RAIRO Operations Research
Will be set by the publisher

MEMOTS: A MEMETIC ALGORITHM INTEGRATING TABU SEARCH FOR COMBINATORIAL MULTIOBJECTIVE OPTIMIZATION*

Thibaut Lust and Jacques Teghem[1]

Abstract. We present in this paper a new multiobjective memetic algorithm scheme called MEMOX. In current multiobjective memetic algorithms, the parents used for recombination are randomly selected. We improve this approach by using a dynamic hypergrid which allows to select a parent located in a region of minimal density. The second parent selected is a solution close, in the objective space, to the first parent. A local search is then applied to the offspring. We experiment this scheme with a new multiobjective tabu search called PRTS, which leads to the memetic algorithm MEMOTS. We show on the multidimensional multiobjective knapsack problem that if the number of objectives increase, it is preferable to have a diversified research rather using an advanced local search. We compare the memetic algorithm MEMOTS to other multiobjective memetic algorithms by using different quality indicators and show that the performances of the method are very interesting.

Keywords: Combinatorial Multiobjective Optimization, Hybrid metaheuristic, Memetic Algorithm, Tabu Search, Knapsack.

Mathematics Subject Classification. 90C29, 90C59

August 24, 2009.

* *T. lust thanks the "Fonds National de la Recherche Scientifique" for a research fellow grant (Aspirant FNRS).*

[1] Laboratory of Mathematics & Operational Research, Faculté Polytechnique de Mons, 9, rue de Houdain 7000 Mons, Belgium, thibaut.lust@fpms.ac.be.

Introduction

A multiobjective combinatorial optimization problem (P) (or multiobjective linear programming problem with binary variables), with K objectives, n variables and q constraints is defined in the following way:

$$\left[\begin{array}{ll} \text{“}\min_{X\in D}\text{”} & z_k(X) = c^k X \qquad k = 1,\ldots,K \\ \text{where} & D = \{X : X \in LD, X \in B^n\} \\ & LD = \{X : AX \leq b, X \geq 0\} \\ \text{with} & A = (q \times n), c^k = (1 \times n), X = (n \times 1), b = (q \times 1) \\ \text{and} & B = \{0,1\} \end{array}\right]$$

Due to the contradictory features of the objectives, it does not exist a solution simultaneously minimizing each objective (and for this reason the notation “ min ” is used), but a set of solutions called *efficient solutions*. A solution $X^* \in D$ is efficient for the problem (P) if there are no other solutions $X \in D$ such as: $z_k(X) \leq z_k(X^*), k = 1,\ldots,K$ with at least a strict inequality. We will indicate by $E(P)$ the whole efficient solution set of the problem (P), which is represented in the objective space by a *Pareto front* or a *trade-off surface*.

In this paper, only a minimal complete set will be sought, i.e. no equivalent efficient solution (two solutions X_1 and X_2 are equivalent if $z_k(X_1) = z_k(X_2)$, $k = 1,\ldots,K$) will be retained, and each solution found will correspond to a distinct non-dominated point in the objective space.

Metaheuristics showed their effectiveness to obtain a good approximation of the solution of difficult optimization problems. So, they were adapted to multiobjective problems [3,8,10], and constitute a research field under full development, since the multiobjective case requires a method which combines different properties at the same time, i.e. able to find solutions located in all the zones of the objective space, but also close to the efficient solutions. This is why it is relevant to hybrid different metaheuristics in order to combine the advantages of each one of them. If we refer to the taxonomy of hybrid metaheuristics of Talbi [30], the memetic algorithms, employed in this paper and based on the cooperation between a genetic algorithm and a local search method, belong to the LTH (Low-level Teamwork Hybrid) class. The LTH class regroups the cooperations resulting from an optimization method with population of solutions where the operator acting on the solutions, in an individual or total way, is replaced by an optimization method (in the memetic algorithms, the mutation operator of the genetic algorithm is replaced by the local search).

This paper is organized as follows: after a short presentation of the existing multiobjective memetic algorithms, we present the MEMOX scheme, a general memetic algorithm scheme for multiobjective optimization. At section 3, we present the local search that will be used in the MEMOX scheme, the original multiobjective tabu search PRTS, what gives rise to the method called MEMOTS. We expose then how the MEMOTS algorithm has been adapted to the multiobjective

multidimensional knapsack problem, how the MEMOTS parameters have been fixed and how the parameter values influence the algorithm performances. Various indicators of quality of an approximation are presented. Finally, we compare the MEMOTS performances with respect to MOGLS [19], a particularly powerful method for this problem, and two other multiobjective memetic algorithms: PMA [18] and IMMOGLS [13].

1. Brief overview of memetic algorithms for multiobjective optimization

A memetic algorithm (or genetic local search) is a genetic algorithm where the mutation operator is replaced by a local search method applied to every new offspring generated [26]. The memetic algorithms are particularly well adapted to the resolution of multiobjective optimization problems since a set of diversified solutions (from where the interest to use a population) but also close to the Pareto front (what is ensured by the local search) is required.

In an interesting survey of memetic algorithms for multiobjective optimization [23], Knowles and Corne distinguish three principal groups of authors who developed memetic multiobjective algorithms: Ishibuchi and Murata with the IMMOGLS (Ishibuchi Murata MultiObjective Genetic Local Search) method [13], Jaszkiewicz with two methods, MOGLS (MultiObjective Genetic Local Search) [15, 19] and PMA (Pareto Memetic Algorithm) [18] and finally, Knowles and Corne with the M-PAES (Memetic Pareto Archived Evolution Strategy) method [22].

The three methods IMMOGLS, MOGLS and PMA are all the three based on the same principle: a scalarizing function is used, defined thanks to a weight vector randomly drawn, to select probabilistically two parents being good on this function. The two parents are then crossed, to generate an offspring. The local search is applied to the offspring and finally, the improved offspring competes with the population for survival to the next generation.

The only point on which the three algorithms differ is the choice of the parents employed for recombination. In IMMOGLS, the parents are selected from the current population using the roulette wheel selection scheme with a linear scaling. The two parents in MOGLS are randomly selected from a temporary population of small size composed of the best solutions on the current scalarizing function. In PMA, the selection is based on a tournament, where the two solutions selected are the winners of a tournament between solutions coming from a sample drawn at random from the population. The selection procedure of PMA is faster than that of MOGLS, but the determination of the sample size of PMA is quite difficult.

