International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(B)**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 727–745

NON-IDENTICAL PARALLEL MACHINE SCHEDULING WITH FUZZY PROCESSING TIMES USING ROBUST GENETIC ALGORITHM AND SIMULATION

Savaş Balin

Department of Industrial Engineering
Yıldız Technical University
Barbaros Bulvarı 34349, İstanbul, Turkey
sbalin@yildiz.edu.tr

Received July 2010; revised November 2010

Abstract. *This paper addresses non-identical parallel machine scheduling problem with fuzzy processing times (FPMSP). A robust genetic algorithm (GA) approach embedded in a simulation model to minimize maximum completion time (makespan) is proposed. The results are compared with those obtained by using LPT rule, known as the most appropriate dispatching rule for such problems. This application illustrates the need for efficient and effective heuristics to solve FPMSPs. The proposed GA approach yields good results and reaches them fast and several times in one run. Moreover, due to its advantage of being a search algorithm, it can explore alternative schedules providing the same results. Thanks to the simulation model, several robustness tests are conducted using different random number sets and it has been shown that the proposed approach is robust.*

Keywords: Non-identical parallel machine scheduling, Fuzzy processing times, Genetic algorithm, Robustness, Simulation

1. Introduction.

Production scheduling consists of planning jobs that need to be performed in an orderly sequence of operations. It is a difficult problem depending on the number of calculations required to obtain a schedule that optimizes the chosen criterion. In modern manufacturing, depending on machine layout and job flow, several kinds of shop exist. As many practical job shop and open shop scheduling problems can be simplified as parallel machine scheduling problem under certain conditions [1-4], the parallel machine scheduling problem (PMSP) receives a great deal of attention in the academic and engineering field [3]. Moreover, it is a generalization of the single machine problem and a particular case of problems arising in flexible manufacturing systems [5,6].

In fact, various factors involved in the scheduling problems are often imprecise or uncertain in nature when formulating scheduling problems of the real-world. This is especially true when human-made factors are incorporated into the problems. Therefore, in the real-world situations, parameters are often encountered with uncertainties. Accordingly, production scheduling problems have been mainly branched into two categories: deterministic scheduling and uncertain scheduling [7]. There are basically two approaches to deal with uncertainties [8], including the stochastic-probabilistic theory and possibility theory or fuzzy set theory [9,10].

In this study, the fuzzy set theory is used to deal with the uncertainties in production scheduling. It provides an alternative and convenient framework for modeling real-world systems mathematically and offers several advantages in the use of heuristic approaches:

- Stochastic-probabilistic theory needs a great deal of knowledge about the statistical distribution of the uncertain parameters. Though, fuzzy theory provides an efficient way to model imprecision even when no historical information is available [11].
- Using stochastic-probabilistic theory, extensive computation is involved and it is necessary to have complete knowledge on the statistical distribution of the uncertain time parameters [12].
- Using fuzzy set theory decreases the scheduling problem computational complexity with respect to the same problem formulated by stochastic-probabilistic theory [13].
- Fuzzy theory allows using fuzzy rules in heuristic algorithms.
- Instead of optimizing average behaviors like in stochastic-probabilistic theory, fuzzy theory rather aims at finding solutions where all constraints are satisfied to some extent, with a sufficient level of confidence.

Fuzzy scheduling has two classes of application: scheduling under flexible constraints and scheduling under incomplete or imprecise information [8]. This study fits into the second class. Considering that, in many real-world applications, the processing time of each job may vary dynamically, processing times are defined as fuzzy variables. Therefore, the FPMSP studied in this paper is to schedule n independent jobs with fuzzy processing times on m non-identical parallel machines in order to minimize makespan. In scheduling problems, makespan ($C_{\max}$) is equivalent to the completion time of the last job leaving the system. The small $C_{\max}$ usually implies a high utilization. Therefore, reducing the $C_{\max}$ should also lead to a higher throughput rate [14,15].

Most of the scheduling problems are characterized as combinatorial optimization problems. They vary widely according to specific production tasks but most are *NP-hard* problems [16-19]. Obviously, they become more difficult to solve when machines are non-identical and some variables are fuzzy. It is quite difficult to achieve a satisfactory result efficiently and effectively with traditional optimization methods. Mathematical optimization techniques can give an optimal solution for a reasonably sized problem; however, in the case of a large scale problem, their application is limited. Dispatching rules (*LPT*, *SPT*, *EDD*, etc.) are suitable only for small scale problems and no single dispatching rule guarantees good result in various problems [20]. Research efforts have, therefore, concentrated on heuristic approaches.

Many heuristics have been proposed such as *Simulated Annealing, Tabu Search, Branch and Bound, Neural Network and Genetic Algorithm.* Among these various approaches to different scheduling problems, there has been an increasing interest in applying *GA*s in view of its characteristic such as high adaptability, near optimization and easy realization. *GA*s have demonstrated considerable success in providing good solutions to fuzzy scheduling problems. They have been well documented in the literature, such as in [21], and have been applied to a wide variety of optimization problems. In particular, Liu and Iwamura [22,23], Liu [24,25], Li et al. [26], Buckley and Hayashi [27] and Buckley and Feuring [28] designed a spectrum of *GA*s to solve fuzzy programming models. For detailed expositions, the readers may consult Liu [24,29] in which numerous *GA*s have been suggested for solving uncertain programming models [7].

GA has been applied with the same success in solving PMSPs in the literature and there are many applications in this field. However, even though it is a common problem in the industry, only a small number of them deal with non-identical parallel machines [30-32]. Moreover, when some variables are fuzzy, the number of studies is even rare. Among the few authors dealing with fuzzy parallel machine scheduling problem (FPMSPs), Peng and Liu [7] consider a multi-objective FPMSP and formulate it as three-objective models, which minimize the fuzzy maximum tardiness, the fuzzy maximum completion time and the fuzzy maximum idleness. They propose three models to formulate fuzzy scheduling problem: fuzzy goal programming, fuzzy chance-constrained programming and fuzzy dependent-chance programming. They propose a hybrid intelligent algorithm designed to solve the proposed scheduling models. Petrovic and Duenas [33] present a new fuzzy logic based decision support system for parallel machine scheduling/rescheduling in the presence of uncertain disruptions. They use fuzzy rules to determine when to reschedule and which rescheduling method to use. Mok et al. [8] try to optimize fault-tolerant fabric-cutting schedules using *GA* by minimizing makespan. They propose a fuzzification scheme to incorporate uncertainties, in terms of both job-specific and human related factors, into the fabric-cutting scheduling problem. One of the rare studies dealing with scheduling non-identical parallel machines with fuzzy processing times using *GA* belongs to Raja et al. [34]. Their objective is to minimize the sum of the absolute deviations of job completion times. They use fuzzy logic to combine earliness and tardiness penalties. Although all these studies deal with FPMSPs and seem similar, they are quite different from each other and from the study presented here. Two scheduling problems arising from the same production layout can be totally different because of their objective function or because of an additional constraint. Consequently, different methods have to be used for each solution. Likewise, the problem studied here, scheduling non-identical parallel machines with fuzzy processing times to minimize the maximum completion time (makespan), differs from the studies presented above in terms of its objective function, constraints and the use of fuzzy theory. Therefore, methods proposed in these studies are not adequate for its solution. However, this problem is frequently encountered in the industry and is of interest from both practical and theoretical points of view. The aim of this study is filling this gap by proposing a *GA* for minimizing the maximum completion time of non-identical parallel machines where processing times are fuzzy.

In order to adapt *GA* to the studied problem, he proposes new *GA* operators and uses two of the most appropriate fuzzy ranking and defuzzification methods of the field. The *GA* is embedded in a simulation model for solving the problem. The use of simulation in implementing *GA* is preferred because of the evolutionary structure of the algorithm and the ability of simulation to perform tests using different random number sets. By this means, the robustness of the approach proposed can be easily tested in a series of numerical experiments.

