Les Annales RECITS (http://www.lrecits.usthb.dz)
Vol. 04, 2017, pages 37-48

A tabu search and a genetic Algorithm for solving a bicriteria Parallel Machine Scheduling Problem

Karima Bouibede-Hocine, Drifa Hetak and Mourad Boudhar

RECITS Laboratory, Faculty of Mathematics, USTHB,
BP 32, El Alia, 16111, Bab Ezzouar, Algiers, Algeria.

ka.hocine@gmail.com, drifa.hettak@yahoo.fr, mboudhar@usthb.dz

Abstract: This paper deal with a uniform parallel machines scheduling problem. The objective is the minimization of two criteria, the maximum completion time and the maximum lateness, and we are interested to solve it by means of a fast and efficient multiple-objective evolutionary algorithm based on NSGA-II. The initial population is either randomly generated or partially generated by using a tabu search algorithm which is based on the minimization of a linear combination of criteria. The aim is to find an approximation of the Pareto frontier. Both the NSGA-II and the tabu search algorithms are tested. Computational results which show the interest of both methods are provided.

Keywords: Scheduling, bicriteria, Tabu Search, NSGA-II.

Résumé : Ce papier traite d'un problème d'ordonnancement sur des machines parallèles uniformes. L'objectif est la minimisation de deux critères : le temps d'achèvement maximal et le retard maximum, et nous nous sommes intéressés à le résoudre au moyen d'un algorithme évolutif multi-objectif rapide et efficace basé sur NSGA-II. La population initiale est générée aléatoirement ou partiellement en utilisant un algorithme de recherche tabou basé sur la minimisation d'une combinaison linéaire de critères. Le but est de trouver une approximation de la frontière de Pareto. Les algorithmes de recherche NSGA-II et tabou sont tous deux testés. Les résultats expérimentaux qui montrent l'intérêt des deux méthodes sont fournis.

Mots clés : Ordonnancement, bicritère, Recherche Tabou, NSGA-II.

1 Introduction

The scheduling problem that we consider can be stated as follows. Consider n jobs to be scheduled without preemption on m uniform parallel machines, such that each machine has its own speed denoted by V_j, $\forall j = 1, ..., m$. Each job J_i has a release time r_i and a due date d_i. Job J_i can be performed by any machine M_j, thus requiring a processing time p_{ij} with $p_{ij} = \frac{p_i}{V_j}$, $\forall i = 1, ..., n, \forall j = 1, ..., m$, where p_i is the absolute processing time of the job J_i. The bicriteria problem involves the makespan and the maximum lateness minimization. For a given schedule, each job J_i completing at time C_i, the two objectives can be computed as follows: $C_{max} = max_{i=1..n}\{C_i\}$ and $L_{max} = max_{i=1..n}\{C_i - d_i\}$. Following the classic three-field notation of scheduling problems introduced in [8] and extended to multicriteria scheduling problems in [15], this problem can be referred to as $Q/r_i, d_i/C_{max}, L_{max}$.

The aim is to find the so-called Pareto optimal solutions or non-dominated solutions which can be formally defined as follows. Let $\mathcal{S}$ be the set of feasible schedules. A schedule $s \in \mathcal{S}$ is a *strict* Pareto optimum iff there does not exist another schedule $s' \in \mathcal{S}$ such that ($L_{max}(s') \leq L_{max}(s)$ and $C_{max}(s') < C_{max}(s)$) nor ($L_{max}(s') < L_{max}(s)$ and $C_{max}(s') \leq C_{max}(s)$). The set of strict Pareto optimum is denoted by $E \subseteq\in \mathcal{S}$. The criteria vector associated to a strict Pareto optimum is called a non dominated criteria vector and the set of such vectors is denoted by ZE. Notice that $|ZE| \leq |E|$. To solve the bicriteria problem we enumerate set ZE by calculating one strict Pareto optimum per non dominated criteria vector. The calculation of the strict Pareto optima for the problem $Q/r_i, d_i/C_{max}, L_{max}$ is NP-hard.

