Information Sciences 181 (2011) 3551–3569

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation

Savaş Balin *

Department of Industrial Engineering, Yıldız Technical University, İstanbul, Turkey

ARTICLE INFO

Article history:
Received 29 July 2010
Received in revised form 5 February 2011
Accepted 7 April 2011
Available online 19 April 2011

Keywords:
Fuzzy parallel machine scheduling problem (FPMSP)
Fuzzy processing times
Genetic algorithm
Robustness
Simulation

ABSTRACT

This paper addresses parallel machine scheduling problems with fuzzy processing times. A robust genetic algorithm (*GA*) approach embedded in a simulation model is proposed to minimize the maximum completion time (makespan). The results are compared with those obtained by using the "longest processing time" rule (*LPT*), which is known as the most appropriate dispatching rule for such problems. This application illustrates the need for efficient and effective heuristics to solve such fuzzy parallel machine scheduling problems (FPMSPs). The proposed *GA* approach yields good results quickly and several times in one run. Moreover, because it is 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 the robustness of the proposed approach is demonstrated.

1. Introduction

Production scheduling consists of planning jobs that need to be performed in an orderly sequence of operations. It is a difficult-to-solve problem, and the number of calculations required to obtain an optimal schedule depends on the chosen criterion. In modern manufacturing, which depends on machine layout and job flow, there are several kinds of shops. As many practical job shop and open shop scheduling problems can be simplified as parallel machine scheduling problem under certain conditions [10,12,29], the parallel machine scheduling problem (PMSP) has received a great deal of attention in the academic and engineering field [21,29]. Moreover, it is a generalization of the single machine problem and of a particular case of problems arising in flexible manufacturing systems [9,19]. In many manufacturing systems, jobs go through several stages, in each of which there are several machines working in parallel. Minimizing the maximum completion time results in shortening the total production time. It helps to minimize the bottlenecks in the production, to improve the throughput rate and to deliver products faster to the customer. In the literature, this problem is known as parallel machine scheduling, but in a larger sense it can be considered as the general problem of scheduling parallel processes. It can be used not only in manufacturing, but also in the service sector (e.g., parallel service providers, transporters, call centers, banking operations...) or in information systems (e.g., parallel information processing, distributed computing...).

In addition, there are various factors involved in real-world scheduling problems that are often imprecise or uncertain in nature. This is especially true when human-made factors are considered into the problems. Therefore, parameters are often encountered with uncertainties. Accordingly, production scheduling problems can be divided into two general categories: deterministic scheduling and uncertain scheduling problems [33]. There are essentially two approaches to deal with uncertainties [30], including the stochastic-probabilistic theory and possibility theory or fuzzy set theory [45,7].

* Tel.: +90 212 3832897; fax: +90 212 2334286.
E-mail address: sbalin@yildiz.edu.tr

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.04.010

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

- The stochastic-probabilistic theory requires significant knowledge about the statistical distribution of the uncertain parameters. In contrast, fuzzy theory provides an efficient way to model imprecision even when no historical information is available [1].
- The use of stochastic-probabilistic theory involves extensive computation and requires complete knowledge on the statistical distribution of the uncertain time parameters [2].
- Using fuzzy set theory decreases the computational complexity of the scheduling problem compared to the stochastic-probabilistic theory [40].
- Fuzzy theory enables the use of fuzzy rules in heuristic algorithms.
- Instead of optimizing the average behaviors, as in stochastic-probabilistic theory, fuzzy theory rather aims to find 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 [30]. 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 parallel machines in order to minimize the makespan. In scheduling problems, the makespan (C_{max}) is equivalent to the completion time of the last job leaving the system. A small value of C_{max} usually implies a high utilization. Therefore, reducing the value of C_{max} should also lead to a higher throughput rate [18,43].

Most of scheduling problems are characterized as combinatorial optimization problems. They vary widely according to the specific production tasks, but most are *NP-hard* problems [36,39,20,8]. Obviously, they become more difficult to solve when 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 large scale problems, their application is limited. Dispatching rules (e.g., longest processing time (*LPT*), shortest processing time (*SPT*), earliest due date (*EDD*),...) are suitable only for small scale problems and no single dispatching rule guarantees a good result in various problems [28]. Most 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* Out of these various approaches to different scheduling problems, *GA* has attracted increasing interest in view of their characteristics such as high adaptability, near optimization and easy realization [31]. *GA* has demonstrated considerable success in providing good solutions to fuzzy scheduling problems. They have been well documented in the literature, such as in Michalewicz [27], and have been applied to a wide variety of optimization problems [5]. In particular, Liu and Iwamura [25,26], Liu [22,23], Buckley and Hayashi [3] and Buckley and Feuring [4] designed a spectrum of *GA* to solve fuzzy programming models. For detailed expositions, the readers may consult Liu [22,24], in which numerous *GA*s have been suggested for solving uncertain programming models [33].

GA has been applied with the same success in solving PMSPs in the literature and there are many applications in this field. However, despite its successes in each of the fields individually, there are only few recent studies on *GA* dealing simultaneously with fuzzy scheduling and parallel machines (FPMSPs) [33]. One of them is the study of Peng and Liu [33] who considered a multi-objective FPMSP and formulate it in terms of three-objective models, which minimize the fuzzy maximum tardiness, the fuzzy maximum completion time and the fuzzy maximum idleness. They proposed three models to formulate fuzzy scheduling problem: fuzzy goal programming, fuzzy chance-constrained programming, and fuzzy dependent-chance programming. They proposed a hybrid intelligent algorithm designed to solve the proposed scheduling models. Petrovic and Duenas [34] presented a new fuzzy logic based decision support system for parallel machine scheduling/rescheduling in the presence of uncertain disruptions. They used fuzzy rules to determine when to reschedule and which rescheduling method to use. Raja et al. [35] aimed to schedule non-identical parallel machines using *GA*. Their objective was to minimize the sum of the absolute deviations of job completion times. Distinctively, they did not face uncertainty using fuzzy sets, but rather used fuzzy logic to combine two objective functions to reduce earliness and tardiness penalties. Mok et al. [30] tried to optimize fault-tolerant fabric-cutting schedules using *GA* by minimizing the 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. Their study is similar to that presented here but differs in terms of the genetic operators, ranking and defuzzification methods used. 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, that of scheduling 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, the 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 the practical and

theoretical points of view. The aim of this study is to fill this gap by proposing a *GA* for minimizing the maximum completion time of parallel machines where processing times are fuzzy.