A numerical example of FPMSP is solved first by using the proposed *GA* approach and then by *LPT* rule, known as the most appropriate dispatching rule for such problems. Results show that the proposed approach surpasses *LPT* rule and highlights the need of using efficient and effective heuristics for FPMSPs.

The paper is structured as follows. First, a quick review of fuzzy set operations, ranking and defuzzification methods is made in Section 2. The formulation of non-identical parallel machine scheduling problem with fuzzy processing times in order to minimize makespan is given in Section 3. The general structure of *GA* is adapted to the studied problem in Section 4. *GA* is embedded in a simulation model because of its computational advantage in running tests. Numerical examples are solved in Section 5. In the first paragraph (5.1), the proposed *GA* is tested with crisp values; in the second paragraph (5.2), processing times are considered as fuzzy numbers; in the third paragraph (5.3), these results are compared to those obtained with *LPT* rule. In Section 6, numerical results are discussed and the robustness of the approach is tested. The paper ends in Section 7 with a brief summary of the main findings.

2. **Review of Related Fuzzy Theory.**

In this section, some concepts of fuzzy theory necessary for formulating FPMSP are reviewed. Given a set of tasks, each with its processing time membership function, the *GA* can get a scheduling result with a final completion-time membership function.

2.1. **Fuzzy numbers and fuzzy set operations.**

Fuzzy ranking and defuzzification methods used in this study are suitable for all types of fuzzy numbers. However, only triangular fuzzy numbers (TFN) are used as example of application because of their computational advantage. Besides, they are largely used for subjective description as they are based on a knowledge of the minimum, maximum and an "inspired guess" as to the modal value. But, note that the proposed fuzzy ranking and defuzzification methods can be performed similarly for all fuzzy numbers with different shapes. Therefore, the proposed *GA* is generic and the results of this study will be valid for all types of fuzzy numbers.

A triangular fuzzy number (TFN) can be denoted as:

$$A = [\mu_{a1}/x_{a1},\ \mu_{a2}/x_{a2},\ \mu_{a3}/x_{a3}] = [0/p, 1/q, 0/r]$$

For a TFN, we have always $\mu_{a1} = \mu_{a3} = 0$ and $\mu_{a2} = 1$. Therefore, a TFN can be denoted as:

$$A = (p, q, r)$$

Let $A_1 = (p_1, q_1, r_1)$ and $A_2 = (p_2, q_2, r_2)$ be two TFN, their sum can be denoted as:

$$A_1 + A_2 = (p_1 + p_2, q_1 + q_2, r_1 + r_2)$$

The multiplication of a TFN and a crisp value k, $k \in \Re_0^+$, is defined as:

$$k \times A \equiv (k \times p, k \times q, k \times r)$$

A TFN can also be denoted by its membership function:

$$\mu_A(x) = \begin{cases} \dfrac{x-p}{q-p} & \text{if } p \le x \le q, \\ \dfrac{x-r}{q-r} & \text{if } q \le x \le r, \\ 0 & \text{elsewhere} \end{cases}$$

Therefore, a TFN can be represented graphically as below:

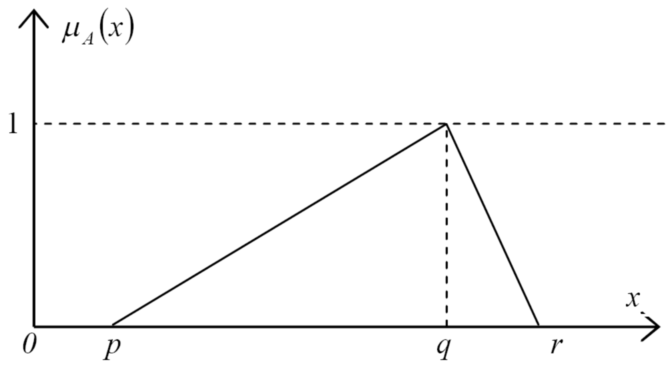

FIGURE 1. Triangular fuzzy number (TFN)

2.2. **Ranking method.** Ranking two numbers is quite important in a scheduling algorithm. Ranking is quite easily performed since the operands are all crisp numbers. However, in this paper, since fuzzy processing times are considered, fuzzy ranking methods have to be used. In literature, several fuzzy comparison methods have been proposed, such as the Hamming distance method, the probability distribution method, pseudoorder fuzzy preference model [35], new fuzzy-weighted average [36], signed distance method [37] and so on. Among these methods, the "signed distance" method is suitable for fuzzy processing time comparison because it is simple in computation and flexible to convert from the interval data [38].

Signed distance of a TFN, $A = (p, q, r)$, from y-axis is denoted by:

$$d(A, 0) = (p + 2q + r)/4$$

TFNs can be ranked by signed distance method as follows [35]:

$$\begin{array}{ll} B \prec A & \text{if } d(B,0) < d(A,0), \\ B = A & \text{if } d(B,0) = d(A,0), \\ B \succ A & \text{if } d(B,0) > d(A,0). \end{array}$$

The signed distance of a TFN can be graphically interpreted as the barycenter of the two parts of its membership function (Figure 2).

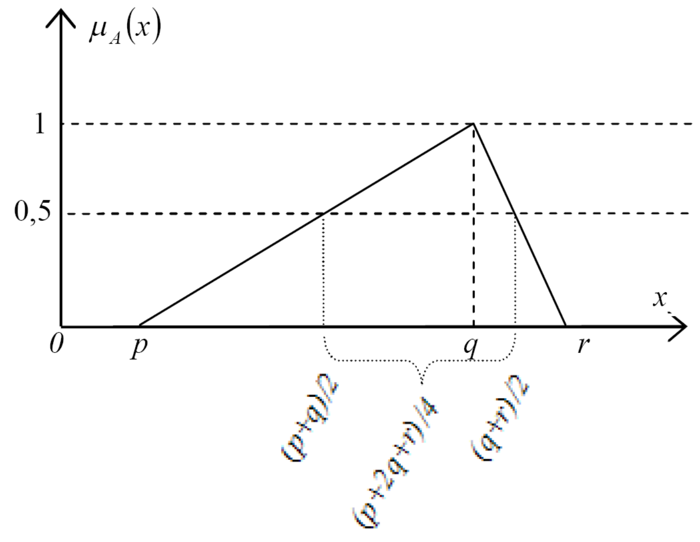

FIGURE 2. Calculation of signed distance for a triangular fuzzy number

The barycenter of the segments $[(p, 0); (q, 1)]$ and $[(q, 1); (r, 0)]$ will be situated at $(p + 2q + r)/4$ on y-axis.

2.3. **Defuzzification method.** Defuzzification is a reverse process of fuzzification and it converts a fuzzy number into a crisp value. The "centroid" defuzzification method is one of the most prevalent and physically appealing defuzzification method. It returns the center of area under the curve. Let A be a fuzzy number and μ_A its membership function. Defuzzification of A by the "centroid" method is denoted by $\delta(A)$ and returns the crisp value of x^*:

$$x^* = \delta(A) = \frac{\int \mu_A(x).xdx}{\int \mu_A(x)dx}$$

For a TFN, $A = (p, q, r)$, $x^* = \delta(A) = (p + q + r)/3$ and it represents the centroid/barycenter of the triangle $((p, 0); (q, 1); (r, 0))$ presented in Figure 3.

