# Fuzzy Programming for Parallel Machines Scheduling: Minimizing Weighted Tardiness/Earliness and Flow Time through Genetic Algorithm

Mohammad Asghari [*,a], Samaneh Nezhadali [b]

[a] *MSc, Department of Industrial Engineering, Ferdowsi University of Mashhad, PO Box 91775-1111, Azadi Sq., Mashhad, Iran*
[b] *MSc, Department of Management, Iran Chamber of Commerce, Industries and Mines, Mashhad, Iran*

Received 17 September, 2012; Revised 18 January, 2013; Accepted 20 February, 2013

**Abstract**

Appropriate scheduling and sequencing of tasks on machines is one of the basic and significant problems that a shop or a factory manager encounters; this is why in recent decades extensive studies have been done on scheduling issues. One type of scheduling problems is just-in-time (JIT) scheduling and in this area, motivated by JIT manufacturing, this study investigates a mathematical model for appraising a multi-objective programing that minimize total weighted tardiness, earliness and total flowtime with fuzzy parameters on parallel machines, simultaneously with respect to the impact of machine deterioration. Besides, in this paper attempted to present a defuzzification approach and a heuristic method based on genetic algorithm (GA) to solve the proposed model. Finally, several dominant properties of optimal solutions are demonstrated in comparison with the results of a state-of-the-art commercial solver and the simulated annealing method that is followed by illustrating some instances for indicating validity and efficiency of the method.

*Keywords:* Mathematical optimization, Fuzzy multi-objective model, Parallel machines scheduling, Weighted tardiness/earliness, Genetic algorithm**.**

## 1. Introduction

In classical scheduling problems, the processing time of jobs has been assumed constant. However, there are many situations that this time may be subject to change due to deterioration and/or learning phenomena (McKay et al. 2002).

Scheduling with costs of earliness and tardiness has received considerable and increasing attention in recent researches. In many practical situations, it is required to guarantee that as many jobs as possible meet their due dates (i.e., to minimize the number of tardy jobs) since in such cases having a job missing its due date is very costly. Thus, minimization of the number of tardy jobs should be the primary concern. On the other hand, it is desirable to minimize the job earliness to minimize the inventory cost. Early/tardy scheduling problems are compatible with the concepts of just-in-time production and supply chain management, which has been adopted by many organizations. Indeed, these production strategies view both early and tardy deliveries as undesirable. By machine deterioration effect, we mean that each machine deteriorates at a different rate. This deterioration is considered in terms of cost that depends on the production rate, the machine operating characteristics, and the kind of work done by each machine. Moreover, job-processing times are increasing functions of their starting times and follow a simple linear deterioration. Browne and Yechiali (1990) first introduced it. Since then, deteriorating job scheduling problems have been widely discussed. Ruat et al. (2008) considered the problem of scheduling a given number of jobs on a single machine with time deteriorating job values and capacity constraints while the objective function is to maximize total revenue. Gawiejnowicz et al. (2006) considered a single machine time-dependent scheduling problem. They introduced two scenarios for a given sequence of job deterioration and formulated a greedy polynomial time approximation algorithm for each scenario.

In recent decades, most of the researches have focused on Just-In-Time (JIT) scheduling models. For example, Sridharan and Zhou (1996) considered a single machine problem with total weighted earliness and tardiness and developed a solution procedure based on decision theory. Cai and Zhou (1999) studied a parallel machine stochastic scheduling problem to minimize expected total cost for early and tardy jobs. Mazzini and Armentano (2001) studied a general single machine problem with early and tardy costs and developed a heuristic. Wan and Yen (2002) studied a general single machine scheduling problem with distinct due windows and weighted

[*] Corresponding E-mail address: Mohammad.Asghari@stu-mail.um.ac.ir

earliness and tardiness costs, and developed a tabu search procedure. Wan and Yen (2009) considered the problem of single machine bicriteria scheduling to minimize the total weighted earliness subject to minimum number of tardy jobs. Guner et al. (1998) considered one machine scheduling to minimize the maximum earliness with minimum number of tardy jobs. They proposed a procedure to minimize the maximum earliness when the set of tardy jobs is specified, then, appraised a branch and bound algorithm to minimum number of tardy jobs.

Minimizing of weighted tardiness penalties is a familiar objective function in parallel machine scheduling. Bilge et al. (2004) used a tabu search method to schedule parallel machines with total weighted tardiness penalties. Yi and Wang (2003) introduced a model for scheduling grouped jobs on identical parallel machines. In their model a set-up time is incurred when one-machine changes from processing one type of component to a different type of component. The objective function here is to minimize the total earliness/tardiness penalties. Radhakrishnan et al. (2000) emphasized the JIT production philosophy, and used simulated annealing for parallel machine scheduling with earliness/tardiness penalties and sequence dependent set-up times.

The fuzzy approach represents an alternative way to model imprecision and uncertainty which is more efficient, especially when no historical information is available (Anglani et al. 2005). First, it was introduced as presentation scheme and calculus for uncertain or vague notions. The fuzzy set theory provides a conceptual framework (Wu and Lee 2008) that performs so efficiently in decreasing the scheduling problem computational complexity with respect to the same problem formulated by the probability theory. It should be noted that such an imprecision is due to the subjective and qualitative evaluations, rather than the effect of uncontrollable events.

The use of the fuzzy sets theory in treating different scheduling problems has been so successful, particularly where judgment and intuition play an important role in, for example, customer demand (Ishii and Tada 1995), processing times (Kuroda and Wang 1996), production due dates (Hong and Chuang 1999), or job precedence relations (Ishii and Tada 1995). Prade et al. (1979) published the earliest paper in fuzzy scheduling. Ishii and Tada (1995) considered a single machine problem minimizing the maximum lateness of jobs with fuzzy precedence relations. Han et al. (1994) proposed same problem with fuzzy due dates. Ishibuchi and Murata (2000) presented a flow shop-scheduling problem with fuzzy parameters, such as fuzzy due dates and fuzzy processing times. Kuroda and Wang (1996) analyzed the fuzzy job shop-scheduling problem. Konno and Ishii (2000) discussed an open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint. Itoh and Ishii (1999) proposed a single machine scheduling model dealing with fuzzy processing times and due dates. Litoiu and Tadei (2001) present some new models for real-time task scheduling with fuzzy deadlines and processing times.

In classical problems of parallel machine scheduling, all the parameters and variables were considered deterministic. However, since multiple sources of uncertainty and complex interrelation-ships at various levels between diverse entities exist in these kinds of problems, it is quite unreliable to set them as precise values.

Some researchers have modeled parallel machine problems by probability distribution that is usually predicted from historical data (Piersma and Romeijn 1996). However, whenever statistical data is unreliable or even unavailable, stochastic models may not be the best choice. In this situation, fuzzy set theory may provide an alternative approach for dealing with the uncertainty. This is why it has found extensive applications in various fields.

