J Intell Manuf (2012) 23:1179–1194
DOI 10.1007/s10845-010-0428-x

Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem

Xiaohui Li · Farouk Yalaoui · Lionel Amodeo · Hicham Chehade

Received: 22 June 2010 / Accepted: 24 June 2010 / Published online: 7 July 2010

Abstract This paper deals with a multiobjective parallel machines scheduling problem. It consists in scheduling n independent jobs on m identical parallel machines. The job data such as processing times, release dates, due dates and sequence dependent setup times are considered. The goal is to optimize two different objectives: the makespan and the total tardiness. A mixed integer linear program is proposed to model the studied problem. As this problem is NP-hard in the strong sense, a metaheuristic method which is the second version of the non dominated sorting genetic algorithm (NSGA-II) is proposed to solve this problem. Since the parameters setting of a genetic algorithm is difficult, a fuzzy logic controller coupled with the NSGA-II (FLC-NSGA-II) is therefore proposed. The role of the fuzzy logic is to better set the crossover and the mutation probabilities in order to update the search ability. After that, an exact method based on the two phase method is also developed. We have used four measuring criteria to compare these methods. The experimental results show the advantages and the efficiency of FLC-NSGA-II.

Keywords Scheduling problem · Parallel machines · Metaheuristics · NSGA-II · Fuzzy logic · Two phase method

X. Li (✉) · F. Yalaoui · L. Amodeo · H. Chehade
Institut Charles Delaunay, LOSI, University of Technology of Troyes, UMR-STMR 6279, 12 Rue Marie Curie, BP 2060, 10000 Troyes Cedex, France
e-mail: xiaohui.li@utt.fr

H. Chehade
e-mail: hicham.chehade@utt.fr

Introduction

The parallel machines scheduling problem is widely studied in the literature because it frequently appears in the industrial production. In this problem, a set $N = \{1, \ldots n\}$ of n jobs is to be scheduled on m identical parallel machines. Each job j is considered with a processing time p_j, a release date r_j, a due date d_j and a sequence dependent setup time s_{ij} which stands for the transfer time between two adjacent jobs i and j (job j is scheduled immediately before job i). Since real life scheduling problems often have several conflicting objectives, two different ones are considered in this work: the minimisation of the makespan and the minimisation of the total tardiness. This problem is denoted by $P_m \mid s_{ij}, r_j \mid C_{\max}, \sum T_j$ in the three field notation.

The most studied criterion in scheduling problems is the makespan. It is the completion time of the job which is finished at last (maximum completion time of jobs). Cheng and Sin (1990) have proved that the problem of minimizing the makespan on two identical parallel machines is NP-hard. Brucker (1998) has proved that if the number of machines is greater than two, then the problem is even strongly NP-hard. In the literature, many methods are proposed to solve it. Gendreau et al. (2001) have proposed a heuristic and a lower bound for the $P_m \mid s_{ij} \mid C_{\max}$ problem. Yalaoui and Chu (2002) proposed a heuristic for the $P_m \mid s_{ij}, split \mid C_{\max}$ problem. An exact method is also developed to solve it. Gharbi and Haouari (2002) have proposed a lower bound and use the branch-and-bound method to solve the $P_m \mid r_j, q_j \mid C_{\max}$ problem, where q_j is the delivery time.

The total tardiness is another concerned criterion which is the sum of tardiness of every job. The parallel machines scheduling problem to minimize the total tardiness is known as NP-hard according to Koulamas (1994). Some exact methods are proposed to solve this problem like the

branch-and-bound method proposed by Azizoglu and Kirca (1998).

Yalaoui and Chu (2002) have also developed a lower bound and a branch-and-bound to solve the same problem. Shim and Kim (2007) have developed some dominance properties and a lower bound for the same problem, and a branch-and-bound method is adopted to solve it. Moreover, many heuristics have been developed, such that of Pritsker et al. (1969), Alidaee and Rosa (1997), Baker and Bertrand (1982).

Many approached methods have been proposed to solve the multiobjective optimization problems (MOPs). Schaffer (1985) proposed an algorithm named VEGA (vector evaluated genetic algorithm) to solve the multiobjective optimization problem. It was the first method which is based on the simple genetic algorithm to solve MOPs. Kursame (1991) proposed the VOES method (vector-optimization evolution strategy) in 1990. Hajela and Lin (1992) proposed the HLGA (Hijela's and Lin's genetic algorithm) in 1993, but these three algorithms are non-Pareto approaches. In the multiobjective optimization problems, there is no single optimal solution, but a set of non-dominated solutions. They are called as the Pareto optimal solutions. Recently, the researchers are interested by the Pareto-based approaches. Multiple objective genetic algorithms (MOGA) (Fonseça and Fleming 1993), Niched Pareto genetic algorithm (NPGA) (Horn et al. 1994), Non-dominated sorting genetic algorithm (NSGA-II) (Deb et al. 2000), Strength Pareto evolutionary algorithm (SPEA-II) (Zitzler 2001), Pareto archived evolution strategy (PAES) (Knowles and Corne 2000) and Pareto envelope based selection algorithm (PESA) (Corne et al. 2000) are proposed for solving MOP, and are widely applied for scheduling problems. For example, Bouibede-Hocine et al. (2006) have used NSGA-II and SPEA algorithms for the problem. Dugardin et al. (2007) have used the NSGA-II for solving a hybrid job shop and a parallel machines scheduling problem. Cochran et al. (2003) have proposed a two-stage multi-population genetic algorithm (MPGA) to minimize three different criteria: makespan, the total weighted tardiness and the total weighted completion time. Chang et al. (2005) have minimized the makespan and the total tardiness for parallel machines scheduling problem by means of a two-phase subpopulation genetic algorithm (SPGA). The numerical results demonstrate that the SPGA is better than NSGA-II and MOGA. After that, they have proposed a modified version of SPGA by using a global archive and an adaptive strategy for solving the same problem (Chang et al. 2006). Eren (2008) has proposed a mathematical model and three heuristic methods for solving a parallel machines problem with the constraints of setup times and remove times.

Since the parameters setting of genetic algorithms is very difficult, we have developed a fuzzy logic controller to better set the crossover and mutation probabilities. Fuzzy logic was first proposed by Zadeh (1965) for system control. In the works of King et al. (2004), Lofti and Kashani, (2004) and Song et al. (1997), the fuzzy logic used to guide a GA for solving their single objective problems. Lau et al. (2009a,b) have proposed the FL-GA, FL-NSGA2, FL-SPEA2 and FL-MICROGA to solve the vehicle routing problems. Yalaoui et al. (2010) have solved a reentrant scheduling problem with FLC-GA and FLC-NSGA2.

The exact methods are also widely applied to solve the parallel machines scheduling problems. However, most of them are destinated to minimize a single objective. We are interested here to apply the exact method for solving the multiobjective optimization problem. A well-known method to solve multiobjective problems is the aggregation (or weighted sum) method, but this latter cannot find the whole Pareto optimal solution set (only the supported solutions, the detail will be explained in section "Resolution methods"). There are several other exact methods which aim to find the whole Pareto optimal set, such as ϵ-constraint, TPM (Two Phase Method) and PPM (Parallel Partitioning Method). The ϵ-constraint method is based on the ϵ-constraint concept to enumerate Pareto solutions (Haimes et al. 1971, 1975). The TPM is firstly proposed for solving a bi-objective assignment problem by Ulungu and Teghem (1995). Lemesre et al. (2007a,b) have proposed the PPM, and they have used the TPM and PPM for a permutation flowshop problem.

Since the $P_m \mid s_{ij}, r_j \mid C_{\max}, \sum T_j$ problem has never been studied, and as it is NP-hard in the strong sense, approximated methods are appropriate for solving it. First, the non dominated sorting genetic algorithm (NSGA-II) is applied and second a fuzzy logic controller coupled to the NSGA-II (FLC-NSGA-II) is developed for the first time in this paper to solve the studied problem. An integer mathematical model is also proposed as well as a special solution encoding proposed in a previous work Li et al. (2009). An exact method TPM is applied to demonstrate the efficiency of the proposed methods. Then the contributions of this work are to propose first a new mathematical model of the problem. Afterthat, the FLC-NSGA-II is used here for solving the problem for the first time. In addition to that, we have applied the TPM with a combination of the aggregation and the ϵ-constraint. This adoption ensures that the problem is always with a single objective optimization search.

The rest of the paper is organized as follows: section "Problem formulation" describes the considered problem and the mathematical model. In section "Resolution methods", we develop the NSGA-II, the FLC-NSGA-II and the TPM method. Computational results are shown in section "Computational results". Our conclusion and perspectives are given in section "Conclusion".