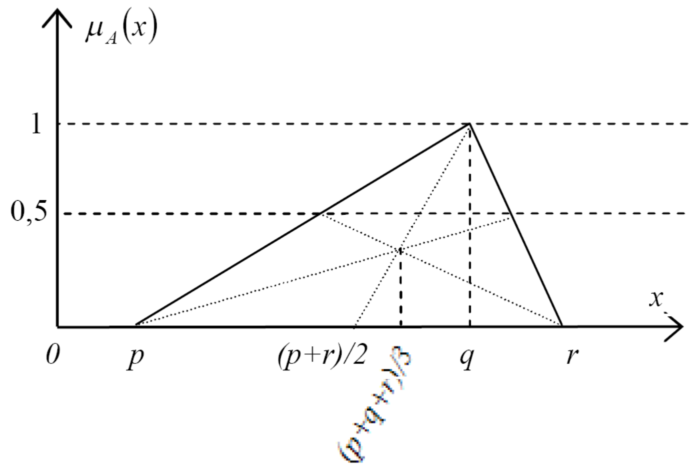

FIGURE 3. Centroid defuzzification method for a triangular fuzzy number

3. **Problem Formulation.** The FPMSP considered in this study can be described as follows: n jobs need to be processed on m non-identical parallel machines and it is desired to minimize the maximum completion time (makespan). Notations, assumptions and constraints used for FPMSP in this study are described as follows:

Notations

n: number of jobs.

m: number of machines.

$i = 1, \ldots, n$: the jobs to be scheduled.

$j = 1, \ldots, m$: the machines.

V_i: speeds of machines.

$P_{i,j}$: processing time of job j on machine i.

$x(i,j)$: boolean variable which determines whether job j is processed by machine i.

$X_{(n\times m)}$: matrix composed of $x(i,j)$ and representing the scheduling of n jobs on m machines.

C_i: fuzzy completion time of machine i.

Assumptions

- Machines have different speeds (V_i).
- All jobs are available simultaneously at time zero.
- Each job has only one operation and can be processed on any machine.
- The processing times are assumed to be fuzzy variables ($P_{i,j}$).
- The execution-time membership function of each task is known and finite.

Constraints

- Each machine can process only one job at a time:

$$x(i,j) \in \{0,1\}$$

- Each job is to be processed without interruption:

$$\sum_{i=1}^{m} x\ (i,j) = 1, \quad j = 1, \ldots, n$$

- All jobs have to be processed to complete the production:

$$\sum_{j=1}^{n} \sum_{i=1}^{m} x(i,j) = n$$

Thus, we have following equations:

- Processing time of a job j on different machines can be expressed as:

$$P_{k,j} \times V_k = P_{l,j} \times V_l, \quad k, l = 1, \ldots, m \tag{1}$$

- C_i, fuzzy completion time of machine i:

$$C_i = \sum_{j=1}^{n} x(i,j) \times P_{i,j}, \quad i = 1, \ldots, m \tag{2}$$

- The maximum completion time (makespan) is equal to:

$$C_{\max} = \max_{i=1}^{m} \{C_i\} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times P_{i,j} \right\} \tag{3}$$

- The objective function can be formulated as follows:

$$\min C_{\max} = \max_{i=1}^{m} \left\{ \sum_{j=1}^{n} x(i,j) \times P_{i,j} \right\} \tag{4}$$

4. **Genetic Optimization of FPSMP.**

The *GA*, inspired by the process of Darwinian evolution, has been recognized as a general search strategy and as an optimization method which is often useful for attacking combinatorial problems. It is introduced in the 1970's by Holland [39] and Davis and Coombs [40] were first to propose *GA* for solving scheduling problems. Since then, a significant number of applications have been appearing.

In contrast to other local search methods, such as *Simulated Annealing* or *Tabu Search*, which are based on handling one feasible solution, *GA* utilizes a population of solutions in its search, giving it more resistance to premature convergence on local minima [41]. Another advantage of *GA* is having a good performance in a large and complex search space. The ability of *GA* to explore and to exploit simultaneously the search space, makes it more efficient than the other methods. Nevertheless, *GA* has also some disadvantages; its performance, in terms of the optimality, is highly dependent on the realization of each of the above steps, and on the values of the *GA* parameters, that are number of solutions in the initial population (i.e., population size), number of generations (i.e., termination criterion), and probability values for genetic operators (i.e., crossover and mutation probabilities) [21,42].

A generic *GA* starts by creating an initial population of chromosomes. Each chromosome encodes a solution of the problem, and its fitness value is related to the value of the objective function for that solution. During each iteration step (or called "generation"), genetic operations, that is, reproduction, natural selection, crossover and mutation are applied in order to search potential better solutions. Crossover combines two chromosomes to generate next-generation chromosomes preserving their characteristics. Mutation reorganizes the structure of genes in a chromosome randomly so that a new combination of genes may appear in the next generation. It serves the search by jumping out of local optimal solutions. Reproduction is to copy a chromosome to the next generation directly so that chromosomes from various generations could cooperate in the evolution and the "quality" of the population may be improved after each generation [43]. In following paragraphs, the general structure of *GA* is adapted to the fuzzy parallel machine scheduling problem in order to minimize maximum completion time (makespan). A specific coding method is used to represent schedules on parallel machines. Fuzzy set operations, fuzzy ranking and defuzzification methods presented above are employed to schedule jobs with fuzzy processing times. Some genetic operators are to be redesigned for the considered problem.

4.1. **Coding.** The raw i of the matrix $\mathbf{X}$ consists of jobs to be processed on machine i. Raws are called "*genes*" $(g_1, \ldots, g_i, \ldots, g_m)$ and they represent jobs to be processed on each machine; jobs to be processed on machine $i$are given by elements non-zero of gene $i(x(i,j) = 1)$. The completion time of machine i, (C_i), is equal to the sum of processing times of jobs to be processed on that machine; it is called as the "*value of gene i*" and it is defined by the following function:

$$f(g_i) = \sum_{j=1}^{n} x(i,j) \times P_{i,j}, \quad i = 1, \ldots, m \tag{5}$$

4.2. **Generation of initial population.** As the matrix $\mathbf{X}$ represents the scheduling of n jobs on m machines, an initial solution can be obtained randomly by having only one non-zero element, $x(i,j) \in \{0,1\}$, in each column. Several initial solutions can be obtained by repeating the same operation and each initial solution is called "chromosome". Chromosomes can be identified by their order of creation, $k \in N$. Initial population is the set consisting of N chromosomes. The number of chromosomes, i.e., population size, is one of the important parameters of GA. Members of the initial population (chromosomes) are the parents of the next generations and the efficiency of the algorithm is highly dependent on their "quality".

4.3. **Calculation of fitness values.** Fitness is the performance evaluation of chromosomes [44]. After the generation of new population, fitness value of each chromosome k is calculated (F_k). The higher the fitness value, the better the performance of the chromosome (i.e., parent). Hence, parents with higher fitness values have more chances to survive. In the proposed formulation, each chromosome represents a schedule of parallel machines. Therefore, the performance of a chromosome has to be measured by the maximum completion time that it provides.

