Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2010, Article ID 943050, 10 pages
doi:10.1155/2010/943050

Research Article

A Multiobjective Optimization Approach to Solve a Parallel Machines Scheduling Problem

Xiaohui Li, Lionel Amodeo, Farouk Yalaoui, and Hicham Chehade

Université de Technologie de Troyes, Institut Charles Delaunay, LOSI (UMR-STMR 6279), 12 Rue Marie Curie BP 2060 10010 Troyes Cedex, France

Correspondence should be addressed to Xiaohui Li, xiaohui.li@utt.fr

Received 23 March 2010; Revised 22 July 2010; Accepted 27 October 2010

Academic Editor: Fakhreddine Karray

A multiobjective optimization problem which focuses on parallel machines scheduling is considered. This problem consists of scheduling n independent jobs on m identical parallel machines with release dates, due dates, and sequence-dependent setup times. The preemption of jobs is forbidden. The aim is to minimize two different objectives: makespan and total tardiness. The contribution of this paper is to propose first a new mathematical model for this specific problem. Then, since this problem is NP hard in the strong sense, two well-known approximated methods, NSGA-II and SPEA-II, are adopted to solve it. Experimental results show the advantages of NSGA-II for the studied problem. An exact method is then applied to be compared with NSGA-II algorithm in order to prove the efficiency of the former. Experimental results show the advantages of NSGA-II for the studied problem. Computational experiments show that on all the tested instances, our NSGA-II algorithm was able to get the optimal solutions.

1. Introduction

The parallel machines scheduling problem is a widely studied optimization problem. This problem considers several available identical machines to execute a set of jobs $N = \{1, \ldots n\}$. Indeed, it can be described as a special hybrid flowshop scheduling problem which has only one stage. Every 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 means the transfer time between two adjacent jobs i and j (job j is scheduled immediately after job i). Two different objectives are minimized at once in this work: makespan and total tardiness. According to the lawler scheduling classification, the studied problem is defined as $P_m|s_{ij}, r_j|(C_{\max}, \sum T_j)$ [1].

The most studied criteria for 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 [2] have proved that the problem of minimizing the makespan on two identical parallel machines is NP hard. Burcker [3] 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. [4] have proposed a heuristic and a lower bound for the $P_m|s_{ij}|C_{\max}$ problem. Yalaoui and Chu [5] have proposed a heuristic for $P_m|s_{ij}, \text{split}|C_{\max}$. An exact method is also developed to solve it. Gharbi and Haouari [6] have proposed a lower bound and use the branch-and-bound method to solve the $P_m|r_j, q_j|C_{\max}$ problem, where q_j is the delivery time. Néron et al. [7] have used two branching schemes for the parallel machines scheduling problem with release dates and tails. Zouba et al. [8] present heuristic algorithms to solve a parallel machines scheduling problem with a period-based changing mode operators to minimize the makespan. Fanjul-Peyro and Ruiz [9] have proposed the size-reduction heuristics for the unrelated parallel machines scheduling problem with the minimization of makespan.

The total tardiness is another concerned criterion which is the sum of tardiness of all jobs. The parallel machines scheduling problem to minimize the total tardiness is known as NP hard according to Koulamas [10]. Some exact methods are proposed to solve this problem. Azizoglu and Kirca [11] have proposed a branch-and-bound method to the parallel machines problem with total tardiness minimization.

Yalaoui and Chu [12] have also developed a lower bound and a branch-and-bound to solve the same problem. In the work of Shim and Kim [13], they developed dominance properties and a lower bound for the same problem, and a branch-and-bound method is adopted to solve it. In 2008, Shim and Kim [14] have used a branch and bound algorithm for solving a parallel machines scheduling problem with a job splitting property. Nguyen et al. [15] proposed a new dynamic programming formulation for solving this problem. Moreover, many heuristics have been developed, such as of Pritsker et al. [16], Alidaee and Rosa [17], Baker and Bertrand [18], and Lamothe et al. [19].

Many approximated methods have been proposed to solve the multiobjective optimization problems (MOPs). In 1985, Schaffer [20] 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 [21] has proposed the VOES method (Vector-Optimization Evolution Strategy) in 1990. Hajela and Lin [22] have 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 nondominated solutions. They are called the Pareto optimal solutions. Recently, the Pareto-based approaches are interested by the researchers. *Multiple Objective Genetic Algorithms* (MOGA) [23], *Niched Pareto Genetic Algorithm* (NPGA) [24], *Nondominated Sorting Genetic Algorithm (*NSGA-II) [25], *Strength Pareto Evolutionary Algorithm* (SPEA-II) [26], *Pareto Archived Evolution Strategy* (PAES) [27] and *Pareto envelope-based selection algorithm* (PESA) [28] are proposed for solving MOP and widely used for scheduling problems.