Problem formulation

Problem description and description

In this parallel machine scheduling problem, a set of n independent jobs should be scheduled on m identical parallel machines. All the machines are available from times zero, and only one job can be executed on the machine at the same time. Each job j has a processing time p_j, it must be processed after its release date r_j and should be completed before the due date d_j. The sequence dependent setup times s_{ij} is also required in this problem. It deals with the setup time when a machine switches the production from job i to job j (job i precedes immediately job j on the same machine). Without loss of generality, we have set $s_{jj} = 0$. We assume that $s_{0j} = 0$, it means that if job j is scheduled at the beginning on a machine, then no setup times is required.

In this problem, some assumptions must be respected: each machine can execute only one job at the same time; each job can be processed only once; each job cannot be interrupted during the processing (no preemption is allowed).

Two different objectives are considered in this work:

- The minimization of the maximum completion time $C_{\max}$, which means the duration of all the processes. The $C_{\max}$ is defined as $C_{\max} = \max C_j, \forall j = 1, \ldots, n$, where C_j is the completion time of job j.
- The minimization of the total tardiness $\sum T_j$ is defined as $\sum T_j = \sum_{j=1}^{n} T_j$, where T_j is the tardiness of job j, in which defined as $T_j = max(0, C_j - d_j)$. If the completion time C_j of job j is greater than its due date d_j, then this job is considered as tardy. Otherwise, there is no tardy and the tardiness T_j of job j is equal to 0.

We have proposed a binary mixed integer linear model for the studied problem. This model is based on the study of Jungwattanakit et al. (2008). They have proposed a mathematical formulation to solve a hybrid flowshop scheduling problem. Since this model is strictly linear, the CPLEX solver can be used here to solve the problem.

Mathematical model

The problem can be formulated as follows:

$$\text{Minimize} \left(C_{\max}, \sum_{j=1}^{n} T_j \right) \tag{1}$$

Subject to:

$$\sum_{k=1}^{m} \sum_{i=0, i\neq j}^{n} X_{kij} = 1 \quad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{k=1}^{m} \sum_{j=1, j\neq i}^{n+1} X_{kij} = 1 \quad \forall i = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{n} X_{k0j} = 1 \quad \forall k = 1, \ldots, m \tag{4}$$

$$\sum_{i=1}^{n} X_{ki(n+1)} = 1 \quad \forall k = 1, \ldots, m \tag{5}$$

$$X_{kjj} = 0 \tag{6}$$

$$C_j - C_j \geq s_{ij} + p_j + \left(\left(\sum_{k=1}^{m} X_{kij} \right) - 1 \right) M \quad \forall j = 1, \ldots, n \quad \forall i = 1, \ldots, n \quad \text{with} \quad C_0 = 0 \quad s_{ij} = 0 \quad i \neq j \tag{7}$$

$$C_j \geq r_j + p_j \quad \forall j = 1, \ldots, n \tag{8}$$

$$T_j \geq C_j - d_j \quad \forall j = 1, \ldots, n \tag{9}$$

$$T_j \geq 0 \quad \forall j = 1, \ldots, n \tag{10}$$

$$C_{\max} \geq C_j \quad \forall j = 1, \ldots, n \tag{11}$$

$$X_{kij} = 0 \quad \text{or} \quad 1 \quad \forall k = 1, \ldots, m \quad \forall i = 1, \ldots, n \quad \forall j = 1, \ldots, n \tag{12}$$

where:

- n: the number of jobs
- m: the number of the machines
- j, i: the index of job $j, i = 1, \ldots, n$
- k: the index of machine $k = 1, \ldots, m$
- r_j: the release of job j, $j = 1, \ldots, n$
- d_j: the due date of job j, $j = 1, \ldots, n$
- p_j: the processing time of job j, $j = 1, \ldots, n$
- s_{ij}: the sequence dependent setup times if job j is the immediate successor of the job i on the same machine
- X_{kij}: the decision variable equal to 1 when the job j is the immediate successor of the job i on the machine k, and 0 otherwise
- C_j: the completion time of job j, $j = 1, \ldots, n$
- T_j: the real tardiness of job j, $j = 1, \ldots, n$
- $C_{\max}$: the makespan
- $\sum_{j=1}^{n} T_j$: the total tardiness
- M: a great constant

Equation (1) presents the objective function. The aim is to minimize two different objectives: makespan and total tardiness. Constraints (2)–(7) ensure that the partial schedule on each machine is not infeasible. Constraints (2) and (3) ensure that each job is processed exactly only once. Constraints (4) and (5) guarantee that only one job can be processed at the first and the last position on each machine. Constraint (6) means that the job cannot be reprocessed before its finish. Constraints (7) and (8) compute the completion time of each job. Constraint (7) means that if job j is scheduled after job

j on the same machine, then job j cannot be started before the finish of job i. Job j is completed by at least the sum of completion time of job i and the setup time from i to j and the processing time of j, if job j is immediately scheduled after job i. The value of M is a very great constant, for example the sum of all jobs processing times and setup times. Constraints (8) ensures that each job must be processed after its release date. Constrains (9) and (10) calculate the tardiness of job j. Constraint (9) determines the value of lateness (Lj) and constraint (10) ensures that only the positive value of lateness can be considered as tardiness ($T_j = \max(0, C_j - d_j)$). Constraint (11) determines the makespan as the maximum of all jobs completion times. Constraint (12) defines the decision variable X_{kij}, it is equal to 1 when job j is immediately scheduled after job i on machine k, 0 otherwise.

Two pseudo jobs are proposed in this study. Job 0 is defined as the beginning of all processes. If any job is scheduled immediately after job 0, it means that this job is scheduled at the first position on the machine. For example, if we have X_{103} (a decision variable, presented in the next section) equal to 1, it means that job 3 is executed first on machine 1 because there is no job before job 1. On the other side, job $n + 1$ means the end of all process. If a job is scheduled immediately before job $n + 1$, it means that this job is finished at last on the machine.

Resolution methods

First, we propose the NSGA-II algorithm (Deb et al. 2000) and an improved NSGA-II based on a fuzzy logic controller (FLC-NSGA-II) to solve the studied problem. The two proposed algorithms are compared in section "Computational results", in order to demonstrate the superiority of FLC-NSGA-II. Then, we develop a two phase method (TPM) which is an exact method in order to identify the set of all non dominated solutions. The TPM is very efficient compared with the full enumeration method, and it is used here to prove the efficiency of FLC-NSGA-II.

NSGA-II

Deb et al. (2000) have proposed the second version of a multiobjective genetic algorithm called non-dominated sorting genetic algorithm (NSGA-II). This algorithm is based on the Pareto dominance relationship. Note that in the multiobjective optimization problem, there is no single optimal solution, but a set of solutions which are not dominated by each others.

In the NSGA-II algorithm, an initial population P_0 is first randomly generated. In each generation t, the following processes are executed. The population of children Q_t (all the offspring chromosomes) is created with the operations of evaluation, selection, crossover and mutation. After that, all the individuals from P_t and Q_t are ranked in different fronts. The non dominated front of level 1 is constituted and includes all the non dominated solutions. In order to find the solutions in the next front, the solutions of previous fronts are not considered. This process is repeated until all the solutions are ranked and assigned to several fronts. Then, the best solutions (in the best front and with the best value of the crowding distance) are chosen for the new population P_{t+1}. This generation is repeated until the stopping criterion is satisfied. The overall structure of the NSGA-II is presented in algorithm 1.

Algorithm 1: Structure of the NSGA-II

Generate the initial population P_0 of size N
Evaluate these solutions
Sort these solutions by non domination and crowding distance
WHILE *stopping criterion is not satisfied* **DO**
  Create the offspring population Q_t (with the operations of selection, crossover and mutation), evaluate all the solutions
  Compose the populations of parents and the children in $R_t = P_t \bigcup Q_t$
  Sort the solutions of R_t in different non dominated fronts F_i by the Pareto dominance
  $P_{t+1} = 0$
  i=1
  WHILE *|Pt+1|+|Fi|<N* ***DO***
    $P_{t+1} \leftarrow P_{t+1} \bigcup F_i$
    i=i+1
  END WHILE
  Rank the solutions of F_i by the crowding distance, add $N - |P_{t+1}|$ solutions in P_{t+1} by descending order of the crowding distance
END WHILE

There are four parameters to set in this algorithm. The population size and the number of generations define the computing time of the algorithm, the probabilities of crossover p_c and mutation p_m define the convergence and diversity of the obtained results. In order to define the best parameters, several tests are made with the combinations of different values with $p_c = 0.6, 0.7, 0.8, 0.9, 1.0$ and $p_m = 0.05, 0.1, 0.15, 0.2, 0.25$. The efficient values of parameters are chosen according to the Riise distance (see following section). Finally, we have set a great crossover probability equal to 0.9 and a small mutation probability equal to 0.1. The population size and the number of generations are set to 100.