In this area, a limited number of the studies in the current literature have been devoted to fuzzy parallel-machine scheduling problems. Peng and Liu (2006) proposed a practical application. In addition to single-objective scheduling models, they considered the multi-objective FPMSPs and formulated as three-objective models, which not only minimize the fuzzy maximum tardiness, but also minimize the fuzzy maximum completion time (i.e., make span) and the fuzzy maximum idleness. Balin (2011) addressed parallel machine scheduling problems with fuzzy processing times and proposed a robust genetic algorithm (GA) approach embedded in a simulation model to minimize the maximum completion time.

In this paper, a fuzzy multi-objective parallel machines scheduling problem would be considered to minimize total weighted tardiness, earliness, aside of minimizing total flowtime and machine deteriorating cost, which is an extension of the problem studied by Mazdeh et al. (2010).

The job-scheduling problem with minimizing weighted tardiness in parallel machines is known NP-hard in the strong sense (Cao et al. 2005; Pfund et al. 2008), this is why the combinational problem with objective functions including minimizing the total tardiness, earliness, flowtime and machine deteriorating cost will also be NP-hard. In the past, a number of various methods such as branch and bound, cutting plane, and other heuristic approaches were introduced. In recent years, researches have used metaheuristic methods such as simulated annealing for same problems.

In this paper, a genetic algorithm is used to solve the model and then the obtained results are compared with the output of commercial software, Lingo solver, to show the effectiveness of the proposed approach

The remainder of this paper is organized as follows. In Section 2, the problem is defined and the objective functions are introduced in details. Next, the mathematical formulation is developed for the problem. In Section 3, we present a new solution method for it and describe an approach in order to consider the four objectives as a single objective. In Section 4, some numerical examples of its occurrence are applied and the feasibility and effectiveness of the proposed method is demonstrated by comparing the simulated annealing method. Finally, concluding remarks are presented in the last Section.

## 2. Problem formulation

The following notations and definitions are used to describe a multi-objective on parallel machines scheduling problem that is an extension of studied problem by Mazdeh et al. (2010).

This problem considers a set of $N$ independent jobs, $J_1, J_2, \ldots, J_n$, on a number of parallel machines selected from a set of $M$ potential machines as each of these jobs exactly need one operation on one machine. Each job $J_i$ has a processing time $\tilde{p}_j$ and a due date $\tilde{d}_j$ that all processing times and due dates are considered as fuzzy numbers. Here machines are supposed to become worse at a different rate by allocating and then doing the jobs on them. This deterioration is a function of production rate, machine's operating characteristics and the kind of work accomplished by each machine, which considered in terms of cost.

A job is early if its completion time is smaller than the common due date. On the other hand a job is tardy if its processing ends after due date. As it is not known in advance whether a job will be completed before or after the due date.

The notations and other assumptions that would be applied in mathematical formulation are followed as below.

### 2.1. Problem assumptions

The following notations are the assumptions considered in the present model.

- Each machine is able to process each job;
- The machine can process at most one job at a time;
- No processing is allowed;
- Associated with job $j$ ($j=1, \ldots, n$) there are a processing time $\tilde{p}_j$ and a due date $\tilde{d}_j$;
- Job processing time may be different by various machines;
- Job processing time is described by a function of the starting time ($\tilde{P}_{jm} = a_{jm} + \tilde{b}_j \tilde{S}_{jm}$);
- The growth rate of the processing time ($\tilde{b}_j$) is independent of machine;
- The jobs are considered independent of each other;

### 2.2. Sets and indices

The following shows nomenclature used in the model.

*Sets*

| | |
|---|---|
| $N$ | The set of jobs that must be scheduled |
| $M$ | The set of available machines |

$i,j \in \{0,1,\ldots,N\}$ are designated job, where job 0 is a dummy job and is always at the first position on a machine

*Parameters*

| | |
|---|---|
| $\gamma_i$ | Earliness weight of job $i$ |
| $\beta_i$ | Tardiness penalty of job $i$ |
| $r_i$ | Arrived time of job $i$ to queue |
| $\tilde{d}_i$ | Due date of job $i$ |
| $\tilde{a}_{im}$ | Processing fix time of job $i$ on machine $m$ |
| $\tilde{b}_i$ | Growth rate of the processing time of job $i$ |
| $\tilde{c}_{im}$ | Cost of machine deterioration |

*Decision variables*

| | |
|---|---|
| $X_{ijm}$ | 1, if job $j$ immediately follows job $i$ in sequence on machine $m$; 0, otherwise |
| $Y_{im}$ | 1, if job $i$ assigned to machine $m$; 0, otherwise |
| $\tilde{T}_i$ | Tardiness value of job $i$ |
| $\tilde{E}_i$ | Earliness value of job $i$ |
| $\tilde{P}_{im}$ | Processing time of job $i$ on machine m |
| $\tilde{S}_{im}$ | Starting time of job $i$ on machine $m$ |
| $\tilde{C}_i$ | Completion time of job $i$ |

### 2.3. Mathematical model

Base on the aforementioned descriptions and indices, a fuzzy nonlinear programming model is developed as follows:

Minimize

$$F1 = \sum_{i \in N} \gamma_i \tilde{E}_i \qquad (1)$$

$$F2 = \sum_{i \in N} \beta_i \tilde{T}_i \qquad (2)$$

$$F3 = \sum_{i \in N} \tilde{C}_i - r_i \qquad (3)$$

$$F4 = \sum_{i \in N} \sum_{m \in M} Y_{im} . \tilde{c}_{im} \qquad (4)$$

Subject to

$$\sum_{i \in N} X_{0im} \leq 1 \qquad \forall m \in M, \qquad (5)$$

$$\sum_{i \in N, i \neq j} \sum_{m \in M} X_{ijm} = 1 \qquad \forall j \in N, \qquad (6)$$

$$\sum_{j\in N, i\neq j} X_{ijm} \le Y_{im} \qquad \forall i \in N, \forall m \in M, \qquad (7)$$

$$\sum_{i\in N, i\neq j} X_{ijm} \le Y_{jm} \qquad \forall j \in N, \forall m \in M, \qquad (8)$$

$$\sum_{m\in M} Y_{im} = 1 \qquad \forall i \in N, \qquad (9)$$

$$\tilde{C}_i \ge \tilde{S}_{im} + \tilde{P}_{im} \qquad \forall i \in N, \forall m \in M, \qquad (10)$$

$$\tilde{P}_{im} + \acute{M}(1-Y_{im}) \ge \tilde{a}_{im} + \tilde{b}_i.\tilde{S}_{im} \qquad \forall i \in N, \forall m \in M, \qquad (11)$$

$$\tilde{S}_{jm} + \acute{M}(1-X_{ijm}) \ge \tilde{C}_i \qquad \forall i \in N, i\neq j, \forall j \in N, \forall m \in M, \qquad (12)$$

$$\tilde{C}_0 = 0 \qquad (13)$$

$$\tilde{S}_{0m} = 0 \qquad \forall m \in M, \qquad (14)$$

$$\tilde{E}_i = \text{Max}\{0, \tilde{d}_i - \tilde{C}_i\} \qquad \forall i \in N, \qquad (15)$$

$$\tilde{T}_i = \text{Max}\{0, \tilde{C}_i - \tilde{d}_i\} \qquad \forall i \in N, \qquad (16)$$

$$\tilde{T}_i, \tilde{E}_i, \tilde{P}_{im}, \tilde{S}_{im}, \tilde{C}_i \in R^+ \qquad \forall i \in N, \forall m \in M, \qquad (17)$$

$$X_{ijm}, Y_{im} \in \{0,1\} \qquad \forall i,j \in N, \forall m \in M, \qquad (18)$$

