A comparative Study of different optimization techniques for a bi-criteria Flow shop Problem

Malek Rahoual [a, *], Mohamed-Hakim Mabed [b], Clarisse Dhaenens [c], El-Ghazali Talbi [c],

[a] *LaMI, Université d'Evry Val d'Essonne, 91000 Evry - France.*
Email: mrahoual@lami.univ-evry.fr

[b] *Université de technologie de Belfort Montbéliard,Laboratoire système et transport, Département Génie Informatique F-90010 Belfort Cedex France.*
E-mail: hakim.mabed@utbm.fr

[c] *Lifl, University of Lille, Bât.M3, 59655 Villeneuve d'Ascq Cedex France.*
E-mail: {dhaenens, talbi}@lifl.fr

Abstract

The resolution of workshop problems as the Flow Shop or the Job Shop has a great importance in industrial areas. Criteria to optimize are generally the minimization of the makespan time or the tardiness time. However, few resolution approaches take into account those different criteria simultaneously. This paper presents a comparative and progressive study of different multicriteria optimization techniques. Several strategies of selection, of diversity maintaining and hybridization will be exposed. Their performances will be compared and tested. A parallel GA model is proposed. It allows to increase the population size and the limit generations number, and leads to better results. In parallel to the work on the optimization technique, we propose here a new bi-criteria flow shop benchmarks, responding to the need to common problem instances in field of multicriteria optimization.

Keywords: Genetic Algorithm, Multicriteria optimization, Flow Shop, Sharing, Ranking, Hybridization, Local Search.

1 Introduction

The Flow Shop problem has received a great attention [11][23] due to its importance in industrial areas [20][22]. The adopted methods for its resolution vary between exact methods as Branch & Bound, Heuristic search [14][23][24][25][27] and metaheuristics [1][2][15][17][21]. However, the majority of these works study the problem in a unicriterion form and aim principally to minimize the makespan time.

Genetic algorithms (GAs) have turned out to be of great efficiency to deal with combinatorial optimization problems; different extensions to the multicriteria case have been elaborated [4][5][11][12][32][34][35]. The difficulty of the multicriteria case lies in the absence of a total order relation between the solutions of the problem. Considering GAs, this insufficiency appears in the difficulty to design a selection operator that assigns selection probabilities, proportional to the desirability degree of the individuals in the population. Another drawback is related to the sensitiveness of the initial population choice as well as the bad sampling during the selection. This sensitiveness frequently induces the premature loss of the diversity as well as the instability of the search. Hence, designing diversity maintaining techniques is necessity.

The second section of this paper presents the multicriteria flow shop problem. It exposes the different parameters to optimize as well as the constraints to satisfy. In the third section, we will expose the different possibilities of extension of GAs to the multicriteria case. We will also present the different choices of coding, the objective functions, and the genetic operators. In the fourth section, different selection strategies will be presented and their performances compared. In the fifth section, we will expose the implemented diversity maintaining methods and their contribution for the solutions quality. The sixth section will be devoted to the presentation of the hybridization of multicriteria GAs with a local search procedure and its contribution will be underlined. In the next section, a comparison of the performances of the proposed method with other works is made. In the last section, we study the possibility of extension to parallel GAs.

2 Multicriteria Flow Shop problem

The flow shop problem is presented as a set of N jobs $\{J_1, J_2, ..., J_N\}$ to be ordered on M machines. Machines are critical resources: one machine cannot be assigned to two jobs simultaneously. Each job is composed of M consecutive tasks $J_i = \{t_{i1}, t_{i2}, .., t_{iM}\}$, where t_{ij} represents the j^{th} task of the job J_i requiring machine m_j. Subsequently, all the jobs have the same processing sequence on the machines. To each task t_{ij} is associated a processing time p_{ij}, a release time r_i, and a due date d_i (deadline of job J_i).
Scheduling tasks on different machines must optimize some criteria [8][18]. These criteria vary according to the particularities of the considered problem, and generally consist in the minimization of one of the following parameters [33], where C_i denotes the completion time of job J_i:

C_{max} : Makespan time ($max_i\ C_i$)
$\overline{C}$: Mean value of jobs completion time
T_{max} : Maximum tardiness $max_i\ \{max\{0, C_i-d_i\}\}$
T : Total tardiness ($\Sigma_i\ max\ \{0, C_i-d_i\}$)
T_{nb} : Number of jobs delayed with regard to their achievement dates d_i.
F_{max} : Maximum job flux ($\Sigma_i\ (C_i-r_i)$)
$\overline{F}$: Mean job fluxes.

A criterion is said to be regular when a delay in the schedule of a task leads to the degradation of the solution quality.
Formally, let C and C' be two vectors defined as:

$$C = \{C_{11}, C_{12}, ..., C_{21}, C_{22}, ..., C_{N1}, C_{N2}, ...C_{NM}\} \text{ and}$$
$$C' = \{C'_{11}, C'_{12}, ..., C'_{21}, C'_{22}, .., C'_{N1}, C'_{N2}, ...C'_{NM}\}$$

where C_{ij} and C'_{ij} designate the achievement dates of the task t_{ij}. If F is a regular criterion, then $C \leq C' \Rightarrow F(C) \leq F(C')$. This property will allow us, in the following, to only take into consideration semi-active schedules, i.e., schedules where each task is programmed as soon as possible [8][18][19].
We are interested in the study of permutation flow shop problem $F/perm,d_i/(C_{max},T)$, where jobs must be ordered in the same order on all the machines.

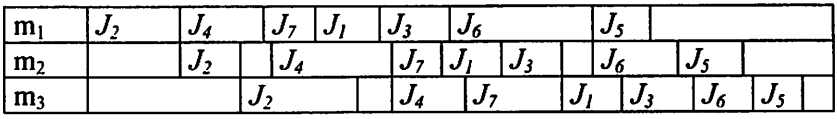

Fig. 1. Schedule example

3 GA and Multicriteria Flow Shop Problem

3.1 Coding

The application of a genetic algorithm to a given problem needs a chromosomal representation of a solution [10] (in our case a representation of a schedule of jobs). As in permutation flow shop jobs are ordered in the same order on every machine, a schedule is completely define by a permutation defining the processing order of the jobs on the machines.

2	4	7	1	3	6	5

Fig. 2. Chromosomal representation of the schedule depicted in Fig. 1

3.2 Determination of the completion times of tasks

The evaluation of a given sequence needs the calculation of the beginning and achievement dates of tasks. Since the criteria to optimize are regular, this calculation is realized by the construction of the schedule, at the earliest, of tasks.