The parallel machines multiobjective scheduling problem is NP hard in the strong sense. Therefore, approximated methods are widely proposed for solving it. In 2003, Cochran et al. [29] have proposed a two-stage multipopulation genetic algorithm (MPGA) to minimize three different criteria: makespan, the total weighted tardiness, and the total weighted completion time. Chang et al. [30] have minimized the makespan and the total tardiness for parallel machines scheduling problem in 2005 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 [31] in 2006. In the works of Baesler et al. [32], a multiobjective optimization algorithm is appropriate to solve the $P_m||(C_{\max}, \sum T_j)$ problem. This algorithm is the combination of a genetic algorithm and a local search and is compared with the MOGA and the MOSA for the $P_m||(C_{\max}, \sum T_j)$ problem. Furthermore, Baesler et al. [33] have proposed *MultiObjective Simulated Annealing with Random Trajectory Search* (MOSARTS) for the same problem. Bouibede-Hocine et al. [34] used NSGA-II and SPEA for the $P|r_i, d_i|C_{\max}$, $L_{\max}$ problem. Dugardin et al. [35] have used the NSGA-II for solving a hybrid job shop and a parallel machines scheduling problems. Gao et al. [36] have presented a new parallel genetic algorithm for the multiobjective parallel machines scheduling problem which is subjected to a special process constraint. The aim is the minimization of the makespan and the earliness/tardiness penalty. Later, Gao [37] have proposed a novel artificial immune system for solving the same problem. Fang [38] proposed an improved weighted-based multiobjective genetic algorithm for the typical parallel machines scheduling problem. Lian [39] have proposed a united search particle swarm optimization algorithm for the parallel machines problem with different due dates. Chyu and Chang [40] have developed a pareto evolutionary algorithm for a biobjective unrelated parallel machines scheduling problem.

Our main contribution in this paper is to consider a special multiobjective parallel machines scheduling problem taking in consideration additional constraints compared to other previous works such as these of Baesler et al. [32, 33] or that of Bouibede et al. [34]. These additional constraints are the release date, the due date and the sequence-dependent setup times. At our knowledge, no other studies take in consideration all these constraints for multiobjective parallel machines scheduling problem, and thus we may consider that the problem studied in this paper is a specific scheduling problem.

Furthermore, as the problem is NP hard in the strong sense, two approximated methods: the *non-dominated sorting genetic algorithm* (NSGA-II) and *Strength Pareto Evolutionary Algorithm* (SPEA-II) are developed in this paper for the first time to solve the studied problem. An integer mathematical model is also proposed for modelling this problem, and we have proposed a special chromosome to encode it. An exact method is applied to demonstrate the efficiency of the proposed methods.

The rest of the paper is organized as follows: Section 2 describes the considered problem and the mathematical model. In Section 3, we develop the NSGA-II and the SPEA-II algorithms. Computational results are shown in Section 4. Our conclusion and perspectives are given in Section 5.

2. Problem Description and Mathematical Formulation

In the parallel machines scheduling problem, a set of n independent jobs should be scheduled on m identical machines without preemption. Each job has a processing time p_j, a release date r_j and a due date d_j. The sequence-dependent setup times s_{ij} are also considered 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 $s_{0j} = 0$, and it means that if job j is scheduled at the beginning on a machine, no setup time is required. All these job data are generated randomly based on a protocol test which is described in Section 4. Some assumption must be respected: each machine can execute only one job at once; each job can be processed only once.

Some notations are defined below:

n: the number of jobs

m: the number of machines

j: the index of jobs $j = 1, \ldots, n$

k: the index of machines $k = 1, \ldots, m$

r: the order of job in the machine

r_j: the release date of job j, $j = 1, \ldots, n$

d_j: the due date 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.

p_j: the processing time of job j, $j = 1, \ldots, n$

C_j: the completion time of job j, $j = 1, \ldots, n$

T_j: the real tardiness of job j, $j = 1, \ldots, n$, $T_j = \max(C_j - d_j, 0)$

$C_{\max}$: make span, $C_{\max} = \max_j C_j$

n_k: the number of jobs assigned to machine k.

The problem can be formulated as follows:

$$\text{Minimize}\left(C_{\max}, \sum T_j\right), \tag{1}$$

subject to

$$\sum_{j=1}^{n} X_{jkr} = 1 \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{2}$$

$$\sum_{k=1}^{m} \sum_{r=1}^{n_k} X_{jkr} = 1, \quad j = 1, 2, \ldots, n, \tag{3}$$

$$p_{[kr]} = \sum_{j=1}^{n} X_{jkr} p_j, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{4}$$

$$s_{[kr]} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{jkr} Y_{ij} s_{ij}, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k \tag{5}$$

$$r_{[kr]} = \sum_{j=1}^{n} X_{jkr} r_j, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{6}$$

$$d_{[kr]} = \sum_{j=1}^{n} X_{jkr} d_j, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{7}$$

$$C_{[kr]} = \max\left(C_{[k,r-1]} + s_{[kr]}, r_{[kr]}\right) + p_{[kr]}, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{8}$$

$$T_{[kr]} = \max\left(C_{[kr]} - d_{[kr]}, 0\right), \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k, \tag{9}$$

$$C_{\max} = \max_{k=1}^{m} \max_{r=1}^{n_k} C_{[kr]}, \tag{10}$$

$$\sum T_j = \sum_{k=1}^{m} \sum_{r=1}^{n_k} T_{[kr]}, \tag{11}$$

$$X_{jkr} = 0 \text{ or } 1, \quad k = 1, 2, \ldots, m,\ r = 1, 2, \ldots, n_k,\ j = 1, 2, \ldots, n, \tag{12}$$

$$Y_{ij} = 0 \text{ or } 1, \quad i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n. \tag{13}$$

Table 1: An example of chromosome.