In FPSMP, completion times are fuzzy numbers and maximum completion time has to be determined using "signed distance" ranking method presented in 2.2. Let C_i, be the completion time of machine i and denoted by a TFN, $C_i = (p_i, q_i, r_i)$. The maximum completion time is equal to:

$$C_{\max} = \max_{i=1}^{m} \{C_i\} = (p_t, q_t, r_t), \quad t = 1, \ldots, m$$
$$\text{with} \left(\frac{p_t + 2 \times q_t + r_t}{4} \right) = \max_{i=1}^{m} \left\{ \frac{p_i + 2 \times q_i + r_i}{4} \right\}$$

The completion time provided by a chromosome k is denoted by $C_{\max}(k)$. As the objective function of the considered problem is the minimization of the makespan, the fitness value of chromosome k can be obtained using the following function [42]:

$$F_k = \alpha \times e^{-\beta \times \delta(C_{\max}(k))}, \tag{6}$$

where α and β are positive real numbers and $\delta(C_{\max}(k))$ is the defuzzified makespan of chromosome k. Let $C_{\max}(k) = (p_t(k),\ q_t(k),\ r_k(k))$ be the maximum completion time given by chromosome k.

$$\delta\left(C_{\max}(k)\right) = \left(p_t(k) + q_t(k) + r_t(k)\right)/3$$

4.4. **Reproduction.** Reproduction is the process in which parents copy themselves according to the probabilities that are proportional to their fitness values. As a result, parents with higher fitness values will have higher probabilities of producing their offspring in the next generation [44]. Here, such parents are selected using the *roulette wheel* method. The roulette wheel selection method ranks the chromosomes based on their fitness function values, and then assigns them a probability distribution in favor of good chromosomes, so as to obtain a better chance of producing good next generations [45]. It has following steps [42]:

- Calculate selection probability of chromosomes k, $P(k) = \dfrac{F_k}{\sum\limits_{i=1}^{N} F(i)}$ $k = 1, \ldots, N$.
- Generate cut points, $S(k)$,

$$S(0) = 0$$
$$S(k) = P(1) +, \ldots, + P(k), \quad k = 1, \ldots, N.$$

- Generate N random number uniformly distributed between 0 and 1, ζ_s for $s = 1, \ldots, N$.
- For each ζ_s, equation $S(k-1) < \zeta_s < S(k)$ gives chromosomes to be selected.

4.5. **Crossover.** According to the coding method used in this paper, crossover operator deals with genes, and not with chromosomes like most of the applications in literature. Each gene i consists of jobs to be processed on machine i, $(x(i,j) \neq 0,\ j = 1, \ldots, n)$; each chromosome contains m genes and each one consists in one feasible schedule. The new proposed crossover operator combines two genes of the same chromosome in a proper order to obtain a new chromosome giving a better feasible solution. The first gene to combine, g_u, indicates the machine with maximum completion time:

$$f(g_u) = \max_{i=1}^{m} \{f(g_i)\} = C_{\max} \tag{7}$$

The job with the shortest processing time of gene g_u is denoted by j' and given by the equation below:

$$P_{u,j'} = \min_{j=1}^{k} \{P_{u,j}\} = (p_{j'}, q_{j'}, r_{j'}), \quad k < m \text{ and } x_{uj} \neq 0 \tag{8}$$

$$\text{with } \left(\frac{p_{j'} + 2 \times q_{j'} + r_{j'}}{4}\right) = \min_{j=1}^{k} \left\{\frac{p_j + 2 \times q_j + r_j}{4}\right\}$$

Crossover operation is carried out by moving job j' to another machine. The new machine is represented by gene g_z and given by the following equation:

$$(f(g_u) \text{-} f(g_z)) \times \frac{V_z}{V_u} = \max_{i=1}^{m} \left\{(f(g_u) - f(g_i)) \times \frac{V_i}{V_u}\right\} \tag{9}$$

The processing time of job j' on machine z, $P_{z,j'}$, can be calculated using Equation (1):

$$P_{z,j'} = P_{u,j'} \times \frac{V_u}{V_z}$$

4.6. **Mutation.** Crossover operator is used to combine existing genes in order to obtain new chromosomes, whereas mutation operator creates new chromosomes by causing small perturbations in genes. Therefore, it helps to maintain the diversity of the population and to extend the solution space. Because of the coding method used in this paper, mutation operator can be applied easily by alternating some elements $x(i,j)$ of matrix $\mathbf{X}$. Every column of matrix $\mathbf{X}$ has only one element valued "1" (one job is processed only on one machine). Mutation can be carried out by moving randomly this element from one raw to another (i.e., moving a job from one machine to another). This operation prevents from getting stuck on local suboptimal solutions and it is very helpful to maintain the richness of the population in dealing with large scale problems.

4.7. **Optimality criterion.** Optimality test is performed as the last step of the algorithm. The optimality criterion for a chromosome k is given by:

$$P_{u,j'} > \left(f\left(g_u\right) - f\left(g_i\right) \times \frac{V_i}{V_u} \right), \quad i = 1, \ldots, m. \tag{10}$$

Two sides of the inequality can be calculated easily using Equations (7) and (8). They are both TFN and can be compared using the “signed distance” ranking method. If the inequality x is satisfied for a chromosome, it means that this chromosome cannot provide better solutions. If it is satisfied for all chromosomes, none of the completion times ($C_{\max}(k)$) can be improved; thus, algorithm ends and the solution is given by the equation:

$$\min C_{\max} = \min_{k=1}^{N} \{C_{\max}(k)\}$$

The minimum makespan is a TFN and can be denoted as: $\min C_{\max} = (p_k, q_k, r_k)$. A final crisp solution can be obtained by defuzzification:

$$x^* = \delta(\min C \max) = (p_k + q_k + r_k)/3$$

5. **Numerical Examples and Simulation Results.** *GA* is embedded in a simulation model and performed using *Simul8*^{TM} simulation software (Figure 4). Each “work center” corresponds to one step of the algorithm. Simulation entity passes through “work centers” to complete the algorithm.

A new population is created at the beginning of each flow of the entity and a new solution is obtained at the end of the flow. Each flow, i.e., iteration, begins with a randomly generated population of 10 chromosomes, half of the chromosomes are exposed to “natural selection” and the rest of them undergo a crossover operation. Crossover operation is repeated until the optimality criterion is satisfied. Between two crossover operations, chromosomes are mutated in order to ensure “genetic diversity”. It helps to extend the solution space by causing small perturbations in genes. The lifetime of each generation is modeled by an entity passing through the system. The initial population survives 100 generations, it means 100 iterations are performed to complete one simulation run. Each generation provides one solution to the problem, at the end of simulation, the solution with the minimum value is kept as the result of the *GA*. The use of simulation allows to perform several iterations by using different initial populations. Accordingly, several tests can be easily performed using different random number sets.

5.1. **Testing with crisp values.** The proposed approach is first tested with crisp processing times. The case study can be described as follows: 9 jobs need to be processed on 4 non-identical parallel machines and it is desired to minimize the maximum completion time (makespan).

The following assumptions are used:

- All jobs are available simultaneously at time zero.
- Processing times are known and finite.
- Each job has only one operation and can be processed on any machine.
- Machines have different speeds.

The constraints which must be respected are:

- Each machine can process only one job at a time.
- Each job is to be processed without interruption.
- All jobs have to be processed to complete the production.