The calculation is made in a recursive manner starting with the first executed tasks as follows:

$$r_{ij} = \begin{cases} 0 : \text{if } J_i \text{ is the first job of the permutation and } j=1 \\ r_{i(j-1)} + p_{i(j-1)} : \text{if } J_i \text{ is the first job of the permutation and } j \neq 1 \\ r_{ij} + p_{ij} \text{ if } J_i : \text{is not the first job of the permutation and } j=1 \\ \max(r_{i(j-1)} + p_{i(j-1)}, r_{ij} + p_{ij}) : \text{otherwise} \end{cases} \quad (1)$$

where r_{ij} is the starting date of the task t_{ij}. $J_{i'}$ is the job preceding J_i in the permutation.
This formula expresses the fact that a task t_{ij} cannot start unless the machine m_j has finished to process the previous task $t_{i'j}$ and the previous task $t_{i(j-1)}$ of the same job is over.

The calculation of r_{ij} is described in the following algorithm (with *S[i]* the i^{th} job of chromosome S, and *free* the availability date of machine):

BEGIN
/ calculation of starting dates of the tasks r_{i1} executed on the first machine */*
free:= 0 ;
for i:= 1 to N do begin $r_{S[i]1}$*:= free; free:=*$r_{S[i]1}$ *+* $p_{S[i]1}$ *; end;*
/ calculation of beginning dates of the tasks t_{ij} executed on other machines */*
for j:= 2 to M / for each machine */*
do begin
free:= 0;
for i:= 1 to N do begin $r_{S[i]j}$*:= maximum(free,* $r_{S[i]j-1}$ *+* $p_{S[i]j-1}$*); free:=* $r_{s[i]j}$ *+* $p_{s[i]j}$*; end;*
end;
END.

3.3 Objective functions

The optimization criteria taken into account in this work are the two following objectives: minimizing the global makespan time as well as the global tardiness.

$$f1 = C_{\max} = \underset{J_i \in jobs}{Max}(l_{iM} + p_{iM}) \quad (2)$$

$$f2 = T = \sum_{J_i \in jobs} \left[Max(0, l_{iM} + p_{iM} - d_i) \right] \quad (3)$$

3.4 Genetic operators

Following Murata and Ishibuchi [17], the mutation and crossover operators are described in the figures below:

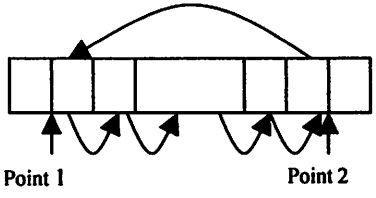

Fig. 3. Mutation Operator

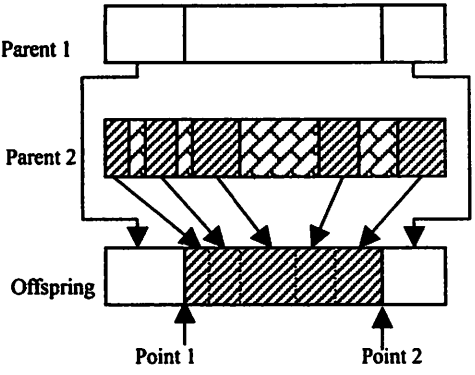

Fig. 4. Crossover operator

The mutation operator consists in choosing randomly two-points of the chromosome, and performing a rotation as shown in fig. 3.
The crossover operator, also called two-points crossover, consists in, first, the random choice of two points. An offspring individual is then generated. In its extremities, the offspring is similar to parent1, and elsewhere remaining jobs are taken in the order encountered in parent 2 (fig. 4). Two-point crossover combine both efficiency and implementation simplicity. One should refer to [10] to a comparative study of different crossover operators for flow shop problem.

4 Selection operator

The absence of a total order relation between the different possible solutions of a multicriteria problem involves confusion in defining the notion of optimality.

4.1 Dominance and Pareto optimality

A well known definition is the dominance notion or Pareto optimality notion. Dominance represents a partial order relation on points of the search space.

$$S_i \text{ domines } S_j \Leftrightarrow \forall k \in [1..nobj] \, f_k(S_i) \leq f_k(S_j) \text{ and } \exists k \in [1..nobj] / f_k(S_i) < f_k(S_j) \quad (4)$$

Where *nobj* designates the number of objectives (f_k) to be optimized.

$$S_i \text{ is Pareto optimal } \Leftrightarrow \forall S_j / S_j \neq S_i, S_j \text{ does not dominate } S_i \quad (5)$$

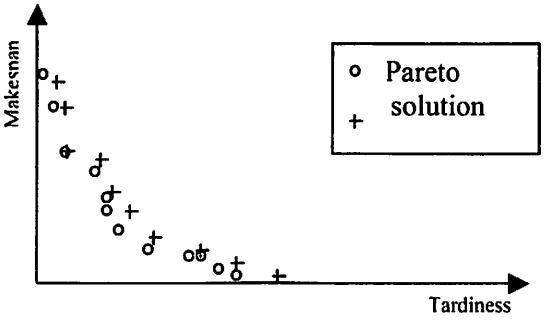

Fig. 5. Non-dominated solutions

Intuitively, a solution S_i is said to dominate the solution S_j only if: S_i is, better or equal to S_j every where (with regard to the objectives), and better than S_j at least for one objective.
The multicriteria optimization purpose is to reach a set of non-dominated solutions that approximates as best as possible the set of Pareto solutions of the problem.

4.2 GAs and multicriteria optimization problems

Many adaptation techniques of GAs to the multicriteria case are found in literature. The principal differences between these methods consist in the way of taking into account the different objectives.
1 – In the selection phase: the multicriteria aspect appears in the ordering of individuals, according to their fitness, in order to be selected.
a– Lexicographical selection: it's based on a pre-established priority order between objectives [28].
b– Ranking selection: it consists in ranking individuals of the population according to different criteria. Therefore, NSGA [29] and NDS [7]

selections use the dominance notion, the WAR selection [7] uses the objective function value.
c– Parallel selection: for each generation, the population is decomposed in *nobj* sub-populations of equal size. To each sub-population is associated an objective to optimize. Individuals of a same sub-population are then selected according to the corresponding objective [26].
d– Selection by weighted sum of variable weight objectives: adopted by Murata and Ishibuchi [17], this method consists, at each generation, in randomly generating a set of *nobj* numbers $\omega_1, \omega_2, \ldots, \omega_{nobj}$ belonging to [0, 1] such as $\omega_1 + \omega_2 + \ldots + \omega_{nobj} = 1$. Individuals are then selected, during this generation, according to the formula $\omega_1 f_1(S) + \omega_2 f_2(S) + \ldots + \omega_{nobj} f_{nobj}(S)$.
2 – In the reproduction phase: The multi-sexual reproduction [6] assigns to every objective a given sex. To each individual of the population is also associated a sex (after generation of the initial population or reproduction). Individuals are allowed to reproduce together only if they are of opposite sexes.

4.3 Implemented selection strategies

In this study, we have implemented and compared six multicriteria selection strategies. The main differences between these methods consist in the way individuals of the population are ordered and selection probabilities are calculated.