Encoding strategy

In our previous work (Li et al. 2009), a special encoding of the same problem was presented. A chromosome with a 3

Table 1 An example of the chromosome

j	1	2	3	4	5
k	1	2	1	1	2
r	2	1	1	3	2

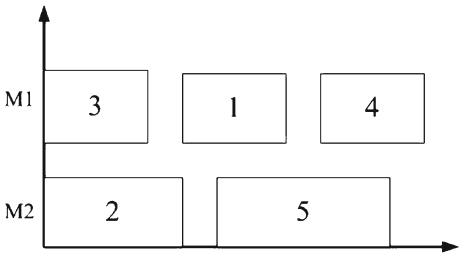

Fig. 1 The Gantt chart of example

lines matrix is proposed to encode the problem as shown in Table 1. In each gene (column), the three elements presented the index of jobs j, the index of machine k (job j is scheduled on machine k) and its position (job j is scheduled at position r on the machine k). An example of this chromosome is presented in Table 1, and the corresponding Gantt chart is shown in Fig. 1.

In the proposed mathematical model, a decision variable X_{kij} is considered. It is equal to 1 when job j is scheduled immediately after job i on machine k, and 0 otherwise. It is easy to count the machine index and the position of each job j with the decision variable values. The presented example in Fig. 1 has the following decision variables that are equal to 1: X_{103}, X_{131}, X_{114}, X_{146}, X_{202}, X_{225} and X_{256} (job 0 and job 6 are two pseudo tasks, the former means the beginning of the process and the latter presents its end). Two examples of the decision variable are to be explained. X_{103} is equal to 1 means that job 3 is executed first on machine 1 because there is no job before job 1 (its immediate former job is job 0 which means the beginning of the process). X_{131} is equal to 1 means that the job 1 is scheduled at position 2 on machine 1 (it follows immediately job 3, and job 3 is at position 1 on the same machine). After then, we can count that job 4 is at position 3 on machine 1, job 2 and job 5 are at position 1 and position 2, respectively, on machine 2. On the other side, jobs 4 and 5 are the last jobs scheduled on the relative machine because their succedent job is job 6 which presents the end of the scheduling process. This result has a good correspondence with the chromosome presented in Table 1.

In the proposed algorithm, all feasible solutions of the initial population are randomly generated. A random decimal value is generated by a uniform distribution [0, 1] for each job, and all jobs are ranked by the decreasing order of their values. After then, the *high priority order transition* policy is adopted, the job with the highest value is scheduled first on the machine which is first available.

Genetic algorithm operators

Crossover and mutation The crossover operation produces two new offspring chromosomes (children) from the parents. Some genes of the two parents are exchanged to create the new chromosome. The well known standard one point crossover is adopted in this work. A crossover point is randomly selected, the part of genes of the two parent chromosomes after the crossover point are exchanged to produce two offspring chromosomes.

Since there may be some errors for the created offspring chromosomes, a repair operation is required after the crossover. The errors may be in these following cases: two jobs are scheduled on the same machine and in the same position, several machines are empty (there is no job scheduled on the machine), the sequence of jobs on the machine is not consecutive. These are not allowed. Therefore, a repair operation is needed to modify these cases.

The mutation operation presented some variations in the offspring chromosomes. Two genes of the chromosome are randomly selected, and be interchanged.

Selection Selection is a process which decides about the choice of chromosomes for the crossover operation depending on the fitness function value or the rank value. In this work, the tournament parent selection is adopted. This selection is one of many selection methods in GA which randomly chooses one few solutions from the parent population and selects the winner (based first on the non-dominated front, and then on the value of the crowding distance) for crossover.

Stopping criteria There are many stopping conditions adopted for GA. In this study, the stopping criterion chosen is a given number of generations.

FLC-NSGA-II

Usually, a genetic algorithm uses two constant parameters: the crossover probability p_c and the mutation probability p_m. Since the setting of the two probabilities is very difficult, a fuzzy logic controller (FLC) is proposed. In this study, the average objective values of the solutions in the non-dominated front and their diversity are considered as the inputs of the fuzzy logic controller. The changes of the two probabilities Δp_c and Δp_m are guided by the output of the FLC. Appropriate crossover and mutation probabilities can produce better results, by avoiding premature convergence and falling into local optimum. This enhancement is carried out every ten consecutive generations (Lau et al. 2009a), in order to provide sufficient required time for the modification in the algorithm.

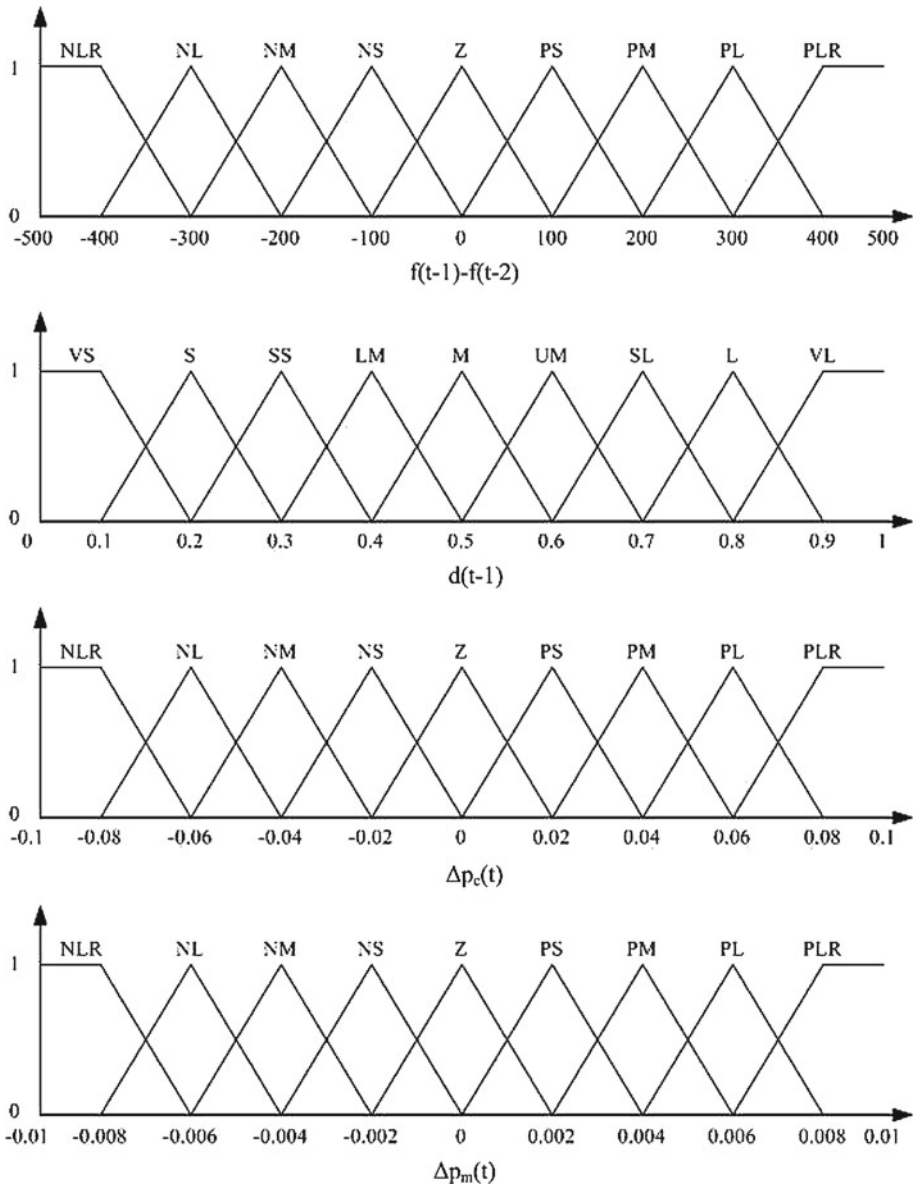

Fig. 2 Membership functions

Fuzzy logic

Fuzzy logic was firstly presented by Zadeh (1965), as a powerful and useful tool applied in many area. A FLC consists of 3 parts: *fuzzification*, *decision making*, and *defuzzification*. We have used here a similar FLC of Lau et al. (2009a) to guide the NSGA-II, with a difference in the parameters of the membership functions. The details are described below.

Fuzzification The membership functions are in order to associate the system input and output values with the fuzzy input and output membership values. In this study, the values of $f_a(t) = f(t-1) - f(t-2)$ and $d(t-1)$ are considered as the system input values where $f(t-1)$ and $f(t-2)$ are the average values of the objective function and $d(t-1)$ is the sum of the hamming distance between each individuals of the entire population. Then $\Delta p_c(t)$ and $\Delta p_m(t)$ are considered as the system output values. The membership functions are presented in Fig. 2. They are usually chosen based on the expert knowledge and experience. The triangular membership functions are used in this study, and the parameters are defined according to several tests. The meaning of each linguistic term is presented in Table 2.