The literature contains a few works dealing with multiccriteria parallel machine scheduling problem. When only criterion C_{max} is minimised, Dell'Amico and Martello [6] propose for the $P||C_{max}$ a branch-and-bound exact algorithm which is capable of solving, in the most favorable cases, instances with up to 10000 jobs in size. The problem with unrelated parallel machines, referred to as $R||C_{max}$, has been solved by Van de Velde [19] and Martello, Soumis and Toth [12] who propose exact algorithms and [7] who propose a new metaheuristics based on the Iterated Greedy methodologiefor scheduling problems [13]. When only criterion L_{max} is minimised, Carlier [2] proposes a branch-and-bound algorithm that can be used to solve the $P|r_i, d_i|L_{max}$ problem. This algorithm has been next extended by Carlier and Pinson [3]. Haouari and Gharbi [9] focus on the design of lower bounds based on network flow formulations that outperform the one proposed by Carlier and Pinson. They also develop branch-and-bound algorithms capable of solving instances of the $P|r_i, d_i|L_{max}$ problem with up to 300 jobs in size. An equivalent problem, referred to as $P|r_i, q_i|C_{max}$, has been tackled by Tercinet et al. [17, 18] who provide a branch-and-bound algorithm with bounds improved over those given by Haouari and Gharbi. An extension of the latter problem has been considered by Lancia [11] who deals with the $R2|r_i, q_i|C_{max}$ problem. A heuristic and an exact method are proposed to solve this problem. To the best of our knowledge no work related to the $Q|r_i, d_i|C_{max}, L_{max}$ problem exists.

We propose to heuristically solve this problem by using a tabu search and a genetic algorithm. The last one is a population-based algorithm which means that it make evolving a

set of solutions towards the set of strict Pareto optima. The initial population of NSGA-II is either randomaly generated or partially generated by using a tabu search algorithm, that minimizes a linear combination of the two criteria considered for our problem.

The remainder is organised as follows. Section 2 presents the solution encoding, *i.e.* how a schedule is modeled by a tabu search and NSGA-II. Section 3 a tabu search that minimizes a linear combinaition of the two criteria. Section 4 presents the details of the genetic algorithm based on NSGA-II scheme. Computational experiments are reported in Section 5 and show the effectiveness of the algorithms on randomly generated instances. At last, Section 6 presents a conclusion and future research directions.

2 Solution encoding

In order to compare the two ways, a tabu search and the NSGA-II algorithms, of performing the enumeration of the set of strict Pareto optima, the same solution representation is used in the two algorithms. In this work we have chosen to implement a direct encoding of solutions. A solution is coded by a matrix $\Re$ of $(n * m)$ where m is the number of machines and n is the number of jobs and $\Re_{i,j}$ indicates the position of the job J_i on the machine M_j such that:

$$\Re\,(i,j) = \begin{cases} k \text{ if job } J_i \text{ is scheduled at position } k \text{ on machine } M_j \\ 0 \text{ otherwise} \end{cases}$$

For example, the solution coded

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}	J_{12}
M_1	1	0	0	2	0	0	3	0	0	4	0	0
M_2	0	1	0	0	2	0	0	3	0	0	4	0
M_3	0	0	1	0	0	2	0	0	3	0	0	4

represents the schedule: $M_1 : (J_1, J_4, J_7, J_{10})$, $M_2 : (J_2, J_5, J_8, J_{11})$, $M_3 : (J_3, J_6, J_9, J_{12})$

3 The tabu search algorithm

The tabu search method was first introduced by Glove [] which is a local search method. It requires an initial solution, a move function and neighborhood. The tabu search work as follows. It proceeds by transiting from one solution to another using a move function. All the neighbors of a current solution are examined in order to find the best one around it local area. Note that the function move can decrease the quality of the solution. A tabu list is introduced to avoid cycling. The components of the tabu list indicate the current tabu moves which are now forbidden. The search procedure stops when a stop criterion holds. In our case, we have chosen a maximum number of iterations without improvement as a stopping criterion.

3.1 Initial solution

The initial solution is obtained by using an heuristic by progressive construction. It can be seen as an alternate application of the Shortest Release Time (SRT) and Earliest Deadline (ED) rules. The SRT is applied, i.e. the job with the smallest value of r_i is scheduled, until the job with the smallest deadline must be scheduled otherwise, it would be infeasible if applying once more the SRT rule. A job is always scheduled on the machine on which it completes at the earlier. This algorithm requires $o\left(nlog(n) + nm\right)$ time.

3.2 Neighborhood

At each iteration of the search procedure we choose the best of two types of moves: the insert moves and the swap moves. The insert moves consist of transferring each job from one machine to all positions of the other machines. The swap moves consist of exchanging each pair of jobs between two machines and inserting them in all positions. So, solutions are generated by removing jobs from a machine M_k and inserting them on a machine M'_k, it is done iteratively as follows. The removal starts from the job placed in the last position of the machine M_k and proceeds to the job placed in the first position. Analogously, the insertion begins by placing each job in the last position of machine and continues until the first position. Solutions reached by swap moves is similar, since such moves are a combination of two insert moves. Note that each job J_i can not start processing before r_i.So, moves must respect this condition. The time complexity to evaluate the solutions generated by the insert moves and The swap moves is $O(n^2m)$. and $O(n^3m)$, respectively.