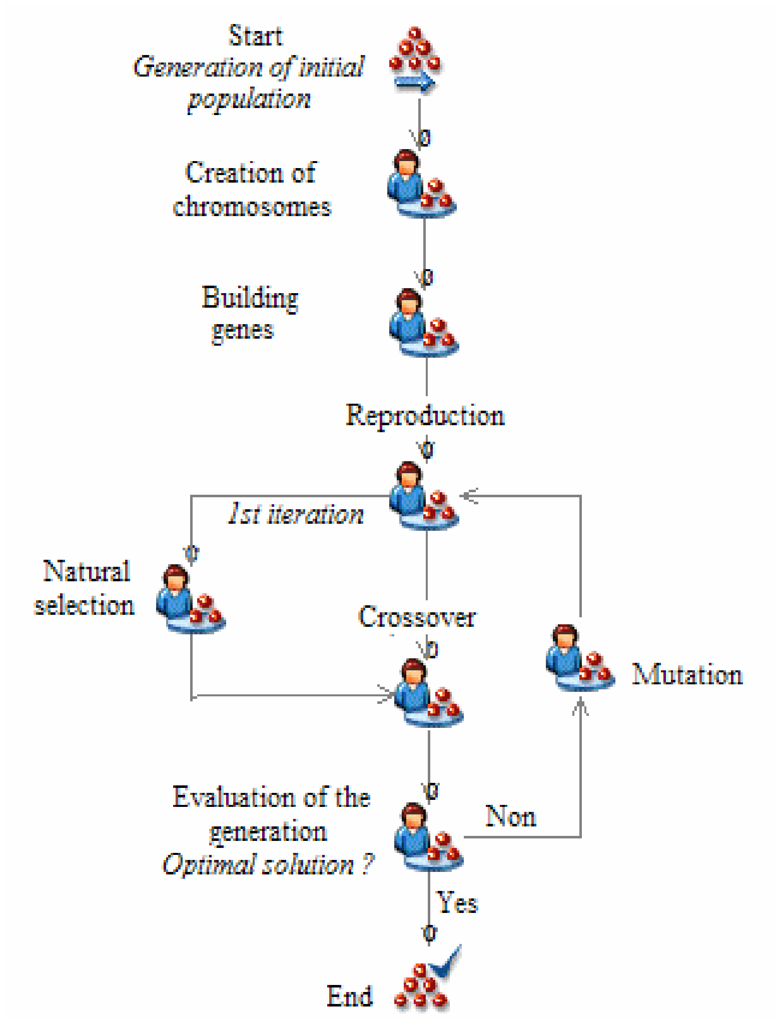

FIGURE 4. Application of GA using simulation

The speed of the first machine is considered as $V_1 = V$. Processing times of each job processed at speed V is noted by P_j (Table 1). Machines have different speeds but, as the speed of each machine can be expressed in term of V, it is no need to present processing times of jobs on each machine in separate tables. In this numerical example, the speeds of machines are considered as follows: $V_1 = V$, $V_2 = 2V$, $V_3 = 2V$, $V_4 = 3V$.

The proposed GA deals with fuzzy numbers. In order to execute the same algorithm with crisp processing times, they have to be expressed as fuzzy numbers (Table 1). For this purpose, it is sufficient to consider TFNs as:

$$A = (p, q, r) \text{ where } p = q = r$$

TABLE 1. Crisp processing times

Processing times (min.)		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
	P	13	19	24	22	28	17	19	24	27
$\boldsymbol{P}_j$	Q	13	19	24	22	28	17	19	24	27
	r	13	19	24	22	28	17	19	24	27

The problem is solved by using the proposed GA approach and the results are given in Table 2. An ideal solution can be defined by supposing that jobs can be interrupted at anytime and it can be calculated as:

$$C_{\max}ideal = \frac{\sum\limits_{j=1}^{n} \delta(P_j)}{\sum\limits_{i=1}^{m} V_i}$$

The ideal solution is found 24.13 minutes. However, as one of the constraint is violated, this solution is not feasible. It only helps to decide on the optimal solution and to compare solutions to the ideal one. According to the ideal solution, the optimal solution must be obviously greater than 24.13 minutes. $C_{\max} = 24.50$ minutes can be accepted as optimal solution. It is obtained 11 times during the simulation (Table 2). However, some of them are provided by the same schedule. In fact, there are 6 schedules giving the same optimal solution. They are presented in Table 3.

If the same problem is formulated as a PSMP and solved by conventional *GA*, the same results are obtained, which confirms the validity of the proposed approach. Another finding of this application is the demonstration of PMSP being a sub-set of FPMSP.

5.2. **Solving FPSMP.** The FPMSP defined in Table 4 is solved by using the proposed *GA* approach, which is the main purpose of the study. The ideal solution is 25.46 minutes. Thus, it can be concluded that the minimum solution found by the approach (Table 5), $C_{\max} = 25.83$ minutes, is optimal. It is obtained 7 times in one simulation run and they are provided by 3 different schedules.

Optimal schedules are given in Table 6. As completion times are TFNs, they are compared by using the "signed distance" method presented in Subsection 2.2. It is found that $C_{\max} = C_1 = (17.50, 24.50, 35.50)$ and using the "centroid" defuzzification method presented in Subsection 2.3, the optimal result is obtained as $\delta(C_{\max}) = 25.83$ minutes.

It should be noted that the machines are loaded differently according to schedules given in Table 6, which illustrates the ability of the proposed *GA* approach to explore alternative optimal schedules.

5.3. **Comparison with LPT.** The same problem defined in Subsection 5.2 is solved by using *LPT* rule, known as the most appropriate dispatching rule for PMSPs. Application of *LPT* rule to FPMSPs is described in [46]. We adapted their method for non-identical parallel machines and obtained following results given below in Table 7.

Application of the "signed distance" ranking method yields $C_{\max} = C_1 = (20.00, 29.50, 41.00)$ and application of "centroid" defuzzification method gives $\delta(C_{\max}) = 30.17$ minutes. Results reveal that the proposed *GA* approach surpasses *LPT* rule and illustrates the need for efficient and effective algorithms for FPMSPs.

6. Result Analysis.

6.1. **Results of GA.** Results obtained by the solution of the FPMSP (Subsection 5.2) show that the proposed GA approach yields good results and reaches them fast. Examination of Table 5 reveals that the optimal solution is reached quite fast only at 9th iteration. In addition, the optimal solution is reached several times in one run. The same Table 5 shows that optimal solution is reached 7 times in 100 iterations, which proves the efficiency of the algorithm. The same success is repeated during the robustness tests performed with 10 different random number sets (Subsection 6.2).

The effectiveness of the algorithm can be evaluated by comparing the result of the GA approach (25.83 minutes, Table 5) to the ideal but infeasible solution of the problem (25.46 minutes, Subsection 5.2), or to the results provided by the *LPT* rule (30.17 minutes,