$N^{\#}$ of jobs	1	2	3	4	5
$N^{\#}$ of machine	1	2	1	1	2
Position on the machine k	2	1	1	3	2

Equation (1) represents the objective function, and the goal of our work is to minimize the makespan and the total tardiness. Constraint (2) ensures that only one job can be scheduled at the rth job position. Constraint (3) means that each job can be scheduled only once. Constraints (4)–(9) denote the data of jobs which are scheduled at the rth job position of kth machine position jobs, such as processing times, setup times, release dates, due dates, completion times, and tardiness times. Constraints (10) and (11) explain how to compute the makespan and the total tardiness. Constraint (12) is a decision variable: if job j is scheduled on machine i in position r, then $X_{jkr} = 1$, otherwise 0. Constraint (13) shows that if job j is the immediate successor of the job i on the same machine, then $Y_{ij} = 1$; otherwise, $Y_{ij} = 0$.

3. Resolution Methods

In this section, two approximated methods and an exact method are developed to solve our problem. Two well-known algorithms have been chosen: NSGA-II and SPEA-II. The experimental results of these algorithms are compared in the following section.

3.1. Encoding Strategy. A special encoding of the studied problem is presented here. It is adopted for our developed methods: NSGA-II and SPEA-II. A chromosome (see Table 1) with a $(N \times 3)$ matrix is proposed to encode the problem. The elements in each column correspond to the job index, the number of the machine where the job is scheduled, and the position of the job on the machine. It is described hereafter

$a(j, 1)$ shows the index of jobs, $j = 1, 2, \ldots, n$,

$$a(j,2) = \begin{cases} k & \text{if job } j \text{ is scheduled on machine } k, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

$$a(j,3) = \begin{cases} r & \text{if job } j \text{ is scheduled at position } r \\ & \quad \text{on the machine,} \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

All feasible solutions of the initial population are randomly generated. A HOT (*High priority Order Transition*) sequence rule is adopted here. A random value is generated in a uniform distribution [0, 1] for each job, and all the jobs are ranked by the decreasing order of this value. Thereafter, the job with the highest value is served first on the machine which is earliest available.

For instance, Table 1 presents an example for which the Gantt chart is illustrated in Figure 1.

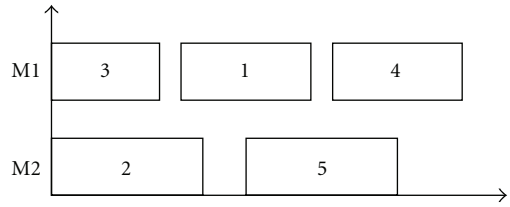

Figure 1: An example of the Gantt chart.

Table 2: The index of configurations.

	TF, RDD			
β, $[A, B]$	0.2	0.2	0.4	0.4
	0.2	0.4	0.2	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

3.2. NSGA-II. The Non-dominated Sorting Genetic Algorithm (NSGA-II) is the second version of the NSGA algorithm. The latter has been widely used to solve multiobjective optimization problems. It was proposed by Deb et al. [25] in 2000 as a fast and efficient multiobjective genetic algorithm to find the non-dominated solutions based on the Pareto dominance relationship. In this algorithm, the procedure begins starting from the creation of an initial population. All the feasible solutions in this population are randomly generated. At each generation, selection, crossover, and mutation operations as well as ranking, the different solutions are realized. These four operations are the main concept of the NSGA-II.

The binary tournament technique is used in the operations of the parents selection. We chose the classic one-point crossover operator in this study, and an operator of reparation is necessary for the created children. In the mutation operator, two columns are randomly chosen and they are interchanged. In the operator of ranking, all the different solutions are assigned to several non-dominated fronts. A computation of the crowding distance has been done to sort the solutions in the same front.

Several parameters are to be set in this algorithm. The population size and the number of generations determine the computational duration. The crossover probability and the mutation probability decide the convergence and the diversity of the results. The parameters setting of NSGA-II is complex. In this work, several tests have been carried out to choose the best parameters of crossover and mutation probabilities. We have tested this problem with a crossover probability equal to {0.6, 0.7, 0.8, 0.9, 1.0} and {0.05, 0.1, 0.15, 0.20, 0.25} for the mutation probability. The final values of the parameters are chosen according to the measuring criteria which are presented in Section 4. 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 generation number are set to 100. For the comparison with another approximated method, a fixed computing time is considered as the stopping criteria. We have chosen 1 second, 2 seconds, and 5 seconds for the different problems (see Section 4).

The main process of NSGA-II is described in Algorithm 1 [25].