A neighbor s is evaluated by the following linear combination of criteria C_{max} and L_{max}: $Z(s) = \alpha$ x $C_{max} + (1 - \alpha)$ x L_{max} The tabu search algorithm is used for a fixed value of α. The best neighbor is the neighbor s^b that is not in the tabu list and with minimum cost function value $Z(s^b) = min_{s \notin tabulist} Z(s)$

3.3 Tabu list

The tabu list is used to prevent the search from cycling between solutions. Our tabu list has a fixed size. When the list is full, the oldest element of the list is replaced by the new element.

4 The NSGA-II algorithm

To explain the used genetic algorithm. We first present the framework of NSGA-II and next crossovor operator and mutation operator are provided.

4.1 Framework

The Non-dominated Sorting Genetic Algorithm (NSGA-II) is a well known and extensively used algorithm based on its predecessor NSGA. It was formulated by Deb ([**?**]) as a fast and very efficient multiple-objective evolutionary algorithm. NSGA-II is based on the notion of *non-dominated sorting* which works as follows. Consider a population P of N chromosomes, each one representing a solution. We refer to F^j as the j-th non dominated set in P (front), *i.e.* F^j is the set of non dominated solutions inside $P - \bigcup_{k=1}^{j-1} F^{j-1}$ with $F^0 = \emptyset$. The idea of NSGA-II is to keep from one generation to another the best fronts F^j assigning to each chromosome a fitness value f depending on the front to which the corresponding chromosome belongs and the spreading of chromosome in solution space. The main lines of NSGA-II are provided in Algorithm 1.

NSGA-II algorithm

Input: P_0, Q_0 (initial populations), N (Population size),
tmax (Max. number of generations).
t=0;
While $(t < tmax)$
  $R_t = P_t \cup Q_t$;
  Perform a non-dominated sorting on R_t to identify fronts F^j;
  $P_{t+1} = P_t$; $j = 1$;
  While $(|P_{t+1}| + |F^j| < N)$
    $P_{t+1} = P_{t+1} \cup F^j$;
    $j = j + 1$
  End while
  Apply the *crowding distance based selection* on F^j to fill P_{t+1};
  $i = 1$; $Q_{t+1} = \emptyset$;
  While $(i \leq N/2)$
    Apply the *crowded tournament selection* on P_{t+1} to select $\mathcal{C}$ and $\mathcal{C}'$;
    Apply the crossover operator under probability p_N^C on $\mathcal{C}$ and $\mathcal{C}'$ to create $\mathcal{O}$ and $\mathcal{O}'$;
    Apply the mutation operator under probability p_N^M on $\mathcal{O}$ and $\mathcal{O}'$;
    $Q_{t+1} = Q_{t+1} \cup \{\mathcal{O}\} \cup \{\mathcal{O}'\}$;
    $i = i + 1$;
  End while
  $t = t + 1$;
End while
Output: Front F_1 from set $P_{tmax} \cup Q_{tmax}$

See [4] for details

Figure 1: The main lines of the NSGA-II algorithm

Now we describe the main components of this algorithm. The initial population P_0 is filled with N randomly generated schedules whilst Q_0 is empty. The crowding distance based selection and the crowded tournament selection, which are applied during the NSGA-II algorithm, are based on the *crowding distance*which is defined as follows. It is an estimate of the perimeter of th cuboid formed by using the nearest neighbors as the vertices. The

diversity is established by this distance. For a chromosome the crowding distance is the sum of the normalized distance (in the criterion space) between the right and the left of neighbors for each criterion. The extreme solutions have a crowding distance equal to infinity.

The binary tournament selection is used as a selection operator : Between two chromosomes, the selected individual is the one with the lowest level. If two chromosomes have the same level, the best chromosome is the one with the greatest crowding distance.

4.2 Crossover and mutation operators

- Crossover: the crossover operator generates offspring by merging two chromosomes which we will call parents. Offsprings are created from parents by using a classic one-point crossover operators. This can result in inconsistencies in offsprings which are corrected by a dedicated sequencing heuristic.

- Mutation: in this operation two columns of the chromosome are randomly selected and they are interchanged.