4.3.1 Selection by Weighted Sum of Objectives

It is one of the first methods used [12] for multicriteria optimization. Based on the transformation of the problem to a single criterion problem, this method consists in combining the different objective functions in one function, as follows.

$$f(S_i) = \sum_{k \in [1..nobj]} \lambda_k \times f_k(S_i) \qquad (6)$$

The weights λ_k are experimentally taken in the interval [0..1] such as $\Sigma_i \lambda_i = 1$. The rank of an individual in this case is equal to $f(S_i)$. Then, the selection probability, $\pi(S_i)$, of individual S_i is equal to:

$$Rank(S_i) = f(S_i) \tag{7}$$

$$\pi(S_i) = Rank(S_i) / \sum_{j \in [1..tp]} Rank(S_j) \tag{8}$$

where *tp* designates the current population size.

4.3.2 Parallel Selection

Half of the selected elements are selected with regard to their makespan time cost. The remaining *tp*/2 individuals are selected with regard to their tardiness cost.

4.3.3 NSGA Selection

In the NSGA selection [29] (Non-dominated Sorted Genetic Algorithm), the ranks of individuals are calculated in a recursive manner, beginning with the non-dominated individuals of the population.

Rank 1 is associated to the non-dominated set of individuals E_1 of the current population.
Rank 2 is associated to the set of individuals E_2 dominated only by individuals belonging to E_1.
.....
Rank k is associated to the set of individuals E_k dominated only by individuals belonging to $E_1 \cup E_2 \cup \ldots \cup E_{k-1}$.

The selection probability of an individual S_i of Rank n in the population follows Baker expression [5]:

$$\pi(S_i) = \frac{S \times (tp + 1 - R_n)}{tp \times (tp - 1)} \tag{9}$$

Where S designates the selection pressure and

$$R_n = 1 + |E_n| + 2 \times \sum_{j \in [1..n-1]} |E_j| \tag{10}$$

4.3.4 NDS Selection

In the NDS selection [7] (Non-Dominated Sorting), the rank of an individual is equal to the number of solutions dominating the individual plus one.

$$Rank(S_i) = |\, S_j \in Population / S_j \text{ dominates } S_i \,| + 1 \qquad (11)$$

The selection probability is calculated using formula (9).

4.3.5 WAR Selection

The weighted Average Ranking [3] consists in calculating the rank of each individual of the population with regard to the different objectives separately. The rank of an individual is calculated as the sum of ranks.

$$Rank(S_i) = Rank\_Makespan(S_i) + Rank\_Tardiness(S_i) \qquad (12)$$

The selection probabilities are calculated as for NSGA selection.

4.3.6 Elitist Selection

The elitist selection consists in maintaining an archive population PO^* that contains so for the best non-dominated solutions encountered during the search. This population will participate to the selection and reproduction stages [10].

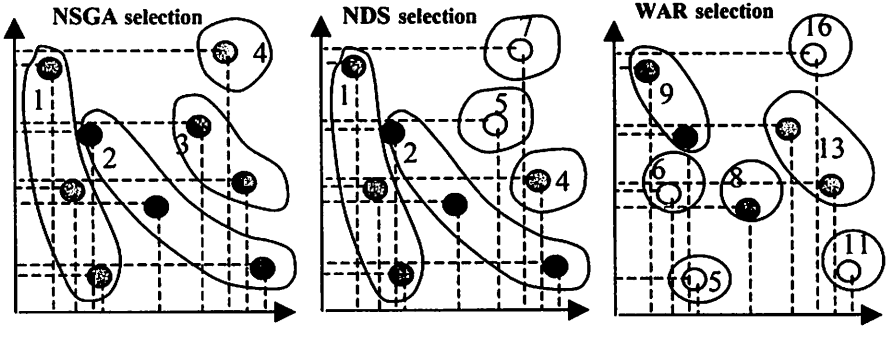

Fig. 6. Comparison between *NSGA, NDS* and *WAR selections*

In this case, the selection probability of an individual S_i of rank n in the current population corresponds to the following expression:

$$\pi(S_i) = \frac{(tp - A)}{tp} \times \frac{S \times (tp + 1 - R_n)}{tp \times (tp - 1)} \quad (13)$$

Then, element of the Pareto population are chosen with a probability A/t_p. Hence, "A" determines the expected number of individuals selected from the PO^* set. Actually, all the selection methods keep an archive containing the best solutions encountered during the search (Pareto set). The particularity of the elitism is to let this population participate during the selection phase. The rank of individuals is calculated with the NSGA technique. The following procedure describes the selection procedure, of an individual of the current population, in the elitist selection case.

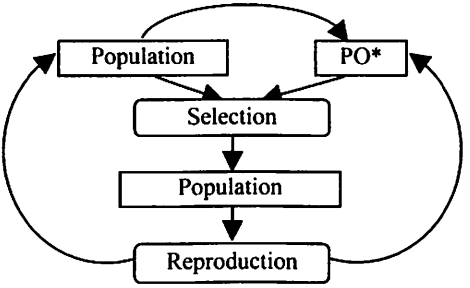

Fig. 7. Elitist selection strategy

4.4 Scheme of the GA

We describe here the principal components of the GA.

BEGIN
1. *Generate randomly the initial population.*
2. *Calculate the PO^* of the current population.*
3. *While the number of limit generations is not reached.*
   Do begin
   a. *Calculate the rank of individuals with regard to the adopted selection type.*
   b. *Calculate the selection probabilities of each individual.*
   c. *Select tp individuals from the current population and eventually from the population PO^* (elitist selection) according to their probabilities, and form the intermediate population.*
   d. *For i:= 1 to tp /* Crossover phase*/*
      Do begin

Select a pair of individuals from the intermediate population. According to a crossover probability Pc
Choose either (a) do the crossover and insert the offspring individual in the new population or (b) insert one of the two parents in the new population.

e. For i:= 1 to tp Do
Mute the individual S_i of the population with a probability P_m.

f. Update PO: PO*=Non-dominated solutions of (PO*+Population.)*

End

END.

4.5 Performances of the different selection strategies

To compare the performances of the 6 implemented selection strategies, tests were effectuated on Heller's [13] problem with 20 jobs * 10 machines. Tests show an effective improvement of the search with the introduction of the elitism in the selection phase. The non-Pareto strategies, represented here by the Weighted Sum Selection and the Parallel Selection, seem to be non-adapted to the multicriteria case. The three selection strategies NSGA, NDS and WAR are of almost identical performances, with a slight supremacy of NSGA and NDS methods. The parameters of the different methods are described in table 1, the population size is 200 and the limit generation number is 15000:

Table 1. Parameters of the different selection strategies

Weighted sum	$\lambda 1 = 0.5$	$\lambda 2 = 0.5$
NSGA	S = 1.7	
NDS	S = 1.7	
WAR	S = 1.7	
Elitist	S = 1.7	A = 5

4.6 Effect of the parameter "A" on the elitist selection strategy

As the contribution of the elitism has been proved, it's interesting to know the impact of the choice of the parameter "A". The choice of this parameter value influences considerably the balance between exploitation and exploration.

Effectively, a high value of A intensifies the exploitation of the good found solutions. A low value of A favors the exploration of new horizons of the search space.
Figures 9 and 10 show the evolution of the Pareto front obtained after different generation numbers for values of the parameter equal to 1 and 4.