3.3. SPEA-II. The second version of the SPEA (Strength Pareto Evolutionary Algorithm) is presented here, which has been introduced by Zitzler et al. [26]. A regular population and an archive mechanism (Pareto archive) are used in SPEA-II. During the generation, all the non-dominated solutions are copied to the Pareto archive. Compared with SPEA, the Pareto archive size is fixed in this algorithm. The Pareto archive is filled with the non-dominated solutions until the considered archive size. If there are not enough non-dominated solutions, the dominated solutions which have the best objective values are also considered. On the other side, if there are too many non-dominated solutions than the archive size, we only kept the best of them with a decreasing order of distance σ (see [26]).

We have used the same operations used in NSGA-II for the selection, crossover, and mutation. In the parameters settings of the algorithm, we have chosen the same crossover probability, mutation probability, and size of the population. We have set the archive size to 50 based on the tested results. The main concept of SPEA-II is described in Algorithm 2 [26].

Table 3: Objective values correspond to the NSGA-II/SPEA-II comparison.

Number of solution	NSGA-II		SPEA-II	
	$C_{\max}$	$\sum T_j$	$C_{\max}$	$\sum T_j$
1	414	1256	416	1449
2	415	1202	423	1350
3	416	1060	431	1338
4	418	990	433	1022
5	420	968	436	1003
6	432	952	469	1001
7	439	932		
8	440	867		

3.4. Exact Method. A full enumeration method is applied here. The aim is to enumerate all the feasible solutions and find the absolute Pareto optimal solutions among them. We have used the same ranking operation of NSGA-II to find the nondominated solutions. Because of the too long computational times, this method can be applied only the small instances. The results of the comparison between the exact method and the NSGA-II algorithm (which has been proved to be better than SPEA-II as shown in the next section) are shown in the following section.

(1) Generate the initial population P_t of size ns
(2) Evaluate these solutions
(3) Sort these solutions by non domination and crowding distance
(4) Creation of the offspring population Q_t with the operators of selection, crossover and mutation
(5) Evaluate all solutions
(6) Sort the solutions of two populations: P_t and Q_t
(7) Choose the best solutions for the new population P_t with the remaining steps (ranking into non dominated front, crowding distance)
(8) If the stopping criteria is satisfied then the algorithm is stopped, the obtained results are all the solutions in the first non dominated front; repeat steps (4) to (7) otherwise

Algorithm 1: Structure of the NSGA-II algorithm [25].

(1) Generate an initial population P_t and an empty Pareto archive Q_t, set t to 0
(2) Evaluate all the solutions in P_t and Q_t
(3) Find the non dominated solutions between P_t and Q_t. If there are too many non dominated solutions for the archive, calculate the distance σ of the solutions, we only kept the best solutions with a decreasing order of distance σ. In contrary, if there are not enough non dominated solutions in the archive, the best dominated solutions must be filled in the archive till this archive is full
(4) If the stopping criteria is satisfied, then stop the algorithm and export the non dominated solutions in the Pareto archive as the result
(5) The operators of selection, crossover and mutation are applied for the new population P_{t+1}, and then return to step (2)

Algorithm 2: Structure of the SPEA-II algorithm [26].

4. Computational Results

4.1. Definition of the Protocol Test. For the computational experiments, a protocol test is developed in which the job data are randomly generated. The job data include the processing times pj, the release dates r_j, the due dates d_j and the sequence-dependent setup times s_{ij}. In a previous work [41], we have presented a protocol test based on the work of [12, 42]. For each job, the processing times p_j are generated using a uniform distribution [1, 100]. After the generation of the processing times, the setup times s_{ij} are set to $\alpha \times \min(p_i, p_j)$, where α is a coefficient randomly generated in $[A, B]$, and $[A, B] \in \{[0.1, 0.2], [0.1, 0.5]\}$. The release dates are generated using a uniform distribution $[0, 50.5 \times N \times \beta/M]$. The value of β is considered from $\{0.6, 0.8\}$ [43]. The due dates of jobs are in the interval $[P\times(1-\mathrm{TF}-(\mathrm{RDD}/2)), P\times(1-\mathrm{TF}+(\mathrm{RDD}/2))]$, where $P = \sum_{j=1}^{n} p_j/m$, and we have chosen TF and RDD between 0.2 and 0.4. Since there are four parameters to set, and two values are proposed to each parameter, we have then 16 different configurations to test, and the index of each configuration is shown in Table 2. For the comparison of two approximated methods, three types of instances have been processed: the problem with 10 jobs and 3 machines, the problem with 20 jobs and 3 machines, and the problem with 50 jobs and 5 machines. The full enumeration method cannot be applied on large problems because of the too long execution times. Therefore, the comparison between the proposed methods and the full enumeration method is realized on the problems with 5, 6, 7, and 8 jobs.