5 Computational Experiments

In this section we provide only final experimentations that aim at comparing the proposed algorithms. Testing of heuristic algorithms is a crucial step since it enables to conclude on their efficiency. This conclusion is necessarily relative to the instances on which they are compared, *i.e.* after experimentations we can only conclude that some algorithms outperform the others on the considered instances. Accordingly, when designing an experimental plan, we focus on the random generation of numerous instances, representing as most configurations as possible. The ideal situation occurs when we are capable of randomly generating instances that are representative of all possible instances, leading thus to absolute conclusions concerning the efficiency of the algorithms. However, in practice, this does not occur and we are reduced to the generation of *estimated representative* instances. The experimental plan we consider for the $Q|r_i, d_i|C_{max}, L_{max}$ problem is defined as follows : We generate problems with $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$ and $m \in \{2, 4, 6\}$. To rule the distribution of deadlines and release dates, two factors, namely L and R are used. Factor L is used to center the distribution of deadlines and release dates whilst factor R rules their spreading. For each problem size $(n; m)$, different values of factor R are tested: $R \in \{0.2; 0.4; 0.6; 0.8; 1.0; 1.2; 1.4; 1.6\}$. The value of factor L in the scheduling uniform parallel machine problem is defined by $L = \frac{\overline{P}}{V_{min}}$ with $V_{min} = min_{j=1,\ldots,m} V_j$, and $\overline{P} = \sum_{i=1}^{n} p_i$. The machine speeds are drawn at random between 1 and 10 using an uniform distribution law. Absolute job processing times p_i are drawn at random 10 and 100 using an uniform distribution law. For each job J_i, its deadline $\tilde{d}_i$ is drawn at random between $\left(L - \frac{R \times \overline{P}}{2\overline{V}}\right)$ and $\left(L + \frac{R \times \overline{P}}{2\overline{V}}\right)$, with $\overline{V} = \sum_{j=1}^{m} V_j$ using an uniform dis-

tribution law. Similary, the release dates r_i is generated between $\left[\tilde{d}_i - 3\frac{p_i}{V_{max}}; \tilde{d}_i - \frac{p_i}{V_{max}}\right]$, with $V_{max} = max_{j=1,\ldots,m} V_j$. In case where $r_i < 0$, we set $\tilde{d}_i = \tilde{d}_i + r_i$ and $r_i = 0$.

To evaluate the efficiency of the genetic algorithms with focus on two aspects. The first one is the CPU time required to return an approximation of set ZE. The results are given in Table 1 where the minimum (t_{min}), the average (t_{avg}) and maximum (t_{max}) CPU times are reported, in seconds, for each algorithm. The second aspect on which we focus is the quality of the approximation computed by the algorithms. Let us refer to ZE_N, resp. ZE_{TS}, as the approximation calculated by NSGA-II algorithm, resp. Tabu search algorithm. We define $ZE^* \subseteq ZE_N \cup ZE_TS$ as the best known approximation, *i.e.* ZE^* contains the non dominated criteria vectors which belongs to $ZE_N \cup ZE_TS$. It is clear that the efficiency evaluation of a heuristic is not as simple in the multicriteria case as in the single criterion case. Jaszkiewicz ([10]) outlines several measures for evaluating the efficiency of multicriteria heuristics. Basically, we distinguish between measures that provide information about the *comparison* between an approximation and the best known approximation (or even exact set ZE), and measures that provide information about the *spreading* of solutions along the approximation. We use four efficiency measures, three of which being relative to the comparison of two fronts and one being relative to the spreading of solutions.
The three measures relative to the comparison between the fronts ZE_N, resp. ZE_TS, and the reference front ZE^* are defined as follows. Let us denote by $Q_1(Z)$ the percentage of potentially non dominated solutions in Z which are also in ZE^*. We have

$$Q_1(Z) = 100 \times \frac{|ZE^* \cap Z|}{|ZE^*|}.$$

For each couple (n, m), over the 240 generated instances, we calculate the minimum, average and maximum values of $Q_1(Z_N)$ and $Q_1(Z_TS)$. The second mesure is referred to $Q_2(Z)$ as the percentage of solutions in the approximation Z which are also in ZE^*. We have

$$Q_2(Z) = 100 \times \frac{|ZE^* \cap Z|}{|Z|}.$$

For each couple (n, m) we calculate the minimum, average and maximum values of $Q_2(Z_N)$ and $Q_2(Z_TS)$. The last comparison measure we use, is referred to $Q_3(Z)$ as the average distance of the front Z from the reference front ZE^*. We have

$$Q_3(Z) = \frac{1}{|ZE^*|} \sum_{z \in ZE^*} \min_{z' \in Z}(d(z, z'))$$

with $d(z, z')$ the euclidean distance between z and z' in $\mathbb{R}^2$. Again, for each couple (n, m) we calculate the minimum, average and maximum values of $Q_3(Z_N)$ and $Q_3(Z_TS)$.