The M-PAES method is rather different from the three preceding ones, since none scalarizing function is used, either in the local search or the parents selection. The solution evaluation is instead based on a form of Pareto ranking. The local search is the (1+1)-PAES method [21], which is a procedure for maintaining a finite size archive of non-dominated solutions. In (1+1)-PAES, a new solution is generated from the current solution, and, to check if this new solution is accepted,

a comparison with the current solution and the population is realized. The rule of acceptation is as follows:

- If the new solution is dominated by the current solution: not accepted.
- If the new solution dominates at least one solution of the population: accepted.
- If the new solution is not dominated by at least one solution of the population but does not dominate any solutions of the population: the solution is accepted at the condition that the solution brings diversity. The diversity is measured by a hypergrid created in the objective space.

In addition, the M-PAES method employs periodically a crossover operator to recombine the solutions found by the (1+1)-PAES procedure.

The M-PAES method is known as being parsimonious in term of number of evaluations because no non-dominated solution is discarded contrarily to algorithms using scalarizing functions, since in these algorithms, solutions are rejected if they are bad on the current scalarizing function. On the other hand, a Pareto ranking evaluation takes more time than applying a scalar acceptance function.

As M-PAES, the algorithm developed in this paper does not use scalarizing functions, which also avoids the necessity of specifying weights sets and normalizing the objectives.

2. MEMOX: a new memetic algorithm scheme for multiobjective optimization

2.1. Presentation of the scheme

We present in this section a new scheme of resolution of multiobjective optimization problems, called MEMOX, and based on a memetic algorithm. The resolution scheme MEMOX combines elements of the algorithms presented at the previous section, without using scalarizing functions for the parents selection.

Indeed, as illustrated at figure 1, a negative point of this selection is that the distribution of the solutions of the population in the objective space is not taken into account; parents being located in a zone already well exploited could thus be selected for the recombination because the weight set λ used to build the scalarizing function on which the selection is based is randomly generated.

The functioning of the MEMOX scheme, presented at figure 2, is as follows.

In an initialization phase, an initial population of non-dominated solutions (or potentially efficient), and ideally diversified, is generated by a heuristic or a local search. At each new solution X_s generated, the set of potentially efficient solutions, noted PE, is actualized by elimination of the potentially efficient solutions of PE which could be found dominated following the addition of the new solution X_s.

Then, two solutions from the population are selected (the parents). The first selected solution, called X_1, is one of the solutions of the population having a minimal density, defined by the number of solutions being in the same hypervolume

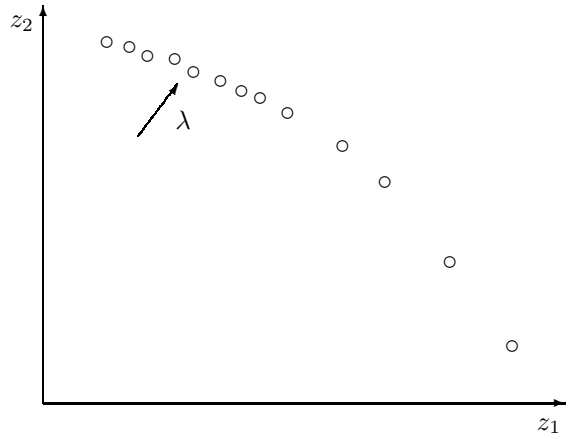

FIGURE 1. Problem of the selection of MOGLS, IMMOGLS and PMA.

as the solution. The technique employed to realize this division is explained at section 2.2 (this is the main difference with respect to the other multiobjective memetic algorithms).

The second one, called X_2, is one of the solutions of PE among the r closest [14] solutions to the first solution according to the euclidean distance in the objective space. It should be noted that if the number of potentially efficient solutions found by the algorithm, noted $|PE|$, is lower or equal to r, the selection of the second solution is realized among the $|PE| - 1$ closest solutions.

The both solutions X_1 and X_2 are then combined by a crossover operator for thus obtaining a new solution called X_3 (the offspring). The solution X_3 is added to the set $|PE|$ of potentially efficient solutions, and this set is consequently actualized to keep only the non-dominated solutions.

Finally, a local search method X (that can be the same than the one used during the initialization phase) is applied from the offspring X_3 until no more improvement in the solution set happens during a certain iterations number it_{stop}. Finally, this process is reiterated by again selecting the minimal density solution.

The role of the local search is to intensify the research by generating new solutions close to the selected solution of minimal density X_1. The diversification of the method is ensured by the improvement of solutions of minimal density, which makes it possible to generate new solutions in little or not exploited zones. An important parameter of the MEMOX scheme is the it_{stop} parameter, i.e. after how many iterations without improvement we have to stop the local search, knowing that the total number of iterations is limited. It can thus be preferable to start again the local search from a solution of minimal density rather than to strongly intensify in a precise zone.

The algorithm has the advantage of being easily adapted to any type of multiobjective optimization problems, provided that a local search (and the corresponding neighborhood), a crossover operator and a means of generating an initial population are defined.

Parameters

it_{stop}: iterations number of the local search without improvement
r: number of solutions taken into account for the selection of the closest solution
S : number of initial solutions
n: maximum number of iterations

Notations

i: current iterations number
n_{ls} : counter of the iterations number of the local search
PE : list of non-dominated solutions, potentially efficient
$|PE|$: number of potentially efficient solutions
$D(X)$: density of a potentially efficient solution X

Initialization

$i \leftarrow 0$
$PE \leftarrow \emptyset$
Generate S admissible solutions thanks to a local search method or a construction heuristic
For each solution X_s generated ($s = 1, \ldots, S$), do:
  $PE \leftarrow PE + \{X_s\}$
  Actualize PE

Iteration i

For each solution of PE calculate the density $D(X_l)$ $l = 1, \ldots, |PE|$
Calculate $D^* = \min D(X_l)$
Choose randomly a solution X_1 among the solutions which respect $D(X_l) = D^*$
Choose a solution X_2 among the $\min(r, |PE| - 1)$ solutions of PE closest to X_1
Cross the two solutions X_1 and X_2 for obtaining a new solution X_3
$PE \leftarrow PE + \{X_3\}$
Actualize PE
Apply a local search method X from X_3 until no more improvement in PE while it_{stop}
$i \leftarrow i + n_{ls}$

Stop Criterion

Iteration count $i = n$

FIGURE 2. MEMOX scheme.