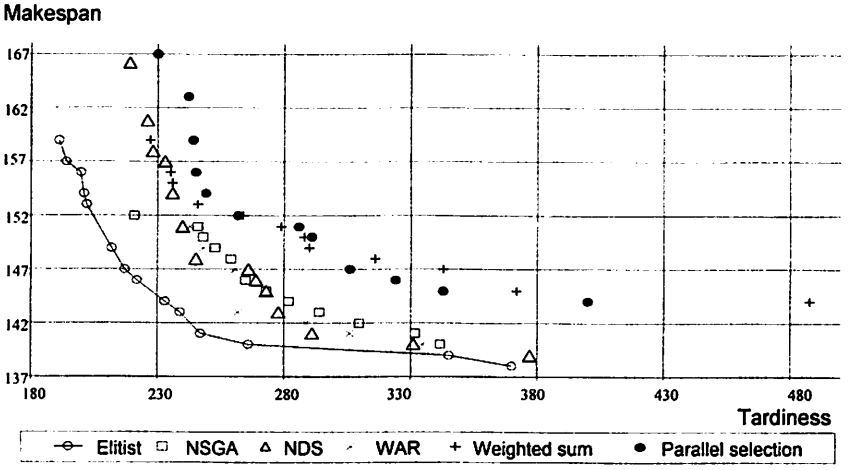

Fig. 8. Comparison of the different selection strategies

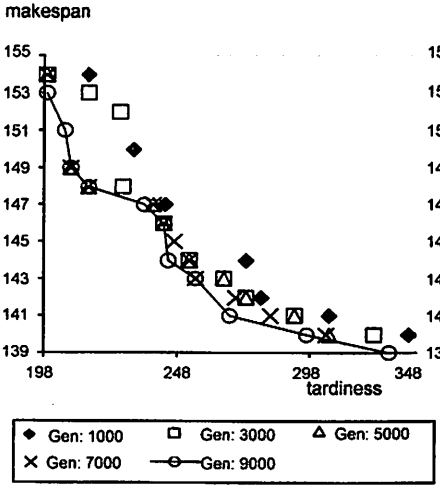

Fig. 9. Search evolution for A = 1

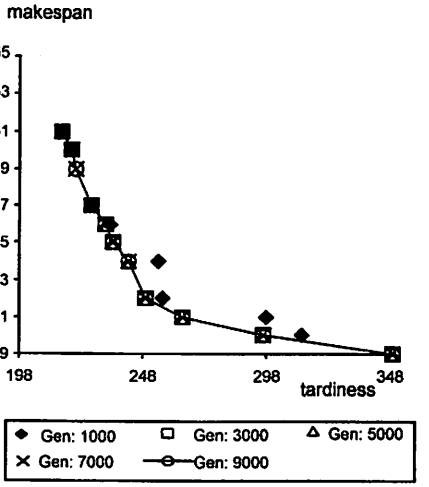

Fig. 10. Search evolution for A = 4

For low values of A (A=1), the Pareto frontier convergence is slow, contrary to results obtained for A=4.

5 The diversity maintaining

Classical GAs are reputed to be very sensitive to the initial population choice and the bad sampling during the selection. This fragility is observable on the diversity loss, or what is also called the genetic drift. To face this drawback, several approaches, aiming to maintain the diversity in the population, were proposed.

1. Introduction of new individuals: this technique consists in generating, randomly and during the search, new individuals and inserts them into current population.
2. Stochastic Universal Sampling (SUS): Baker [32] proposed a technique that aims to eliminate the drifts of the probabilistic choice that make the effective offspring number of an individual different from those expected theoretically. The SUS selection proceeds by choosing probabilistically the first individual only (by turning a biased wheel). Other individuals are then selected by turning the wheel, from its current position, with a constant angle each time.
3. Distance maintaining: Proposed by Mauldin [16], this technique defines a value k as a minimal distance authorized between two individuals of the population. When an individual is generated, its distance with regard to the rest of the population is calculated. Then, if an individual is judged similar to another individual (distance < k), then it will be mutated until the distance k is respected.
4. Crowding [16]: Holland was the first to suggest the use of this operator in the replacement phase of GAs. After the generation of an individual, this operator replaces, in the population, the individual most similar to it.
5. Neighborhood restriction: Other works, proposed for the diversity maintaining are based on neighborhood restriction [9]. The principle is to allow reproduction between two individuals only if they are similar. Other works prevent reproduction between similar individuals to avoid incest.
6. Ecological niching: The sharing principle consists in the degradation of the fitness of individuals belonging to search space regions with a high concentration of solutions. This process has the effect to:

- change the problem scenery,
- favor the dispersion of the solutions in the search space,
- form sub-populations of similar individuals.

The degradation of the fitness of an individual is realized thanks to a function called sharing function *sh*. In a minimization scheme, the revised objective function f of an individual S_i noted $f'(S_i)$ is equal to the original function f multiplied by the sharing counter (niching counter) of the individual $m(S_i)$.

$$f'(S_i) = f(S_i) \times m(S_i) \tag{14}$$

The sharing counter calculates the similarity degree of an individual with the remaining individuals of the population.
The sharing function *sh* is defined as follows:

$$m(x) = \sum_{y \in pop} sh(dist(x,y)) \tag{15}$$

$$sh(dist(x,y)) = \begin{cases} 1 - \left(\dfrac{dist(x,y)}{\gamma} \right)^{\alpha} & \text{if } dist(x,y) < \gamma \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

In formula (16) the constant γ designates the non-similarity threshold (niche size), i.e., the distance from which two individuals x and y are not considered as belonging to the same niche any more. The constant α allows to control and regulate the form of the function *sh*. Depending on whether the distance between two individuals is calculated in the decision space (chromosomal representation of an individual) or in the objective space (fitness of individuals), three approaches are possible: genotypic sharing, phenotypic sharing and combined sharing.

5.1 Genotypic Sharing

In this case the distance between individuals is calculated according to the difference between chromosomes. Since a schedule is represented by a permutation, the distance between two schedules (two permutations) is then equal to:

$$dist1(x, y) = |\{(i, j) \in J \times J / i \text{ precedes } j \text{ in the solution } x \text{ and } j \text{ precedes } i \text{ in } y\}| \quad (17)$$

This means that the distance between two individuals x, and y is equal to the number of order inversions between them.