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

A numerical example of a FPMSP is solved first by using the proposed *GA* approach and then by the *LPT* rule, which is known as the most appropriate dispatching rule for such problems. The results show that the proposed approach surpasses the *LPT* rule and highlight the need to use 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 provided in Section 2. The formulation of the FPMSP to minimize the makespan is given in Section 3. The general structure of the *GA* is adapted to the FPMSP in Section 4. The *GA* is embedded in a simulation model because of its computational advantages in running tests. Numerical examples are solved in Section 5. In the first paragraph (Section 5.1), the case study is described; in the second paragraph (Section 5.2), the proposed *GA* is tested with real numbers; in the third paragraph (Section 5.3), processing times are considered as fuzzy numbers, and in the fourth paragraph (Section 5.4), these results are compared to those obtained with the *LPT* rule. In Section 6, the numerical results are discussed and the robustness of the approach is tested. The paper ends in Section 6 with a brief summary of the main findings.

2. Review of related fuzzy theory

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

2.1. Fuzzy numbers

The 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 an example application because of their computational advantages. They are also commonly used for subjective description, as they are based on knowledge of the minimum, maximum and an "inspired guess" as to the modal value. However, note that the proposed fuzzy ranking and defuzzification methods can be similarly applied for all fuzzy numbers of any 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:

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

A TFN can also be denoted by its membership function:

$$\mu_A(x) = \begin{cases} \frac{x-p}{q-p}, & \text{if } p \leqslant x \leqslant q, \\ \frac{x-r}{q-r}, & \text{if } q \leqslant \ \mathrm{x} \leqslant \mathrm{r}, \\ 0, & \text{elsewhere}. \end{cases}$$

Therefore, a TFN can be represented graphically as below (Fig. 1):

2.2. Ranking method

Ranking two numbers is quite important in a scheduling algorithm. Ranking is quite easy when the operands are all real numbers. However, in this paper, because fuzzy processing times are considered, fuzzy ranking methods have to be used. In the literature, several fuzzy comparison methods have been proposed, such as the Hamming distance method, the probability distribution method, the pseudo-order fuzzy preference model [37], the new fuzzy-weighted average [42], the signed distance method [44] and so on. Among these methods, the "signed distance" method is most suitable for fuzzy processing time comparison because it is simple in computation and flexible to convert from interval data [41].

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

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

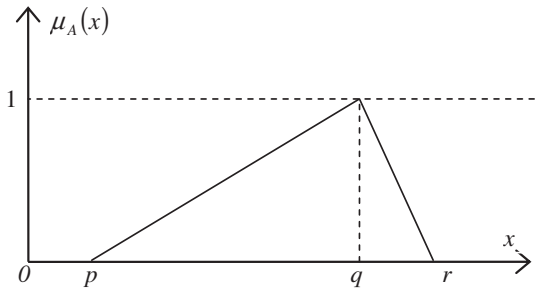

Fig. 1. Triangular fuzzy number (TFN).

TFNs can be ranked by signed distance method as follows [44]:

$$B \prec A \text{ if } d(B) < d(A),$$
$$B = A \text{ if } d(B) = d(A),$$
$$B \succ A \text{ if } d(B) > d(A).$$

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

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

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

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 Fig. 3.

3. Problem formulation

The FPMSP considered in this study can be described as follows: n jobs must be processed on m identical parallel machines, and the goal is to minimize the maximum completion time (makespan). The machines are identical, and each machine can process only one job at a time. 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. The execution-time membership function of each task is known and finite (P_j).

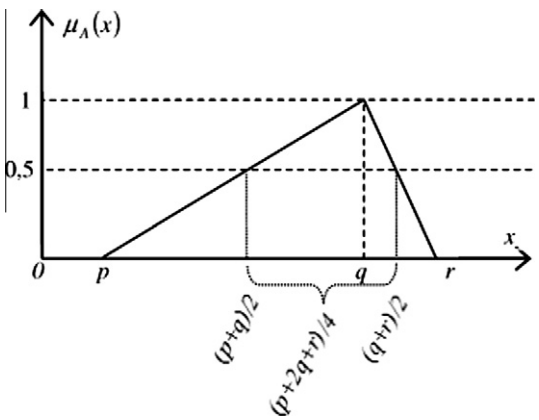

Fig. 2. Calculation of signed distance for a triangular fuzzy number.

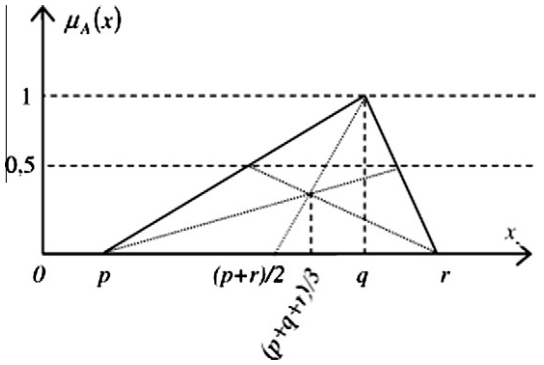

Fig. 3. Centroid defuzzification method for a triangular fuzzy number.

The variable $x(i,j)$ is the boolean variable that determines whether job j is processed by machine i (if $x(i,j) = 1$) or not (if $x(i,j) = 0$). The matrix **X** is composed of the elements $x(i,j)$, $x(i,j) \in \{0,1\}$. C_i is the fuzzy completion time of machine i:

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

Each job is to be processed without interruption and each machine can process only one job at a time:

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

All jobs have to be processed to complete the production:

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

The maximum completion time (makespan) is equal to:

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

Thus, the objective function can be defined as the minimization of this maximum completion time:

$$\textit{Objective function} = \textit{Minimization of} \ \max_{i=1}^{m}\left\{\sum_{j=1}^{n} x(i,j) \times P_j\right\}. \tag{ii}$$

4. Genetic optimization of FPMSP

The *GA*, inspired by the process of Darwinian evolution, has been recognized as a general search strategy and as an optimization method that is often useful for solving combinatorial problems. It was introduced in the 1970's by Holland [14], and Davis and Coombs [6] were the first to propose its application to solving scheduling problems. Since then, many other applications have appeared. In contrast to other local search methods, such as *Simulated Annealing* or *Tabu Search*, which work with one feasible solution at a time, *GA* utilizes a population of solutions in its search, giving it more resistance to premature convergence to local minima [32]. Another advantage of *GA* is its good performance in a large and complex search space. The ability of *GA* to explore and to exploit the search space simultaneously makes it more efficient than the other methods. Nevertheless, *GA* has also some disadvantages; the optimality of its performance is highly dependent on the realization of each of the above steps, and on the values of the *GA* parameters, namely the 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) [11,27].