An important parameter of the MEMOX scheme is the it_{stop} parameter, i.e. after how many iterations we have to stop the local search, knowing that the total number of iterations is limited. It can thus be preferable to start again the local search from a solution of minimal density rather than strongly intensifying in a precise zone.

2.2. The dynamic hypergrid

An another important point not explained in this scheme is how to measure the density of a potentially efficient solution.

Two authors have already used the concept of density within the framework of multiobjective optimization:

- Deb with the concepts of hyperniche [28] and hypercube [6], introduced into the genetic algorithms NSGA (Non dominated Sorting Genetic Algorithm), for the selection of individuals through a population.
- Knowles and Corne with the hypergrid, introduced into the (1+1)-PAES method [21], employed in the rule of acceptance of a new solution and also used to limit the number of solutions generated.

An inconvenient of the hyperniche and hypercube measures is the high calculation computational time, because each solution has to be compared to all the other to obtain the density of all the solutions. Another argument in discredit of the hyperniche measure is that this measure is extremely dependent on the size of the niche [7].

Consequently, we use a hypergrid in MEMOX, created by a division of the objective space in hypervolumes. The density of a solution is thus defined by the number of solutions being in the same hypervolume as the solution. In this way, it is quite easy to measure the density of a solution by always having in memory the number of solutions in each hypervolume and the coordinates of the solutions in the hypergrid. Thus, if we want to know the density of a solution, it is enough to read the number of solutions present in the hypervolume identified by the coordinates of the solution.

The hypergrid size has to be managed. Indeed, if the number of hypervolumes that compose the hypergrid is to high, all the solutions will be on different hypervolumes. On the other hand, if there are not enough hypervolumes, a lot of solutions will be on the same hypervolumes (see figure 3).

In these two cases, it will be difficult to distinguish a solution of minimal density. That is why the hypergrid size is dynamically updated. As the method starts from a low number of solutions and this number is, for the majority of the cases, in constant increase when a multiobjective problem is solved, the number of divisions of the hypergrid is also constantly increased. The rule is as follows: if the average density of the solutions becomes higher than a certain threshold, the number of divisions of the hypergrid is increased by a certain step. In this way, a good measure of density is guaranteed. It should be noted that the average density is calculated by only considering the hypervolumes containing at least one solution.

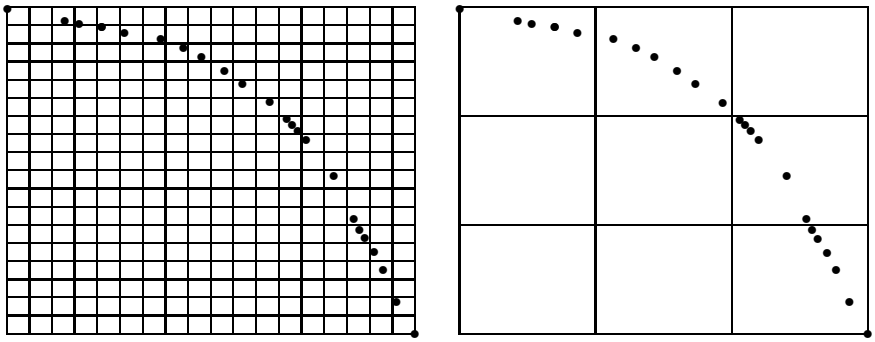

FIGURE 3. On the left: too many divisions, a high number of solutions have a density equal to one. On the right: not enough divisions, a solution of minimal density can not be distinguished.

The illustration of this technique is represented at figure 4, where the evolution of the number of divisions of the hypergrid according to the iterations number is represented.

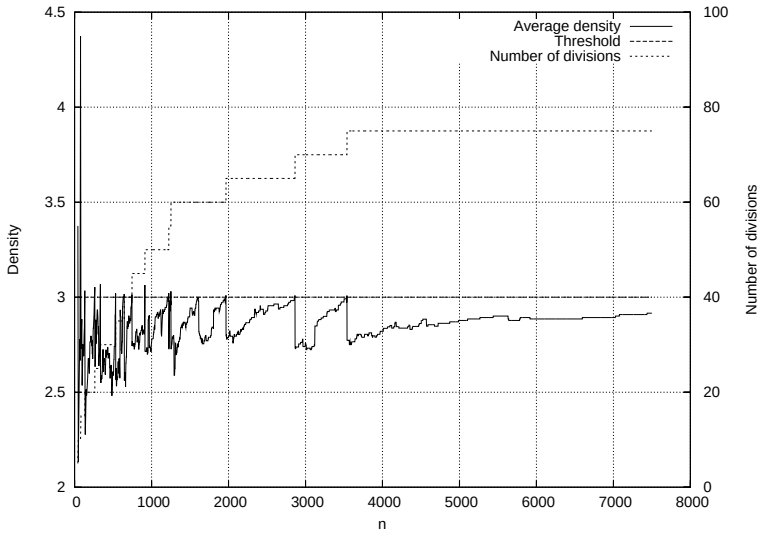

FIGURE 4. Evolution of the number of division of the hypergrid.

The hypergrid is also updated as soon as a new solution being outside of the hypergrid is produced. Between each update, we keep in memory the hypergrid

and the coordinates of the solutions, which makes it possible to reduce the computational time, but however, increases the necessary memory place, which could appear critical for big size problems.

This way of managing the hypergrid implies to set the mean density threshold from which the number of divisions of the hypergrid is increased. The influence of this parameter is experimented at section 6.2.2, and it is shown that the value of this parameter remains relatively easy to set, since the sensitivity of the algorithm with respect to the value taken by this parameter is not strong.

3. The Pareto Ranking Tabu Search

We present in this section the PRTS method, the local search method that we will use in the MEMOX scheme for the resolution of the knapsack problem. The PRTS method follows the classical framework of tabu search [11]: at each iteration, L neighbors non-tabu are generated starting from the current solution. The best neighbor is then selected and becomes the next current solution. To evaluate the neighbors in a multiobjective context, we employ an original evaluation function of neighbors, based on a recent paper of Elaoud et al. [9]. In this paper, a genetic algorithm (called PFGA for Pareto Fitness Genetic Algorithm) using a new fitness function based on individuals rank and density, is developed. This approach giving promising results, the fitness function of PFGA has been adapted within a tabu search for the evaluation and the selection of the neighbors.