4.2. Measuring Criteria. The comparison of two different Pareto fronts for solving the same MOP is very complicated, since each front is not a single scalar value, but a set of non-dominated solutions. An approximated method involves the following objectives [44]: 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 fronts. Ranjithan et al. [45] 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 [43] 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. Van Veldhuizen [46] has proposed an error-ratio criterion and a generational distance (GD). The former one represents the proportion of nontrue Pareto solutions in the obtained front. It measures the general progress to the reference set. Zitzler and Thiele [44] have presented a Zitzler measure, a μ distance is presented by Riise [47]. The two latter measures were adopted in this work, because they do not require to compute the absolute Pareto optimal front (reference set). The number of nondominated 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 nondominated solutions in fronts A and B, respectively. The μ distance is

Table 4: The comparison results for 10 jobs and 3 machines.

	n_s NSGA-II	n_s SPEA-II	μ	max_μ	min_μ	$C1$	$C2$
1	4	3.2	−10.719	−0.427	−32.998	0.020	0.941
2	3.7	3.7	−23.331	−5.572	−57.541	0	1
3	4.5	3.6	−12.328	−6.213	−29.621	0.033	0.960
4	4.8	3.7	−8.414	−0.236	−36.070	0.020	0.820
5	4.3	3.1	−12.079	−0.437	−25.091	0.040	0.850
6	4.2	3.6	−11.865	−7.612	−20.759	0	1
7	4.8	3.3	−16.530	−5.301	−39.432	0	1
8	4.5	4.3	−16.052	−4.020	−37.314	0	0.980
9	3.6	3.4	−28.844	−1.658	−36.948	0	0.967
10	3	3.6	−29.923	−24.467	−40.082	0	1
11	4.2	3.5	−21.510	−4.712	−63.821	0	0.917
12	4.7	3.3	−17.275	−1.108	−36.915	0	0.967
13	3.3	3.2	−20.658	−2.105	−44.051	0.033	0.883
14	3.4	3.6	−20.507	−5.982	−38.358	0	1
15	3.4	3.1	−6.748	−0.369	−25.938	0	1
16	3.3	3	−2.677	−1.189	−7.184	0	0.900

Table 5: The comparison results for 20 jobs and 3 machines.

	n_s NSGA-II	n_s SPEA-II	μ	max_μ	min_μ	$C1$	$C2$
1	5.2	3.6	−57.783	−26.900	−157.165	0	1
2	6.2	3.8	−64.773	−19.398	−183.824	0	1
3	6.7	3.8	−43.357	−9.967	−116.936	0	1
4	6.7	3.7	−80.565	−11.515	−158.075	0.020	0.986
5	5.6	3.9	−38.786	−5.863	−65.553	0	1
6	6.3	4.5	−84.840	−15.780	−150.437	0	1
7	6.3	3.3	−70.763	−13.980	−121.599	0	1
8	6.6	4.2	−130.905	−39.507	−207.397	0	1
9	4.4	4.4	−89.322	−33.199	−242.742	0	1
10	5.6	3.6	−90.343	−19.670	−177.626	0	1
11	5.4	3.5	−69.500	−30.652	−162.870	0	1
12	6.6	3.7	−102.433	−16.751	−226.405	0	1
13	5	3.9	−83.883	−19.717	−160.027	0	1
14	5.2	3.6	−61.100	−3.506	−122.710	0.025	0.950
15	6.8	3.7	−56.728	−18.379	−96.926	0	1
16	4.5	3.5	−125.261	−33.281	−199.404	0	1

defined as $\mu = \sum_{i=1}^{n_A} d_i/n_A$, where d_i is the distance between 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 to the optimal solutions than front B. The lower of the value μ we find, the better quality of front A we will get.

The Zitzler measure has been presented by Zitzler and Thiele [44], where $C1$ measures the ratio of solutions in A which are dominated by at least one solution in B. The equation is defined by $C1 = |\{a \in A \mid \exists b \in B : b \succ a\}|/|A|$, where a and b are the non-dominated solutions in fronts A and B respectively. When $C1 = 1$, it means that all of the solutions in A are dominated by solutions in B. However, when $C1 = 0$, it means that there are no solutions in A that are dominated by at least one solution in B. On the other side, $C2$ measures the ratio of solutions in B which are dominated by at least one solution in A.

4.3. Results

4.3.1. Comparison of NSGA-II and SPEA-II. First, the experimental results of NSGA-II and SPEA-II are compared. Figure 2 shows an example where 20 jobs should be scheduled on 3 identical parallel machines. The objective values

Table 6: The comparison results for 50 jobs and 5 machines.

	n_s NSGA-II	n_s SPEA-II	μ	max_μ	min_μ	$C1$	$C2$
1	5.4	4.4	−121.605	−71.3029	−276.454	0	1
2	6.9	3.9	−95.738	−28.279	−251.580	0.057	0.933
3	6.9	5.7	−147.188	−63.880	−253.701	0	1
4	7	4.7	−131.867	−26.957	−255.181	0	1
5	6	3.2	−151.970	−58.576	−281.431	0	1
6	5.2	2.9	−168.611	−101.356	−280.068	0	1
7	7	3.8	−194.197	−73.031	−302.200	0	1
8	6.7	3.4	−132.477	−51.033	−212.641	0	1
9	5.2	4	−161.820	−75.712	−253.037	0	1
10	7.1	3.5	−177.886	−46.687	−287.345	0	1
11	7.4	4	−147.448	−58.326	−305.017	0	1
12	7.1	4.4	−163.878	−22.844	−362.630	0	1
13	5	4.1	−149.905	−71.089	−269.733	0	1
14	5.7	3.8	−207.573	−63.686	−291.703	0	1
15	7	3.4	−189.336	−103.498	−277.892	0	1
16	7.4	3.5	−201.379	−62.870	−325.050	0	1

Table 7: NSGA-II/Full enumeration comparison.