In the presented model, Eqs. (1) to (4) are the objective functions, namely minimizing total weighted earliness, tardiness, total flowtime and minimizing the cost of machine deterioration, respectively. More precisely, Eq. (2) states if $C_i$-$d_i$ > 0 then delivering job *i* has been delayed and it causes tardiness. Otherwise, if $C_i$-$d_i$ < 0 it causes earliness that showed in Eq. (1). Eq. (4) states if a job is processed on one machine, machine deteriorating will happen. Constraint (5) ensures that only a job can follow the dummy job (*i* = 0) on each assigned machine. Sometimes, in a job-scheduling scheme, it is likely to schedule only one job on a machine, in this situation a dummy job helps us to define $X_{ijm}$, which will be equal to zero when no job is scheduled and any job does not follow the dummy job on the machine. Eq. (6) assures a job must be processed on a machine and just followed by one job. Eqs. (7) and (8) state that if job *i* is immediately followed by *j* on machine *m* then both jobs *i* and *j* belong to machine *m*. Eq. (9) ensures that each job is assigned to only one machine. Eq. (10) relates the completion time of each job to its starting and processing time. Eq. (11) expresses the relation between the job processing time, its starting time and fix part of the processing time. Eq. (12) expresses that the job starting time is at least equal to the completion time of previous preceding job. Eq. (13) states that the completion time of the dummy job is equal to zero. Eq. (14) states that the starting time of the dummy job on each machine is equal to zero. Eq. (15) expresses the relation between the completion time of job, its due date and earliness variable and constraint (16) specifies the job tardiness. Finally, constraints in set (17) enforce the non-negativity restrictions on the corresponding decision variables and constraints in set (18) enforce the integrality restrictions on the binary variables.

## 3. Deterministic linear programming

The presented model is known as a fuzzy nonlinear programming. To solve this model, at first, the fuzzy numbers will be converted into the interval type by α-cut. After that, the interval numbers transmit in deterministic format through applying the convex conversion (Appendix A). In this approach, a fix value for α is determined by decision maker for showing risk as a result of which larger α means that the decision maker has accepted more risk since he has not considered the uncertainty of fuzzy numbers and vice versa.

### *3.1. Conversion fuzzy numbers into interval form*

Fuzzy numbers used in the paper such as processing time have been assumed to be triangular that has been illustrated in Fig 1. This figure shows fuzzy number $\tilde{P}_{im}$ with triangular membership function that is noted as $(P_{im}^l, P_{im}^m, P_{im}^u)$.

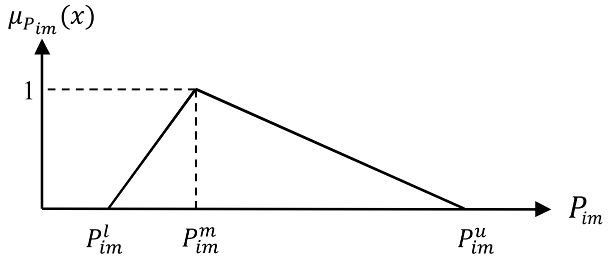

Fig. 1. Membership function of triangular fuzzy number

The membership function of this number is as follows:

$$\mu_{\tilde{P}_{im}}(P) = \begin{cases} \dfrac{P - P_{im}^l}{P_{im}^m - P_{im}^l}, & P_{im}^l \le P \le P_{im}^m \\ \dfrac{P - P_{im}^u}{P_{im}^m - P_{im}^u}, & P_{im}^m \le P \le P_{im}^u \\ 0, & P \le P_{im}^l \ , P \ge P_{im}^u \end{cases} \qquad (19)$$

**Lemma 1:** The α-cut on this membership function for $\alpha \in \{0,1\}$ presents close interval $[P_{im}^l, P_{im}^u]$ which can result: $\tilde{P}_{im_\alpha} = [P^l_{im}, P^u_{im}] = [\alpha.P^m_{im} + (1-\alpha)P^l_{im}$ , $\alpha.P^m_{im} + (1-\alpha)P^u_{im}]$

(We define $\tilde{P}_{im_\alpha}$ as Interval number (1))

**Proof:** With consideration of the α-cut on membership function of number $\tilde{P}_{im}$ that shown in Fig 2, we will have a new upper and lower bound for this number.

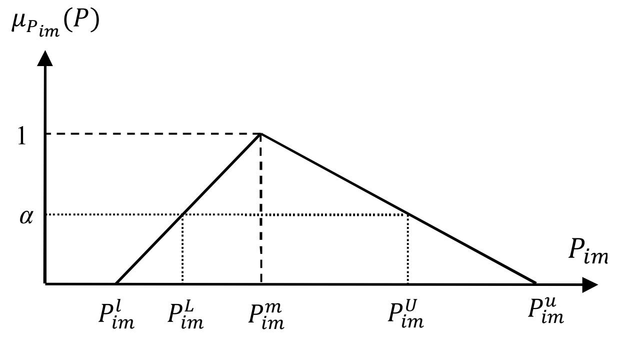

Fig. 2. α-cut on membership function of triangular fuzzy number $\tilde{P}_{im}$

According to Fig 2 following equations can be resulted:

$$\frac{\alpha}{P^L_{im} - P^l_{im}} = \frac{1}{P^m_{im} - P^l_{im}} \Rightarrow \alpha(P^m_{im} - P^l_{im}) = P^L_{im} - P^l_{im} \Rightarrow P^L_{im} = \alpha.P^m_{im} + (1-\alpha)P^l_{im} \quad (20)$$

$$\frac{\alpha}{P^u_{im} - P^U_{im}} = \frac{1}{P^u_{im} - P^m_{im}} \Rightarrow \alpha(P^u_{im} - P^m_{im}) = P^u_{im} - P^U_{im} \Rightarrow P^U_{im} = \alpha.P^m_{im} + (1-\alpha)P^u_{im} \quad (21)$$

Others like $\tilde{P}_{im}$ are converted into the interval form through α-cut on these numbers. However, due date $(\tilde{d}_i)$ defined different with membership function as following that has been introduced same Hwang and Yoon (1981) and has been illustrated in Fig 3.

$$\mu_{\tilde{d}_i}(d) = \begin{cases} 1, & d \le d^*_i \\ \dfrac{d - d^m_i}{d^*_i - d^m_i}, & d^*_i \le d \le d^m_i \\ 0, & d \ge d^m_i \end{cases} \quad (22)$$

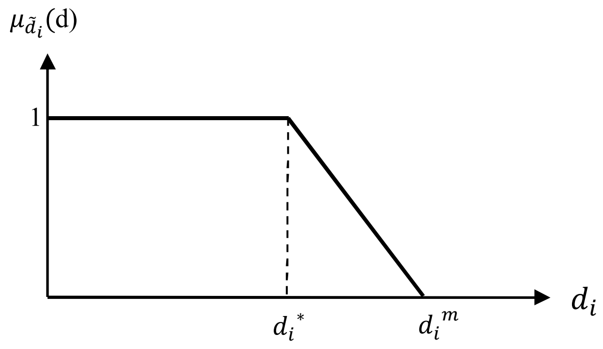

Fig. 3. Membership function of due date

This number can be converted to the interval form by applying $\alpha$-cut as follows:

$$\tilde{d}_{i_\alpha} = [d^L_i, d^U_i] = [0 \ , \ \alpha.d^*_i + (1-\alpha)d^m_i] \quad \text{Interval number (2)}$$

3.2. Conversion to deterministic programming

If we substitute interval numbers in the model, fuzzy programming is converted to interval programming. Now, the numbers should be defuzzified. In this paper, deterministic numbers are obtained via applying convex conversion. Hence, the interval programming converts into deterministic programming.