5.2 Phenotypic Sharing

The distance in this case is taken as the difference between the costs of the two individuals. f_i designates the i^{th} objective function.

$$dist2(x,y) = \sum_{i=1}^{nobj} |f_i(x) - f_i(y)| \quad (18)$$

5.3 Combined Sharing

This case represents the combination of the two first approaches cited above. Both, the distances (genotypic and phenotypic) are used. The function *sh*, in this case, takes the following form:

$$sh(x,y) = \begin{cases} 1 - \dfrac{dist1(x,y)}{\gamma 1} & \text{if } dist1(x,y) < \gamma 1,\ dist2(x,y) \geq \gamma 2 \\ 1 - \dfrac{dist2(x,y)}{\gamma 2} & \text{if } dist1(x,y) \geq \gamma 1,\ dist2(x,y) < \gamma 2 \\ 1 - \dfrac{dist1(x,y)\,dist2(x,y)}{\gamma 1 \gamma 2} & \text{if } dist1(x,y) < \gamma 1,\ dist2(x,y) < \gamma 2 \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

5.4 Experiments

To show the contribution of the different diversification methods on the search, many tests were performed on the Heller problem 20*10, with a limit generation number equal to 50000 and using the same parameters

as those described above for the elitist selection strategy. Results are presented on figure 11.
The parameters concerning the diversification strategy are: α=0.9, γ1=4 and γ2=1. We notice that the contribution of the genotypic diversification is slightly appreciable compared to the results obtained by the phenotypic diversification. However, the diversification in decision space is distinguished by an efficient set of solutions not found by the phenotypic diversification. The composed diversification presents a best solution quality than the two previous methods. Noting that the diversification contribution appears only after a considerable number of generations, hence the necessity to take a very high limit generation number (>40000).

6 Hybridization with a Local Search

We were interested by the use of a Local Search (LS) procedure [32] as a mean of acceleration and refinement of the Genetic Search. In this case, the idea is to run the GA first in order to approach the Pareto frontier.
Then the local search will be used for the refinement of the found solutions (see fig. 12).
In order to generate the neighborhood of a given solution, we were inspired by the mutation operator. Thus, the neighborhood of a solution corresponds to the set of permutations generated after the rotation of a pair of jobs (see fig. 13).

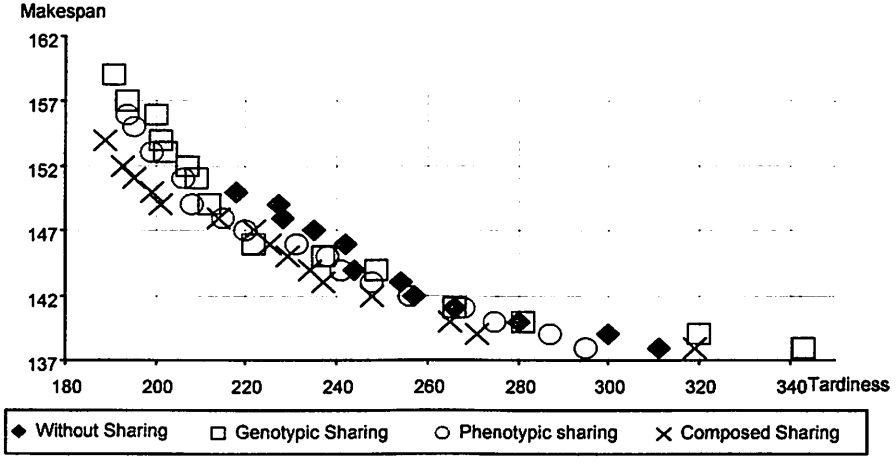

Fig. 11. Diversification contribution

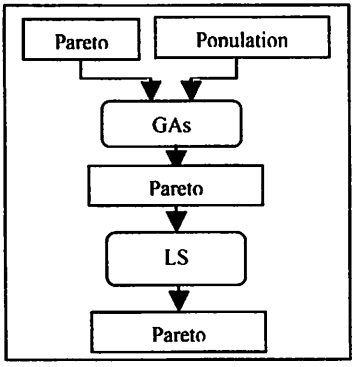

Fig. 12. Hybridization of the GA with LS

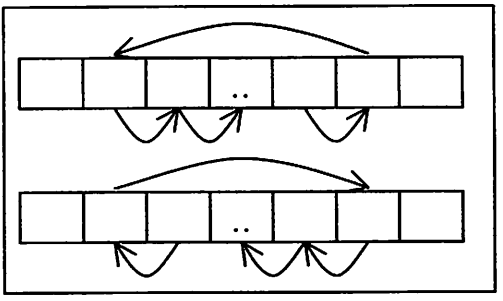

Fig. 13. Generation of neighbors in LS

The hybridization [32] process consists in generating the neighborhood for each individual of the Pareto population. The non-dominated neighbors in the Pareto population are inserted in this one. The solutions belonging to Pareto and dominated by the neighborhood of one solution are suppressed. This process is iterated until no neighbor of any Pareto solution is inserted in the Pareto population. Fig. 14 describes schematically the Local Search process.

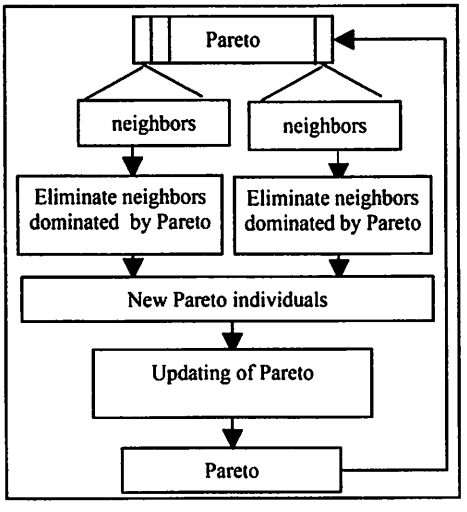

Fig. 14. Local Search in action

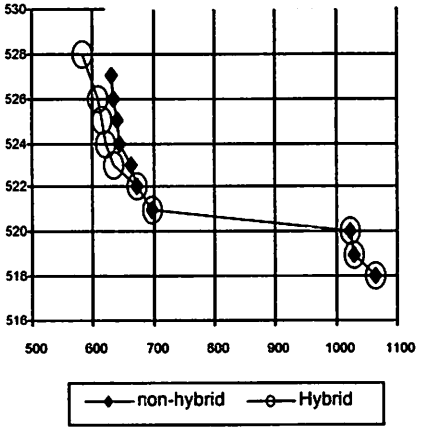

Fig. 15. Local search contribution

The tests of performance of the hybrid GA with the local search show that LS is of no interest for little size problems, notably Heller 20*10. However, the hybridization contribution appears as soon as the size of the problem increases. The tests presented in fig. 15 are realized on Heller problem with 100 jobs and 10 machines.

7 Tests & performances

Comparative studies of multicriteria optimization works, as the Flow shop problem, are generally confronted to the absence of benchmarks. This is due to the difference between the criteria to optimize and the specificities of the treated problem. For the Flow Shop problem, only the makespan may be considered as an effective comparison criterion, because such benchmarks exist. Hence, one quality of a multicriteria algorithm may be its ability to find extrema solutions (that are at the extremities of the Pareto front) approaching those found by the unicriterion algorithm.

Problems "taxx", due to Taillard [5], are classical problems for minimizing the makespan in a flow Shop context. They offer a good comparison support. These problems are often cited in literature [2][21] and each one is characterized by a value LB, corresponding to the makespan lower bound, and a value UB corresponding to the best solution obtained for this problem.

