., Wang, L., Jin, Y.H., (2008). An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers, Computers & Operations Research, 35,2791–2806.
7. Ke Pan, Q., Wang, L., Gao, L., (2011). A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers, Applied Soft Computing, 11,5270–5280.
8. Ke Pan Q., Wang, L., Gao, L., Li, W.D., (2011) An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, Information Sciences, 181, 668–685.
9. Gupta, J.N.D., Gupta, S.K., (1988). Single facility scheduling with nonlinear processing times, Computers & Industrial Engineering, 14, 387–393.
10. Mosheiov, G., (2000). Complexity analysis of job-shop scheduling with deteriorating jobs, Discrete Applied Mathematics, 117, 195–209.
11. Wang, J.B., Ng, D.C.T., Cheng, T.C.E., Liu, L.L., (2006). Minimizing total completion time in a two-machine flow shop with deteriorating jobs, Applied Mathematics and Computation, 180, 185–193.
12. Wang, J.B., Xia, Z.Q., (2006). Flow shop scheduling with deteriorating jobs under dominating machines, Omega, 34, 327–336.
13. Shiau, Y.R., Lee, W.C., Wu, C.C., Chang, C.M., (2007). Two-machine flow shop scheduling to minimize mean flow time under simple linear deterioration, International Journal of Advanced Manufactured Technology, 34,774–782.
14. Lee, W.C., Wu, C.C., Wen, C.C., Chung, Y.H., (2008). A two-machine flow shop makespan scheduling problem with deteriorating jobs, Computers & Industrial Engineering, 54,737–49.
15. Ng, C.T., Wang, J.B., Cheng, T.C.E., Liu, L.L., (2010). A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs, Computers & Operations Research, 37,83–90.
16. Yang, S.H., Wang, J.B., (2011). Minimizing total weighted completion time in a two-machine flow shop scheduling under simple linear deterioration, Applied Mathematics and Computation, 217,4819–4826.
17. Bank, M., Fatemi-Ghomi S.M.T., Jolai,F., Behnamian, J., (2012). Application of particle swarm optimization and simulated annealing algorithms in flow shop scheduling problem under linear deterioration, Advances in Engineering Software, 47, 1–6.
18. Bank, M., Fatemi-Ghomi, S.M.T., Jolai, F., Behnamian, J., (2012). Two-machine flow shop total tardiness scheduling problem with deteriorating jobs, Applied Mathematical Modelling, 36(11), 5418–5426.
19. Balin, S., (2011). Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation, Information Sciences, 181,3551–3569**.**
20. Anglani, A., Grieco, A., Guerriero, E., Musmanno, R., (2005). Robust scheduling of parallel machines with sequence-dependent setup costs, European Journal of Operational Research, 161,704–720.
21. Balasubramanian, J., Grossmann, I.E., (2003). Scheduling optimization under uncertainty – an alternative approach, Computers and Chemical Engineering,27, 469–490.
22. Slowinski, R, Hapke, M., (2000). Scheduling Under Fuzziness, Physica-Verlag Editions, New York.
23. Mok, P.Y., Kwong, C.K., Wong, W.K., (2007). Optimization of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory, European Journal of Operations Research,177, 1876–1893.
24. Zadeh, L. A., (1965). Fuzzy sets. Information and Control, 8, 338–353.
25. Wu, H.C., (2010). Solving the fuzzy earliness and tardiness in scheduling problems by using genetic algorithms, Expert Systems with Applications, 37, 4860–4866.
26. Lai, P.J., Wu, H.C., (2011). Evaluate the fuzzy completion times in the fuzzy flow shop scheduling problems using the virus-evolutionary genetic algorithms, Applied Soft Computing, 11, 4540–4550.
27. Fortemps, P., Roubens, M., (1996). Ranking and defuzzification methods based on area compensation. Fuzzy Sets and Systems, 82, 319–330.

28. Kononov, A., Gawiejnowicz, S., (2001). NP-hard cases in scheduling deteriorating jobs on dedicated machines. Journal of the Operational Research Society, 52 (6), 708–717.
29. Ruiz, R., Stützle, T., (2008). An iterated greedy heuristic for the sequence dependent setup times flow shop problem with makespan and weighted tardiness objectives, European Journal of Operational Research, 187(3),1143–59.
30. Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., Sassani, F., (2009). Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints, Computers & Operations Research, 36, 3224 – 3230.
31. Lin, S.W., Ying, K.C., Lee, Z.J., (2009). Meta-heuristics for scheduling a non-permutation flow line manufacturing cell with sequence dependent family setup times, Computers & Operations Research, 36(4), 1110–1121.
32. Behnamian, J &Zandieh, M., (2011). A discrete colonial competitive algorithm for hybrid flow shop scheduling to minimize earliness and quadratic tardiness penalties, Expert Systems with Applications, 38, 14490–14498.
33. AtashpazGargari, E., & Lucas, C., (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. IEEE Congresson Evolutionary Computation, Singapore, 4661–4667.
34. Moslehi, G., Mirzaee, M., Vasei, M., Modarres, M., Azaron, A., (2009). Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness, International Journal of Production Economics, 122,763–773.