Decision making When the system input values are transferred into the fuzzy input by the membership functions, a number of IF-THEN rules are applied to get the output membership values. These rules are based on the expert knowledge and experience. We have chosen the decision tables of Lau et al. (2009a) in our work, they are presented in Tables 3 and 4.

Defuzzification The aim of the defuzzification is to calculate the deviation values of the crossover and mutation probabilities by the fuzzy output membership values and the

Table 2 Meanings of linguistic terms

Linguistic terms for $f_a(t)$, $\Delta p_c(t)$, $\Delta p_m(t)$	Meaning	Linguistic terms for $d(t-1)$	Meaning
NLR	Negative larger	VS	Very small
NL	Negative large	S	Small
NM	Negative medium	SS	Slightly small
NS	Negative small	LM	Lower medium
Z	Zero	M	Medium
PS	Positive small	UM	Upper medium
PM	Positive medium	SL	Slightly large
PL	Positive large	L	Large
PLR	Positive larger	VL	Very large

membership functions of Δp_c and Δp_m. There are many defuzzification methods such as AI (adaptive integration), COA (center of area), CDD (constraint decision defuzzification) and COG (center of gravity). The COG defuzzification method is adopted here, as it is the best known defuzzification operator (Leekwijck and Kerre 1997). This method computes the gravity center of the area under the membership function.

Example An example is provided here to explain the three steps of the fuzzy logic. In this example, $f(t-1)-f(t-2) = -330$, which means that the difference of average values of makespan and total tardiness over the entire population at iteration $t-1$ and $t-2$ is negative and equal to 330. We have therefore NLR with a proportion of 0.3 and NL with a proportion of 0.7 by the membership functions of $f(t-1)-f(t-2)$ (see Fig. 2). The value of $d(t-1)$ is equal to 0.25 which also provides the inputs of S with 0.5 and SS with 0.5. These steps are shown in Fig. 3.

After obtaining the input membership values, the decision tables are used to get the output membership values. We only explain the computation of Δp_c here, which is guided by the decision table of $\Delta p_c(t)$ (Table 3). The same steps are applied for Δp_m. There exists four possibilities: NLR(0.3) and S(0.5), NLR(0.3) and SS(0.5), NL(0.7) and S(0.5), NL(0.7) and SS(0.5). For the first case, we can get that the combination of NLR and S provides PS. Moreover the minimum value between two proportion values (0.3 and 0.5) is kept. Then, we have obtained PS(0.3). We can also obtain PS(0.3), Z(0.5) and PS(0.5) for the 3 remaining cases.

The final step is the defuzzification by using the membership fuction of Fig. 4. We should compute the sum of PS(0.3), PS(0.3), Z(0.5) and PS(0.5). The COG (center of gravity) method is adopted. We have obtained the value

Table 3 Decision table of $\Delta p_c(t)$

$d(t-1) \setminus f_a(t)$	NLR	NL	NM	NS	Z	PS	PM	PL	PLR
VL	PLR	PLR	PL	PL	PM	PM	PS	PS	Z
L	PLR	PL	PL	PM	PM	PS	PS	Z	NS
SL	PL	PL	PM	PM	PS	PS	Z	NS	NS
UM	PL	PM	PM	PS	PS	Z	NS	NS	NM
M	PM	PM	PS	PS	Z	NS	NS	NM	NM
LM	PM	PS	PS	Z	NS	NS	NM	NM	NL
SS	PS	PS	Z	NS	NS	NM	NM	NL	NL
S	PS	Z	NS	NS	NM	NM	NL	NL	NLR
VS	Z	NS	NS	NM	NM	NL	NL	NLR	NLR

Table 4 Decision table of $\Delta p_m(t)$

$d(t-1) \setminus f_a(t)$	NLR	NL	NM	NS	Z	PS	PM	PL	PLR
VL	NLR	NLR	NL	NL	NM	NM	NS	NS	Z
L	NLR	NL	NL	NM	NM	NS	NS	Z	PS
SL	NL	NL	NM	NM	NS	NS	Z	PS	PS
UM	NL	NM	NM	NS	NS	Z	PS	PS	PM
M	NM	NM	NS	NS	Z	PS	PS	PM	PM
LM	NM	NS	NS	Z	PS	PS	PM	PM	PL
SS	NS	NS	Z	PS	PS	PM	PM	PL	PL
S	NS	Z	PS	PS	PM	PM	PL	PL	PLR
VS	Z	PS	PS	PM	PM	PL	PL	PLR	PLR

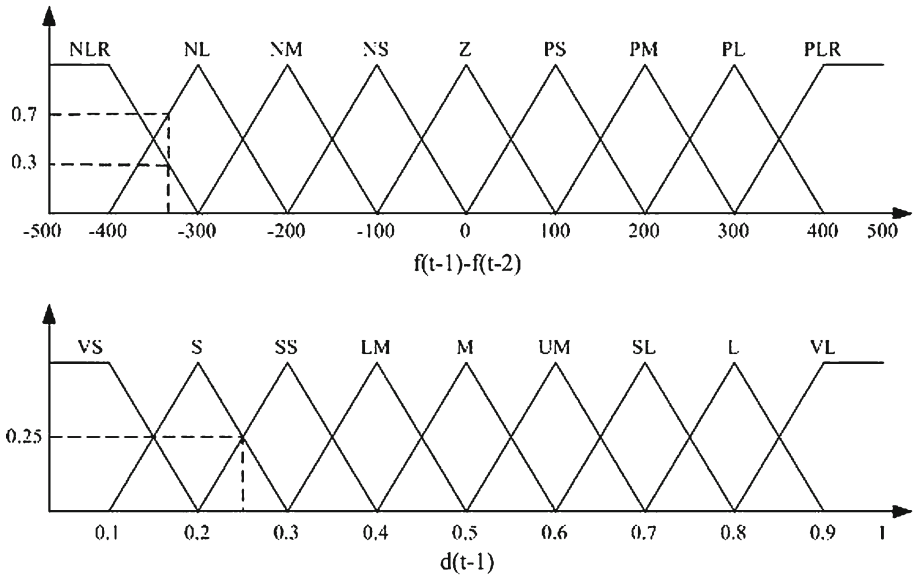

Fig. 3 Example of fuzzification

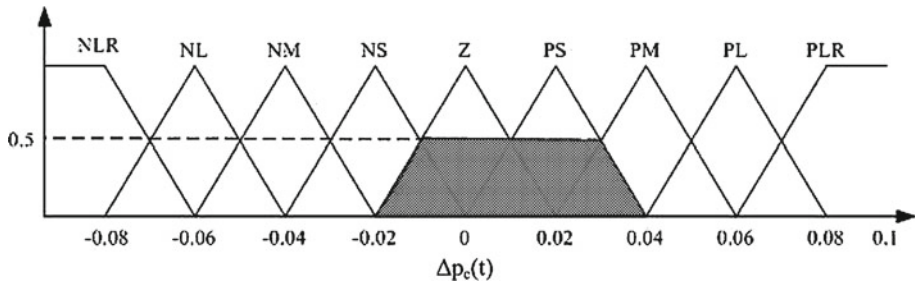

Fig. 4 Example of defuzzification

of Δp_c which is equal to 0.01. It means that if an initial value of the crossover probability at iteration $t-1$ is equal to 0.5, then the crossover probability is set to 0.51.

Finally, bounds have been set to the two probabilities: the p_c cannot be less than 0.5, and p_m must be less than 0.25.

Fuzzy logic guided NSGA-II

In order to improve the performances of the NSGA-II, a fuzzy logic controller (FLC) is proposed here to change the crossover and mutation probabilities each ten consecutive generations. The aim is to provide sufficient times for the NSGA-II to respond the changes. The average fitness value of the population and the population diversity are taken as the input of the fuzzy controller. In fact, appropriate crossover and mutation probabilities can provide good convergence and diversity of the obtained results. Since the studied problem is a multiobjective optimization problem, the average fitness value $f(t)$ is the average value of the makespan and the total tardiness of the population at iteration t. The value of $d(t)$ is the degree of population diversity at iteration t. It is the average of the difference of all pairs of chromosomes of all the solutions in the obtained population and it can be computed as shown equation 13.

$$d(t) = \frac{1}{\frac{N(N-1)}{2}} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{n} \frac{\delta(g_{ik}, g_{jk})}{n} \tag{13}$$

where N is the population size, the value of n is the chromosome length and g_{ik} is the values of three elements of the kth gene of the ith chromosome (Lau et al. 2009a), and $\delta(g_{ik}, g_{jk}) = 1$ if the two genes are not the same, 0 otherwise.

The same parameters of the NSGA-II are used for setting the FLC-NSGA-II. The population size and the number of generations are fixed at 100. The initial crossover probability is 0.9, and the initial mutation probability is 0.1.

The structure of the FLC-NSGA-II is shown in Algorithm 2.