TABLE 2. Results with crisp processing times

Iteration No.		$C_{\max}$		$\delta(C_{\max})$	***Iteration No.***		$C_{\max}$		$\delta(C_{\max})$
	p	***q***	***r***			***p***	***q***	***r***	
1	25,00	25,00	25,00	25,00	51	27,00	27,00	27,00	27,00
2	24,67	24,67	24,67	24,67	52	24,67	24,67	24,67	24,67
3	26,00	26,00	26,00	26,00	53	24,50	24,50	24,50	**24,50**
4	27,00	27,00	27,00	27,00	54	25,50	25,50	25,50	25,50
5	25,50	25,50	25,50	25,50	55	25,50	25,50	25,50	25,50
6	25,00	25,00	25,00	25,00	56	28,00	28,00	28,00	28,00
7	24,50	24,50	24,50	**24,50**	57	25,00	25,00	25,00	25,00
8	25,50	25,50	25,50	25,50	58	25,00	25,00	25,00	25,00
9	27,00	27,00	27,00	27,00	59	25,00	25,00	25,00	25,00
10	26,00	26,00	26,00	26,00	60	25,33	25,33	25,33	25,33
11	27,00	27,00	27,00	27,00	61	25,00	25,00	25,00	25,00
12	26,00	26,00	26,00	26,00	62	24,50	24,50	24,50	**24,50**
13	24,50	24,50	24,50	**24,50**	63	25,50	25,50	25,50	25,50
14	25,50	25,50	25,50	25,50	64	25,50	25,50	25,50	25,50
15	25,67	25,67	25,67	25,67	65	25,50	25,50	25,50	25,50
16	27,00	27,00	27,00	27,00	66	25,50	25,50	25,50	25,50
17	26,00	26,00	26,00	26,00	67	27,00	27,00	27,00	27,00
18	25,33	25,33	25,33	25,33	68	24,50	24,50	24,50	**24,50**
19	24,50	24,50	24,50	**24,50**	69	26,00	26,00	26,00	26,00
20	25,50	25,50	25,50	25,50	70	26,00	26,00	26,00	26,00
21	26,00	26,00	26,00	26,00	71	27,00	27,00	27,00	27,00
22	27,00	27,00	27,00	27,00	72	25,50	25,50	25,50	25,50
23	27,00	27,00	27,00	27,00	73	27,00	27,00	27,00	27,00
24	27,00	27,00	27,00	27,00	74	27,00	27,00	27,00	27,00
25	24,50	24,50	24,50	**24,50**	75	25,67	25,67	25,67	25,67
26	25,33	25,33	25,33	25,33	76	27,00	27,00	27,00	27,00
27	26,00	26,00	26,00	26,00	77	25,33	25,33	25,33	25,33
28	25,50	25,50	25,50	25,50	78	27,50	27,50	27,50	27,50
29	25,50	25,50	25,50	25,50	79	27,00	27,00	27,00	27,00
30	25,50	25,50	25,50	25,50	80	26,00	26,00	26,00	26,00
31	25,67	25,67	25,67	25,67	81	26,00	26,00	26,00	26,00
32	25,00	25,00	25,00	25,00	82	27,50	27,50	27,50	27,50
33	27,00	27,00	27,00	27,00	83	27,00	27,00	27,00	27,00
34	25,00	25,00	25,00	25,00	84	24,50	24,50	24,50	**24,50**
35	25,67	25,67	25,67	25,67	85	27,00	27,00	27,00	27,00
36	26,00	26,00	26,00	26,00	86	26,00	26,00	26,00	26,00
37	25,50	25,50	25,50	25,50	87	26,00	26,00	26,00	26,00
38	27,00	27,00	27,00	27,00	88	25,50	25,50	25,50	25,50
39	26,00	26,00	26,00	26,00	89	25,50	25,50	25,50	25,50
40	25,50	25,50	25,50	25,50	90	26,33	26,33	26,33	26,33
41	24,50	24,50	24,50	**24,50**	91	27,00	27,00	27,00	27,00
42	25,50	25,50	25,50	25,50	92	26,00	26,00	26,00	26,00
43	26,00	26,00	26,00	26,00	93	27,50	27,50	27,50	27,50
44	27,00	27,00	27,00	27,00	94	25,50	25,50	25,50	25,50
45	24,50	24,50	24,50	**24,50**	95	25,00	25,00	25,00	25,00
46	25,50	25,50	25,50	25,50	96	25,33	25,33	25,33	25,33
47	25,50	25,50	25,50	25,50	97	27,50	27,50	27,50	27,50
48	25,33	25,33	25,33	25,33	98	24,50	24,50	24,50	**24,50**
49	25,00	25,00	25,00	25,00	99	27,00	27,00	27,00	27,00
50	26,00	26,00	26,00	26,00	100	26,00	26,00	26,00	26,00

Table 3. Optimal schedules with crisp processing times

Iterations No.7 & No.19		
Machines	***Scheduled jobs***	C_i
M.1	job 8	24,00
M.2	job 1 job 7 job 6	24,50
M.3	job 9 job 4	24,50
M.4	job 5 job 2 job 3	23,67

Iterations No.41 & No.68		
Machines	***Scheduled jobs***	C_i
M.1	job 3	24,00
M.2	job 4 job 9	24,50
M.3	job 1 job 6 job 2	24,50
M.4	job 5 job 8 job 7	23,67

Iterations No.13		
Machines	***Scheduled jobs***	C_i
M.1	job 3	24,00
M.2	job 4 job 9	24,50
M.3	job 5 job 2	23,50
M.4	job 8 job 6 job 1 job 7	24,33

Iterations No.45 & No.62 & No.98		
Machines	***Scheduled jobs***	C_i
M.1	job 3	24,00
M.2	job 5 job 2	23,50
M.3	job 7 job 1 job 6	24,50
M.4	job 9 job 4 job 8	24,33

Iterations No.25 & No.53		
Machines	***Scheduled jobs***	C_i
M.1	job 3	24,00
M.2	job 5 job 7	23,50
M.3	job 9 job 4	24,50
M.4	job 6 job 1 job 8 job 2	24,33

Iterations No.84		
Machines	***Scheduled jobs***	C_i
M.1	job 3	24,00
M.2	job 5 job 7	23,50
M.3	job 2 job 1 job 6	24,50
M.4	job 9 job 4 job 8	24,33

Table 4. Fuzzy processing times

Processing times (min.)		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
	p	10	12	22	16	20	13	12	21	18
P_j	q	13	19	24	22	28	17	19	24	27
	r	22	24	38	25	30	25	44	30	36

Table 7). The optimal result of the algorithm is only 1.5% far from the ideal solution and 16.8% better then the result of the *LPT* rule, known as the best dispatching rule for the minimization of the maximum completion time of parallel machines. Moreover, even the worst result (28.00 minutes, Table 5) of the algorithm is much more better (7.8%) than the result obtained by the *LPT* rule. Another conclusion provided by the examination of these results is that non-optimal results are not too "bad" either. The average result of 100 iterations is 26.59 minutes (Table 5), which is only 2.94% far from the optimal solution. These findings shows the effectiveness of the proposed approach.

Another benefit of this approach, thanks to its advantage of being a search algorithm, unlike other methods proposed for such scheduling problems as *Linear Programming*, *Branch and Bound*, *LPT*, the proposed approach can explore alternative schedules providing the same results. Three alternative schedules, both optimal, are proposed in Table 6.

6.2. **Testing the robustness.** Roy [47] defines the term "robust" as an adjective referring to a capacity for withstanding "vague approximations" and/or "zones of ignorance" in order to prevent undesirable impacts, notably the degradation of the properties to be maintained. Considering the definition of Roy [47], a "robust heuristic" can be defined as an algorithm providing consistent results (i.e., being resistant) in multiple runs performed using different random numbers that correspond to the uncertain parameters of the algorithm and form a "zone of ignorance".