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 (so called "generation"), the genetic operations of reproduction, natural selection, crossover and mutation are applied to search for potentially 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 helps the search process to jump out of locally optimal solutions. Reproduction

copies a chromosome to the next generation directly, so that chromosomes from various generations can cooperate in the evolution and improve the "quality" of the population after each generation [16]. In the following paragraphs, the general structure of *GA* is adapted to the fuzzy parallel machine scheduling problem in order to minimize the maximum completion time (makespan). A specific coding method is used to represent schedules on parallel machines. The fuzzy set operations, fuzzy ranking and defuzzification methods presented above are employed to schedule jobs with fuzzy processing times. Some genetic operators must be redesigned for this problem.

4.1. Coding

Row i of the matrix $\mathbf{X}$ consists of jobs to be processed on machine i. Rows 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 non-zero elements of gene i ($x(i,j) = 1$). The completion time (C_i) of machine i is equal to the sum of the processing times of jobs to be processed on that machine; it is called the "*value of gene i* " and it is defined by the following function:

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

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: each initial solution is called a "chromosome". Chromosomes can be identified by their order of creation, $k = 1,\ldots,N$. The initial population is the set of N chromosomes. The number of chromosomes, i.e. population size, is one of the important parameters of the *GA*. The 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 [45]. After the generation of a new population, the 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 FPMSP, completion times are fuzzy numbers, and the maximum completion time is determined using the "signed distance" ranking method presented in 2.2. Let C_i be the completion time of machine i, represented by a TFN, $C_i = (p_i, q_i, r_i)$. The maximum completion time is equal to:

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

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 [11]:

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

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.

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

The values of parameters α and β are determined according to the given problem. To solve the problem defined in Section 5, several tests have been performed in order to determine their optimal values; α is set equal to 100 and β to 10%. If β is greater than 20%, the fitness values of chromosomes approach zero regardless of the value of α is (independently of α); if β is less than 5%, the fitness values are too big. For values between 10% and 15%, the fitness values are reasonably acceptable and the results of the algorithm do not vary dramatically, so the value of 10% is selected. The parameter α determines the scale of the fitness values, it is set equal to 100 in order to produce fitness values between zero and 100.

4.4. Reproduction

Reproduction is the process by which parents copy themselves according to probabilities that are proportional to their fitness values. As a result, parents with higher fitness values will have higher probabilities of producing offspring in the next

generation [46]. In the proposed algorithm, 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 that favors good chromosomes, so as to obtain a better chance of producing good next generations [17]. It has the following steps [11]:

– Calculate the selection probability of chromosomes

$$P(k) = \frac{F_k}{\sum_{i=1}^{N} F(i)}, \quad k = 1, \ldots, N. \tag{v}$$

– Generate cut points, $S(k)$

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

– Generate N random numbers 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 the chromosomes to be selected.

4.5. Crossover

According to the coding method used in this paper, the crossover operator deals with genes rather than with chromosomes as in most of the applications in the 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 consists of 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 be combined, g_u, indicates the machine with the maximum completion time:

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

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

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

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

The 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:

$$f(g_u) - f(g_z) = \max_{i=1}^{m}\{f(g_u) - f(g_i)\}. \tag{viii}$$

4.6. Mutation

The crossover operator is used to combine existing genes in order to obtain new chromosomes, whereas the mutation operator creates new chromosomes by causing small perturbations in genes, thus helping to maintain the diversity of the population and to extend the solution space. Because of the coding method used in this paper, the mutation operator can be easily applied by alternating some elements $x(i,j)$ of matrix $\mathbf{X}$. Each column of matrix $\mathbf{X}$ has only one element with the value "1" (one job is processed only on one machine). Mutation can be carried out by moving this element randomly from one row to another (i.e. moving a job from one machine to another). This operation prevents the algorithm from getting stuck on local suboptimal solutions and is very helpful to maintain the richness of the population in dealing with large scale problems. When solving the problem considered in Section 5, only one chromosome is mutated in each generation. As the initial population consists only of 10 chromosomes, more mutations would add too much randomness to the algorithm.

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:

$$P_{j'} > f(g_u) - f(g_i), \quad i = 1, \ldots, m. \tag{ix}$$

The two sides of the inequality can be calculated easily using formulas *vi* and *vii*. They are both TFN and can be compared using the "signed distance" ranking method. If the inequality *ix* is satisfied for a chromosome, it means that this chromosome cannot be improved to provide better solutions. If it is satisfied for all chromosomes, none of the completion times ($C_{max}(k)$) can be improved; thus, the algorithm terminates 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 $minC_{max} = (p_k, q_k, r_k)$. A final quantifiable solution can be obtained by defuzzification:

$$x^* = \delta(minCmax) = (p_k + q_k + r_k)/3.$$

5. Numerical examples and simulation results

The *GA* is embedded in a simulation model and implemented using the *Simul8*^{TM} simulation software (Fig. 4). Each "work center" corresponds to one step of the algorithm. The simulation entity passes through the "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 each flow. Each flow, i.e. each iteration, begins with a randomly generated population of 10 chromosomes, half of which are exposed to "natural selection" while the other half undergoes a crossover operation. The crossover operation is repeated until the optimality criterion is satisfied. Between two crossover operations, the chromosomes are mutated in order to ensure "genetic diversity". It helps to extend the solution space by causing small perturbations in the genes. The lifetime of each generation is modeled by an entity passing through the system. The initial population survives 50 generations, meaning that 50 iterations are performed in 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 us to perform several iterations by using different initial populations. Accordingly, several tests can be easily performed using different random number sets.

5.1. Description of the case study

The case study can be described as follows: 9 jobs are to be processed on 4 parallel machines, the goal is 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.
- All machines are identical.

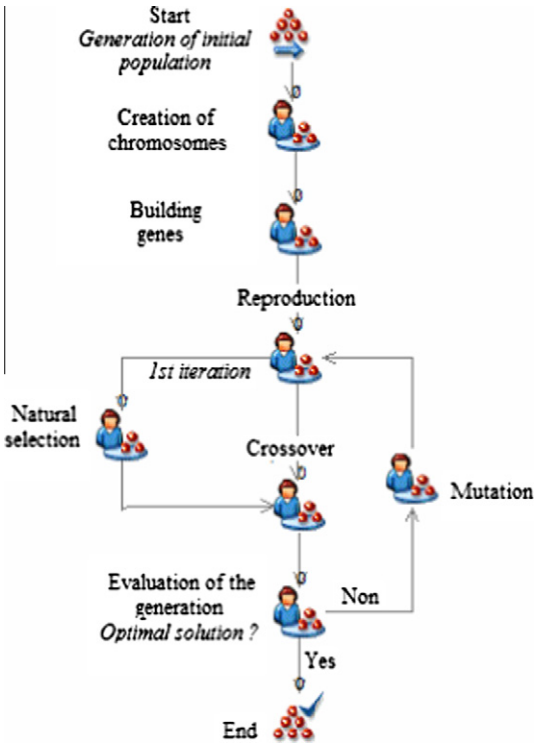

Fig. 4. Application of *GA* using simulation.