Two phase method

In this section, an exact method is applied in order to identify all the Pareto optimal solutions. There are two types of solutions in the Pareto front: supported solutions and non-supported solutions. The supported solutions are these located in the convex hull of the Pareto front, and which can be find by linear aggregation methods. The other solutions are also Pareto optimal solutions but not located in the convex hull. Hence, we can obtain a set of all the supported

Algorithm 1: Structure of the NSGA-II

Generate the initial population P_0 of size N
Evaluate these solutions
Sort these solutions by non-dominated front and crowding distance
WHILE *stopping criterion is not satisfied* **DO**
Create the offspring population Q_t (with the operations of selection, crossover and mutation), evaluate all the solutions
Compose the populations of parents and the children in $R_t = P_t \bigcup Q_t$
Sort the solutions of R_t in different non dominated fronts F_i by the Pareto dominance
$P_{t+1} = 0$
i=1
WHILE *|Pt+1|+|Fi|<N* ***DO***
$P_{t+1} \leftarrow P_{t+1} \bigcup F_i$
i=i+1
END WHILE
Rank the solutions of F_i by the crowding distance, add $N-|P_{t+1}|$ solutions in P_{t+1} by descending order of the crowding distance
IF *t mod 10=0* ***THEN***
The values of f(t-1)-f(t-2) and d(t-1) are considered as the input of FLC, update the p_c and p_m with the FLC
END IF
END WHILE

solutions by different aggregations, but it is not possible for non-supported solutions. A well known exact method TPM (two-phase method) is adopted here for solving the bi-objective scheduling problem. This method was first proposed by Ulungu and Teghem (1995). In the first phase of this method, the aggregation method is applied to enumerate all supported solutions. After that, a limitation is applied in the second phase to find the non-supported solutions on the triangle areas which are constituted by each two adjacent supported solutions.

The ϵ-constraint method is used in the second step. Therefore, the TPM in our works is a combination of two exact methods: the aggregation and the ϵ-constraint. Hence, the search is always transferred to a single objective optimization. The CPLEX solver is used in our work. The details of this method are described in the following.

First phase

The first phase of TPM aims to find all the supported solutions with an aggregation of the two objectives: $f' = \lambda_1 f_1 + \lambda_2 f_2$. The two objectives f_1 and f_2 are the makespan and the total tardiness, respectively. It starts by searching the extreme solutions. A single objective is first considered: the makespan is to be minimized concerning one extreme solution (solution R in Fig. 5). In this case, we have chosen the two coefficients with $\lambda_1 = 1$ and $\lambda_2 = 0$($\lambda_1 = 0$ and $\lambda_2 = 1$ for minimizing the total tardiness). The search in the horizontal direction is to minimize the makespan, and the search in the vertical direction is for minimizing the total tardiness (see Fig. 5). R is the extreme solution which has the best value of the makespan, and S is another extreme with the best value of the total tardiness. However this is not appropriated for our studied problem, so an improvement has been adopted in this work.

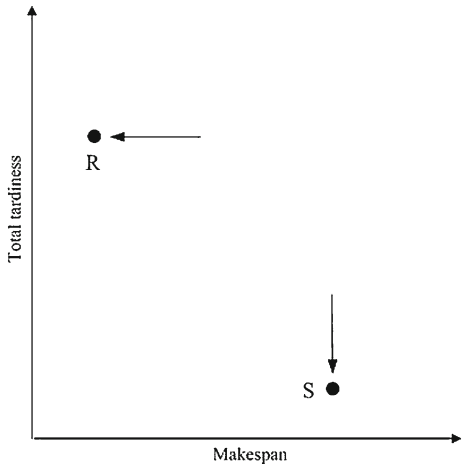

Fig. 5 The search for extreme solutions with the TPM

In fact, in a scheduling problem, several solutions may give the same value for one objective. Based on that, the minimization of only one criterion cannot guarantee finding the real extreme solution (maybe another solution has the same makespan or total tardiness, but the value of another objective is larger than that of the extreme solution). Lemesre et al. (2007a) have proposed to find the extreme solutions in a lexicographical way. For each extreme solution, two searches have been done. In this work, a search with the direction which is almost horizontal (or a vertical direction for minimizing the total tardiness) is applied to find the extreme solution of the makaspan with only one search (Przybylski 2006). For example in Fig. 6, R is the real extreme solution with the best objective value of the makespan. There are however several solutions above R. Since they have the same value of the makespan, the horizontal search cannot guarantee the finding R. Hence, we have set the two coefficients as $\lambda_1 = T$ and $\lambda_2 = 1$. If the value of T is enough great, it ensures that there are no Pareto optimal solutions in the triangle which is presented in Fig. 6. T must be more large than the value of $f_2(R)$, we have set T as the sum of the processing times in this study. This search is almost parallel with the horizontal line. On the other side, a search with the almost vertical direction is applied while searching for S.

Once the two extreme solutions are found, different aggregations $f' = \lambda_1 f_1 + \lambda_2 f_2$ are applied to explore the supported solutions between the given supported solutions. The search starts between two extreme solutions (they are also supported solutions). The search direction will be orthogonal to the line connecting R and S. The two coefficients are fixed to $\lambda_1 = f_2(R) - f_2(S)$ and $\lambda_2 = f_1(S) - f_1(R)$ (see Fig. 7). This search will find a supported solution with the

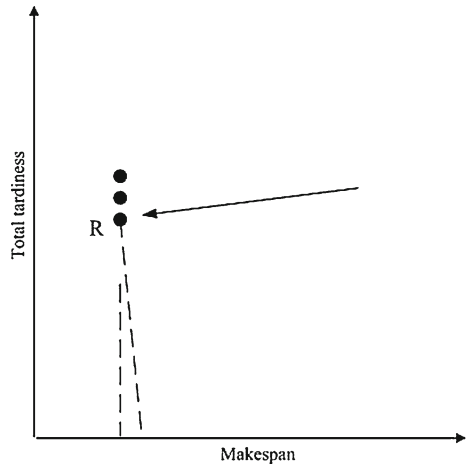

Fig. 6 The improved search for the extreme solutions

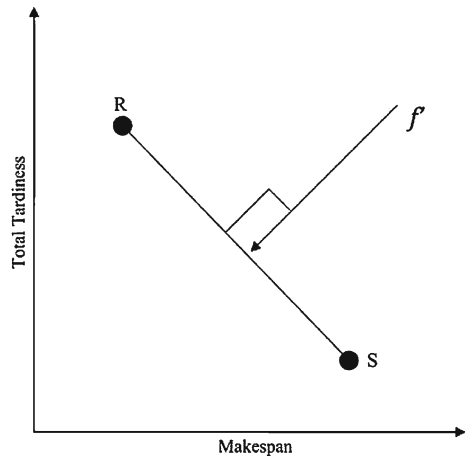

Fig. 7 Search between two extreme solutions

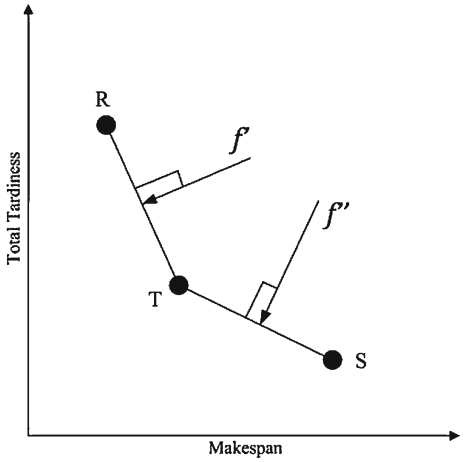

Fig. 8 New searchs

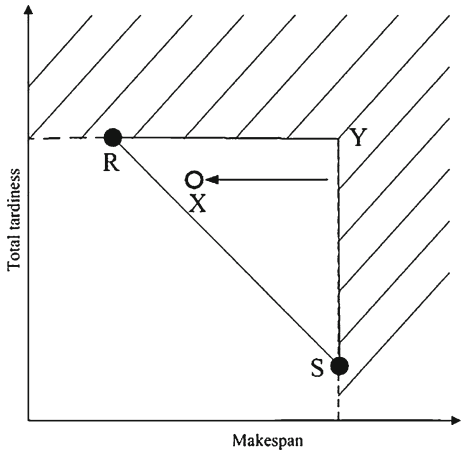

Fig. 9 Search for non supported solutions

minimum value of $\lambda_1 f_1 + \lambda_2 f_2$, if there exists at least one supported solution T between R and S. Two new searches are to be executed between R and T and between T and S (see Fig. 8). Hence, the search will be finished when no new supported solutions are found.