Table 5. Results with fuzzy processing times

Iteration		$C_{\max}$			***Iteration***		$C_{\max}$		
No.	p	q	r	$\delta(C_{\max})$	***No.***	p	q	r	$\delta(C_{\max})$
1	20,00	28,00	30,00	26,00	51	19,50	26,00	36,00	27,17
2	17,67	25,33	35,67	26,22	52	20,00	28,00	30,00	26,00
3	21,00	26,00	34,00	27,00	53	19,00	27,50	33,00	26,50
4	18,33	25,67	33,67	25,89	54	20,00	28,00	30,00	26,00
5	18,00	27,00	36,00	27,00	55	18,00	27,00	36,00	27,00
6	20,00	28,00	30,00	26,00	56	17,50	24,50	35,50	**25,83**
7	20,00	25,50	37,00	27,50	57	20,00	28,00	30,00	26,00
8	21,00	26,00	34,00	27,00	58	20,00	28,00	30,00	26,00
9	17,50	24,50	35,50	**25,83**	59	19,00	26,33	36,00	27,11
10	20,00	28,00	30,00	26,00	60	22,00	24,00	38,00	28,00
11	19,00	27,50	33,00	26,50	61	19,00	27,50	33,00	26,50
12	19,00	26,33	36,00	27,11	62	20,00	25,50	37,00	27,50
13	19,00	27,00	35,50	27,17	63	20,00	28,00	30,00	26,00
14	19,00	24,33	36,33	26,56	64	21,00	26,00	34,00	27,00
15	21,00	25,33	32,67	26,33	65	18,00	27,00	36,00	27,00
16	19,00	27,50	33,00	26,50	66	17,33	23,33	39,33	26,67
17	20,00	25,50	37,00	27,50	67	19,00	27,50	33,00	26,50
18	19,50	26,00	36,00	27,17	68	19,00	27,50	33,00	26,50
19	19,00	24,33	36,33	26,56	69	17,50	24,50	35,50	**25,83**
20	19,00	27,50	33,00	26,50	70	21,00	26,00	34,00	27,00
21	20,00	28,00	30,00	26,00	71	19,50	25,50	33,00	26,00
22	21,00	26,00	34,00	27,00	72	18,00	27,00	36,00	27,00
23	20,00	28,00	30,00	26,00	73	20,00	28,00	30,00	26,00
24	21,00	26,00	34,00	27,00	74	17,50	24,50	35,50	**25,83**
25	21,00	26,00	34,00	27,00	75	19,33	27,33	33,67	26,78
26	20,00	28,00	30,00	26,00	76	20,00	25,50	37,00	27,50
27	19,00	27,50	33,00	26,50	77	20,00	25,50	37,00	27,50
28	20,00	28,00	30,00	26,00	78	21,00	26,00	34,00	27,00
29	19,50	26,00	36,00	27,17	79	19,00	24,33	36,33	26,56
30	17,50	24,50	35,50	**25,83**	80	19,00	27,50	33,00	26,50
31	21,00	25,33	32,67	26,33	81	18,67	24,33	40,33	27,78
32	19,50	26,00	36,00	27,17	82	20,00	28,00	30,00	26,00
33	22,00	24,00	38,00	28,00	83	19,00	27,50	33,00	26,50
34	20,00	28,00	30,00	26,00	84	19,00	27,50	33,00	26,50
35	20,00	28,00	30,00	26,00	85	19,00	27,50	33,00	26,50
36	18,00	27,00	36,00	27,00	86	17,50	24,50	35,50	**25,83**
37	20,00	28,00	30,00	26,00	87	19,00	24,33	36,33	26,56
38	17,67	25,33	35,67	26,22	88	20,00	28,00	30,00	26,00
39	20,00	28,00	30,00	26,00	89	20,00	28,00	30,00	26,00
40	22,00	24,00	38,00	28,00	90	20,00	28,00	30,00	26,00
41	18,00	27,00	36,00	27,00	91	17,67	25,33	35,67	26,22
42	20,00	28,00	30,00	26,00	92	20,33	25,33	36,67	27,44
43	18,00	27,00	36,00	27,00	93	20,00	28,00	30,00	26,00
44	20,00	28,00	30,00	26,00	94	17,67	25,33	35,67	26,22
45	20,00	28,00	30,00	26,00	95	22,00	24,00	38,00	28,00
46	18,00	27,00	36,00	27,00	96	20,00	28,00	30,00	26,00
47	17,50	24,50	35,50	**25,83**	97	21,00	26,00	34,00	27,00
48	20,00	28,00	30,00	26,00	98	20,00	25,50	37,00	27,50
49	18,00	27,00	36,00	27,00	99	19,00	27,50	33,00	26,50
50	22,00	24,00	38,00	28,00	100	20,00	28,00	30,00	26,00

TABLE 6. Optimal schedules with fuzzy processing times

Iterations No.9 & No.56						
Machines	***Scheduled jobs***			***C_i***		
M.1	job 8			21,00	24,00	30,00
M.2	job 5	job 7		16,00	23,50	37,00
M.3	job 2	job 1	job 6	**17,50**	**24,50**	**35,50**
M.4	job 9	job 4	job 3	18,67	24,33	33,00

Iterations No.30 & No.86						
Machines	***Scheduled jobs***			***C_i***		
M.1	job 8			21,00	24,00	30,00
M.2	job 3	job 7		17,00	21,50	41,00
M.3	job 6	job 1	job 2	**17,50**	**24,50**	**35,50**
M.4	job 9	job 5	job 4	18,00	25,67	30,33

Iterations No.47 & No.69 & No.74						
Machines	***Scheduled jobs***			***C_i***		
M.1	job 8			21,00	24,00	30,00
M.2	job 1	job 6	job 2	**17,50**	**24,50**	**35,50**
M.3	job 7	job 9		15,00	23,00	40,00
M.4	job 4	job 3	job 5	19,33	24,67	31,00

TABLE 7. Results of *LPT*

Machines	***Scheduled jobs***			***C_i***		
M.1	Job 8			21,00	24,00	30,00
M.2	Job 5	job 4		18,00	25,00	27,50
M.3	job 9	job 2	job 1	**20,00**	**29,50**	**41,00**
M.4	job 3	job 7	job 6	15,67	20,00	35,67

Robustness test should not be performed by causing changes in processing times which are the part of the input of the problem. Changing the processing times would result in the modification of the problem. This analysis is rather called "sensitivity analysis". In this study, as the problem considered is just a numerical example, there is no need to perform a sensitivity analysis. Besides, sensitivity analysis measures how the solution (here the optimal schedule) responses to the changes in the inputs.

10 different tests using 10 different random number sets are performed. The summary of the results are presented in Table 8. The second column of the table shows how many time the optimal solution ($\delta(C_{\max}) = 25.83$ minutes) is reached and the third column shows the average results of the 100 iterations in one test.

According the results, the proposed *GA* approach proves its efficiency and effectiveness in solving FPMSPs. Optimal solutions are again reached fast and several times (6.6 times in average) in one run. Non-optimal schedules are not "bad" either, the average of all solutions found in 100 iterations of 10 tests is 26.71 minutes (Table 8). Besides, all the alternative optimal schedules have been explored.

7. **Conclusion.**

The parallel machine scheduling problem receives a considerable attention in both academic and industrial field. Various factors involved in the scheduling problems are often imprecise or uncertain. The fuzzy set theory provides an alternative

Table 8. Summary table of test results

Test No	***Number of Optimal Solution Reached***	***Average*** $\boldsymbol{\delta(C_{\max})}$
1	6 times	26.77 minutes
2	6 times	26.67 minutes
3	8 times	26.69 minutes
4	7 times	26.74 minutes
5	7 times	26.73 minutes
6	6 times	26.70 minutes
7	5 times	26.71 minutes
8	7 times	26.64 minutes
9	6 times	26.76 minutes
10	8 times	26.69 minutes
Average	*6.6 times*	*26.71 minutes*

and convenient framework for modeling real-world systems mathematically and offers several advantages in the use of heuristic approaches. In this study, non-identical parallel machine scheduling problem with fuzzy processing times (FPMSP) is considered and a *GA* approach is proposed to minimize maximum completion time (makespan).

New *GA* operators and two of well-known ranking and defuzzificaton methods are used in order to adapt the general structure of *GA* to the considered problem. The *GA* is embedded in a simulation model for solving the problem. The use of simulation in implementing *GA* is preferred because of the evolutionary structure of the algorithm and the ability of simulation to perform several tests using different random number sets. By this means, the robustness of the proposed approach can be easily tested in a series of numerical experiments.

The proposed approach is first tested for scheduling non-identical parallel machines with crisp processing times (Subsection 5.1). Results were identical to those obtained by using conventional *GA* for PMSPs. This exercise served to validate the proposed *GA* approach and proved that PMSPs form a sub-set of FPMSPs. Then, a numerical example of FPMSP is solved by using the proposed *GA* approach (Subsection 5.2) and results are compared to those obtained by *LPT* rule (Subsection 5.3), known as the most appropriate dispatching rule for such problems. Results showed that the proposed approach surpasses *LPT* rule and illustrates the need of using efficient and effective heuristics for FPMSPs.

The discussion of numerical results leads to the following conclusions (Subsection 6.1):

- The proposed approach is quite efficient since it can find optimal results fast and several times in one run.
- Non-optimal results are not too "bad" either which shows the effectiveness of the approach. The worst result obtained by the proposed approach is better than the result of *LPT* rule.
- Thanks to its advantage of being a search algorithm, other alternative schedules giving the same optimal result can be explored.

The robustness of the approach is tested by running the simulation using 10 different random number sets (Subsection 6.2). Results revealed that the proposed approach is robust enough to provide same results efficiently and effectively in different circumstances.

The motivation of the author is to contribute to the literature of scheduling parallel machines under uncertainty, which can help to consider more realistic production scheduling problems in industrial field. This problem is not addressed before for non-identical parallel machines with fuzzy processing times. This study highlights the advantage of using fuzzy set theory for modeling such problems and emphasizes the need for efficient and effective heuristics to solve them. Accordingly, in this study, a robust *GA* approach is proposed to solve parallel machine scheduling problem with fuzzy processing times.