The algorithm used by Taillard to generate those benchmarks, consists in the generation of a problem instance from the three entry data: N, M and an integer number SEED. We extended this process in order to generate instances with tardiness. The following algorithm describes our bicriteria Flow shop instances generator, where N, M, SEED and UB are global variables given by the user. This algorithm generates instances as Taillard ones with a time-limit d_i for every job.

```
function unif1: integer
begin
    m = 2147483647;
    a = 16807;
    b = 127773;
```

```
    c = 2836;
    k = SEED/b ;                    /*integer division*/
    SEED = a * (SEED modulo b) - k * c;
    if (SEED < 0) SEED = SEED + m;
    valeur_0_1 = SEED/m ;          /*non-integer div.*/
    return inf + (valeur_0_1 * (sup - inf + 1));
end

function unif2: integer
begin
    m = 2147483647;
    a = 16807;
    b = 127773;
    c = 2836;
    k = SEED2/b;                    /*integer division*/
    SEED2 = a × (SEED2 modulo b) - k × c;
    if ( SEED2 < 0)  SEED2 = SEED2 + m;
    valeur_0_1 = SEED2/m          /*non-integer div.*/
    return (12 × UB/30 + value_0_1 × 17 × UB/30);
end

procedure Generate_instance
begin
    SEED2 = SEED * 2;
    for j = 1 to M
      for i = 1 to N do p_ij = unif1;
                                    /*p_ij: duration of t_ij */
    for i = 1 to N do d_i=unif2;
                                /*d_i: deadline of job J_i*/

 end.
```

Table 2 presents results of tests, performed on 8 Taillard instances extended to the bicriteria case, noted *ma_taxx_bi*. *MM* designates the best obtained makespan and *MR* the minimal registered tardiness. *Dev* measures the difference between the best obtained makespan and *UB*. $|PO|$ calculates the number of solutions of the obtained Pareto frontier.

Results in table 2 show the algorithm capacity to find extremum solutions. The obtained Pareto set is sufficiently spread, what offers a good sampling of the Pareto frontier. The deviations, although slight, are justifiable. Indeed, the proposed algorithm is very independent of the

treated problem, since no heuristic, guaranteeing minimizing the makespan or the tardiness is used.

The small size of the |PO| set is due to the strongly correlated characteristics of the two objectives to optimize.

The proposed algorithm, not guided by any heuristic adapted to the flow shop problem, converges very slowly. The parallelization of the algorithm is then imposed as a sensible mean to fill this deficiency. The reduction of the processing time of a generation will compensate the augmentation of the population size or of the limit generations number.

Table 2. Performance of the hybrid genetic algorithm

Problem	Dim. N x M	SEED	LB	UB	MM	Dev %	MR	|PO|	Nb of gener.
ma_ta01_bi	20x5	873654221	1232	**1278**	**1278**	0	453	4	$50\ 10^3$
ma_ta02_bi	20x5	379008056	1290	**1359**	**1359**	0	491	6	$50\ 10^3$
ma_ta11_bi	20x10	587595453	1448	**1582**	**1586**	0.25	1508	28	$80\ 10^3$
ma_ta12_bi	20x10	1401007982	1479	**1659**	**1674**	0.9	1342	21	$80\ 10^3$
ma_ta21_bi	20x20	479340445	1911	**2297**	**2330**	1.43	1062	32	$20\ 10^4$
ma_ta31_bi	50x5	1328042058	2712	**2724**	**2735**	0.4	3629	11	$20\ 10^4$
ma_ta41_bi	50x10	1958948863	2907	**3037**	**3126**	2.93	6653	24	$20\ 10^4$
ma_ta51_bi	50x20	1539989115	3480	**3886**	**3990**	2.67	11379	32	$30\ 10^4$

8 A parallel genetic algorithm

The parallelization mechanism, we have adopted, is based on the distributed model [31]. This approach favors the subdivision of the population into sub-populations of equal size. Each processor executes the GA on the sub-population assigned to it. On determined laps of time, the different GAs exchange their best individuals.

We have opted for a ring communication topology (fig. 16) in order to minimize the interprocess communication rate and also to maintain the connexity of the graph.

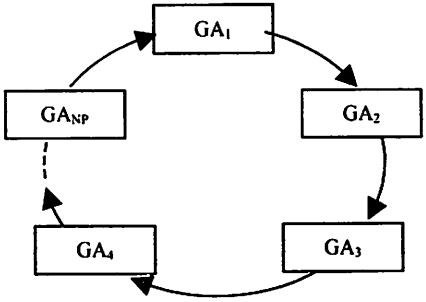

Fig. 16. Ring communication topology

With this scheme, good individuals may be spread to all sub-populations after a certain number of generations.

We describe below the migration process.

BEGIN
1. Generate the initial sub-population P_0, i = 0,
2. While the limit generation number is not reached
do - selection;
- Reproduction
- If (i modulo Period) = 0 then
- if the sent individual is received then choose a non-dominated individual and send it to the process on the right
- if the process is available then receive an individual from the left process
- acknowledge reception to the left process.
- insert the received individual in the population by randomly choosing a victim
- i:= i + 1
Done
END.

The advantages of such a model are:

- Reliability and stability: The blocking of a process or its slowness doesn't influence the behavior of other processes.
- Reduced communication rate: during each migration wave, we will have a maximum of 2 x NC exchanged messages – NC individuals transmitted and NC acknowledgements of receipt.
- Independence in regard to the environments: the algorithm is independent of the execution platform (homogeneity, number of machines, characteristics).

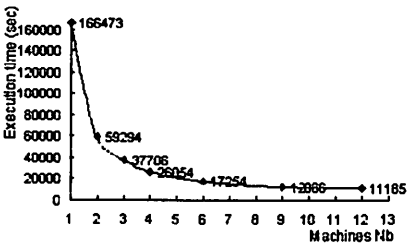

Fig. 17. Execution time of the parallel GA for ma_ta21_bi in function of the number of machines

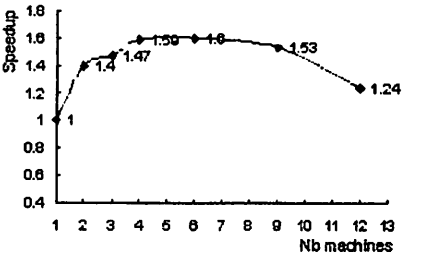

Fig. 18. Efficiency of the parallel GA for ma_ta21_bi

Figures 17 and 18 show respectively the variation of the processing time (in seconds) and the efficiency of the parallel GA with regard to the machines (processes) number. The tests were effectuated on a cluster of stations SUN ULTRA1 and the parallel program are implemented using C/PVM (Parallel Virtual Machine) Library. The execution time gain being proved, this may encourage us to increase the population size as well as the limit generation number in order to reach better solutions. Results are shown in table 3 for the two instances ma_ta11_bi and ma_ta21_bi.