Two measures appear in the evaluation function of PFGA: a classification of the individuals according to an original rank and a division of the objective space in hypervolumes allowing to measure the individuals density. These two concepts are then aggregate into a single evaluation function.

When the PRTS method is used as a local search in the MEMOX scheme, since its role is to intensify the research, it is preferable to use only the rank for the selection of the best neighbor. Indeed, the local search is already applied starting from a solution of low density, and the integration of the density concept in the selection of the neighbors would slow down the intensification of the research, and thus the generation of good solutions in the minimal density zone.

However, if the tabu algorithm is used in an independent way, i.e. without integrating it in a genetic algorithm, it is preferable to use the density measure in the neighbors evaluation since diversification and intensification are seeking at the same time. The addition of the density measure leads to the PRTS+D algorithm, not presented in this paper but already showed in [24].

In PRTS, the evaluation of each neighbor generated from the current solution is consequently only based on the Double Pareto Ranking (DPR) [9].

The DPR rank of a neighbor i is determined in two times. Initially, the Pareto Ranking (PR) is calculated by counting the number of individuals of the population which dominate the neighbor:

$$PR(i) = \Big|\{j | j \in P, i \prec j\}\Big|,$$

where P represents the population and the symbol $\prec$ corresponds to the Pareto dominance relation. In PRTS, the considered population includes the neighbors and the potentially efficient solutions found by the algorithm.

The DPR rank of a neighbor i is then defined by the sum between its $PR(i)$ rank and the $PR(j)$ rank of its dominators j:

$$DPR(i) = PR(i) + \sum_{j \in P,\, j \succ i} PR(j)$$

By way of example, the values of the PR and DPR ranks for a population of 13 individuals are presented at figure 5. If we compare the individuals I_1 (of PR and DPR ranks equal to 4 and 5) and I_2 (of PR and DPR ranks equal to 4 and 9), we note that their PR ranks are equal since each one is dominated by four individuals. However, the individual I_1 seems more interesting since it is enough that it becomes better than only one of the individuals that dominate it to be non-dominated whereas the individual I_2 must become better than at least three individuals to be non-dominated.

Thus, by the use of the DPR, the individual I_1 will be preferred since the PR ranks of the individuals that dominate it are lower, which is translated by a lower DPR rank.

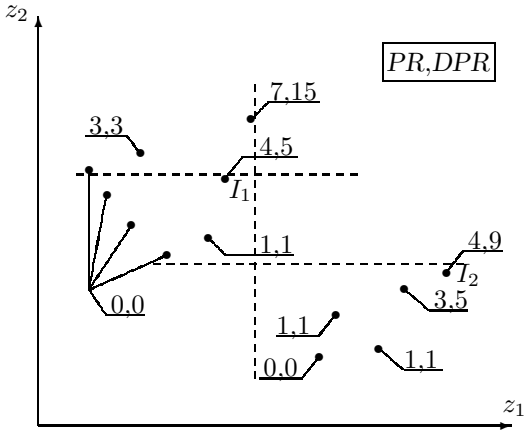

FIGURE 5. Illustration of the DPR rank.

After the best neighbor selection, the set of potentially efficient solutions PE is improved by adding the non-dominated neighbors, i.e. those presenting a zero value for the DPR rank and different in the objective space from the solutions of PE. In addition, the set PE is actualized by elimination of the potentially efficient solutions of PE which could be found dominated following the addition of the new solutions.

Parameters

t: tabu tenure
L: number of neighbors generated at each iteration (neighborhood size)
n: maximum number of iterations

Notations

$PE \equiv$ list of non-dominated solutions, potentially efficient
$T \equiv$ tabu list of tabu tenure t
$X \equiv$ current solution
$N(X) \equiv$ neighborhood of X: set of solutions $\in D$ obtained by a movement starting from X
$DPR(Y) \equiv$ Double Pareto Ranking of a neighbor Y

Initialization

$i \leftarrow 0$
Choose randomly a feasible solution $X_0 \in D$
$PE \leftarrow \{X_0\}$
$X \leftarrow X_0$
$T \leftarrow \emptyset$

Iteration i

Generate randomly L neighbors $Y_1, \ldots, Y_l, \ldots, Y_L$ non-tabu in $N(X)$
For each neighbor Y_l calculate $f(Y_l) = \mathrm{DPR}(Y_l)$
Calculate $f^* = \min f(Y_l)$
Choose randomly a neighbor Y^* among the neighbors which respect $f(Y_l) = f^*$
$X \leftarrow Y^*$
Give to the movement $X \rightarrow Y^*$ the tabu statute
For each neighbor Y_l, if $DPR(Y_l) = 0$ and Y_l is different in the objective space from the solutions of PE then
    $PE \leftarrow PE + \{Y_l\}$
    Actualize PE
$i \leftarrow i + 1$

Stop criterion

Iteration count $i = n$

FIGURE 6. PRTS algorithm.

The algorithm of the PRTS method is given at figure 6. The algorithm presents three parameters, which depend on the problem and on the user requirements: the tabu tenure t, the number L of neighbors generated at each iteration and the maximum number of iterations n.

4. Quality indicators

In single optimization, it is quite easy to measure the quality of a solution or to compare the solutions obtained by various methods. That is more difficult in the multicriteria case [35], because the solutions are represented by a trade-off surface composed of a solution set, and the problem of the quality measurement of such a set is thus also and logically a multicriteria problem.

Consequently, we use several indicators to measure the quality of an approximation of a solution set:

- the hypervolume $\mathcal{H}$ (to be maximized) [34]: approximation of the volume included under the curve formed by the evaluation of the solutions. This indicator is only calculated for the two objective instances since it takes high computational time if the number of objectives and solutions increase, even if works are led to reduce this time [32].
- the spacing out metric SM (to be minimized) [27]: measure the uniformity of the distribution of the solutions composing the trade-off surface. More the value is close to zero, the best will be the distribution of the solutions on the trade-off surface.
- the R measure (to be minimized) [17], correlated to the distance between a reference point and the solutions of the approximation. The reference point used can be found in [17].
- the average distance D_1 and maximal distance D_2 (to be minimized) [4, 31] between the solutions of the reference set and the solutions of the approximation, by using the euclidean distance. Ideally, the reference set is the Pareto front.
- the percentage of efficient solutions found, only calculated when the efficient solutions are available, noted $|EP|(\%)$.
- the number of potentially efficient solutions found, noted $|PE|$.