Problem (n, m)	n_s NSGA-II	n_s FE	μ	$C1$	$C2$	CT FE	CTNSGA-II
(5, 2)	5	5	0	0	0	0.016s	0.688s
(5, 3)	4	4	0	0	0	0.016s	0.743s
(6, 2)	3	3	0	0	0	0.922s	0.688s
(6, 3)	4	4	0	0	0	0.890s	0.766s
(7, 2)	4	4	0	0	0	132s	0.672s
(7, 3)	5	5	0	0	0	206s	0.750s
(8, 2)	6	6	0	0	0	38004s	0.672s
(8, 3)	3	3	0	0	0	42608s	0.687s

of all the obtained solutions are shown in Table 3. We can observe that the front obtained by NSGA-II dominates the other front obtained by SPEA-II. Indeed, in this example, we have $C1 = 0$, which means that there is no solution in the NSGA-II front that is dominated by at least a solution from SPEA-II. $C2 = 1$ means that all the solutions of SPEA-II are dominated by at least one solution of NSGA-II front. Finally, we have $\mu = -38.071$; μ is negative because the NSGA-II front dominates the SPEA-II front.

Tables 4, 5, and 6 show the comparison results of NSGA-II and SPEA-II to solve three different problems with 16 different configurations. The index of the configuration is shown in Table 2. For each configuration, we have tested 10 instances. The mean value of μ, the minimum value and the maximum value of μ (max_μ and min_μ), the mean values of $C1$ and $C2$ are presented. In this work, 480 instances are therefore tested.

The first line of Table 4 is explained. This problem is defined with 10 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 3). The next 7 elements show the comparison results. Since each configuration is tested for ten instances, the value of n_s shows that the NSGA-II has obtained more non-dominated solutions than SPEA-II with respective mean values of 4 and 3.2. The mean value of the Riise distance $\mu = -10.719$ indicates that the obtained front of NSGA-II dominates the SPEA-II front, since this value is negative. The minimum and maximum values of μ are both negatives, which means that the Riise distance is always negative during the ten tests. The results show that $C1 = 0$; that is, there are no solutions provided by NSGA-II that are dominated by at least one solution of the SPEA-II. On the other side, $C2 = 1$ means that all solutions provided by SPEA-II are dominated by at least one solution of NSGA-II.

In this work, we have chosen a fixed computing time as a stopping criterion for NSGA-II and SPEA-II. One second, two seconds, and five seconds are the execution times that

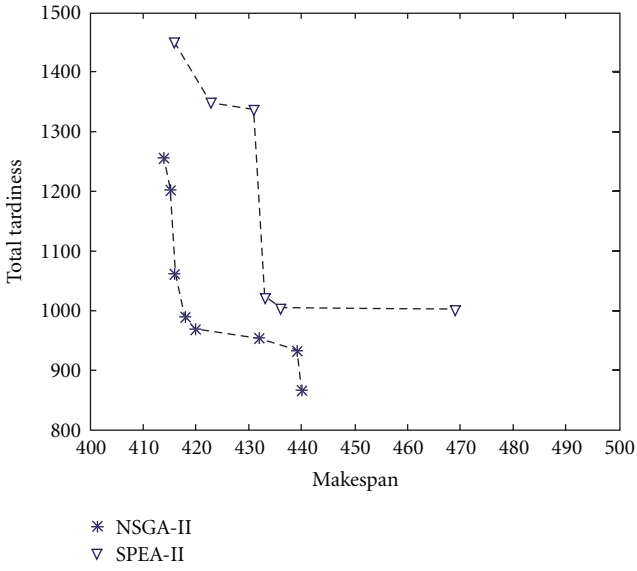

Figure 2: Example of NSGA-II/SPEA-II comparison.

are fixed for the problems with 10 jobs, 20 jobs, and 50 jobs, respectively. Tables 4, 5, and 6 show a comparison between NSGA-II and SPEA-II for the tested instances.

The values of μ are always negative, which means that the NSGA-II front always dominates the SPEA-II front. The absolute value of μ distance is not high for small-size problems. However, we can observe that the absolute values of the μ distance are higher for large-size problems. That means that for small problems, the two fronts of NSGA-II and SPEA-II are both very close to the absolute Pareto front (the efficiency of NSGA-II is demonstrated in the following section). So, the difference between the two fronts is not very obvious, and the number of non-dominated solutions in the obtained fronts are quiet similar. When solving complex problems, the advantage of NSGA-II is obvious. The NSGA-II front is under the SPEA-II front and the distance between the two fronts is larger than before. The NSGA-II has obtained more non-dominated solutions than SPEA-II. The value of $C1$ is always less than $C2$. In fact, the values of $C1$ are nearly equal to 0, while those of $C2$ are nearly equal to 1. Based on all the tested instances, the comparison between the different numerical results show the advantages and the efficiency of NSGA-II.