Follows, interval numbers convert into deterministic form by applying convex conversion context that present in Appendix A.

3.3. Solving multi-objective optimization

I use a general form of multi-objective programming that is a family of $L_p$-metrics and is adopted from Hwang & Yoon, (1981). This method considers the minimum deviation from the ideal solution as follows:

$$\begin{aligned} &\text{Min} \quad f_1(x), f_2(x), \ldots, f_n(x) \\ &\text{S.t:} \quad x \in X \end{aligned} \quad (23)$$

That $f_1(x), f_2(x), \ldots, f_n(x)$ are the objective functions and $x$ is the feasible region. First, an ideal solution for each objective function separately will be obtained by following problems solving:

$$\begin{aligned} &f_i^* = \text{Min} f_i(x) \qquad (i=1, \ldots, n) \\ &\text{S.t:} \quad x \in X \end{aligned} \quad (24)$$

Then, will be obtained without unit function with dividing the each function in its optimum value. Thus, multi-objective programming problem can satisfactorily solve by following new objective function:

$$\begin{aligned} &\text{Min}\left[\sum_i \omega_i^p\left(\frac{f_i(x)-f_i^-}{f_i^+-f_i^-}\right)^p\right]^{\frac{1}{p}} \\ &\text{S.t:} \qquad x \in X \end{aligned} \tag{25}$$

Where each function is weighted using ''$\omega$'' to denote the importance of objective functions. This weight adjustment is used for alimenting and balancing between functions that will be determined by decision makers just as following relationship can be established.

$$\begin{aligned} &\sum_i \omega_i = 1 \\ &\omega_i \geq 0 \qquad i \in (1, \ldots, n) \end{aligned} \tag{26}$$

Obviously, the result is dependent on the value of $p$. Generally, $p$ is 1 or 2. However, other values of $p$ also can be used.

## 4. Solution algorithm

### *4.1. Genetic algorithms*

This section presents a genetic algorithm with real number encoding which uses an adaptive mutation process to change the probability. Because in practical application a small fixed mutation probability can only result in a premature convergence, although, the mutation probability has been usually considered stable throughout the whole search process.

*Step 1. (Initialization)*

For getting the initial population, at first, all constraints of the problem excluding the follower's objective function are considered and some individuals satisfying these constraints are randomly generated by Lingo solver to guarantee that all the individuals are feasible to the problem. Note that before proceeding with the GA it is necessary to code the state of the network and determine the chromosome structure. Hence, coding of the potential solution is the first important aspect of a correct implementation of the GA. This is while; a proper coding will result in better solutions with less time consumption (Fig 4).

| *Machine* | 1 | | | | 2 | | | | … |
|---|---|---|---|---|---|---|---|---|---|
| *Job* | 1 | 2 | 3 | … | 1 | 2 | 3 | … | |
| *Random array* | **0** | **1** | **0** | | **1** | **0** | **1** | | |

Fig. 4. The chromosome structure

*Step 2. (Fitness function)*

The objective function as given by Eq. (25) is used also in the GA as fitness function.

*Step 3. (Crossover operator)*

The crossover operator recombines the genes of two selected chromosomes to generate a new crossover child to be formed in the next generation. It aims to take the best features of each parent and mix the remaining features in forming the offspring. Crossover helps genetic algorithm to converge to the best individual.

In this study has been used uniform crossover, which is widely used in the literature because of its efficiency in identifying, inheriting and protecting common genes and recombining non-common genes (Sywerda 1989; Page et al. 1999; Falkenauer 1999).

This crossover function creates a random vector and selects the genes where the vector is a 1 from the first parent, the genes where the vector is a 0 from the second parent, and combines the genes to form the child. For example, if the "1 0 1 0 1 0 0" and "0 1 0 0 0 1 1" string are defined as parents and "1 0 0 1 0 0 0" the binary vector, the function returns "1 1 0 0 0 1 1" as the child.

*Step 4. (Mutation operator)*

Mutation is another commonly used operator that let the GA explore a wider region of the solution space. This operator is carried with a fixed probability as a transition from current solution to its neighborhood in a local search algorithm. Mutation operates alternatively on the first or on the second chromosome, usually by introducing random gene changes.

In this paper, a special mutation operator named as scramble mutation (Leu et al. 1994) is employed with an adaptive process for changing the probability. All the vector elements are mutated according to the below mutation procedure and minimum mutation probability of 0.05.

$$\begin{aligned} &p_{m+1} \\ &= \begin{cases} p_m - p_{step} & ; \quad \text{if } F_m \text{ remained unchanged} \\ p_m & ; \quad \text{if } F_m \text{ declined} \\ p_{final} & ; \quad \text{if } p_m - p_{step} < p_{final} \end{cases} \\ &p_0 = 1.0 \\ &p_{step} = 0.001 \\ &p_{final} = 0.05 \end{aligned} \tag{27}$$

where, $m$ denotes the generation number and $p$ stands for the probability of applying the mutation operator to an individual.

*Step 5. (Selection)*

During this phase, the mating pool for the reproduction step is formed by choosing a set of individuals from the current population to keep good individuals and eliminate the bad ones from one generation to another (Oguz and Ercan 2005). In the proposed algorithm we used the best known selection strategy Roulette Wheel (Holland 1975), also known as fitness proportionate selection for mating.

The fitness values of the members within the population in this selection strategy are scaled as the total rescaled fitness values equals to 1. First, the fitness values of all population members divide by the members sum in order to calculate expected values of each individual in the population then generate a uniform random number within the interval (0, 1), and finally the individual whose cumulative rescaled fitness value is greater than or equal to the generated number is the one selected parent.

*Step 6. (Termination condition)*

We use the maximum number of generations as a stopping rule and record the optimum in every generation, $N \times M$, where $N$ and $M$ are the number of jobs and machines, respectively.

*4.2. Simulated annealing*

In this paper, for comparison, simulated annealing (SA) (Davis 1987; Kirkpatrick et al. 1983) is adopted as an another search method for the problem. Here, observe that SA searches for solutions by exchanging the job processing order for each machine.