Moreover, as shown in figures 19 and 20, the use of large populations distributed on different GAs and the increase of the maximal generation number improve the quality of solutions obtained.

Table 3. Improvement of the quality of the solutions for (1): ma_ta11_bi and (2): ma_ta21_bi

Pb.	UB	Sequential GA with tp=200					Parallel GA with tp=300				
		MM	Dev %	MR	|PO|	Nb gen	MM	Dev %	MR	|PO|	Nb gen
(1)	1582	1586	0.25	1508	28	80000	1583	0.06	1431	32	300000
(2)	2297	2330	1,43	1062	32	200000	2305	0.34	1057	29	300000

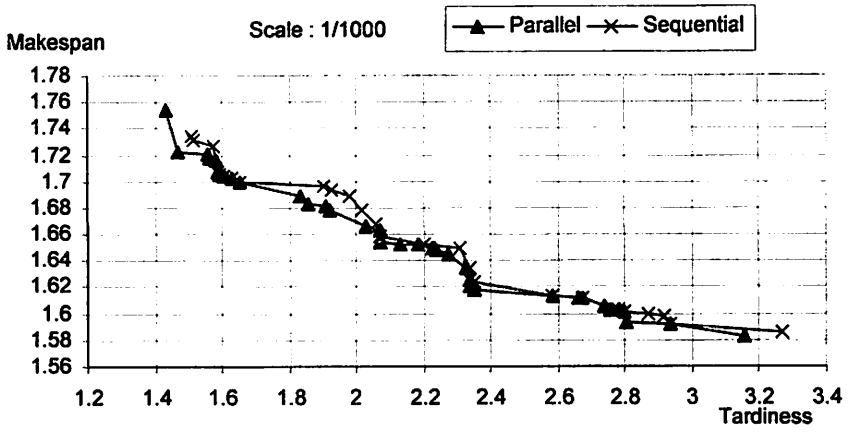

Fig. 19. Parallelization impact on the Pareto frontier for ma_ta11_bi

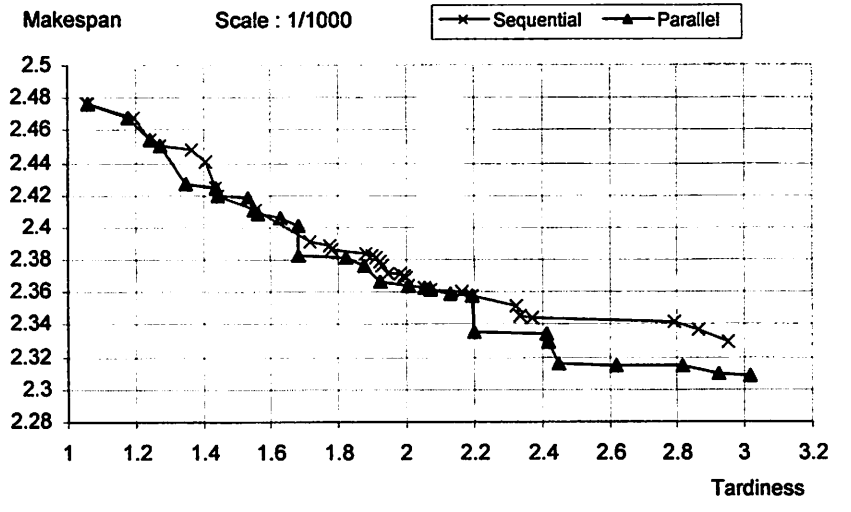

Fig. 20. Parallelization impact on the Pareto frontier for ma_ta21_bi

9 NEH heuristic for the generation of the initial population

In this section, we propose a probabilistic method of NEH heuristic, due to Nawaz and Ham [2], which generates a population of individuals of

good quality. In this method a solution is generated by a progressive elaboration of several partial sequences. At each iteration, a job J_i is selected in order to be inserted, in the partial sequence δ, with a probability equals to:

$$\left(\sum_{j=1}^{M} p_{ij}\right)^{\alpha} \Big/ \sum_{k\in\Phi}\left(\sum_{j=1}^{M} p_{kj}\right)^{\alpha} \qquad (20)$$

where ϕ corresponds to the set of jobs not yet planned.
Once a job is selected, it's inserted in the sequence δ at the position that provides the best makespan. α represents a control parameter of the heuristic. It belongs to the interval $]0,+\infty[$. A small value of α leads to modest quality solutions well spread in the search space, while a high value generates good quality solutions concentrated in attractive regions of the search space.
We give below the initial population generation procedure based on NEH heuristic.

BEGIN
For each individual of the initial population
do begin
Φ = *{set of all jobs}*; δ = *nil;*
$\forall J_i$ *calculate the total time* $P[i] = \Sigma_j(P_{ij})$
end;

choose job J_m *with probability* $P[m]^{\alpha}/\Sigma(P[k]^{\alpha})$;
$P[m] = 0$; $\Phi = \Phi - J_m$;
choose job J_n *with probability* $P[n]^{\alpha}/\Sigma(P[k]^{\alpha})$;
$P[n] = 0$; $\Phi = \Phi - J_n$;
If $J_m \rightarrow J_n$ *is better than* $J_n \rightarrow J_m$ *then* $\delta = J_m \rightarrow J_n$ *else* $\delta = J_n \rightarrow J_m$

For $k = 3$ *to* N
Do begin
choose J_m *with probability* $P[m]^{\alpha}/\Sigma(P[k]^{\alpha})$;
$P[m] = 0$; $\Phi = \Phi - J_m$;
Generate the set of the constructed sequences by introduction of J_m *in* δ *at one of the k possible positions;*
δ = *the best new sequence;*
end;
END.

Conclusion and perspectives

In this work, we have constructed our algorithm by a progressive introduction of concepts such as selection, diversity maintaining, hybridization and parallelization. At each stage we have shown the contribution of the introduced mechanism, what allows us to formulate the following conclusions.

Pareto selection strategies (NSGA, NDS, WAR) are well adapted to the multicriteria case. Efficiency of such methods is still improved with the introduction of Elitism during the selection phase. Elitism may lead to a premature convergence of the search, so, it is necessary to adequately choose parameters A and S. However, the risk of a genetic drift and of instability of the search is always present. Diversification strategies seem to be the privileged mean to prevent such problems. Three variants of the method, based on the ecological niching, were developed. The phenotypic sharing appears to be the most interesting. This interest is related to the fact that a large and well spread Pareto frontier is desirable. However, the genotypic diversification may yield good results. The combination of both concepts improves quality of solutions found.

Then, the local search was used as a mean of refinement and acceleration of the search (fig. 21). The hybridization scheme consists in first, executing the GA in order to get a first approximation of the Pareto frontier and then executing the local search which has the merit to improve the solutions (find the local optima of the search regions). Moreover, experiments have shown that an iterative execution of the local search, after each generation of the GA, doesn't give better results and doesn't justify the important cost of the induced processing time. The contribution of hybridization appears for problems of important size only.