An illustration of these indicators for a two objectives example is given at figure 7.

The two distances D_1 and D_2 are rather good indicators, provided that the reference set is of good quality. The distance D_1 reflects the capacity of an algorithm to reach solutions close to the reference set, so it can be interpreted as an indicator to measure the intensification property of a multiobjective algorithm.

Being given that the solution set obtained by the algorithms are in general uniform (no zones without solutions), the maximum distance between the solutions of the reference set and the solutions of the approximation appears generally in the extreme zones. In this way, the distance D_2 seems a good indicator to measure the diversification property of an algorithm, interpreted here by the capacity to find solutions in the extreme zones (which are important solutions since they optimize one of the objectives).

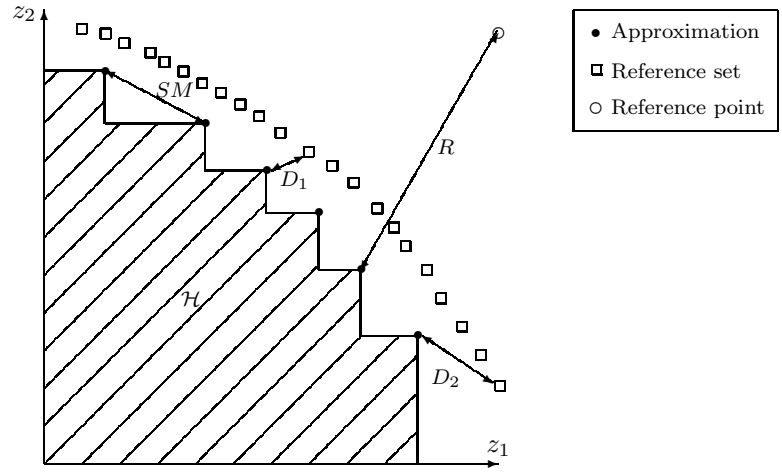

FIGURE 7. Quality indicators.

5. The knapsack problem

We apply the MEMOX scheme to the knapsack problem. The local search method used is the PRTS method. The integration of this tabu search to the MEMOX scheme leads to a method called MEMOTS. We define in this section the knapsack problem and how the MEMOTS method has been adapted to this problem.

5.1. Problem definition

The knapsack problem consists, in its unidimensional and unicriteria case, to choose a set of objects to put in a bag. Each object having a weight and a profit, the objects should be selected to maximize the total profit while not exceeding the bag capacity. In the multiobjective case, K profits are associated at each object and in the multidimensional case, q bags are considered, which means that q constraints are added. The multidimensional multiobjective knapsack problem is defined as follows:

$$\left[\begin{array}{lll} \max & z^k(X) = \sum_{i=1}^{n} c_i^k x_i & k = 1, \ldots, K \\ \text{s.t} & \sum_{i=1}^{n} w_i^l x_i \leq b_l & l = 1, \ldots, q \\ & x_i \in \{0,1\} & i = 1, \ldots, n \end{array}\right]$$

with n the number of objects available, X the decision vector, formed of the binary variables x_i ($x_i = 1$ means that the object i is in the knapsack), c_i^k the profit k of the object i, w_i^l the weight l of the object i, b_l the capacity of the knapsack l and z^k the value taken by the objective k.

5.2. Data sets

We use the knapsack instances published by Zitzler [33]. The data sets include 250, 500 and 750 objects, with two objectives and two constraints or three objectives and three constraints.

We have generated a minimal complete set of the efficient solutions of the two objectives instances which is particularly useful to measure the quality of the approximations, by applying a lexicographic method [29] coupled with the commercial CPLEX solver. We did not use the solutions published by Zitzler [33], because we noted that they were some mistakes in these solutions (solutions are missing), probably due to a bad parameterization of the solver employed by Zitzler.

For the three objectives instances, we approximate the efficient solutions by applying several times the MEMOTS and MOGLS methods during a great iterations number. We then retain only the non-dominated solutions.

However, these sets are not optimal and a new non-dominated solution could be found by one of the algorithms tested in this chapter, what can very often arrive since the number of non-dominated solutions of the three objectives instances of the knapsack problem tested in this paper is very high. The quality indicators of the algorithm would be penalized because the distance between the new solution and the reference set would be non-null. Thus, we improve in an artificial way these reference sets by adding a certain value ϵ to the evaluation of each solution of the reference sets. The value is determined experimentally: the value must be small but sufficiently large in order to avoid the problem of generation of non-dominated solutions not being in the reference sets.

It should be noted that, thereafter, when we will speak about, for example, the 250-2 instance, it will mean the instance with 250 objects and two objectives.

5.3. Adaptation of the MEMOTS algorithm to the knapsack problem

To adapt the MEMOTS algorithm to the knapsack problem, we have to define a crossover operator, the neighborhood and the tabu list management of the PRTS method and also a means of generating an initial population.

5.3.1. *Crossover operator*

We must define the crossover operator and also a repair procedure for a solution that does not respect the constraints of the knapsack problem, what can arrive after the recombination of the two solutions X_1 and X_2.

The crossing is carried out thanks to a one-point operator starting from the two solutions X_1 and X_2. As this operation can produce two solutions, we keep only

that which has the left part of X_1 and the right part of X_2, which gives a new solution called X_3.

The solution X_3 can then not respect the constraints problem. We thus apply a repair procedure, inspired by the Michalewicz and Arabas procedure [25] for the single objective knapsack problem, which consists in removing the objects j in the increasing order of the following ratio:

$$\frac{\sum_{k=1}^{K} \lambda_k c_j^k}{\sum_{l=1}^{q} \left(w_j^l \cdot \max\left(0, \sum_{i=1}^{n} w_i^l x_i - b_l\right)\right)}$$

until satisfaction of the constraints.

For an object j, this ratio corresponds to the linear aggregation of the profits generated by the object on the sum of the weights of the object in the different bags, by only considering the bags that violate the capacity constraints and by giving more importance to the bags whose constraints are strongly transgressed (measured by $\sum_{i=1}^{n} w_i^l x_i - b_l$). This procedure is also used in the MOGLS method [19].