The following constraints must be respected:

- 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.

The *GA* is first tested for scheduling parallel machines with real values of processing times (Section 5.2). This exercise will serve to validate the proposed approach. Then, a numerical example using fuzzy processing times is solved by using the proposed *GA* approach (Section 5.3). The results are compared to those obtained with the *LPT* rule (Section 5.4), which is known as the most appropriate dispatching rule for such problems. The results are analyzed and the robustness of the approach is tested in Section 6.

5.2. Testing with real numbers

The *GA* is first applied to schedule jobs with real values of processing times ($P_j \in \mathbf{R}$, Table 1). The proposed *GA* deals with fuzzy numbers. In order to execute the same algorithm with real values, they have to be expressed as fuzzy numbers (Table 1). For this purpose, it is sufficient to consider TFNs as:

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

An ideal solution can be defined by supposing that jobs can be interrupted at anytime and can be calculated as:

$$C_{max}ideal = \sum_{j=1}^{n} \delta(P_j)/m.$$

The ideal solution is found to be 47.5 min. However, as one of the constraints is violated, this solution is not feasible. It only helps to determine the optimal solution and to compare solutions to the ideal one. Consequently, the optimal solution must be obviously greater than 47.5 min. Moreover, the sum of the processing times does not give 2 times 48 min in order to obtain $\sum P_j = \sum C_i = 48 + 48 + 47 + 47$. Thus, the optimal solution cannot be 48 min either. It can be concluded that $C_{max} = 49$ is the optimal solution.

The optimal solution is obtained 7 times during the simulation: iterations 4, 6, 19, 27, 32, 34, 41 (indicated by bold fonts in Table 2). However, some of them are provided by the same schedule. In fact, there are two schedules giving the same optimal solution. They are presented in Table 3 and 4.

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

5.3. Solving FPMSP

The same problem defined in Section 5.1 is considered, but with the new fuzzy processing times given in Table 5. This FPMSP is solved by using the proposed *GA* approach, which is the main purpose of the study.

The results are presented in Table 6. They are compared using the signed distance method and it is found that $C_{max} = (41, 52, 60)$ is the minimum. The defuzzification of the result by the centroid method gives $\delta(C_{max}) = 51$ min. Moreover, the calculation of the ideal solution, with the same violation of the constraint used above, leads to 50.5 min. Therefore, the result obtained by the proposed *GA* approach can be accepted as optimal.

The optimal solution is obtained 8 times in one simulation run and is produced by 2 different schedules, given in Table 7 and 8. These schedules lead to the same result but to different machine loads, which shows the ability of the proposed *GA* approach to explore alternative optimal schedules.

5.4. Comparison with LPT

The same problem defined in Section 5.3 is solved using the *LPT* rule, which is known as the most appropriate dispatching rule for PMSPs. The application of the *LPT* rule to FPMSPs is described in Hong et al. [15]. The results are given below in Table 9.

Table 1
Real values of processing times.

Processing times (min)		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
P_j	p	13	18	24	22	28	15	19	24	27
	q	13	18	24	22	28	15	19	24	27
	r	13	18	24	22	28	15	19	24	27

Table 2
Results with real values of processing times.

Iteration No.	C_{max}			$\delta(C_{max})$	*Iteration No.*	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	50	50	50	50	26	50	50	50	50
2	52	52	52	52	27	49	49	49	**49**
3	52	52	52	52	28	52	52	52	52
4	49	49	49	**49**	29	51	51	51	51
5	51	51	51	51	30	51	51	51	51
6	49	49	49	**49**	31	50	50	50	50
7	51	51	51	51	32	49	49	49	**49**
8	50	50	50	50	33	52	52	52	52
9	50	50	50	50	34	49	49	49	**49**
10	51	51	51	51	35	51	51	51	51
11	52	52	52	52	36	52	52	52	52
12	51	51	51	51	37	52	52	52	52
13	51	51	51	51	38	50	50	50	50
14	50	50	50	50	39	51	51	51	51
15	52	52	52	52	40	50	50	50	50
16	52	52	52	52	41	49	49	49	**49**
17	52	52	52	52	42	51	51	51	51
18	51	51	51	51	43	50	50	50	50
19	49	49	49	**49**	44	52	52	52	52
20	51	51	51	51	45	51	51	51	51
21	50	50	50	50	46	51	51	51	51
22	52	52	52	52	47	52	52	52	52
23	50	50	50	50	48	52	52	52	52
24	51	51	51	51	49	50	50	50	50
25	51	51	51	51	50	51	51	51	51

Table 3
The first optimal schedule with real values of processing times.

Machines	Scheduled jobs			C_i
Iterations No. 4 & No. 19				
M.1	job 3	job 8		48.00
M.2	job 5	job 7		47.00
M.3	job 4	job 9		49.00
M.4	job 2	job 1	job 6	46.00

Table 4
The second optimal schedule with real values of processing times.

Machines	Scheduled jobs			C_i
Iterations No.6 & No.27 & No.32 & No.34 & No.41				
M.1	job 8	job 3		48.00
M.2	job 4	job 9		49.00
M.3	job 1	job 7	job 6	47.00
M.4	job 2	job 5		46.00

Table 5
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_j	p	10	12	22	16	20	12	12	21	18
	q	13	18	24	22	28	15	19	24	27
	r	22	24	38	25	30	24	44	30	36

The application of the "signed distance" method yields $C_{max} = C_1 = (44,52,84)$, and the application of "centroid" defuzzification method gives $\delta(C_{max}) = 60$ min. The results reveal that the proposed *GA* approach surpasses the *LPT* rule and illustrate the need for efficient and effective algorithms for FPMSPs.

Table 6
Results with fuzzy processing times.

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	38	50	71	53	26	38	50	71	53
2	42	52	68	54	27	41	52	60	**51**
3	38	50	71	53	28	40	51	74	55
4	43	48	68	53	29	42	52	68	54
5	41	52	60	**51**	30	38	50	71	53
6	38	50	71	53	31	43	48	68	53
7	39	51	66	52	32	38	53	71	54
8	42	52	68	54	33	38	53	71	54
9	43	48	68	53	34	38	55	66	53
10	38	50	71	53	35	39	51	66	52
11	42	52	68	54	36	42	52	68	54
12	39	51	66	52	37	42	52	68	54
13	39	51	66	52	38	43	48	68	53
14	43	48	68	53	39	38	50	71	53
15	38	53	71	54	40	41	52	60	**51**
16	38	50	71	53	41	38	50	71	53
17	41	52	60	**51**	42	38	55	66	53
18	41	52	60	**51**	43	38	55	66	53
19	43	48	68	53	44	41	52	60	**51**
20	39	51	66	52	45	38	50	71	53
21	39	51	66	52	46	38	50	71	53
22	40	51	74	55	47	38	55	66	53
23	43	48	68	53	48	42	52	68	54
24	41	52	60	**51**	49	38	50	71	53
25	41	52	60	**51**	50	38	53	71	54

Table 7
The first optimal schedule with fuzzy processing times.