The last measure we use to evaluate the efficiency of the heuristics estimates the spreading of solutions along the approximation and, consequently, does not require set ZE^*. Let us

refer to $d'(z)$ as the absolute distance of $z \in Z$ and defined by $d'(z) = \min_{z' \in Z, z' \neq z}\{\sum_{k=1}^{2} |z_k - z'_k|\}$. Besides, we denoted by $\bar{d}'$ the average value of the d' distances over set Z: $\bar{d}' = \frac{1}{|Z|}\sum_{z \in Z} d'(z)$. The spreading measure of set Z is thus defined by:

$$Q_4(Z) = \sqrt{\frac{1}{|Z|-1}\sum_{z \in Z}(\bar{d}' - d'(z))^2}$$

Lower is the value of Q_4 for an approximation Z and more uniformly are spread the criteria vector along the approximation. This measure must be definitely used in conjunction with the three previous ones in order to evaluate the quality of the approximation. For the two quality measures, the distance is not computed directly from the calculation of the objective value; instead, $d(z, z')$ is computed indirectly after normalization. The normalization procedure is to convert all objective values between 0 and 1.

Table 1: Computational times required by NSGA-II and Tabu search

Problem size		NSGA-II			SPEA		
n	m	t_{min} (s)	t_{avg} (s)	t_{max} (s)	t_{min} (s)	t_{avg} (s)	t_{max} (s)
10	2	0.4360	0.5095	0.6240	0.1090	0.1447	0.2340
10	4	0.8890	0.9612	1.0450	0.2190	0.2852	0.3750
10	6	1.2640	1.3895	1.6530	0.3270	0.4312	0.5310
20	2	3.4170	3.9114	4.5400	0.8430	0.9449	1.1080
20	4	6.8790	7.9816	9.1570	1.5910	3.6593	434.3080
20	6	10.2020	12.3853	14.9760	2.3710	2.9125	3.4950
30	2	12.8240	16.6635	337.0660	2.6990	3.0250	3.3390
30	4	28.4070	36.2259	45.2860	5.2420	5.9562	6.7390
30	6	44.8500	59.6212	78.9050	7.9710	9.1183	11.3260
40	2	34.2420	48.4803	61.3710	5.9440	6.7443	7.5350
40	4	82.5240	124.0115	169.7900	11.9650	13.3969	20.8110
40	6	156.3750	217.1778	304.1530	18.1580	20.1414	24.1800
50	2	77.6880	130.0120	385.0630	12.1520	13.4617	15.5220
50	4	272.2360	309.6482	408.8770	25.6314	27.4521	31.2470
50	6	311.6800	527.8431	619.2381	37.9030	51.9350	66.7950
60	2	220.6625	432.8893	538.0295	20.9414	23.2478	44.4890
60	4	409.2387	885.1679	1176.9270	34.3506	48.0398	54.3286
60	6	720.4710	1766.3951	2081.8305	50.3585	72.3892	93.1399
70	2	880.2435	1532.1080	2218.3643	47.1055	57.7203	79.4555
70	4	1009.1800	3005.2182	3841.7036	71.0470	98.0459	135.3141
70	6	2279.6215	4116.2081	4499.0305	90.3958	148.4253	180.7300
80	2	2115.2279	3882.2081	4207.3319	104.6900	123.0823	166.1225
80	4	2306.4410	4262.6210	4957.3780	143.5310	197.2940	214.6003
80	6	2658.0947	4973.1598	6001.4584	171.3908	246.0811	320.8520
90	2	2408.1410	4608.2405	5101.7300	152.2697	239.2546	250.7990
90	4	3107.1600	4891.1510	5050.5900	280.8240	399.3001	501.3621
90	6	4301.8030	5309.3600	5759.1000	501.4061	529.9387	590.8120
100	2	3753.7119	5104.0990	5502.8030	331.7504	474.3091	578.4559
100	4	4691.1001	6133.3080	5943.4950	572.3110	681.7902	700.2064
100	6	5001.8150	6627.1449	7129.5500	643.1920	801.2904	1030.1200

Table 2: Efficiency measures for NSGA-II and Tabu search (1)