The algorithm of SA used in this paper is summarized as follows.

*Step 1.* Generate one solution (schedule) through the random selection in Step 4 of an active scheduling generating algorithm and denote it by $X^c$. Set an initial temperature $T_0$.

*Step 2.* Represent the job process sequence for each machine of a solution $X^c$ by the corresponding matrix, and select one machine at random. Select two jobs of the machine and exchange them. For example in the problem of 3 jobs and 3 machines, when the first job ($J_3$) and the second job ($J_1$) of machine 3 ($M_3$) are selected and the result after exchange becomes as shown in Fig 5.

$$\begin{array}{c} M_1 \\ M_2 \\ M_3 \end{array} \begin{pmatrix} J_1 & J_2 & J_3 \\ J_2 & J_1 & J_3 \\ \boxed{J_3} & \boxed{J_1} & J_2 \end{pmatrix} \Rightarrow \begin{array}{c} M_1 \\ M_2 \\ M_3 \end{array} \begin{pmatrix} J_1 & J_2 & J_3 \\ J_2 & J_1 & J_3 \\ \boxed{J_1} & \boxed{J_3} & J_2 \end{pmatrix}$$

Fig. 5. Example of job processing order and job exchange

*Step 3.* Based on the job processing sequence after job exchange, dissolve the conflict occurred in Step 4 of an active scheduling generating algorithm, and generate a new solution. If the obtained solution is different from the solution before job exchange, set the solution as a neighborhood solution $X$ and go to Step 4. Otherwise, return to Step 2, and select a new exchange pair.

*Step 4.* If the objective function value of the solution through exchange is improved, accept the exchange, and set $X^c = X$. Otherwise, determine the acceptance by the following substeps.

1. Using the decrement $\Delta f$ of the objective function value and temperature $T$, calculate exp ($-\Delta f / T$).
2. Generate a uniform random number on the open interval (0, 1) and compare it with the value of exp ($-\Delta f / T$).
3. If the value of exp ($-\Delta f / T$) is greater than the random number, accept the exchange, and set $X^c = X$. Otherwise, the exchange is not accepted.

When the exchange is accepted, go to Step 5. Otherwise, return to Step 2 to find the next exchange pair.

*Step 5.* The equilibrium state test is performed by checking whether the change of the objective function value obtained through the exchanges in the prescribed number of times is small enough or not. The number for the equilibrium state test is called the epoch. Here, the test is performed in the following substeps.

1. Repeat the procedures from Step 2 to Step 4, until the exchanges are performed by the number of epoch. When reached the epoch number, perform the following substeps (2)-(4).
2. Calculate the average value $\bar{f}_e$ of the objective function values during the current epoch and the average value $\bar{f}'_e$ of the objective function values through the exchanges thus far.
3. Check whether the relative error between the average value $\bar{f}'_e$ in the whole and the average value $\bar{f}_e$ during the epoch is smaller than the prescribed tolerance value $\varepsilon$ or not, i.e., check whether $\left(\left|\bar{f}_e - \bar{f}'_e\right| / \bar{f}'_e\right) < \varepsilon$ holds or not.
4. When the relative error is smaller than the tolerance value, regard the equilibrium state is reached at this temperature, and go to Step 6 to decrease the temperature. Otherwise, clear the counter of the epoch, and return to Step 2 to repeat the job exchange process.

*Step 6.* Starting with an initial temperature $T_0$, decrease the temperature with the predetermined ratio $\alpha$, i.e., $T_{\text{new}} = \alpha \times T_{\text{old}}$.

*Step 7.* If the number of the pair exchanges reaches the predetermined number, stop the algorithm.

Repeating this process, when the algorithm is terminated, select the solution with the best objective function value among the obtained solutions.

## 5. Numerical example

According to our observations, there is no comparable mathematical model in the literature to compare with the proposed model. So, first a small test problem has been solved only to investigate behavior of proposed mathematical model. Tables 1, 2 and 3 summarize the data used for two numerical examples with 10 jobs.

Here, the behavior of the proposed fuzzy model is appraised for different α (α ϵ [0, 1]) through two solving methods. In Table 4, the model has been solved by the Lingo 13.0 solver and its computational times compared with the GA. The experiments were run in an Intel(R) core(TM) i3 CPU, at 2.13GHz and with 4.00 GB of RAM memory.

To solve the example problem, gave the common data of due dates, fixed part of the processing times and deteriorating cost. In our experiments, the population size is 30 and the new individuals are created with a crossover rate of 0.45. The termination criterion is the completion of 30 generations of individuals. The problem is solved for *p* = 1 and the importance weight of objective functions are determined as $\alpha_1$=0.27, $\alpha_2$=0.23, $\alpha_3$=0.29 and $\alpha_4$=0.21.

Table 1
Some parameters for generation of problem instances (weights, arrival times and due dates).

| Jobs | $\beta_i$ | $\gamma_i$ | $r_i$ | $\tilde{d}_i$ |
|---|---|---|---|---|
| 1 | 2 | 1 | 125 | (220, 244, 256) |
| 2 | 3 | 1 | 70 | (119, 132, 143) |
| 3 | 1 | 1 | 20 | (63, 78, 82) |
| 4 | 1 | 2 | 15 | (47, 62, 74) |
| 5 | 1 | 3 | 105 | (170, 184, 196) |
| 6 | 1 | 1 | 90 | (122, 133, 144) |
| 7 | 3 | 3 | 50 | (157, 160, 172) |
| 8 | 2 | 2 | 0 | (27, 41, 53) |
| 9 | 2 | 1 | 95 | (123, 146, 177) |
| 10 | 2 | 3 | 110 | (180, 194, 206) |

Table 2
The growth rate and fixed part of the processing time

| Jobs | $\tilde{b}_i$ | $\tilde{a}_{i1}$ | $\tilde{a}_{i2}$ | $\tilde{a}_{i3}$ |
|---|---|---|---|---|
| 1 | (0.55, 0.60, 0.62) | (21, 22, 23) | (11, 18, 21) | (11, 20, 26) |
| 2 | (0.24, 0.26, 0.27) | (26, 28, 35) | (25, 26, 34) | (20, 29, 31) |
| 3 | (0.75, 0.75, 0.78) | (44, 46, 53) | (47, 50, 56) | (44, 50, 54) |
| 4 | (0.39, 0.42, 0.44) | (43, 48, 51) | (43, 50, 56) | (45, 48, 56) |
| 5 | (0.32, 0.33, 0.34) | (32, 36,37) | (29, 34, 42) | (33, 35, 43) |
| 6 | (0.17, 0.18, 0.19) | (14, 18, 22) | (10, 18, 23) | (9, 18, 24) |
| 7 | (0.60, 0.65, 0.66) | (38, 46, 54) | (37, 46, 50) | (42, 46, 49) |
| 8 | (0.03, 0.04, 0.04) | (40, 44, 49) | (40, 46, 53) | (42, 43, 49) |
| 9 | (0.09, 0.10, 0.10) | (9, 12, 19) | (7, 11, 19) | (5, 11, 14) |
| 10 | (0.14, 0.15, 0.16) | (32, 37, 45) | (25, 34, 37) | (30, 34, 39) |