In MOGLS, the weight set λ is the same weight set than the one employed to build the scalarizing function used for the selection of X_1 and X_2. In the MEMOX scheme, a scalarizing function is not employed in the selection, and the use of a random weight set could lead to a solution X_3 being enough far from X_1 and X_2, and thus not being located in a zone of minimal density.

So, the weight set is determined according to the efficient solution X_1 selected, in order to guarantee that the better the evaluation of the solution X_1 according to an objective is, the more the value of the weight according to this objective will be high. The computation formula of the weight set λ is thus the following one:

$$\lambda_k = \frac{R\big(k, z_k(X_1)\big)}{\sum_{k=1}^{K} R\big(k, z_k(X_1)\big)} \qquad k = 1, \ldots, K$$

where the function $R\big(k, z_k(X_1)\big)$ gives the number of potentially efficient solutions whose evaluation according to the objective k is lower (or higher if the objective must be minimized) than $z_k(X_1)$. The value of this function is ranged between 0 and $|PE| - 1$, where $|PE|$ represents the number of potentially efficient solutions.

In this way, we obtain a weight set for the repair procedure which goes in the direction of the minimal density solution X_1.

5.3.2. *Neighborhood*

The neighborhood $N(X)$ used in the PRTS algorithm, which must allow to generate several good solutions from only one, is defined in the following way:

- Generate a weight set λ.
- Create a list containing the objects j present in the current solution X (identified by $x_j = 1$) sorting out it in the increasing order of the following ratio:

$$\frac{\sum_{k=1}^{K} \lambda_k c_j^k}{\sum_{l=1}^{q} w_j^l}$$

  The objects placed at the beginning of the list are thus the objects of low ratio profit/weight. The size of the list is limited at β percent of n_o, with n_o equal to the number of objects present in the solution X.
- Generation of the neighbors:
  - Remove a combination of m objects from the solution X by randomly selecting them from the list.
  - Fill to the maximum the bags with the objects j not in the solution X (identified by $x_j = 0$) taken in the decreasing order of the following ratio:

$$\frac{\sum_{k=1}^{K} \lambda_k c_j^k}{\sum_{l=1}^{q} \left(\frac{w_j^l}{b_l - \sum_{i=1}^{n} w_i^l x_i} + 1 \right)}$$

    This procedure is also used in MOGLS [19], and consists in filling the bags with the objects of high profit and low weight, by giving higher influence to the bags whose the load (measured by $\sum_{i=1}^{n} w_i^l x_i$) is close to the maximum capacity (b_l).

So, roughly speaking, the neighbors are generated by removing the 'worst' objects from the current solution and by adding the 'best' objects not in the current solution. We fix the number of neighbors generated at each iteration to the minimum between $\beta \cdot n_o$ (size of the list) and $C_m^{\beta \cdot n_o}$ (number of combinations of objects that can be removed). If the number m of objects that have to be removed is higher than $\beta \cdot n_o$, we only remove $\beta \cdot n_o$ objects.

For the first iteration of the tabu search PRTS, the weight set λ used is the weight set determined in the crossover operator (this weight set is thus determined even if the solution X_3 respects the constraints). The weight sets for the next iterations are randomly generated.

The neighborhood parameters are thus β, the percentage of objects present in the solution X appointed at being removed and m, the number of objects removed. The influence of these two parameters are studied at section 6.2.1.

The movement can thus be characterized by an array giving the objects removed and by the weight set used to select the objects filling the bags. These two variables are considered as the movement attribute.

5.3.3. *Tabu list management*

At each movement $X \leftarrow Y^*$, we record only a part of the movement attribute, namely the array giving the objects removed. The choice of these objects to fill the bags is then forbidden for the t (tabu tenure) next iterations.

5.3.4. *Initial population generation*

The initial population is simply generated by application of the PRTS method starting from a solution randomly generated, and by using for the neighborhood a random weight set. The generation of the initial population is stopped when $r+1$ non-dominated solutions has been generated.

6. Results

Initially, we present the influence of the selection based on the density measure. Then, a study of the influence of the parameter values on the performances of the MEMOTS algorithm is carried out, which gives a very useful help to set the parameters employed for the performances comparison, presented at the end of this section.

To realize these experimentations, the basic values employed for the various parameters are presented in table 1, where n corresponds to the iterations number, β and m are the neighborhood parameters, *density* is the density threshold from which the number of divisions of the hypergrid is increased by the value taken by the *step* parameter (the starting number of divisions is fixed at 5), r is the number of solutions taken into account for the selection of the closest solution to the first parent and it_{stop} is the iterations number of the local search without improvement after which the local search is stopped; it_{stop} equal to 0 means that only one iteration of the tabu search is realized. The computer used for the experiments is a 3 GHz Pentium IV and 20 runs of the algorithms are performed each time.

Table 1. Parameter values of MEMOTS used for the study of the parameters influence.

Instance	n	β	m	*density*	*step*	t	r	it_{stop}
250-2	7500	10	1	3	5	5	20	0
500-2	10000	10	1	3	5	5	20	0
750-2	12500	10	1	3	5	5	20	0
250-3	10000	10	1	5	5	5	20	0
500-3	12500	5	2	5	5	5	20	0
750-3	15000	5	3	5	5	5	20	0

6.1. Influence of the selection based on the density measure

We compare in this section the performances of two ways of selection of the first parent, an important point of the MEMOX scheme:

- random selection of the first parent.
- selection based on the density measure.

We represent at figure 8 the influence of the way to select the first parent on the performances of the algorithm.