Second phase

After the searching of all the supported solutions, the second phase of TPM consists of exploring the non supported solutions. The ϵ-constraint method in the horizontal direction is here applied. The aim is to explore the non supported solutions with a single objective, so we can execute the search with CPLEX solver. As shown in Fig. 9, a search for a non supported solution X (if it exists on the triangle which is constituted by R, Y and S (where Y is the point with the positions of $f_1(S)$ and $f_2(R)$). This search is similar to the search of the extreme solution which has the best value of the makespan, but several constraints are added. It can be presented as follows: $\min f_1(X)$, with $f_1(R) < f_1(X) < f_1(S)$, $f_2(S) < f_2(X) < f_2(R)$. If a non-supported is explored, a new search will be started on the zone UQSY (see Fig. 10, U is a non dominated solutions on the triangle, Y is the point with $f_1(S)$ and $f_2(U)$ and Q is the vertical projection of U on the line connecting R and S). This means that the new search is under the following constraints: $f_1(U) < f_1(X) < f_1(S)$, $f_2(S) < f_2(X) < f_2(U)$. This operation is repeated until no new non supported solution is found.

Computational results

Definition of the protocol test

Since there are not existent instances for the studied problem in the literature, the job data are randomly generated by a protocol test. Two types of instances are tested here: problems

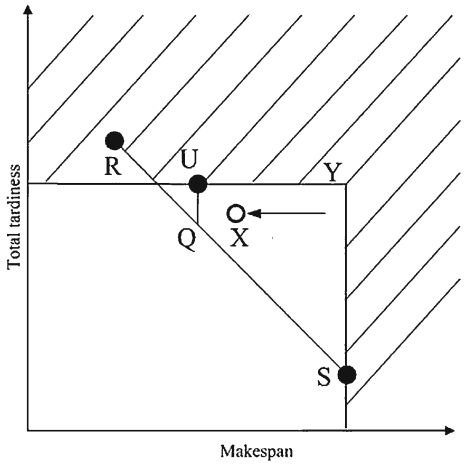

Fig. 10 New search for non supported solutions

Table 5 Index of configurations

β, [$W1$, $W2$]	TF,RDD			
	0.2,0.2	0.2,0.4	0.4,0.2	0.4,0.4
0.6, [0.1, 0.2]	1	2	3	4
0.8, [0.1, 0.2]	5	6	7	8
0.6, [0.1, 0.5]	9	10	11	12
0.8, [0.1, 0.5]	13	14	15	16

with $n = 20$ jobs and $m = 3$ machines and those with $n = 50$ jobs and $m = 5$ machines.

The following rules are carried out to generate the job data (Nessah et al. 2005; Yalaoui and Chu 2002). The processing times p_j of job j is generated between a uniform distribution [1, 100]. The sequence dependent setup times of job j are chosen with regards to its processing times, after the generation of the processing times p_j, the setup times s_{ij} are set to. The value of α is a coefficient randomly generated in $[W1, W2]$, where $[W1, W2] \in \{[0.1, 0.2], [0.1, 0.5]\}$. The release dates are generated between $[0, 50.5 \times n \times \beta/m]$. The parameter β has 2 changed values which are set to 0.6 and 0.8 (Yalaoui and Chu 2002). The due dates of jobs are in the interval $[max(0, P \times (1 - TF - RDD/2)), P \times (1 - TF + RDD/2)]$, where $P = \sum_{j=1}^{n} p_j/m$; and we have set the two parameters TF and RDD (Yalaoui and Chu 2002) to 0.2 and 0.4, respectively.

There are four parameters to set and two choices for each parameter. Therefore, 16 different configurations have been tested in this work. Table 5 shows the index of each configuration. We have tested two different problems: 20 and 50 jobs should be scheduled on 3 and 5 machines, respectively. The exact method cannot be applied on large size problems because of too long computational times. Therefore, the exact method is applied for the problems with maximum 10 jobs.

Measuring criteria

The comparison of two different Pareto fronts is difficult, since each front is not a single scalar value, but a set of non-dominated solutions. An approximated method involves the following objectives Zitzler and Thiele (1999): the distance to the absolute Pareto optimal front is to be minimized, the diversity of the generated solutions is to be maximized and the spread of the obtained front is to be maximized.

Recently, several measuring criteria have been proposed for comparing two non dominated front. Ranjithan et al. (2001) have proposed a spread metric which determines the maximum range represented by the non-dominated solutions in the objective space, and a coverage metric that characterizes the distribution of solutions. Kumar and Ranjithan (2002) have proposed a D factor that represents the degree of dominance of non dominated solutions from two obtained fronts which are produced by two different algorithms. VanVeldhuizen (1999) has proposed an error ratio criterion and a generational distance (GD). The former one represents the proportion of non true Pareto solutions in the obtained front. The latter one measures general progress to the reference set. Zitzler (2001) have presented the Zitzler measure. The μ distance is presented by Riise (2002) while Dugardin et al. (2010) have proposed a new measure based on the μ distance. The three latter measures are adopted in our study, because they do not require to compute the absolute Pareto optimal front (reference set) while other measures need to. The number of non dominated solutions n_s of the Pareto front is also considered in our work.

Let A and B be two Pareto fronts obtained by different methods. n_A and n_B are the number of non dominated solutions in front A and B, respectively. The μ distance is defined as, where d_i is the distance between a solution i of A and its orthogonal projection on B. The value of d_i is negative, when solution i is below the front B (or the extrapolated front). The μ distance shows if front A is closer from the optimal solutions than front B. The more negative the value of μ is, the better quality of front A would be.

On the base of μ distance, Dugardin et al. (2010) have developed a new measure μ_d distance. The new measure is defined as $\mu_d = \frac{\mu}{D}$, where $D = \sqrt{(f_{1\max} - f_{1\min})^2) + (f_{1\max} - f_{1\min})^2}$ is the cross of the rectangle which are constituted by the fronts A and B, μ is the value of μ distance. This measure presents the proportional improvement to the diagonal of the rectangle.

The Zitzler measure was presented by Zitzler (2001), where C1 measures the ratio of solutions in A which are dominated by at least one solution in B. The equation is defined by $C1 = \frac{|\{a \in A | \exists b \in B : b \succ a\}|}{|A|}$, where a and b are the non-dominated solutions in the set A and B, respectively. When $C1 = 1$, it means that all of the solutions in A are dominated by solutions in B when $C1 = 0$, it means that

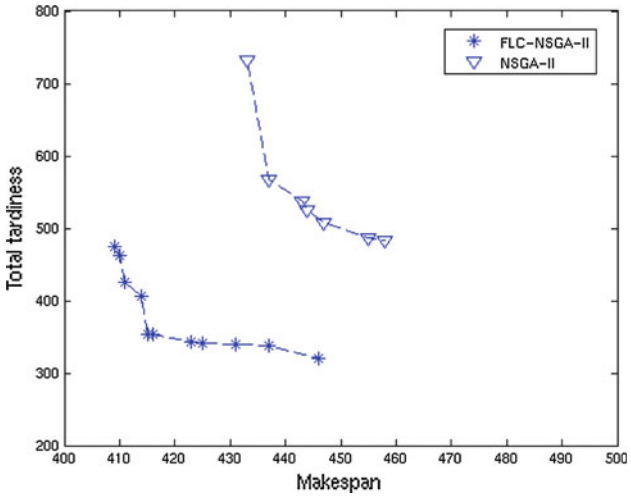

Fig. 11 An example of FLC-NSGA-II and NSGA-II front

Table 6 The solutions objective values of the considered example

Index	FLC-NSGA-II		NSGA-II	
	$C_{\max}$	$\sum T_j$	$C_{\max}$	$\sum T_j$
1	409	475	433	732
2	410	463	437	568
3	411	425	443	537
4	414	406	444	525
5	415	354	447	508
6	416	353	455	487
7	423	343	458	484
8	425	341		
9	431	340		
10	437	338		
11	446	320		

there are no solutions in A dominated by at least one solution in B. On the contrary, $C2$ measures the ratio of solutions in B which are dominated by at least one solution in A.

Comparison of FLC-NSGA-II and NSGA-II

First, we have compared the FLC-NSGA-II algorithm with the classical NSGA-II. An example is shown in Fig. 11 where 20 jobs should be scheduled on 3 machines. Table 6 shows the objective values of the obtained solutions of the two algorithms. We can observe that the obtained front of FLC-NSGA-II dominates the other front of NSGA-II. There are 11 non dominated solutions in the FLC-NSGA-II front against 7 non dominated solutions in the NSGA-II front. The value of the μ distance is equal to -101.29, and μ_d is -0.244. These two values are negative because the FLC-NSGA-II front dominates the NSGA-II front. Furthermore, we have $C1 = 0$ which means that that there is no solution in the FLC-NSGA-II front dominated by any solution from NSGA-II, and $C2 = 1$ means that all the solutions in the NSGA-II front are dominated by at least one solution in the FLC-NSGA-II front.