Table 3
Machines deteriorating cost

| Jobs | $\tilde{c}_{i1}$ | $\tilde{c}_{i2}$ | $\tilde{c}_{i3}$ |
|---|---|---|---|
| 1 | (1.1, 1.2, 1.2) | (1.7, 2.0, 2.1) | (1.6, 1.7, 1.8) |
| 2 | (5.2, 5.4, 5.6) | (4.0, 4.3, 4.3) | (3.1, 3.2, 3.5) |
| 3 | (2.4, 2.4, 2.5) | (2.0, 2.2, 2.3) | (2.2, 2.4, 2.5) |
| 4 | (0.7, 0.8, 0.8) | (1.2, 1.3, 1.4) | (0.9, 1.1, 1.1) |
| 5 | (8.7, 9.5, 9.9) | (7.1, 7.9, 8.2) | (8.0, 8.7, 9.0) |
| 6 | (5.9, 6.0, 6.2) | (4.3, 4.4, 4.7) | (6.5, 7.1, 7.4) |
| 7 | (0.7, 0.8, 0.8) | (0.7, 0.8, 0.8) | (0.9, 1.0, 1.0) |
| 8 | (5.6, 6.1, 6.3) | (4.4, 4.8, 5.2) | (4.5, 4.8, 5.2) |
| 9 | (8.4, 8.7, 8.8) | (8.6, 9.6, 10.0) | (11.6, 12.2, 12.7) |
| 10 | (3.4, 3.7, 3.8) | (2.9, 3.1, 3.2) | (2.9, 3.1, 3.2) |

Table 4 summarizes the evaluation results obtained by heuristic method which is coded in C++ as a sub-algorithm, according to a group of parameters defined in Table 1 to Table 3. In this dimension of the problem, the acquired results by the methods show same amount for the objective function. Therefore, to evaluate the GA solutions, the reported computation times are used as performance measure. Fig 6 illustrates the Gantt chart of this problem in α = 0.2.

In the next step, different illustrative examples have been developed to discuss the effectiveness of the proposed method in dealing with different problem sizes. To compare the performance of proposed GA on problems with high size, fourteen instances have been generated (Table 5). Other parameters are generated randomly using uniform distributions specified in Table 6.

Table 4
Evaluation of results

| | α | Objective value | CPU time (Sec.) | | Improvement (%) |
|---|---|---|---|---|---|
| | | | GA | Lingo | |
| (1) | 0 | 0.272 | 208 | 251 | 17.1 |
| (2) | 0.2 | 0.222 | 205 | 247 | 17.0 |
| (3) | 0.4 | 0.260 | 56 | 251 | 77.7 |
| (4) | 0.5 | 0.251 | 99 | 249 | 60.2 |
| (5) | 0.6 | 0.267 | 248 | 248 | 0.0 |
| (6) | 0.8 | 0.217 | 202 | 245 | 17.6 |
| (7) | 1.0 | 0.273 | 205 | 248 | 17.3 |

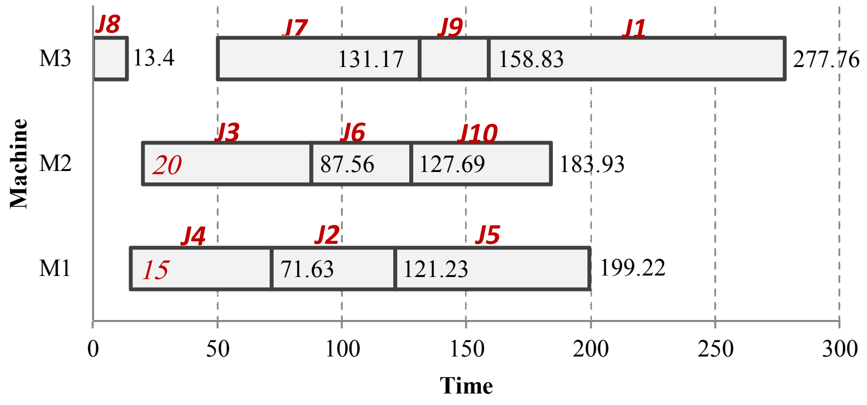

Fig. 6. Gantt chart for α = 0.2

Table 5
Test problems' dimensions.

| Problem number | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| The number of jobs | N | 10 | 15 | 20 | 25 | 30 | 40 | 50 | 70 | 100 | 150 | 200 | 300 | 400 | 500 |
| The number of machines | M | 4 | 5 | 10 | 10 | 15 | 20 | 25 | 30 | 40 | 50 | 70 | 100 | 150 | 200 |

Table 6
The values of the parameters used in the test problems

| Parameter | Symbol | Range |
|---|---|---|
| Earliness weight | $\gamma_i$ | ~ uni[1,3]* |
| Tardiness penalty | $\beta_i$ | ~ uni[1,3] |
| Processing fix time | $\tilde{a}_{im}$ | ~ uni[0,60] |
| Processing variable time | $\tilde{b}_i$ | ~ uni[0,1] |
| Arrived time | $r_i$ | ~ uni[0,100] |
| Due date | $\tilde{d}_i$ | ~ uni[100.200] |
| Deteriorating cost | $\tilde{c}_{im}$ | ~ uni[0,15] |

* Uniform distribution [lower bound, upper bound]

In Table 7, the evaluation results have been summarized through different α values (α ϵ {0.2, 0.5 and 1}), according to a group of parameters defined in Table 6. The parameter values of SA are set as 0.5, 0.9, 10'000, 5 and 0.2 that denote the initial temperature ($T_0$), the changing ratio ($\alpha$), the number of search ($S$), the number of epoch and the tolerance value ($\varepsilon$), respectively. Here, an initial temperature is set to be 277 for the last instance when solving the problems which minimize the objective function using SA.

It should be emphasized here that these parameter values are found through a lot of experiences and these values are used in all of the trials of GA and SA. All of the trials of GA and SA are performed 10 times for each of the problems. The average times required for computation are respectively shown in Table 7.