REFERENCES

[1] F. Glover, Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research*, vol.13, no.5, pp.533-549, 1986.

[2] S. Gürsel and E. Baez, Minimizing the number of tardy jobs in identical machine scheduling, *Computers & Industrial Engineering*, vol.25, no.4, pp.243-246, 1993.

[3] L. Min and W. Cheng, Scheduling algorithm based on evolutionary computing in identical parallel machine production line, *Robotics and Computer-Integrated Manufacturing*, vol.19, pp.401-407, 2003.

[4] R. Zhang and C. Wu, Bottleneck machine identification based on optimization for the job shop scheduling problem, *ICIC Express Letters*, vol.2, no.2, pp.175-180, 2008.

[5] R. Masuchun, W. Masuchun and T. Thepmanee, Integrating production scheduling and material requirements planning, *ICIC Express Letters*, vol.3, no.3(A), pp.501-506, 2009.

[6] A. H. Gharehgozli, R. Tavakkoli-Moghaddam and N. Zaerpour, A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates, *Robotics and Computer-Integrated Manufacturing*, vol.25, pp.853-859, 2009.

[7] J. Peng and B. Liu, Parallel machine scheduling models with fuzzy processing times, *Information Sciences-Informatics and Computer Science: An International Journal*, vol.166, pp.49-66, 2004.

[8] P. Y. Mok, C. K. Kwong and W. K. Wong, Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory, *European Journal of Operations Research*, vol.177, pp.1876-1893, 2007.

[9] L. A. Zadeh, Fuzzy set as a basis for a theory of possibility, *Fuzzy Sets and Systems*, vol.1, pp.3-28, 1978.

[10] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York, 1988.

[11] A. Anglani, A. Grieco, E. Guerriero and R. Musmanno, Robust scheduling of parallel machines with sequence-dependent set-upcosts, *European Journal of Operational Research*, vol.161, pp.704-720, 2005.

[12] J. Balasubramanian and I. E. Grossmann, Scheduling optimization under uncertainty – An alternative approach, *Computers and Chemical Engineering*, vol.27, pp.469-490, 2003.

[13] R. Slowinski and M. Hapke, *Scheduling under Fuzziness*, Physica-Verlag, New York, USA, 2000.

[14] A. H. Kashan, B. Karimi and M. Jenabi, A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes, *Computers & Operations Research*, vol.35, pp.1084-1098, 2008.

[15] L. Y. Tseng, Y. H. Chin and S. C. Wang, A deadline-based task scheduling with minimized makespan, *International Journal of Innovative Computing, Information and Control*, vol.5, no.6, pp.1665-1680, 2009.

[16] A. H. G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity and Computations*, Martinus Nijhoff, The Hague, 1976.

[17] R. Sethi, On the complexity of mean flow time scheduling, *Mathematics of Operations Research*, vol.2, no.4, pp.320-330, 1997.

[18] J. K. Lenstra and A. H. G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operational Research*, vol.26, pp.22-35, 1978.

[19] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-completenes*, W. H. Freeman, San Francisco, CA, 1979.

[20] L. Min and W. Cheng, A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines, *Artificial Intelligence in Engineering*, vol.13, pp.399-403, 1999.

[21] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, Springer-Verlag, New York, USA, 1992.

[22] B. Liu and K. Iwamura, Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems*, vol.94, pp.227-237, 1998.

[23] B. Liu and K. Iwamura, A note on chance constrained programming with fuzzy coefficients, *Fuzzy Sets and Systems*, vol.100, pp.229-233, 1998.

[24] B. Liu, *Uncertain Programming*, John Wiley & Sons, New York, USA, 1999.

[25] B. Liu, Dependent-chance programming with fuzzy decisions, *IEEE Transactions on Fuzzy Systems*, vol.7, pp.354-360, 1999.

[26] Y. Li, Y. Yang, L. Zhou and R. Zhu, Observations on using problem-specific genetic algorithm for multiprocessor real-time task scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.9, pp.2531-2540, 2009.

[27] J. J. Buckley and Y. Hayashi, Fuzzy genetic algorithm and applications, *Fuzzy Sets and Systems*, vol.61, pp.129-136, 1994.

[28] J. J. Buckley and T. Feuring, Evolutionary algorithm solution to fuzzy problems: Fuzzy linear programming, *Fuzzy Sets and Systems*, vol.109, pp.35-53, 2000.

[29] B. Liu, *Theory and Practice of Uncertain Programming*, Physica-Verlag, Heidelberg, 2002.

[30] N. Van Hop and N. N. Nagarur, The scheduling problem of PCBs for multiple non-identical parallel machines, *European Journal of Operational Research*, vol.158, pp.577-594, 2004.

[31] A. Özalp, A genetic algorithm for scheduling of jobs on lines of press machines, *Lecture Notes in Computer Science*, vol.3743, pp.535-543, 2006.

[32] T. Çakar, R. Köker and H. I. Demir, Parallel robot scheduling to minimize mean tardiness with precedence constraints using a genetic algorithm, *Advances in Engineering Software*, vol.39, pp.47-54, 2008.

[33] D. Petrovic and A. Duenas, A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions, *Fuzzy Sets and Systems*, vol.157, pp.2273-2285, 2006.

[34] K. Raja, C. Arumugam and V. Selladurai, Non-identical parallel-machine scheduling using genetic algorithm and fuzzy logic approach, *International Journal of Services and Operations Management*, vol.4, no.1, pp.333-346, 2008.

[35] B. Roy and P. Vincke, Relational systems of preference with one or more pseudo-criteria: Some new concepts and results, *Management Science*, vol.30, pp.1323-1335, 1984.

[36] L. V. Vanegas and A. W. Labib, Application of new fuzzyweighted average method to engineering design evaluation, *International Journal of Production Research*, vol.39, no.6, pp.1147-1162, 2001.

[37] J. S. Yao and K. Wu, Ranking fuzzy numbers based on decomposition principle and signed distance, *Fuzzy Sets and Systems*, vol.116, pp.275-288, 2000.

[38] N. Van Hop, A heuristic solution for fuzzy mixed-model line balancing problem, *European Journal of Operational Research*, vol.168, pp.798-810, 2006.

[39] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[40] L. Davis and S. Coombs, Genetic algorithms and communication link speed design: Theoretical considerations, *Proc. of the 2nd International Conference on Genetic Algorithms and Their Application*, pp.252-256, 1987.

[41] B. J. Park, H. R. Choi and H. S. Kim, A hybrid genetic algorithm for the job shop scheduling problems, *Computers & Industrial Engineering*, vol.45, pp.597-613, 2003.

[42] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, MA, 1989.

[43] P. Ji, M. T. Sze and W. B. Lee, A genetic algorithm of determining cycle time for printed circuit board assembly lines, *European Journal of Operations Research*, vol.128, pp.175-184, 2001.

[44] H. Zhou, Y. Feng and L. Han, The hybrid heuristic genetic algorithm for job shop scheduling, *Computers & Industrial Engineering*, vol.40, no.3, pp.191-200, 2001.

[45] C. Jou, A genetic algorithm with sub-indexed partitioning genes and its application to production scheduling of parallel machines, *Computers & Industrial Engineering*, vol.48, pp.39-54, 2005.

[46] T. P. Hong, C. M. Huang and K. M. Yu, LPT scheduling for fuzzy tasks, *Fuzzy Sets and Systems*, vol.97, pp.277-286, 1988.

[47] B. Roy, Robustness in operational research and decision aiding: A multi-faceted issue, *European Journal of Operational Research*, vol.200, pp.629-638, 2010.