Tables 7 and 8 show the comparison results of the FLC-NSGA-II and the NSGA-II algorithms for the two studied problems. The index in both tables stands for the configuration number among the 16 configurations that are tested. For each configuration, we have tested 30 instances. The mean value of μ and μ_d, the best values and the worst values of μ and μ_d (best μ, best μ_d, worst μ and worst μ_d), the mean values of $C1$ and $C2$, the mean values of the number of non dominated solutions of two obtained front (n_{s1} presents the non-dominated solutions number in the FLC-NSGA-II front, n_{s2} for NSGA-II front) are presented. In this work, 960 instances are therefore tested for the comparison of FLC-NSGA-II and NSGA-II.

The first line of Table 7 is explained. This problem is defined with 20 jobs and 3 machines. Index 1 means that the jobs data are randomly generated with $\beta = 0.6$, $[A, B] = [0.1, 0.2]$, RDD $= 0.2$ and TF $= 0.2$ (see Table 2). The next elements show the comparison results. Since each configuration is tested for 30 instances, the values of non dominated solutions show that the FLC-NSGA-II has obtained more non dominated solutions than NSGA-II with respective mean values of 5.5 and 4.4. The mean value of the Riise distance $\mu = -100.877$ indicates that the obtained front of FLC-NSGA-II dominates the NSGA-II front, since this value is negative. The best (maximum) and worst (minimum) values of μ are both negative which means that the Riise distance is always negative during the 30 tests. The mean value, best value and worst value of μ_d proved the same decision. The results show that $C1 = 0.007$, i.e. there are few solutions provided by FLC-NSGA-II that are dominated by at least one solution from NSGA-II. On the other side, $C2 = 0.989$ means that almost all the solutions provided by NSGA-II are dominated by at least one solution from FLC-NSGA-II.

In Tables 7 and 8, the values of μ and μ_d are always negative which means that the FLC-NSGA-II front always dominates the NSGA-II front. The absolute value of the μ distance is not high for small size problems. However, we can observe that the absolute values of the μ distance for large size problems are larger than those for small size problems. That means that for small problems, the two fronts of FLC-NSGA-II and NSGA-II are both very close to the absolute Pareto front (the efficiency of FLC-NSGA-II is demonstrated in the following section). So the difference between the two fronts is not very significant. When solving complex problems, the advantage of FLC-NSGA-II is obvious. The FLC-NSGA-II front is under the NSGA-II front and the distance between the two fronts is larger than the first case. The

Table 7 Comparison of FLC-NSGA-II/NSGA-II for 20 jobs on 3 machines

Index	μ	Best μ	Worst μ	μ_d	Best μ_d	Worst μ_d	C1	C2	n_{s1}	n_{s2}
1	−100.877	−193.511	−2.702	−0.329	−0.820	−0.013	0.007	0.989	5.5	4.4
2	−88.590	−199.318	−2.554	−0.235	−0.512	−0.009	0	1	7.9	4.7
3	65.210	−178.797	−5.053	−0.199	−0.516	−0.030	0	1	6.3	4.0
4	−77.687	−189.469	−5.804	−0.186	−0.463	−0.027	0	1	7.7	5.5
5	−55.337	−128.356	−10.347	−0.191	−0.428	−0.030	0	1	6.0	4.8
6	−71.735	−198.721	−12.715	−0.175	−0.501	−0.032	0	0.993	6.1	5.6
7	−61.711	−150.897	−16.559	−0.128	−0.328	−0.025	0	1	5.1	3.7
8	−87.772	−193.046	−24.712	−0.158	−0.392	−0.024	0	1	6.8	4.3
9	−86.665	−175.757	−9.060	−0.285	−0.678	−0.059	0	0.987	4.2	4.6
10	−95.537	−190.584	−20.007	−0.215	−0.502	−0.032	0	1	6.1	5.5
11	−94.129	−175.857	−4.375	−0.218	−0.473	−0.026	0	1	6.3	4.7
12	−92.702	−199.488	−24.222	−0.199	−0.611	−0.030	0	1	5.3	3.5
13	−69.579	−162.163	−1.803	−0.262	−0.614	−0.003	0	1	5.0	3.9
14	−59.691	−123.098	−15.793	−0.155	−0.311	−0.029	0	1	6.5	4.9
15	−107.757	−188.572	−20.305	−0.244	−0.488	−0.035	0	1	6.4	5.7
16	−80.604	−198.179	−19.245	−0.164	−0.405	−0.040	0	1	6.6	4.1

Table 8 Comparison of FLC-NSGA-II/NSGA-II for 50 jobs on 5 machines

Index	μ	Best μ	Worst μ	μ_d	Best μ_d	Worst μ_d	C1	C2	n_{s1}	n_{s2}
1	−181.302	−350.982	−66.861	−0.123	−0.239	−0.044	0	1	9.0	4.5
2	−167.560	−376.556	−33.453	−0.116	−0.247	−0.018	0	1	6.8	4.6
3	−187.809	−396.978	−20.839	−0.112	−0.444	−0.013	0	1	8.3	5.2
4	−174.049	−390.042	−59.603	−0.082	−0.197	−0.025	0	1	7.7	5.7
5	−96.999	−218.605	−32.830	−0.058	−0.105	−0.019	0	1	5.8	3.5
6	−160.750	−377.596	−21.816	−0.098	−0.331	−0.016	0	1	6.2	4.5
7	−170.889	−305.597	−36.694	−0.070	−0.123	−0.021	0	1	7.0	5.1
8	−136.165	−381.779	−31.590	−0.069	−0.264	−0.014	0	1	7.5	4.5
9	−162.439	−321.890	−17.314	−0.116	−0.262	−0.035	0	1	7.0	4.6
10	−192.545	−378.492	−47.432	−0.122	−0.286	−0.038	0	1	6.7	4.5
11	−174.997	−349.638	−60.716	−0.094	−0.221	−0.024	0	1	7.8	4.8
12	−184.703	−306.552	−44.591	−0.084	−0.132	−0.030	0	1	7.6	5.0
13	−197.049	−363.772	−86.951	−0.079	−0.145	−0.028	0	1	7.8	5.9
14	−208.787	−342.780	−70.282	−0.086	−0.166	−0.038	0	1	8.1	4.8
15	−234.965	−345.660	−119.208	−0.081	−0.133	−0.029	0	1	7.4	5.1
16	−192.109	−349.358	−34.528	−0.092	−0.225	−0.013	0	1	7.2	4.4

FLC-NSGA-II has obtained more non-dominated solutions than NSGA-II. The value of $C1$ is always less than $C2$. In fact, the values of $C1$ are equal to 0, while those of $C2$ are equal to 1. Based on the 960 tested instances, the comparison between the different numerical results show the advantage and the efficiency of FLC-NSGA-II.

Comparison of the TPM and the full enumeration

In order to prove the advantages of the proposed exact method (TPM), it has been compared with a full enumeration method. The aim of the latter is to enumerate all the feasible solutions and find the absolute Pareto optimal solutions

Table 9 The computing times of the two exact methods

Problem (n, m)	Full enumeration	TPM
(5, 2)	0.047s	0.15s
(6, 2)	1.015s	1.47s
(7, 2)	153s	13.97s
(8, 2)	38004s	98s
(9, 2)	–	933s
(10, 2)	–	4186s

Table 10 FLC-NSGA-II/TPM comparison

Problem (n, m)	n_s1	n_s2	μ	C1	C2	CT1	CT2
(5, 2)	5	5	0	0	0	0.15s	0.688s
(6, 2)	4	4	0	0	0	1.47s	0.743s
(7, 2)	3	3	0	0	0	13.97s	0.688s
(8, 2)	4	4	0	0	0	98s	0.766s
(9, 2)	4	4	0	0	0	933s	0.672s
(10, 2)	5	5	0	0	0	4186s	0.750s

among them. This method can be applied with maximum 8 jobs for the studied problem, while the proposed TPM can solve this problem with maximum 10 jobs. Moreover, the two phase method requires less computing times. The computational results are shown in Table 9.

Comparison of the FLC-NSGA-II and the TPM

In this section, the FLC-NSGA-II is compared with the two phase method (TPM). The latter method is provided to obtain the absolute Pareto optimal solutions. Since too long computing times are required for the exact method, the comparisons are executed on the small instances (maximum 10 jobs). The number of generations of FLC-NSGA-II is fixed at 200. We can observe that this algorithm has obtained the whole Pareto optimal solutions as it is shown by the values of the measuring criteria in Table 10 (μ, $C1$ and $C2$). Furthermore, the CT1 and CT2 present, respectively, the computing times of FLC-NSGA-II and TPM algorithms. As we may notice, the FLC-NSGA-II requires less computing times than the TPM. The difference between the two computing times is much more obvious starting by the problems with 7 jobs where the CT1 is always less than 1 s.