Table 7
The performance of the solution methods on the problem instances

| Instance | α | No. of best value | | Best objective value | | CPU time (Sec.) | | |
|---|---|---|---|---|---|---|---|---|
| | | GA | SA | GA | SA | GA | SA | Lingo |
| | 0.2 | 10 | 3 | 0.158 | 0.158 | 271 | 344 | 1,906 |
| 1 | 0.5 | 10 | 9 | 0.247 | 0.247 | 298 | 358 | 1,947 |
| | 1 | 10 | 7 | 0.190 | 0.190 | 281 | 334 | 1,837 |
| | 0.2 | 10 | 3 | 0.143 | 0.143 | 313 | 369 | 806 |
| 2 | 0.5 | 10 | 4 | 0.148 | 0.148 | 314 | 399 | 797 |
| | 1 | 10 | 4 | 0.170 | 0.170 | 299 | 365 | 806 |
| | 0.2 | 8 | 4 | 0.313 | 0.313 | 316 | 370 | 5,529 |
| 3 | 0.5 | 7 | 1 | 0.312 | 0.312 | 338 | 429 | 5,657 |
| | 1 | 7 | 1 | 0.207 | 0.207 | 344 | 423 | 5,604 |
| | 0.2 | 7 | 5 | 0.526 | 0.526 | 418 | 514 | 28,358 |
| 4 | 0.5 | 6 | 6 | 0.777 | 0.777 | 414 | 530 | 28,159 |
| | 1 | 8 | 5 | 0.793 | 0.793 | 430 | 516 | 26,655 |
| | 0.2 | 9 | 7 | 0.375 | 0.375 | 428 | 535 | - |
| 5 | 0.5 | 10 | 4 | 0.392 | 0.392 | 426 | 541 | - |
| | 1 | 9 | 5 | 0.328 | 0.328 | 446 | 540 | - |
| | 0.2 | 9 | 6 | 0.543 | 0.543 | 460 | 575 | - |
| 6 | 0.5 | 9 | 5 | 0.518 | 0.518 | 438 | 561 | - |
| | 1 | 9 | 4 | 0.456 | 0.456 | 460 | 593 | - |
| | 0.2 | 8 | 3 | 0.166 | 0.166 | 542 | 683 | - |
| 7 | 0.5 | 8 | 3 | 0.275 | 0.275 | 514 | 648 | - |
| | 1 | 7 | 3 | 0.207 | 0.207 | 536 | 697 | - |
| | 0.2 | 5 | 1 | 0.547 | 0.547 | 833 | 1075 | - |
| 8 | 0.5 | 9 | 2 | 0.533 | 0.533 | 807 | 1049 | - |
| | 1 | 8 | 0 | 0.477 | 0.481 | 832 | 1073 | - |
| | 0.2 | 4 | 1 | 0.588 | 0.588 | 1485 | 1856 | - |
| 9 | 0.5 | 5 | 0 | 0.596 | 0.604 | 1437 | 1839 | - |
| | 1 | 5 | 2 | 0.518 | 0.518 | 1506 | 2003 | - |
| | 0.2 | 7 | 0 | 0.268 | 0.275 | 2677 | 3507 | - |
| 10 | 0.5 | 7 | 2 | 0.224 | 0.224 | 2626 | 3493 | - |
| | 1 | 6 | 0 | 0.194 | 0.217 | 2475 | 3069 | - |
| | 0.2 | 8 | 0 | 0.316 | 0.330 | 6464 | 8403 | - |
| 11 | 0.5 | 7 | 0 | 0.291 | 0.362 | 7119 | 10109 | - |
| | 1 | 3 | 1 | 0.342 | 0.342 | 6447 | 8510 | - |
| | 0.2 | 2 | 0 | 0.433 | 0.452 | 10452 | 14215 | - |
| 12 | 0.5 | 0 | 1 | 0.311 | 0.303 | 11304 | 15147 | - |
| | 1 | 2 | 0 | 0.310 | 0.363 | 12183 | 16691 | - |
| | 0.2 | 5 | 0 | 0.759 | 0.796 | 18909 | 26284 | - |
| 13 | 0.5 | 3 | 0 | 0.748 | 0.756 | 22594 | 31180 | - |
| | 1 | 2 | 0 | 0.650 | 0.734 | 18165 | 25613 | - |
| | 0.2 | 3 | 0 | 0.552 | 0.627 | 28460 | 39844 | - |
| 14 | 0.5 | 1 | 0 | 0.536 | 0.618 | 25936 | 36570 | - |
| | 1 | 4 | 0 | 0.607 | 0.708 | 26152 | 37397 | - |

According to was done sensitive analysis fronts of different number of α for presented heuristic method, it can be seen that the computation times for GA are dramatically lower than for Lingo as its computation time rapidly increases with the growing number of binary variables.

Number of best value in this table denotes the number of the best solution obtained among 10 trials. For the fourteen different problems the proposed GA reaches the best solution for almost all of the trials. On the contrary, although SA gives much more stable solutions compared to GA, the number of best solution is quite small and it is extremely low especially for Problem 2. Therefore, through our numerical experiences, it may be concluded that the proposed GA gives much more efficient search and stability than SA.

## 6. Conclusion

In this paper, a fuzzy scheduling problem of parallel machines has been studied in minimizing total weighted tardiness/earliness, jobs flowtime and machine deteriorating cost. First, a fuzzy mathematical formulation was developed and then after defuzzified, the proposed multi-objective optimization model was transferred to a single-objective form using a method that minimized the distance to the ideal vector. The final model was solved and the results were compared by two heuristics methods, genetic algorithm and simulating annulling, for different illustrative examples to analyze and validate the approach. Computational results show efficiency and effectiveness of the developed heuristic solution methods when time complexity is addressed.

This study developed several issues that could be further investigated in future research. For example, the accuracy and efficiency of the proposed method could be improved. A number of verification and validation methods may be helpful in testing the accuracy and consistency of the process.

## 7. Appendix A: (Noor approach)

Consider general forms of interval programming:
Model A-1:

$$\text{Min} \quad Z=\sum_{j=1}^{n}[c_j^L,c_j^U]x_j$$

s.t:

$$\sum_{j=1}^{n}[a_{ij}^L,a_{ij}^U]x_j \geq [b_i^L,b_i^U] \qquad i=1,\dots,m$$

$$x_j \geq 0 \qquad j=1,\dots,n$$

With applying convex conversion as following:

$$c_j^L \leq c_j \leq c_j^U \Rightarrow c_j=\lambda_j c_j^U+(1-\lambda_j)c_j^L$$
$$=c_j^L+\lambda_j(c_j^U-c_j^L)$$
$$0 \leq \lambda_j \leq 1 \qquad j=1,\dots,n$$
$$a_{ij}^L \leq a_{ij} \leq a_{ij}^U \Rightarrow a_{ij}=\beta_{ij}a_{ij}^U+(1-\beta_{ij})a_{ij}^L$$
$$=a_{ij}^L+\beta_{ij}(a_{ij}^U-a_{ij}^L)$$
$$0 \leq \beta_{ij} \leq 1 \qquad j=1,\dots,n \quad i=1,\dots,m$$
$$b_i^L \leq b_i \leq b_i^U \Rightarrow b_i=\beta_i b_i^U+(1-\beta_i)b_i^L$$
$$=b_i^L+\beta_i(b_i^U-b_i^L)$$
$$0 \leq \beta_i \leq 1 \qquad i=1,\dots,m$$

Moreover, with substitute preceding number in model A-1 we result:

$$\text{Min} \quad Z=\sum_{j=1}^{n}[c_j^L+\lambda_j(c_j^U-c_j^L)]x_j$$

s.t:

$$\sum_{j=1}^{n}[a_{ij}^L+\beta_{ij}(a_{ij}^U-a_{ij}^L)]x_j \geq b_i^L+\beta_i(b_i^U-b_i^L)$$
$$i=1,\dots,m$$
$$x_j \geq 0 \qquad j=1,\dots,n$$
$$0 \leq \beta_{ij} \leq 1 \qquad i=1,\dots,m$$
$$j=1,\dots,n$$
$$0 \leq \beta_i \leq 1 \qquad i=1,\dots,m$$
$$0 \leq \lambda_j \leq 1 \qquad j=1,\dots,n$$