Machines	Scheduled jobs			C_i		
Iterations No. 5 & No. 24						
M.1	job 8	job 5		41.0	52.0	60.0
M.2	job 2	job 6	job 1	34.0	46.0	70.0
M.3	job 4	job 9		34.0	49.0	61.0
M.4	job 3	job 7		34.0	43.0	82.0

Table 8
The second optimal schedule with fuzzy processing times.

Machines	Scheduled jobs			C_i		
Iterations No. 17 & No. 18 & No. 25 & No. 27 & No. 40 & No. 44						
M.1	job 7	job 9		30.0	46.0	80.0
M.2	job 5	job 8		41.0	52.0	60.0
M.3	job 3	job 4		38.0	46.0	63.0
M.4	job 2	job 1	job 6	34.0	46.0	73.0

Table 9
Results of *LPT*.

Machines	Scheduled jobs			C_i		
M.1	job 3	job 6	job 1	44.0	52.0	84.0
M.2	job 9	job 2		30.0	45.0	60.0
M.3	job 5	job 4		36.0	50.0	55.0
M.4	job 8	job 7		33.0	43.0	74.0

Table 10
Summary table of test results.

Test No.	Number of optimal solution reached	Average $\delta(C_{max})$ (min)
1	7 times	53.20
2	10 times	52.86
3	7 times	53.34
4	10 times	53.00
5	8 times	53.10
6	4 times	53.14
7	6 times	53.34
8	10 times	52.92
9	4 times	53.12
10	6 times	53.32
Average	7.2 times	53.13

6. Result analysis

6.1. Results of GA

The results show that the proposed *GA* approach yields good results efficiently. An examination of Table 6 reveals that the optimal solution is reached at only the 5th iteration. In addition, the optimal solution is reached several times in one run. The same Table 6 shows that the optimal solution is reached 8 times in 50 iterations, which demonstrates the efficiency of the algorithm. The same success is repeated during the robustness tests performed with 10 different random number sets (Section 6.2).

The effectiveness of the algorithm can be evaluated by comparing the result of the *GA* approach (51 min, given in Table 6) to the ideal but infeasible solution of the problem (50.5 min, Section 5.3), or to the results provided by the *LPT* rule (60 min, Section 5.4). The optimal result of the algorithm differs by only 0.99% from the ideal solution and is 17.65% better then the result of the *LPT* rule, which is known as the best dispatching rule for minimizing the maximum completion time of parallel machines. Moreover, even the worst result (55 min, Table 6) of the algorithm is much more better (9.09%) than the result obtained by the *LPT* rule. Another conclusion suggested by the examination of these results is that the non-optimal results are not too "bad" either. The average result of 50 iterations is 52.86 min (Table 6), which differs by only 3.65% from the optimal solution. These findings show the effectiveness of the proposed approach.

Another benefit of this approach is that, thanks to its being a search algorithm, unlike other methods proposed for such scheduling problems as *Linear Programming*, *Branch* & *Bound*, and *LPT*, the proposed approach is able to explore alternative schedules providing the same results. Two alternative schedules, both optimal, are proposed in Table 7 and 8.

6.2. Testing the robustness

Roy [38] defines the term "robust" as an adjective referring to the capacity of withstanding "vague approximations" and/or "zones of ignorance" in order to prevent undesirable impacts, notably the degradation of the properties to be maintained. The robustness of the *GA* is considered in only a few studies in the literature [13]. Considering the definition of Roy [38], 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".

A 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 is a different type of analysis, known as "sensitivity analysis". In this study, as the problem considered is just a numerical example, there is no need to perform a sensitivity analysis. Moreover, sensitivity analysis measures how the solution (here the optimal schedule) responses to the changes in the inputs.

10 different tests were performed using 10 different random number sets. The detailed results are given in the Appendix and a summary of the results is presented in Table 10. The second column of the table shows how many times the optimal solution ($\delta(C_{max}) = 51$ min) is reached, and the third column shows the average results of the 50 iterations in one test.

According to the results, the proposed *GA* approach has demonstrated its efficiency and effectiveness in solving FPMSPs. Optimal solutions are again reached quickly and repeatedly (7.2 times on average) in one run. The non-optimal schedules are not "bad" either; the average of all solutions found in 50 iterations of 10 tests is 53.13 min. Moreover, the algorithm has the advantage of exploring all the alternative optimal schedules.

7. Conclusion

The parallel machine scheduling problem receives considerable attention in both academic and industrial fields. Various factors involved in scheduling problems are often imprecise or uncertain. The fuzzy set theory provides a convenient alternative framework for modeling real-world systems mathematically and offers several advantages in the use of heuristic

approaches. In this study, the parallel machine scheduling problem with fuzzy processing times (FPMSP) is considered and a *GA* approach is proposed to minimize the maximum completion time (makespan).

New *GA* operators and two of the more well-known ranking and defuzzification methods are used 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 to implement the *GA* is preferred because of the evolutionary structure of the algorithm and the ability of the 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 was first tested for scheduling parallel machines with real values of processing times (Section 5.2). The 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 was solved by using the proposed *GA* approach (Section 5.3), and the results were compared to those obtained by the *LPT* rule (Section 5.4), which is known as the most appropriate dispatching rule for such problems. The results showed that the proposed approach surpasses the *LPT* rule and illustrated the need of using efficient and effective heuristics for FPMSPs.

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

- The proposed approach is quite efficient because it can find optimal results quickly and several times in one run.
- The 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 the *LPT* rule.
- Because it is a search algorithm, the algorithm can explore other alternative schedules giving the same optimal result.

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

The motivation of the author was to contribute to the literature on scheduling parallel machines under uncertainty. 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 was proposed to solve the parallel machine scheduling problem with fuzzy processing times. Future research can be directed toward scheduling parallel machines under additional constraints, such as setup times or ready times. This can help to consider more realistic scheduling problems in the industrial field. The problem considered here will then be a special case of this problem, with the setup times and ready times equal to zero.

Appendix A. Detailed results of the robustness test

Tests 1–10.