To test the efficiency of such a method, we are faced to two problems: multicriteria flow-shop problems are not standard (criteria to optimize are not identical), and the absence of benchmarks. In this work, we have proposed to extend Taillard instances [30] to the bicriterion case. The tests performed show a great ability of the hybrid GA to reach solutions with a small makespan.

Due to the selection and diversification mechanisms, the proposed GA is quite slow. Parallelization is then presented as an interesting mean to overcome this drawback. The reduction of the required time for a generation incites to use larger populations and to increase the limit generations number, which improves the quality of the Pareto frontier.

As perspectives of this work, we may study other hybridization mechanisms of the genetic algorithm, with other methods, like tabu search, for example as this method has been successfully used for the Flow shop problem

We also need to test the strength of the method for Flow Shop problem with more than two criteria. A graphical comparison being impossible in this case, more elaborated tests of performance methods must be used, such as the contribution notion [5] and entropy [5][32][36]. The extension of the method to the job shop problems may also be studied.

Bibliography

1. Adenso-Diaz, B.: Restricted neighborhood in the tabu search for flow-shop problem. European Journal of Operational Research, Vol 61, 1991, page 318.
2. Allahverdi, A.: The two- and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime, *European Journal of Operational Research*, Volume 147, Issue 2, 1 June 2003, Pages 373-396.
3. Allahverdi, A., Aldowaisan, T.: No-wait flowshops with bicriteria of makespan and maximum lateness, European Journal of Operational Research, Volume 152, Issue 1, 1 January 2004, Pages 132-147.
4. Bagchi, T.P.: Multiobjective Scheduling by Genetic Algorithms. Kluwer Academic Publishers, Boston, 1999.
5. Ben-Daya, M., Al-Fawzan, M.: A tabu search approach for the Flow shop scheduling problem. European Journal of Operational Research, Vol 109, Page 88, 1998.
6. Bentley, P.J., Wakefield, J.P.: Find an acceptable Pareto-optimal solutions using multiobjective Genetic Algorithms. Springer Verlag, London Page 231, June 1997.
7. Brah, S.B., Hunsucker, J.L.: Branch & Bound algorithm for the flow-shop with multiple processors. European Journal of Operational Research, Vol 51, 1991; p. 88.
8. Coello, C.A.C., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.

9. Coello, C.A.C.: An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends, Congress on Evolutionary Computation, 1999.
10. Cotta, C., Troya, J.M.: Genetic Forma Recombination in Permutation Flowshop Problems, journal Evolutionary Computation, volume 6, n°1,pp 25-44, 1998.
11. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK, 2001.
12. Ehrgott, M., Gandibleux, X.: (editors). Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, Kluwer Academic Publishers, Boston, 2002.
13. Eshelman, L.J., Shaffer, J.D.: Preventing premature convergence in genetic algorithms by preventing incest. Fourth international conference on genetic algorithms, San Mateo, California, Morgan Kaufmann Pub, 1991, p. 115.
14. Fonseca, C. M., Fleming, P.J.: Multiobjective genetic algorithms made easy: selection, sharing and mating restrictions. In IEEE Int. Conf On Genetic Algorithms in Engineering System: Innovations and Applications, Page 45, Sheffield, UK, 1995.
15. Fujita, K., Hirokawa, N., Akagi, S., Kimatura, S., Yokohata, H.: Multi-objective optimal design of automotive engine using genetic algorithm. In Proceedings of DETC'98 - ASME Design Engineering Technical Conferences, page 11, 1998.
16. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In second Int. Conf. on Genetic Algorithms ICGA'2, page 41, 1987.
17. Gonzalez, T., Sahni, S.: Flowshop and Job-shop Schedules: Complexity and Approximation. Operational Research, Vol 26, N°1, 1978, page 36.
18. Hajela, P., Lin, C.Y.: Genetic search strategies in multicriterion optimal design. Structural Optimization, (4) Page 99, 1992.
19. Heller, J.: Some Numerical Experiments For a MxJ Flow Shop And Its Decision Theoretical Aspects. Operational Research, Vol 8, 1960, page 178.
20. Ishibuchi, H., Misaki, S., and Tanaka, H.: Modified simulated annealing algorithms for the flow shop sequencing problems. European Journal of Operational Research, 81, 388-398, 1995.
21. Köksalan, M., Keha, A.B.: Using genetic algorithms for single-machine bicriteria scheduling problems, *European Journal of*

Operational Research, Volume 145, Issue 3, 16 March 2003, Pages 543-556.
22. Mabed, M-H, Rahoual, M, Talbi, E-G, and Dhaenens, C.: Algorithmes génétiques multicritères pour les problèmes de Flowshop, MOSIM'01, p.848-849, Troyes, 2001.
23. Murata, T., Ishibuchi, H.: A Multi-objectives Genetic Local Search Algorithm and Its Application Flow-shop Scheduling. IEEE Transaction System. Vol 28, N°3, 1998, Page 392.
24. Nagar, A., Heragu, S. S., and Haddock, J.: Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research, 81, 88-104, 1995.
25. Nawaz, M., Enscore, E. E. Jr., and Ham, I.: A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. OMEGA, 11, 91-95, 1983.
26. Nowicki, E.: The permutation Flow shop with buffers: A tabu search approach. European Journal of Operational Research, Vol 116, Page 205, 1999.
27. Rajendran, C.: Two-stage flowshop scheduling problem with bicriteria. Journal of Operational Research Society, 43, 871-884, 1992.
28. Rajendran, C.: Heuristics for scheduling in flowshop with multiple objectives. European Journal of Operational Research, 82, 540-555, 1995.
29. Sayin, S., and Karabati, S.: A bicriteria approach to the two-machine flow shop scheduling problem. European Journal of Operational Research, 113, 435-449, 1999.
30. Sivrikaya-Serifoglu, F., and Ulusoy, G., A bicriteria two-machine permutation flowshop problem. European Journal of Operational Research, 107, 414-430, 1998.
31. Sridhar, J., and Rajendran, C., Scheduling in flowshop and cellular manufacturing systems with multiple objectives – A genetic algorithmic approach. Production Planning & Control, 7, 374-382, 1996.
32. Srinivas, N., Deb, K.: Multiobjective optimisation using non-dominated sorting in genetic algorithms. Evolutionary Computation 2(8): Page 221, 1995.
33. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research, Vol 64, Page 278, 1993.
34. Talbi, E-G., Rahoual, M., Mabed, M-H., and Dhaenens, C.: New genetic approach for multicriteria optimization problems: Application to the flow-shop, Lecture Notes in Computer Science, LNCS No. 1993, Springer Verlag, pp. 416-428, 2001.

35. T'kindt, V., Billaut, J-C.: Multicriteria Scheduling. Theory, Models and Algorithms, Springer, Berlin, 2002.
36. Zitzler, E., Thiele, L.: Multiobjective evolutionary Algorithms: A comparative case study and the strength Pareto approach, IEEE Transaction on Evolutionary Computation, 3(4): 257-271, November 1999.