Pb. size		NSGA-II						SPEA					
		Q_1			Q_2			Q_1			Q_2		
n	m	min	avg	max	min	avg	max	min	avg	max	min	avg	max
10	2	0.0000	0.5217	1.0000	0.0000	0.7228	1.0000	0.0000	0.5854	1.0000	0.0000	0.7430	1.0000
10	4	0.0000	0.6958	1.0000	0.0000	0.8167	1.0000	0.0000	0.6958	1.0000	0.0000	0.8167	1.0000
10	6	0.0000	0.7375	1.0000	0.0000	0.8458	1.0000	0.0000	0.7375	1.0000	0.0000	0.8458	1.0000
20	2	0.0000	0.4395	1.0000	0.0000	0.5784	1.0000	0.0000	0.5750	1.0000	0.0000	0.6202	1.0000
20	4	0.0000	0.5129	1.0000	0.0000	0.6997	1.0000	0.0000	0.5711	1.0000	0.0000	0.7076	1.0000
20	6	0.0000	0.5611	1.0000	0.0000	0.7211	1.0000	0.0000	0.5771	1.0000	0.0000	0.7245	1.0000
30	2	0.0000	0.5937	1.0000	0.0000	0.4022	1.0000	0.0000	0.7556	1.0000	0.0000	0.4257	1.0000
30	4	0.0000	0.5566	1.0000	0.0000	0.5206	1.0000	0.0000	0.6957	1.0000	0.0000	0.5501	1.0000
30	6	0.0000	0.4699	1.0000	0.0000	0.6936	1.0000	0.0000	0.5255	1.0000	0.0000	0.6968	1.0000
40	2	0.0000	0.7889	1.0000	0.0000	0.2239	1.0000	0.0000	0.7316	1.0000	0.0000	0.2387	1.0000
40	4	0.0000	0.6559	1.0000	0.0000	0.3873	1.0000	0.0000	0.7944	1.0000	0.0000	0.4120	1.0000
40	6	0.0000	0.5474	1.0000	0.0000	0.5086	1.0000	0.0000	0.6645	1.0000	0.0000	0.5382	1.0000
50	2	0.0000	0.5135	1.0000	0.0000	0.5636	1.0000	0.0000	0.6194	1.0000	0.0000	0.7111	1.0000
50	4	0.0000	0.3040	1.0000	0.0000	0.7466	1.0000	0.0000	0.3806	1.0000	0.0000	0.8889	1.0000
50	6	0.0000	0.3778	1.0000	0.0000	0.6467	1.0000	0.0000	0.4858	1.0000	0.0000	0.7301	1.0000
60	2	0.0000	0.5455	1.0000	0.0000	0.6115	1.0000	0.0000	0.5142	1.0000	0.0000	0.7699	1.0000
60	4	0.0000	0.4736	1.0000	0.0000	0.7106	1.0000	0.0000	0.5675	1.0000	0.0000	0.6096	1.0000
60	6	0.0000	0.4500	1.0000	0.0000	0.8579	1.0000	0.0000	0.6325	1.0000	0.0000	0.6904	1.0000
70	2	0.0000	0.4338	1.0000	0.0000	0.6503	1.0000	0.0000	0.6516	1.0000	0.0000	0.7011	1.0000
70	4	0.0000	0.4575	1.0000	0.0000	0.8961	1.0000	0.0000	0.6484	1.0000	0.0000	0.7989	1.0000
70	6	0.0000	0.6181	1.0000	0.0000	0.5128	1.0000	0.0000	0.6966	1.0000	0.0000	0.8192	1.0000
80	2	0.0000	0.5706	1.0000	0.0000	0.7318	1.0000	0.0000	0.4034	1.0000	0.0000	0.6808	1.0000
80	4	0.0000	0.6228	1.0000	0.0000	0.8741	1.0000	0.0000	0.6160	1.0000	0.0000	0.6384	1.0000
80	6	0.0000	0.5996	1.0000	0.0000	0.6294	1.0000	0.0000	0.5840	1.0000	0.0000	0.7616	1.0000
90	2	0.0000	0.5810	1.0000	0.0000	0.7756	1.0000	0.0000	0.4568	1.0000	0.0000	0.5079	1.0000
90	4	0.0000	0.6665	1.0000	0.0000	0.7246	1.0000	0.0000	0.5432	1.0000	0.0000	0.7921	1.0000
90	6	0.0000	0.6254	1.0000	0.0000	0.8451	1.0000	0.0000	0.6512	1.0000	0.0000	0.6741	1.0000
100	2	0.0000	0.5885	1.0000	0.0000	0.8894	1.0000	0.0000	0.5488	1.0000	0.0000	0.5259	1.0000
100	4	0.0000	0.4209	1.0000	0.0000	0.6421	1.0000	0.0000	0.5752	1.0000	0.0000	0.7833	1.0000
100	6	0.0000	0.5347	1.0000	0.0000	0.7497	1.0000	0.0000	0.7248	1.0000	0.0000	0.8167	1.0000

6 Conclusions and future directions

In this paper we have solved the bicriteria uniform parallel machine scheduling Problem $Q|r_i, d_i|C_{max}, L_{max}$. Because this problem is NP-hard, two genetic algorithms which are commonly used for solving multicriteria optimization problems have been applied for providing a set of non-dominated solutions. One of these algorithms is based on the non-dominated sorting genetic algorithm-II and the other on the strength Pareto evolutionary algorithm. Computational experiments have been conducted in order to compare the efficiency of the two algorithms. Random data has been generated following a strategy so as to cover as many configurations as possible. To evaluate NSGA-II and SPEA, we have focused on the CPU time required to return an approximation of Pareto front on the one hand, and on the quality of the approximation obtained on the other. Therefore, some useful comparison metrics which take into account the number of Pareto optimal solutions found by the two algorithms, spacing and diversity of these solutions have been