Test 1

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	38	50	71	53	26	39	51	66	52
2	38	53	71	54	27	41	52	60	**51**
3	39	51	66	52	28	42	52	68	54
4	39	51	66	52	29	43	48	68	53
5	43	48	68	53	30	38	53	71	54
6	42	52	68	54	31	38	50	71	53
7	38	50	71	53	32	39	51	66	52
8	38	50	71	53	33	42	52	68	54
9	43	48	68	53	34	39	51	66	52
10	38	55	66	53	35	41	52	60	**51**
11	41	52	60	**51**	36	38	50	71	53
12	43	48	68	53	37	43	52	76	57
13	43	52	76	57	38	43	52	76	57
14	43	48	68	53	39	41	52	60	**51**
15	43	48	68	53	40	34	47	90	57
16	42	52	68	54	41	42	52	68	54
17	42	52	68	54	42	41	52	60	**51**
18	41	52	60	**51**	43	42	52	68	54
19	39	51	66	52	44	40	51	74	55
20	38	50	71	53	45	38	53	71	54
21	38	50	71	53	46	38	55	66	53
22	38	53	71	54	47	42	52	68	54
23	43	48	68	53	48	38	55	66	53
24	41	52	60	**51**	49	42	52	68	54
25	39	51	66	52	50	38	50	71	53

Test 2

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	38	50	71	53	26	38	55	66	53
2	38	55	66	53	27	40	51	74	55
3	42	52	68	54	28	39	51	66	52
4	38	53	71	54	29	42	52	68	54
5	41	52	60	**51**	30	38	55	66	53
6	38	50	71	53	31	41	52	60	**51**
7	41	52	60	**51**	32	39	51	66	52
8	40	51	74	55	33	38	50	71	53
9	42	52	68	54	34	40	51	74	55
10	41	52	60	**51**	35	41	52	60	**51**
11	43	48	68	53	36	39	51	66	52
12	38	55	66	53	37	40	51	74	55
13	43	48	68	53	38	41	52	60	**51**
14	38	53	71	54	39	39	51	66	52
15	40	51	74	55	40	41	52	60	**51**
16	43	48	68	53	41	42	52	68	54
17	38	55	66	53	42	43	48	68	53
18	38	50	71	53	43	38	55	66	53
19	38	50	71	53	44	41	52	60	**51**
20	41	52	60	**51**	45	38	50	71	53
21	38	55	66	53	46	38	50	71	53
22	38	55	66	53	47	43	48	68	53
23	41	52	60	**51**	48	38	53	71	54
24	38	50	71	53	49	38	50	71	53
25	39	51	66	52	50	42	52	68	54

Test 3

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	42	52	68	54	26	42	52	68	54
2	38	53	71	54	27	34	47	90	57
3	39	51	66	52	28	42	52	68	54
4	40	51	74	55	29	38	55	66	53
5	40	51	74	55	30	41	52	60	**51**
6	41	52	60	**51**	31	38	50	71	53
7	43	48	68	53	32	38	53	71	54
8	38	53	71	54	33	38	50	71	53
9	41	52	60	**51**	34	38	53	71	54
10	40	51	74	55	35	38	53	71	54
11	38	55	66	53	36	39	51	66	52
12	38	55	66	53	37	34	47	90	57
13	38	55	66	53	38	41	52	60	**51**
14	38	50	71	53	39	38	55	66	53
15	38	55	66	53	40	38	53	71	54
16	38	50	71	53	41	38	53	71	54
17	43	48	68	53	42	40	51	74	55
18	38	50	71	53	43	38	50	71	53
19	41	52	60	**51**	44	40	51	74	55
20	43	48	68	53	45	39	51	66	52
21	42	52	68	54	46	41	52	60	**51**
22	42	52	68	54	47	38	55	66	53
23	38	53	71	54	48	38	50	71	53
24	34	47	90	57	49	43	48	68	53
25	41	52	60	**51**	50	39	51	66	52

Test 4

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	38	50	71	53	26	38	50	71	53
2	41	52	60	**51**	27	41	52	60	**51**
3	38	55	66	53	28	38	55	66	53
4	39	51	66	52	29	38	55	66	53
5	41	52	60	**51**	30	38	55	66	53
6	43	48	68	53	31	39	51	66	52
7	38	55	66	53	32	41	52	60	**51**
8	40	51	74	55	33	38	55	66	53
9	43	48	68	53	34	42	52	68	54
10	42	52	68	54	35	41	52	60	**51**
11	38	55	66	53	36	38	53	71	54
12	41	52	60	**51**	37	42	52	68	54
13	41	52	60	**51**	38	42	52	68	54
14	41	52	60	**51**	39	38	55	66	53
15	39	51	66	52	40	38	53	71	54
16	42	52	68	54	41	38	55	66	53
17	34	47	90	57	42	38	55	66	53
18	38	53	71	54	43	43	48	68	53
19	38	55	66	53	44	38	50	71	53
20	41	52	60	**51**	45	38	53	71	54
21	38	50	71	53	46	39	51	66	52
22	34	47	90	57	47	42	52	68	54
23	38	50	71	53	48	38	55	66	53
24	41	52	60	**51**	49	43	48	68	53
25	38	55	66	53	50	43	52	76	57

Test 5

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	40	51	74	55	26	42	52	68	54
2	41	52	60	**51**	27	39	51	66	52
3	41	52	60	**51**	28	38	50	71	53
4	38	53	71	54	29	38	55	66	53
5	42	52	68	54	30	42	52	68	54
6	39	51	66	52	31	41	52	60	**51**
7	42	52	68	54	32	38	50	71	53
8	38	55	66	53	33	38	50	71	53
9	40	51	74	55	34	42	52	68	54
10	38	53	71	54	35	38	50	71	53
11	38	53	71	54	36	41	52	60	**51**
12	40	51	74	55	37	39	51	66	52
13	38	50	71	53	38	38	50	71	53
14	38	55	66	53	39	39	51	66	52
15	41	52	60	**51**	40	39	51	66	52
16	40	51	74	55	41	43	48	68	53
17	38	53	71	54	42	38	55	66	53
18	40	51	74	55	43	38	55	66	53
19	43	48	68	53	44	41	52	60	**51**
20	38	53	71	54	45	40	55	73	56
21	38	50	71	53	46	42	52	68	54
22	38	53	71	54	47	38	50	71	53
23	38	53	71	54	48	42	52	68	54
24	41	52	60	**51**	49	41	52	60	**51**
25	38	50	71	53	50	39	51	66	52