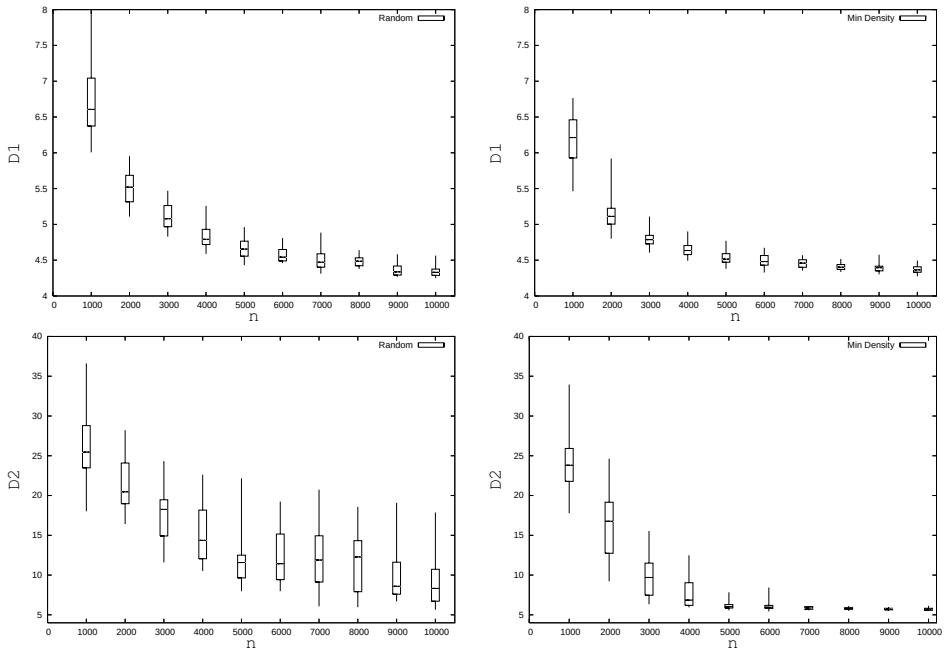

Figure 8. Influence of the selection based on the density measure for the 250-3 instance. On the left: random selection. On the right: selection based on the density measure.

The graphs represent the evolution of the indicators D_1 and D_2 according to the iterations number n for the 250-3 instance, and for the two ways of selecting the first parent. We chose to represent the data in the form of 'Box-and-Whisker Plot', allowing to reproduce the distribution of the variable studied. The ends of the box represent the first and third quartiles of the distribution and the horizontal line in the box gives the median. We also added segments at the ends of the box up which go up to the extreme values of the distribution. We note that if the selection of the first parent is carried out in a random way, the speed of convergence of the indicator D_1 is slightly lower than if the first parent selected is that of minimal density, although the end value is of the same quality. On the

other hand, for the D_2 indicator, convergence is not only slower, but moreover, the final distribution is of worse quality (higher value of the median and strong variations in the distribution). The interest to select the parent of minimal density is thus rather large, but essentially in order to reduce the D_2 indicator.

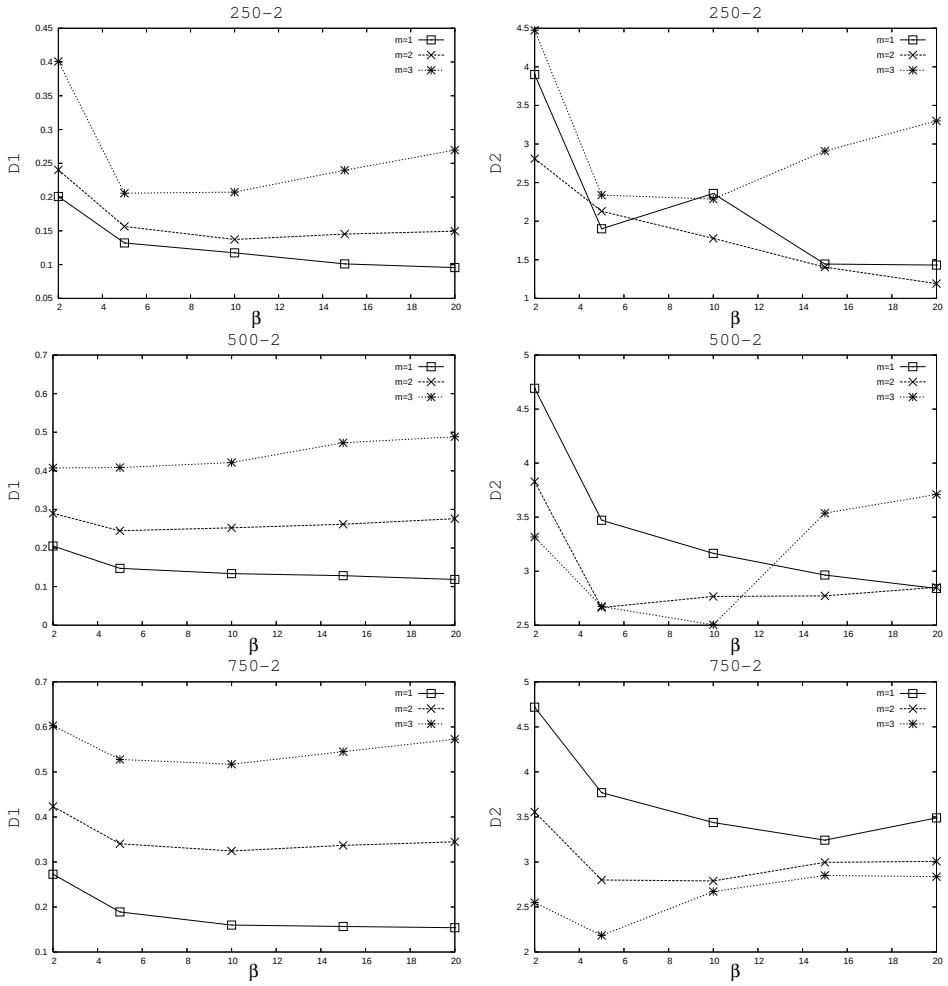

FIGURE 9. Neighborhood - 2 objectives instances. $\square : m = 1$, $\times : m = 2$, $* : m = 3$.

6.2. Parameters tuning

6.2.1. *Neighborhood parameters*

We have to set the β parameter (related to the number of objects present in the current solution that can be removed from the current solution) and the m

parameter (number of objects removed to generate a neighbor). We can see at figures 9 and 10 the influences of the β and m parameters on the D_1 and D_2 indicators for all the instances treated in this paper. To ensure the visibility of the graphs, we only indicate the average value obtained by the D_1 and D_2 indicators.