Table 3: Efficiency measures for NSGA-II and Tabu search (2)

Pb. size		NSGA-II						SPEA					
		Q_3			Q_4			Q_3			Q_4		
n	m	min	avg	max	min	avg	max	min	avg	max	min	avg	max
10	2	0.0000	0.4087	1.0000	0.0000	0.2747	1.0000	0.0000	0.0083	1.0000	0.0000	0.1626	1.0000
10	4	0.0000	0.3042	1.0000	0.0000	0.1833	1.0000	0.0000	0.0041	0.0000	0.0000	0.0013	1.0000
10	6	0.0000	0.2625	1.0000	0.0000	0.1542	1.0000	0.0000	0.0209	0.0000	0.0000	0.1176	0.0000
20	2	0.0000	0.4016	1.0000	0.0000	0.4359	1.0000	0.0000	0.0621	1.0000	0.0000	0.2546	1.0000
20	4	0.0000	0.4073	1.0000	0.0000	0.3194	1.0000	0.0000	0.0207	1.0000	0.0000	0.1335	1.0000
20	6	0.0000	0.4248	1.0000	0.0000	0.2847	1.0000	0.0000	0.0042	1.0000	0.0000	0.0251	1.0000
30	2	0.0000	0.2124	1.0000	0.0000	0.6843	1.0000	0.0000	0.1166	1.0000	0.0000	0.2756	1.0000
30	4	0.0000	0.2979	1.0000	0.0000	0.5000	1.0000	0.0000	0.0679	1.0000	0.0000	0.2835	1.0000
30	6	0.0000	0.4662	1.0000	0.0000	0.3296	1.0000	0.0000	0.0128	1.0000	0.0000	0.1513	1.0000
40	2	0.0000	0.1754	1.0000	0.0000	0.7457	1.0000	0.0000	0.3155	1.0000	0.0000	0.1507	1.0000
40	4	0.0000	0.1801	1.0000	0.0000	0.6533	1.0000	0.0000	0.1505	1.0000	0.0000	0.2694	1.0000
40	6	0.0000	0.3094	1.0000	0.0000	0.5241	1.0000	0.0000	0.0441	1.0000	0.0000	0.2883	1.0000
50	2	0.0000	0.4941	1.0000	0.0000	0.5767	1.0000	0.0000	0.0546	1.0000	0.0000	0.0190	1.0000
50	4	0.0000	0.6858	1.0000	0.0000	0.2830	1.0000	0.0000	0.1807	1.0000	0.0000	0.0274	1.0000
50	6	0.0000	0.5910	1.0000	0.0000	0.3848	1.0000	0.0000	0.0422	1.0000	0.0000	0.2554	1.0000
60	2	0.0000	0.5233	1.0000	0.0000	0.3407	1.0000	0.0000	0.0168	1.0000	0.0000	0.2226	1.0000
60	4	0.0000	0.5170	1.0000	0.0000	0.2537	1.0000	0.0000	0.0919	1.0000	0.0000	0.1599	1.0000
60	6	0.0000	0.4153	1.0000	0.0000	0.1792	1.0000	0.0000	0.0848	1.0000	0.0000	0.0641	1.0000
70	2	0.0000	0.5930	1.0000	0.0000	0.3851	1.0000	0.0000	0.0149	1.0000	0.0000	0.1549	1.0000
70	4	0.0000	0.4463	1.0000	0.0000	0.1965	1.0000	0.0000	0.0824	1.0000	0.0000	0.0280	1.0000
70	6	0.0000	0.3208	1.0000	0.0000	0.5386	1.0000	0.0000	0.1981	1.0000	0.0000	0.2060	1.0000
80	2	0.0000	0.4615	1.0000	0.0000	0.2385	1.0000	0.0000	0.0973	1.0000	0.0000	0.0414	1.0000
80	4	0.0000	0.3531	1.0000	0.0000	0.1941	1.0000	0.0000	0.0446	1.0000	0.0000	0.0224	1.0000
80	6	0.0000	0.4139	1.0000	0.0000	0.3537	1.0000	0.0000	0.0677	1.0000	0.0000	0.0821	1.0000
90	2	0.0000	0.3955	1.0000	0.0000	0.2792	1.0000	0.0000	0.0401	1.0000	0.0000	0.1589	1.0000
90	4	0.0000	0.4060	1.0000	0.0000	0.2385	1.0000	0.0000	0.1359	1.0000	0.0000	0.1845	1.0000
90	6	0.0000	0.3780	1.0000	0.0000	0.1647	1.0000	0.0000	0.0451	1.0000	0.0000	0.0483	1.0000
100	2	0.0000	0.5059	1.0000	0.0000	0.1986	1.0000	0.0000	0.0720	1.0000	0.0000	0.0714	1.0000
100	4	0.0000	0.6142	1.0000	0.0000	0.3660	1.0000	0.0000	0.1456	1.0000	0.0000	0.1442	1.0000
100	6	0.0000	0.5090	1.0000	0.0000	0.2169	1.0000	0.0000	0.0586	1.0000	0.0000	0.0365	1.0000