Test 6

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	39	51	66	52	26	38	50	71	53
2	38	53	71	54	27	38	50	71	53
3	42	52	68	54	28	39	51	66	52
4	39	51	66	52	29	38	53	71	54
5	38	53	71	54	30	38	55	66	53
6	38	53	71	54	31	43	48	68	53
7	43	48	68	53	32	38	53	71	54
8	43	48	68	53	33	38	55	66	53
9	41	52	60	**51**	34	41	52	60	**51**
10	38	50	71	53	35	38	53	71	54
11	38	55	66	53	36	41	52	60	**51**
12	39	51	66	52	37	38	53	71	54
13	42	52	68	54	38	38	50	71	53
14	38	55	66	53	39	38	55	66	53
15	38	55	66	53	40	38	55	66	53
16	38	50	71	53	41	38	55	66	53
17	39	51	66	52	42	34	47	90	57
18	38	50	71	53	43	38	50	71	53
19	38	55	66	53	44	40	51	74	55
20	42	52	68	54	45	38	55	66	53
21	38	53	71	54	46	38	55	66	53
22	41	52	60	**51**	47	40	51	74	55
23	38	55	66	53	48	38	55	66	53
24	39	51	66	52	49	42	52	68	54
25	42	52	68	54	50	38	50	71	53

Test 7

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	43	48	68	53	26	38	55	66	53
2	38	55	66	53	27	41	52	60	**51**
3	38	55	66	53	28	42	52	68	54
4	42	52	68	54	29	41	52	60	**51**
5	42	52	68	54	30	38	55	66	53
6	41	52	60	**51**	31	38	55	66	53
7	34	47	90	57	32	38	50	71	53
8	43	48	68	53	33	38	55	66	53
9	41	52	60	**51**	34	38	50	71	53
10	40	51	74	55	35	41	52	60	**51**
11	40	51	74	55	36	38	53	71	54
12	38	53	71	54	37	42	52	68	54
13	38	50	71	53	38	38	55	66	53
14	38	53	71	54	39	38	50	71	53
15	38	50	71	53	40	42	52	68	54
16	39	51	66	52	41	38	55	66	53
17	38	55	66	53	42	42	52	68	54
18	38	55	66	53	43	38	55	66	53
19	34	47	90	57	44	42	52	68	54
20	40	51	74	55	45	39	51	66	52
21	41	52	60	**51**	46	38	53	71	54
22	42	52	68	54	47	38	55	66	53
23	40	51	74	55	48	40	51	74	55
24	39	51	66	52	49	42	52	68	54
25	39	51	66	52	50	40	51	74	55

Test 8

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	42	52	68	54	26	38	55	66	53
2	38	55	66	53	27	41	52	60	**51**
3	41	52	60	**51**	28	41	52	60	**51**
4	41	52	60	**51**	29	38	55	66	53
5	38	55	66	53	30	40	51	74	55
6	42	52	68	54	31	43	48	68	53
7	41	52	60	**51**	32	38	55	66	53
8	42	52	68	54	33	38	50	71	53
9	38	50	71	53	34	38	50	71	53
10	41	52	60	**51**	35	39	51	66	52
11	38	55	66	53	36	38	53	71	54
12	38	55	66	53	37	42	52	68	54
13	39	51	66	52	38	38	55	66	53
14	38	50	71	53	39	42	52	68	54
15	38	50	71	53	40	38	55	66	53
16	42	52	68	54	41	34	47	90	57
17	43	48	68	53	42	41	52	60	**51**
18	41	52	60	**51**	43	42	52	68	54
19	38	50	71	53	44	43	48	68	53
20	41	52	60	**51**	45	39	51	66	52
21	39	51	66	52	46	42	52	68	54
22	34	47	90	57	47	38	55	66	53
23	41	52	60	**51**	48	43	48	68	53
24	38	50	71	53	49	38	50	71	53
25	42	52	68	54	50	38	55	66	53

Test 9

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	38	55	66	53	26	38	50	71	53
2	38	55	66	53	27	39	51	66	52
3	41	52	60	**51**	28	38	50	71	53
4	43	48	68	53	29	38	55	66	53
5	38	53	71	54	30	38	53	71	54
6	38	55	66	53	31	38	55	66	53
7	34	47	90	57	32	43	48	68	53
8	41	52	60	**51**	33	38	55	66	53
9	42	52	68	54	34	39	51	66	52
10	38	55	66	53	35	42	52	68	54
11	38	55	66	53	36	38	50	71	53
12	38	50	71	53	37	42	52	68	54
13	41	52	60	**51**	38	38	50	71	53
14	38	50	71	53	39	40	51	74	55
15	43	48	68	53	40	42	52	68	54
16	41	52	60	**51**	41	38	55	66	53
17	39	51	66	52	42	38	53	71	54
18	39	51	66	52	43	39	51	66	52
19	38	55	66	53	44	38	50	71	53
20	42	52	68	54	45	38	55	66	53
21	38	55	66	53	46	43	48	68	53
22	42	52	68	54	47	42	52	68	54
23	42	52	68	54	48	43	48	68	53
24	38	53	71	54	49	39	51	66	52
25	40	51	74	55	50	38	50	71	53

Test 10

Iteration No.	C_{max}			$\delta(C_{max})$	Iteration No.	C_{max}			$\delta(C_{max})$
	p	q	r			p	q	r	
1	40	51	74	55	26	38	50	71	53
2	38	53	71	54	27	43	48	68	53
3	38	50	71	53	28	38	53	71	54
4	40	51	74	55	29	41	52	60	**51**
5	41	52	60	**51**	30	38	50	71	53
6	41	52	60	**51**	31	38	53	71	54
7	42	52	68	54	32	39	51	66	52
8	38	53	71	54	33	41	52	60	**51**
9	38	53	71	54	34	34	47	90	57
10	41	52	60	**51**	35	40	51	74	55
11	38	55	66	53	36	39	51	66	52
12	38	50	71	53	37	38	50	71	53
13	43	48	68	53	38	38	55	66	53
14	43	48	68	53	39	40	51	74	55
15	38	55	66	53	40	38	50	71	53
16	42	52	68	54	41	43	48	68	53
17	43	48	68	53	42	43	48	68	53
18	42	52	68	54	43	39	51	66	52
19	39	51	66	52	44	42	52	68	54
20	38	55	66	53	45	42	52	68	54
21	38	50	71	53	46	34	47	90	57
22	40	51	74	55	47	38	50	71	53
23	41	52	60	**51**	48	42	52	68	54
24	38	50	71	53	49	40	55	73	56
25	43	48	68	53	50	38	55	66	53