4.3.2. Comparison with the Pareto Optimal Solutions. In this section, the absolute Pareto optimal solutions obtained by an exact method are compared to the solutions of the NSGA-II, which has been proved to be better than SPEA-II. This exact method consists of enumerating all the possible solutions of the problem. Since the full enumeration requires too much execution time, we cannot solve large-size problems with this exact method. In this work, we are able to solve the problem with a maximum number of 8 jobs. The results in Table 7 have shown that the NSGA-II has obtained the same solutions ($\mu = 0$, $C1 = C2 = 0$) than the exact method for the tested instances. The main advantage of the NSGA-II is that it requires less computing time (CT NSGA-II) than the full enumeration (CT FE).

5. Conclusion

In this work, an identical parallel machines 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 to develop a new mathematical model and two metaheuristics which are the NSGA-II and SPEA-II. The computational results show the advantage of NSGA-II over the SPEA-II on the 480 tested problems based on the adopted measuring criteria. After that, the NSGA-II is compared to the full enumeration for small-size problems. We can observe that NSGA-II is able to obtain the absolute optimal solutions. The full enumeration method cannot solve problems with more than 8 jobs. Hence, the perspective is to develop another exact method to compare the results of the two methods with larger and more complex problems. On the other side, the setting of crossover and mutation probabilities is very difficult. Therefore, it would be interesting to develop a self-adopted GA which can set the probabilities during the generations.

References

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, no. C, pp. 287–326, 1979.

[2] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *European Journal of Operational Research*, vol. 47, no. 3, pp. 271–292, 1990.

[3] P. Brucker, *Scheduling Algorithm*, Springer, Berlin, Germany, 1998.

[4] M. Gendreau, G. Laporte, and E. M. Guimarães, "A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times," *European Journal of Operational Research*, vol. 133, no. 1, pp. 183–189, 2001.

[5] F. Yalaoui and C. Chu, "An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times," *IIE Transactions*, vol. 35, no. 2, pp. 183–190, 2003.

[6] A. Gharbi and M. Haouari, "Minimizing makespan on parallel machines subject to release dates and delivery times," *Journal of Scheduling*, vol. 5, no. 4, pp. 329–355, 2002.

[7] E. Néron, F. Tercinet, and F. Sourd, "Search tree based approaches for parallel machine scheduling," *Computers and Operations Research*, vol. 35, no. 4, pp. 1127–1137, 2008.

[8] M. Zouba, P. Baptiste, and D. Rebaine, "Scheduling identical parallel machines and operators within a period based changing mode," *Computers and Operations Research*, vol. 36, no. 12, pp. 3231–3239, 2009.

[9] L. Fanjul-Peyro and R. Ruiz, "Size-reduction heuristics for the unrelated parallel machines scheduling problem," *Computers and Operations Research*, vol. 38, no. 1, pp. 301–309, 2011.

[10] C. Koulamas, "Total tardiness problem: review and extensions," *Operations Research*, vol. 42, no. 6, pp. 1025–1041, 1994.

[11] M. Azizoglu and O. Kirca, "Tardiness minimization on parallel machines," *International Journal of Production Economics*, vol. 55, no. 2, pp. 163–168, 1998.

[12] F. Yalaoui and C. Chu, "Parallel machine scheduling to minimize total tardiness," *International Journal of Production Economics*, vol. 76, no. 3, pp. 265–279, 2002.

[13] S.-O. Shim and Y.-D. Kim, "Scheduling on parallel identical machines to minimize total tardiness," *European Journal of Operational Research*, vol. 177, no. 1, pp. 135–146, 2007.

[14] S.-O. Shim and Y.-D. Kim, "A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property," *Computers and Operations Research*, vol. 35, no. 3, pp. 863–875, 2008.

[15] N. Huynh Tuong, A. Soukhal, and J.-C. Billaut, "A new dynamic programming formulation for scheduling independent tasks with common due date on parallel machines," *European Journal of Operational Research*, vol. 202, no. 3, pp. 646–653, 2010.

[16] A. A. B. Pritsker, L. J. Walters, and P. M. Wolf, "Multi-project scheduling with limited resources: a zero-one programming approach," *Management Science*, vol. 16, pp. 93–108, 1969.

[17] B. Alidaee and D. Rosa, "Scheduling parallel machines to minimize total weighted and unweighted tardiness," *Computers and Operations Research*, vol. 24, no. 8, pp. 775–788, 1997.

[18] K. R. Baker and J. W. M. Bertrand, "A dynamic priority rule for scheduling against due-dates," *Journal of Operations Management*, vol. 3, no. 1, pp. 37–42, 1982.

[19] J. Lamothe, F. Marmier, M. Dupuy, P. Gaborit, and L. Dupont, "Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints," *Computers and Operations Research*. In press.

[20] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100, 1985.

[21] F. Kursame, "A variant of evolution strategies for vector optimization," in *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSNI '91)*, H. P. Schwefel and R. Manner, Eds., vol. 496 of *Lecturer Notes in Computer Science*, pp. 193–197, 1991.