applied. Experimental results have shown that SPEA requires a short time to return an approximation of Pareto optima relative to NSGA-II. We also conclude that SPEA provides more efficient solutions than NSGA-II according to the performance measures, except for some instances for which NSGA-II presents better values. As a conclusion, we have shown in this paper that on the average SPEA outperforms NSGA-II.

In terms of future research, we could consider as extensions to this work, combining the algorithms we have used with local search algorithms which may yield shorter CPU time and better solution quality. We can also consider other criteria for this problem.

References

[1] Bagchi, T.P.: Multiobjective scheduling by genetic Algorithms. Kluwer (1999).

[2] Carlier, J.: Scheduling jobs with release dates and tails on identical machines to minimize the makespan. Eur. J. of Oper. Res. **29** (1987) 298–306.

[3] Carlier, J., Pinson, E.: Jackson's pseudo preemptive schedule for the $Pm|r_j, q_j|C_{max}$ scheduling problem. Ann. of Oper. Res. **83** (1998) 41–58.

[4] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley (2002).

[5] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elistit non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI) (2000) 849–858.

[6] Dell'Amico, M., Martello, S.: Optimal scheduling of tasks on identical parallel processors. ORSA J. on Comp. **7** (1995) 191–200.

[7] Fanjul-Peyro, L., Ruiz, R.: Iterated greedy local search methods for unrelated parallel machine scheduling European Journal of Operational Research **205** 2010 55–69.

[8] Graham, R.-L., Lawler, E.-L., Lenstra, J.-K, Rinnooy Kan, A.: Optimization and approximation in deterministic sequencing and scheduling : a survey. 5, (1979) 287–326.

[9] Haouari, M., Gharbi, A.: An improved max-flow-based lower bound for minimizing maximum lateness on identical parallel machines. Oper. Res. Let. **31** (2003) 49–52.

[10] Jaszkiewicz, A.: Evaluation of Multiple Objective Metaheuristics. In Gandibleux, X., Sevaux, M., Sorensen, K., T'kindt, V. (Eds): Multiple Objective Metaheuristics, Lecture Notes in Economics and Mathematical Systems, **535** (2004) 1–26.

[11] Lancia, G.: Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize makespan. Eur. J. of Oper. Res. **120** (2000) 277–288.

[12] Martello, S., Soumis, F., Toth, P.: Exact and approximation algorithms for makespan minimization on unrelated parallel machines. Disc. App. Math. **75** (1997) 169–188.

[13] R. RUIZ and T. Stutzle An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives European Journal of Operational Research **3(187)** (2008) 1143–1159.

[14] Srinivas, D., Deb, K.: Multi-objective function optimization using non dominated sorting genetic algorithms. Evol. Comp. J. **2(3)** (1994) 221–248.

[15] T'kindt, V., Billaut, J.-C.: Multicriteria Scheduling: Theory, Models and Algorithms. Springer, Berlin (2005).

[16] T'kindt, V., Bouibede-Hocine, K., Esswein, C.: Counting and enumeration complexity with application to multicriteria scheduling. 4'OR. **3(1)** (2005) 1–21.

[17] Tercinet, F.: Méthodes arborescentes pour la résolution des problèmes d'ordonnancement, conception d'un outil d'aide au développement. Ph.D.Thesis in French, Laboratoire d'Informatique, Université François-Rabelais de Tours (2004).

[18] Tercinet, F., Lenté, C., Néron, E.: Mixed satisfiability tests for multiprocessor scheduling with release dates and deadlines. Oper. Res. Let. 32(4) (2004) 326–330.

[19] Van de Velde, S.L.: Duality-based algorithms for scheduling unrelated parallel machines. ORSA J. on Computing **3(1)** (2005) 1–21.

[20] Zitzler, E., Thiele, L.: An evolutionary algorithm for multiobjective optimisation: The strength Pareto approach. Tech. Report **43** (1998), Computer Engineering and Networks Laboratory (TIK), Zurich (Switzerland).