References

[1] A. Anglani, A. Grieco, E. Guerriero, R. Musmanno, Robust scheduling of parallel machines with sequence-dependent set-upcosts, European Journal of Operational Research 161 (2005) 704–720.
[2] J. Balasubramanian, I.E. Grossmann, Scheduling optimization under uncertainty – an alternative approach, Computers and Chemical Engineering 27 (2003) 469–490.
[3] J.J. Buckley, Y. Hayashi, Fuzzy genetic algorithm and applications, Fuzzy Sets and Systems 61 (1994) 129–136.
[4] J.J. Buckley, T. Feuring, Evolutionary algorithm solution to fuzzy problems: fuzzy linear programming, Fuzzy Sets and Systems 109 (2000) 35–53.
[5] C.T. Cheng, C.P. Ou, K.W. Chau, Combining a fuzzy optimal model with a genetic algorithm to solve multiobjective rainfall-runoff model calibration, Journal of Hydrology 1–4 (268) (2002) 72–86.
[6] L. Davis, S. Coombs, Genetic algorithms and communication link speed design: theoretical considerations, Grefenstette (1987) 252–256.
[7] D. Dubois, H. Prade, Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, 1988.
[8] M.R. Garey, D.S. Johnson, Computer and Intractability: A Guide to the Theory of NP-Completeness, WH Freeman, San Francisco, 1979.
[9] A.H. Gharehgozli, R. Tavakkoli-Moghaddam, 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 25 (2009) 853–859.
[10] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 5 (13) (1986) 533–549.
[11] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, MA, 1989.
[12] S. Gürsel, E. Baez, Minimizing the number of tardy jobs in identical machine scheduling, Computers & Industrial Engineering 4 (25) (1993) 243–246.
[13] W.H. Ho, S.H. Chen, T.K. Liu, J.H. Chou, Design of robust-optimal output feedback controllers for linear uncertain systems using LMI-based approach and genetic algorithm, Information Sciences 23 (180) (2010) 4529–4542.
[14] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[15] T.P. Hong, C.M. Huang, K.M. Yu, LPT scheduling for fuzzy tasks, Fuzzy Sets and Systems 97 (1988) 277–286.
[16] P. Ji, M.T. Sze, W.B. Lee, A genetic algorithm of determining cycle time for printed circuit board assembly lines, European Journal of Operations Research 128 (2001) 175–184.
[17] C. Jou, A genetic algorithm with sub-indexed partitioning genes and its application to production scheduling of parallel machines, Computers & Industrial Engineering 48 (2005) 39–54.
[18] A.H. Kashan, B. Karimi, M. Jenabi, A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes, Computers & Operations Research 35 (2008) 1084–1098.
[19] W.C. Lee, C.C. Wu, Some single-machine and m-machine flowshop scheduling problems with learning considerations, Information Sciences 22 (179) (2009) 3885–3892.
[20] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, Operational Research 26 (1978) 22–35.
[21] J.Y. Lin, C.T. Cheng, K.W. Chau, Using support vector machines for long-term discharge prediction, Hydrological Sciences Journal 4 (51) (2006) 599–612.
[22] B. Liu, Uncertain Programming, John Wiley & Sons, New York, 1999.
[23] B. Liu, Dependent-chance programming with fuzzy decisions, IEEE Transactions on Fuzzy Systems 7 (1999) 354–360.
[24] B. Liu, Theory and Practice of Uncertain Programming, Physica-Verlag, Heidelberg, 2002.
[25] B. Liu, K. Iwamura, Chance constrained programming with fuzzy parameters, Fuzzy Sets and Systems 94 (1998) 227–237.
[26] B. Liu, K. Iwamura, A note on chance constrained programming with fuzzy coefficients, Fuzzy Sets and Systems 100 (1998) 229–233.
[27] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1992.
[28] L. Min, W. Cheng, A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines, Artificial Intelligence in Engineering 13 (1999) 399–403.
[29] L. Min, W. Cheng, Scheduling algorithm based on evolutionary computing in identical parallel machine production line, Robotics and Computer-Integrated Manufacturing 19 (2003) 401–407.

[30] P.Y. Mok, C.K. Kwong, W.K. Wong, Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory, European Journal of Operations Research 177 (2007) 1876–1893.
[31] N. Muttil, K.W. Chau, Neural network and genetic programming for modelling coastal algal blooms, International Journal of Environment and Pollution 3–4 (28) (2006) 223–238.
[32] B.J. Park, H.R. Choi, H.S. Kim, A hybrid genetic algorithm for the job shop scheduling problems, Computers & Industrial Engineering 45 (2003) 597–613.
[33] J. Peng, B. Liu, Parallel machine scheduling models with fuzzy processing times, Information Sciences-Informatics and Computer Science: An International Journal 166 (2004) 49–66.
[34] D. Petrovic, A. Duenas, A fuzzy logic based production scheduling/rescheduling in the presence of uncertain disruptions, Fuzzy Sets and Systems 157 (2006) 2273–2285.
[35] K. Raja, C. Arumugam, V. Selladurai, Non-identical parallel-machine scheduling using genetic algorithm and fuzzy logic approach, International Journal of Services and Operations Management 1 (4) (2008) 333–346.
[36] A.H.G. RinnooyKan, Machine Scheduling Problems: Classification, Complexity and Computations, Martinus Nijhoff, The Hague, 1976.
[37] B. Roy, P. Vincke, Relational systems of preference with one or more pseudo-criteria: Some new concepts and results, Management Science 30 (1984) 1323–1335.
[38] B. Roy, Robustness in operational research and decision aiding: a multi-faceted issue, European Journal of Operational Research 200 (2010) 629–638.
[39] R. Sethi, On the complexity of mean flow time scheduling, Mathematics of Operations Research 4 (2) (1977) 320–330.
[40] R. Slowinski, M. Hapke, Scheduling Under Fuzziness, Physica-Verlag Editions, New York, 2000.
[41] N. Van Hop, A heuristic solution for fuzzy mixed-model line balancing problem, European Journal of Operational Research 3 (168) (2006) 798–810.
[42] L.V. Vanegas, A.W. Labib, Application of new fuzzy weighted average method to engineering design evaluation, International Journal of Production Research 6 (39) (2001) 1147–1162.
[43] Y. Wen, H. Xu, J. Yang, Information Sciences, A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system 3 (181) (2011) 567–581.
[44] J.S. Yao, K. Wu, Ranking fuzzy numbers based on decomposition principle and signed distance, Fuzzy Sets and Systems 116 (2000) 275–288.
[45] L.A. Zadeh, Fuzzy set as a basis for a theory of possibility, Fuzzy Sets and Systems 1 (1978) 3–28.
[46] H. Zhou, Y. Feng, L. Han, The hybrid heuristic genetic algorithm for job shop scheduling, Computers & Industrial Engineering 3 (40) (2001) 191–200.

Savaş Balin completed his industrial engineering undergraduate degree at the University of Galatasaray. He then continued his education with a master's degree in decision making at the University of Paris Dauphine. He gained his Ph.D. degree in production management from the same University. His Ph.D. dealt with the system design and quality improvement in service industry. His research interests include production scheduling, evolutionary heuristics, system analysis & design, modelling and simulation. He is now teaching in the Industrial Engineering Department of Yıldız Technical University-Istanbul and charged with establishing a new System Engineering Department at the same University.