Conclusion

In this work, an identical parallel machine scheduling problem with release dates, due dates and sequence dependant setup times is considered. The goal is to minimize the makespan and the total tardiness. The contribution of this paper is first to develop a new mathematical model and to develope two metaheuristics which are the NSGA-II and a fuzzy logic guided NSGA-II (FLC-NSGA-II). The computational results show the advantage of FLC-NSGA-II over the NSGA-II on the 960 tested problems based on the adopted measuring criteria. The second contribution is the adoption of the TPM (two phase method) to solve the problem. The FLC-NSGA-II is compared with the TPM method for the small size problems. We can observe that FLC-NSGA-II is able to obtain the absolute optimal solutions. The TPM method can solve the problems with maximum 10 jobs.

For our further works, it is interested to develop a hybrid metaheuristic which is a combination of the proposed approximated method and an exact method. Since the results of approximated method are very close to the Pareto optimal solutions, a search with ϵ-constraints method will be explored which is based on these results.

References

Alidaee, B., & Rosa, D. (1997). Scheduling parallel machines to minimize the total weighted and un-weighted tardiness. *Computers and Operations Research, 24*(4), 775–788.

Azizoglu, M., & Kirca, O. (1998). Tardiness minimization on parallel machines. *International Journal of Production Economics, 55*, 163–168.

Baker, K. R., & Bertrand, J. W. (1982). A dynamic priority rule for sequencing against due-date. *Journal of Operational Management, 3*, 37–42.

Bouibede-Hocine, K., T'kindt, V., & Tran, D. (2006). Two evolutionary algorithms for a uniform parallel machines. In *7th International Conference on Multi-Objective Programming and Goal Programming, Loire valley, France*.

Brucker, P. (1998). *Scheduling algorithm*. Berlin: Springer.

Chang, P. C., Chen, S. H., & Lin, K. L. (2005). Two-phase sub population genetic algorithm for parallel machine-scheduling problem. *Expert Systems with Applications, 29*(3), 705–712.

Chang, P. C., Chen, S. H., & Hsieh, J. C. (2006). A global archive sub-population genetic algorithm with adaptive strategy in multi-objective parallel-machine scheduling problem. In *Proceedings of International Conference on Natural Computation* (pp. 730–739)

Cheng, T., & Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research, 47*, 271–292.

Cochran, J. K., Horng, S. M., & Fowler, J. W. (2003). A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers and Operations Reaserch, 30*, 1087–1102.

Corne, D. W., Knowles, J. D., & Oates, M. J. (2000). The Pareto envelope based selection algorithm for multiobjective optimization. In *Proceedings of the Parallel Problem Solving from Nature VI Conference* (pp. 839–848).

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective

optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference* (pp. 849–858).

Dugardin, F., Chehade, H., Amodeo, L., Yalaoui, F., & Prins, C. (2007). Hybrid Job Shop and parallel machine scheduling problems: minimization of total tardiness criterion. In E. Levner (Ed.), *Multiprocessor scheduling: theory and applications* (pp. 436), ISBN 978-3-902613-02-8.

Dugardin, F., Yalaoui, F., & Amodeo, L. (2010). New multi-objective method to solve re-entrant hybrid flow shop scheduling problem. *European Journal of Operation Research, 203*, 22–31.

Eren, T. (2008). A bicriteria parallel machine scheduling with a learning effect of setup and removal times. *Applied Mathematical Modeling, Information Systems and Operational Research, 45*, 75–81.

Fonseça, C. M. & Fleming, P. J. (1993) Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California* (pp. 416–423).

Gendreau, M., Laporte, G., & Guimaraes, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Reasearch, 133*, 183–189.

Gharbi, A., & Haouari, M. (2002). Minimizing makespan on parallel machines subject to release dates and delivery times. *Journal of Scheduling, 5*, 329–355.

Haimes, Y., Ladson, L., & Wismer, D. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transaction on System, 1*, 296–297.

Haimes, Y., Hall, W., & Freedman, H. (1975). *Multiobjective optimization in water resource systems: the surrogate worth trade off method*. New Yaok: Elsevier.

Hajela, P., & Lin, C. Y. (1992). Genetic Search Strategies in Multicriterion Optimal Design. *Structural Optimization, 4*, 99–107.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation* (Vol. 1, pp. 82–87).

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology, 37*, 354–370.

King, R. T. F. A., Radha, B., & Rughooputh, H. C. S. (2004). A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration. In *Proceedings of 2004 IEEE international conference on networking, sensing and control* (pp. 577–582).

Knowles, J. D., & Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation, 8*(2), 149–172.

Koulamas, C. (1994). The total tardiness problem: review and extension. *Operations Research, 42*, 764–775.

Kumar, S. V. & Ranjithan, S. R. (2002). Evaluation of the Constraint Method-based Multiobjective Evolutionary Algorithm (CMEA) for a three objective optimization problem, In W.B. Langdon et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO 2002), Morgan Kaufmann, New York* (pp. 431–438).

Kursame, F. (1991) A variant of Evolution Strategies for Vector Optimization. In Schwefel, H.P. Manner, R. (Eds.), *Parallel problem solving from nature, 1st workshop, PPSNI*, (Vol. 496 of Lecture Notes in Computer Science, pp. 193–197).

Lau, H. C. W., Chan, T. M., Tsui, W. T., & HO, G. T. S. (2009). Cost optimization of the supply chain network using genetic algorithms. In *IEEE Transactions on Knowledge and Data Engineering,* (Vol. 99(1))

Lau, H. C. W., Chan, T. M., Tsui, W. T., Chan, F. T. S., Ho, G. T. S., & Choy, K. L. (2009). A fuzzy guided multi-objective evolutionay algorithm model for solving transportation problem. *Expert Systems with Applications, 36*, 8255–8268.

Leekwijck, W. V., & Kerre, E. E. (1997). Defuzzification: criteria and classification. *Fuzzy Sets and Systems, 108*, 159–178.

Lemesre, J., Dhaenens, C., & Talbi, E. (2007a). An exact parallel method for a bi-objective permutation flowshop problem. *European Journal of Operational Research, 177*, 1641–1655.

Lemesre, J., Dhaenens, C., & Talbi, E. (2007b). Parallel Partitioning method (PPM): A new exact method to solve bi-objective problems. *Computers and Operations Research, 34*, 2450–2462.

Li, X., Yalaoui, F., Amodeo, L., & Hamzaoui, A. (2009). Multiobjective method to solve a parallel machine scheduling problem. In *7th International Logistics & Supply Chain Congress'09, Istanbul, Turkey, November* 5–6.

Lofti, A. A. & Kashani, F. H. (2004). Bandwidth optimization of the E-shaped microstrip antenna using the genetic algorithm based on fuzzy decision making, In *Proceeding of 2004 IEEE antennas and propagation society international symposium* (pp. 2333–2336).

Nessah, R., Chu, C., & Yalaoui, F. (2005). An exact method for $P_m/sds, r_i/\sum_{i=1}^{n} C_i$ problem. *Computers & Operations Research, 34*, 2840–2848.

Pritsker, A. A. B., Walters, L. J., & Wolf, P. M. (1969). Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science, 16*, 93–108.

Przybylski, A. (2006). *Méthode en deux phases pour la resolution exacte de problèmes d'optimisation combinatoire comportant plusieurs objectifs: nouveaux développements et application au problème d'affectation linéaire*, Ph.D thesis, University of Nantes.

Ranjithan, S. R., Chetan, S. K., & Dakshina, H. K. (2001). Constraint method-based evolutionary algorithm (CMEA) for multiobjective optimization, In *Evolutionary Multi-Criteria Optimization* (pp. 299–313).

Riise, A. (2002). Comparing genetic algorithms and tabu search for multi-objective optimization, In *Abstract conference proceedings, Edinburgh, UK, July* 2002 (pp. 29).

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms* (pp. 93–100).

Shim, A.-O., & Kim, Y.-D. (2007). Scheduling on parallel identical machine to minimize total tardiness. *European Journal of Operational Research, 177*, 135–146.

Song, Y. H., Wang, H. S., Wang, P. Y., & Johns, A. T. (1997). Environmental/economic dispatch using fuzzy logic controlled genetic algorithms. *IEE Proceeding of Generation Transmission and Distribution, 144*(4), 377–382.

Ulungu, E., & Teghem, J. (1995). The two phase method: an efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences, 20*(2), 149–165.

VanVeldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classification, Analyses, and New Innovations*, PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Yalaoui, F., & Chu, C. (2002). An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. *IIE Transactions, 35*, 183–190.

Yalaoui, F., & Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *International Journal Production Economics, 76*, 265–279.

Yalaoui, N., Dugardin, F., Yalaoui, F., Amodeo, L., & Mahdi, H. (2010). *Fuzzy Project Scheduling, Production Engineering and Management under Fuzziness*. Berlin: Springer.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control, 8*, 338–353.

Zitzler, E., & Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zitzler, E., Laumanns, M., & Thiele, L., *SPEA2 : Improving the Strength Pareto Evolutionary Algorithm* Swiss Federal Institute of Technology, Lausanne, Switerland, Tech. Rep. TIK-Rep, 103.