And or:
Model A-2:

$$\text{Min} \quad Z=\sum_{j=1}^{n}c_j^L x_j+\sum_{j=1}^{n}\lambda_j x_j(c_j^U-c_j^L)$$

s.t:

$$\sum_{j=1}^{n}a_{ij}^L x_j+\sum_{j=1}^{n}\beta_{ij}x_j(a_{ij}^U-a_{ij}^L)-\beta_i(b_i^U-b_i^L)$$
$$\geq b_i^L \qquad i=1,\dots,m$$
$$x_j \geq 0 \qquad j=1,\dots,n$$
$$0 \leq \beta_{ij} \leq 1 \qquad i=1,\dots,m$$
$$j=1,\dots,n$$
$$0 \leq \beta_i \leq 1 \qquad i=1,\dots,m$$
$$0 \leq \lambda_j \leq 1 \qquad j=1,\dots,n$$

Model A-2 is known as a programming problem with deterministic numbers.

## 8. References

[1] Anglani A, Grieco A, Guerriero E, Musmanno R (2005) Robust scheduling of parallel machines with sequence-dependent set-up costs. European Journal of Operational Research 161:704-720.

[2] Balin S (2011) Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Information Sciences 181:3551-3569.

[3] Bilge Ü, Kıraç F, Kurtulan M, Pekgün P (2004) A tabu search algorithm for parallel machine total tardiness problem. Computers & Operations Research 31:397-414.

[4] Browne S, Yechiali U (1990) Scheduling Deteriorating Jobs on a Single Processor. Operations Research 38:495-498.

[5] Cai X, Zhou S (1999) Stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy jobs. Operations Research 47:422-437.

[6] Cao D, Chen M, Wan G (2005) Parallel machine selection and job scheduling to minimize machine cost and job tardiness. Computers & Operations Research 32:1995-2012.

[7] Davis L (1987) Genetic Algorithms and Simulated Annealing. Morgan Kaufmann (ed), Los Altos, CA.

[8] Falkenauer E (1999) The worth of uniform crossover. In: Proceedings of the 1999 Congress on Evolutionary Computation CEC99, USA.

[9] Gawiejnowicz S, Kurc W, Pankowska L (2006) Analysis of a time-dependent scheduling problem by signatures of deterioration rate sequences. Discrete Applied Mathematics 154:2150-2166.

[10] Guner E, Erol S, Tani K (1998) One machine scheduling to minimize the maximum earliness with minimum number of tardy jobs. International Journal of Production Economics, 55:213-219.

[11] Han S, Ishii H, Fujii S (1994) One machine scheduling problem with fuzzy due date. European Journal of Operational Research 79:1-12**.**

[12] Holland JH, (1975) Adaptation in Natural and Artificial Systems. Arbor A (ed) The University of Michigan Press, MI.

[13] Hong TP, Chuang TN, (1999) A new triangular fuzzy Johnson algorithm. Computers & Industrial Engineering 36:179-200.

[14] Hwang CL, Yoon K (1981) Multiple attribute decision making: Methods and applications. Springer-Verlag, Berlin and New York.

[15] Ishibuchi H, Murata T (2000) Flow shop scheduling with fuzzy due date and fuzzy processing time. In: Slowinski R, Hapke M (ed) Scheduling under Fuzziness.

[16] Ishii H, Tada M, (1995) Single machine scheduling problem with fuzzy precedence relation. European Journal of Operational Research 87:284-288.

[17] Itoh T, Ishii H (1999) Fuzzy due-date scheduling problem with fuzzy processing time. International Transactions in Operational Research 6:639-647.

[18] Kirkpatrick S, Gelatt CD, Vecchi MP, (1983) Optimization by simulated annealing. Science 220:671-680.

[19] Konno T, Ishii H (2000) An open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint. Fuzzy Sets and Systems 109:141-147.

[20] Kuroda M, Wang Z (1996) Fuzzy job shop scheduling. International Journal of Production Economics 44:45-51.

[21] Leu YY, Matheson LA, Rees LP (1994) Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria. Decision Sciences 15:581–606.

[22] Litoiu M, Tadei R (2001) Real-time task scheduling with fuzzy deadlines and processing times. Fuzzy Sets and Systems 117:35-45.

[23] Mazdeh MM, Zaerpour F, Zareei A, Hajinezhad A (2010) Parallel machines scheduling to minimize job tardiness and machine deteriorating cost with deteriorating jobs. Applied Mathematical Modeling 34:1498-1510.

[24] Mazzini R, Armentano VA, (2001) A heuristic for single machine scheduling with early and tardy costs. European Journal of Operational Research 128:129-146.

[25] McKay K, Pinedo M, Webster S (2002) Practice-focused research issues for scheduling systems. Production and Operations Management 11:249-258.

[26] Oguz C, Ercan MF, (2005) A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. J Scheduling 8:323-351.

[27] Page J, Poli P, Langdon WB, (1999) Smooth uniform crossover with smooth point mutation in genetic programming: a preliminary study. In: Genetic Programming, Proceedings of EuroGP'99, Sweden.

[28] Peng J, Liu B (2004) Parallel machine scheduling models with fuzzy processing times. Information Sciences 166:49-66.

[29] Pfund M, Fowler JW, Gadkari A, Chen Y (2008) Scheduling jobs on parallel machines with setup times and ready times. Computers & Industrial Engineering 54:764-782.

[30] Piersma N, Romeijn HE, (1996) Parallel machine scheduling: A probabilistic analysis. Naval Research Logistics (NRL) 43:897-916.

[31] Prade H (1979) Using fuzzy set theory in a scheduling problem: A case study. Fuzzy Sets and Systems 2:153-165.

[32] Radhakrishnan S, Ventura JA (2000) Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. International Journal of Production Research 38:2233-2252.

[33] Raut S, Gupta JND, Swami S (2008) Single machine scheduling with time deteriorating job values. Journal of the Operational Research Society 59:105-118.

[34] Sridharan V, Zhou Z (1996) A decision theory based scheduling procedure for single machine weighted earliness and tardiness problems. European Journal of Operational Research 94:292-301.

[35] Syswerda G (1989) Uniform crossover in genetic algorithms In: Schaffer DJ (ed) Proceedings of the 3rd International Conference on Genetic Algorithms, USA, pp. 2-9.

[36] Wan G, Yen BPC (2002) Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. European Journal of Operational Research 142:271-281.

[37] Wan G, Yen BPC (2009) Single machine scheduling to minimize total weighted earliness subject to minimal

number of tardy jobs. European Journal of Operational Research 195:89-97.

[38] Wu CC, Lee WC, (2008) Single machine group-scheduling problems with deteriorating setup times and job-processing times. International Journal of Production Economics 115:128-133.

[39] Yi Y, Wang DW, (2003) Soft computing for scheduling with batch setup times and earliness-tardiness penalties on parallel machines. Journal of Intelligent Manufacturing 14:311-322.