[22] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural Optimization*, vol. 4, no. 2, pp. 99–107, 1992.

[23] C. M. Fonseça and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 416–423, San Mateo, Calif, USA, 1993.

[24] J. Horn, N. Nafpliotis, and D. E. Goldberg, "Niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation*, vol. 1, pp. 82–87, 1994.

[25] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "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, 2000.

[26] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength pareto evolutionary algorithm," Tech. Rep. 103, Swiss Federal Institute of Technology, Lausanne, Switerland, 2001.

[27] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto Archived Evolution Strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[28] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope based selection algorithm for multiobjective optimization," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 839–848, 2000.

[29] J. K. Cochran, S.-M. Horng, and J. W. Fowler, "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines," *Computers and Operations Research*, vol. 30, no. 7, pp. 1087–1102, 2003.

[30] P.-C. Chang, S.-H. Chen, and K.-L. Lin, "Two-phase sub population genetic algorithm for parallel machine-scheduling problem," *Expert Systems with Applications*, vol. 29, no. 3, pp. 705–712, 2005.

[31] P.-C. Chang, S.-H. Chen, and J.-C. Hsieh, "A global archive sub-population genetic algorithm with adaptive strategy in multi-objective parallel-machine scheduling problem," in *Proceedings of the 2nd International Conference on Natural Computation (ICNC '06)*, vol. 4221 of *Lecture Notes in Computer Science*, pp. 730–739, Xi'an, China, September 2006.

[32] F. Baesler, R. Moraga, and O. Cornejo, "Introduction of memory elements in simulated annealing method to solve multiobjective parallel machine scheduling problems," *Ingeniare*, vol. 16, no. 3, pp. 428–437, 2008.

[33] F. Baesler, L. Ceballos, and M. Ramírez, "Multiobjecttve molding machine scheduling using memetic algorithms," *Maderas: Ciencia y Tecnologia*, vol. 8, no. 3, pp. 183–192, 2006.

[34] K. Bouibede-Hocine, V. T'kindt, and D. Tran, "Two evolutionary algorithms for a uniform parallel machines," in *Proceedings of the 7th International Conference on Multi-Objective Programming and Goal Programming*, Loire valley, France, June 2006.

[35] F. Dugardin, H. Chehade, L. Amodeo, F. Yalaoui, and C. Prins, "Hybrid Job Shop and parallel machine scheduling problems: minimization of total tardiness criterion," in *Multiprocessor Scheduling: Theory and Applications*, E. Levner, Ed., p. 436, 2007.

[36] J. Gao, G. He, and Y. Wang, "A new parallel genetic algorithm for solving multiobjective scheduling problems subjected to special process constraint," *International Journal of Advanced Manufacturing Technology*, vol. 43, no. 1-2, pp. 151–160, 2009.

[37] J. Gao, "A novel artificial immune system for solving multiobjective scheduling problems subject to special process constraint," *Computers and Industrial Engineering*, vol. 58, no. 4, pp. 602–609, 2010.

[38] Z. Fang, "An improved weight-based multiobjective genetic algorithm and its application to parallel machine scheduling," in *Proceedings of the 2nd International Conference on Intelligent Computing Technology and Automation (ICICTA '09)*, vol. 1, pp. 161–164, Changsha, China, October 2009.

[39] Z. Lian, "A united search particle swarm optimization algorithm for multiobjective scheduling problem," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3518–3526, 2010.

[40] C.-C. Chyu and W.-S. Chang, "A Pareto evolutionary algorithm approach to bi-objective unrelated parallel machine scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 5–8, pp. 697–708, 2010.

[41] X. Li, F. Yalaoui, L. Amodeo, and A. et Hamzaoui, "Multiobjective method to solve a parallel machine scheduling problem," in *Proceedings of the 7th International Logistics & Supply Chain Congress*, Istanbul, Turkey, November 2009.

[42] R. Nessah, C. Chu, and F. Yalaoui, "An exact method for Pm/sds, ri/$\sum_{i=1}^{n}$Ci problem," *Computers and Operations Research*, vol. 34, no. 9, pp. 2840–2848, 2007.

[43] S. V. Kumar and S. R. Ranjithan, "Evaluation of the Constraint Method-based Multiobjective Evolutionary Algorithm (CMEA) for a three objective optimization problem," in *Proceedings of the Genetic and Evolutionary Computation*

Conference (GECCO '02), W. B. Langdon et al., Ed., pp. 431–438, Morgan Kaufmann, 2002.

[44] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[45] S. R. Ranjithan, S. K. Chetan, and H. K. Dakshina, "Constraint method-based evolutionary algorithm (CMEA) for multiobjective optimization," in *Evolutionary Multi-Criteria Optimization*, Lecture Notes in Computer Science, pp. 299–313, 2001.

[46] D. A. VanVeldhuizen, *Multiobjective evolutionary algorithms: classification, analyses, and new innovations*, Ph.D. thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 1999.

[47] A. Riise, "Comparing genetic algorithms and tabu search for multi-objective optimization," in *Abstract Conference Proceedings*, p. 29, Edinburgh, UK, July 2002.