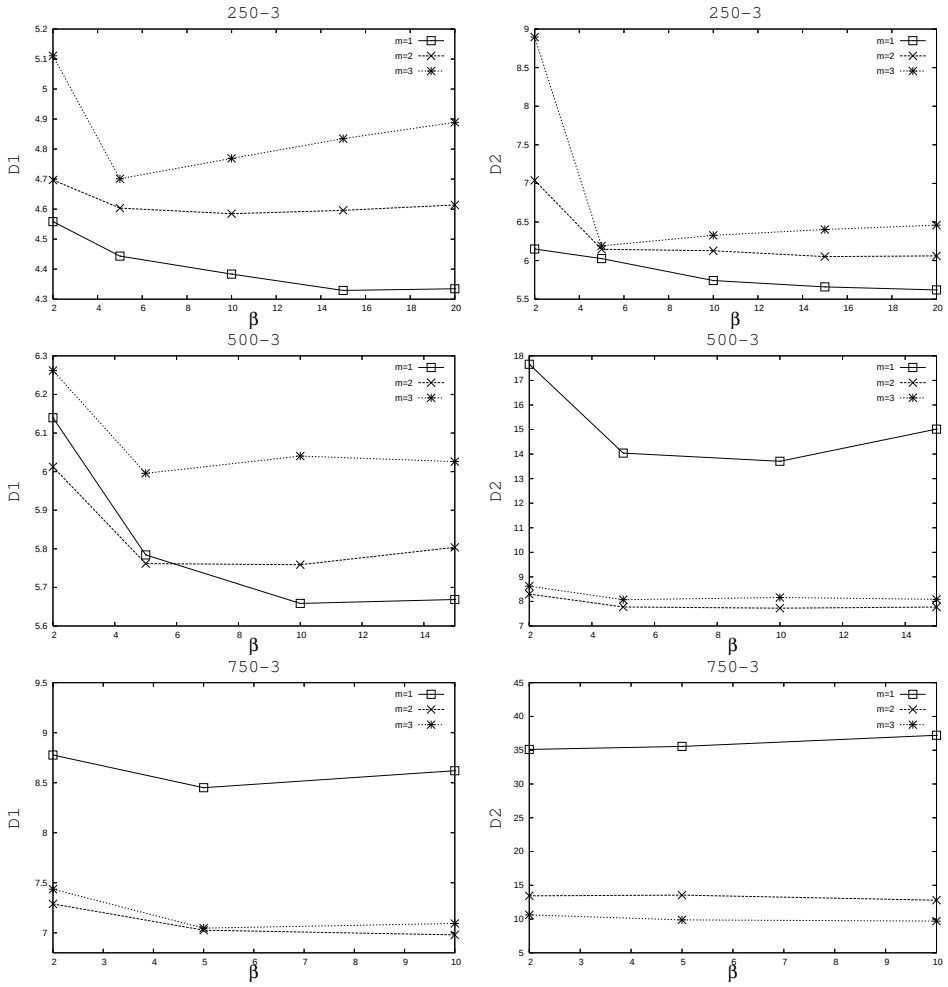

FIGURE 10. Neighborhood - 3 objectives instances. $\square$: $m = 1$, $\times$: $m = 2$, $*$: $m = 3$.

For the β parameter, a value equal to 10 seems good, since there are no more significant improvements after 10. For the m parameter, it appears that to minimize the D_1 indicator, it is preferable to remove only one object ($m = 1$), except for the 750-3 instance. We can explain that by the fact that removing only one object improves the intensification property of the algorithm, although, for the 750-3 instance, removing only one object is not sufficient. On the other hand, for

instances with more than 250 objects, to minimize D_2, it is better to remove more than one object ($m > 1$). But removing one object in the place of removing more than one object has only large influence on the D_2 indicator for the instances with more than 2 objectives and more than 250 objects (500-3 and 750-3).

So, it is more difficult to set parameters of multiobjective metaheuristics, because various contradictory criteria occur in the performance measure. Consequently, we will use for the parameter values, a compromise between intensification (D_1 minimization) and diversification (D_2 minimization).

6.2.2. *Influence of the density threshold*

We study in this section the influence of the mean density threshold from which the number of division of the hypergrid is increased. We can see at figure 11 the influence of this threshold on the D_1 and D_2 indicators for the 750-2 and 250-3 instances.

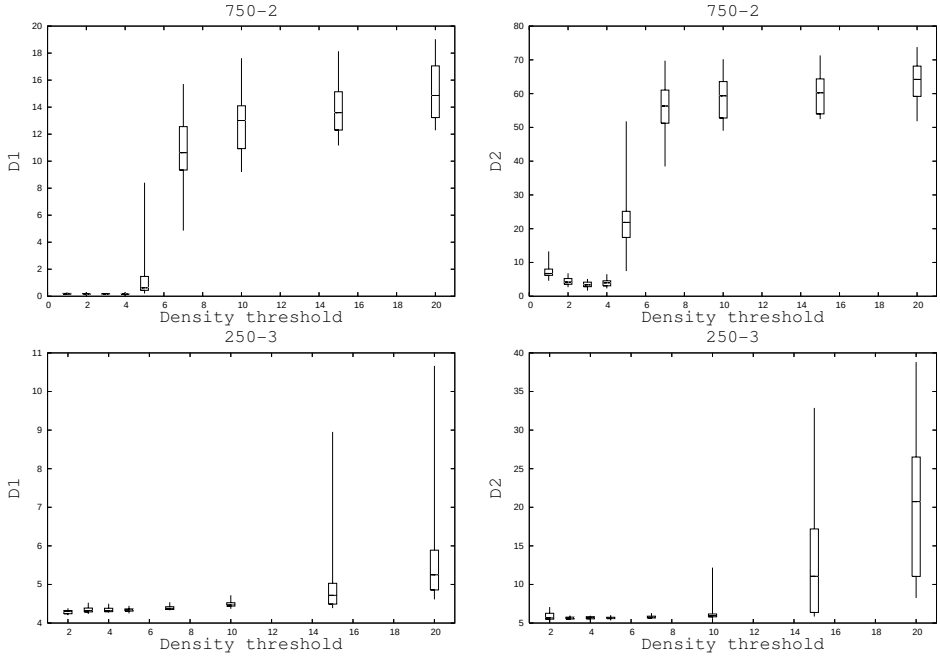

FIGURE 11. Influence of the density threshold.

We note that a value equal to 3 seems the best value for both instances and using a value upper to 4 for the two objectives instance and upper to 10 for the three objectives instance strongly degrades the performances. We also studied the influence of the threshold on the other instances and we obtained the same conclusion.

So as mention at section 2.2, it's quite easy to set this parameter, since a value around 3 gives good performances for all the instances.

6.2.3. *Influence of it_{stop}*

The influence of the iteration count it_{stop} without improvement after which the local search is stopped is represented at figure 12 for the 500-2 and 500-3 instances.

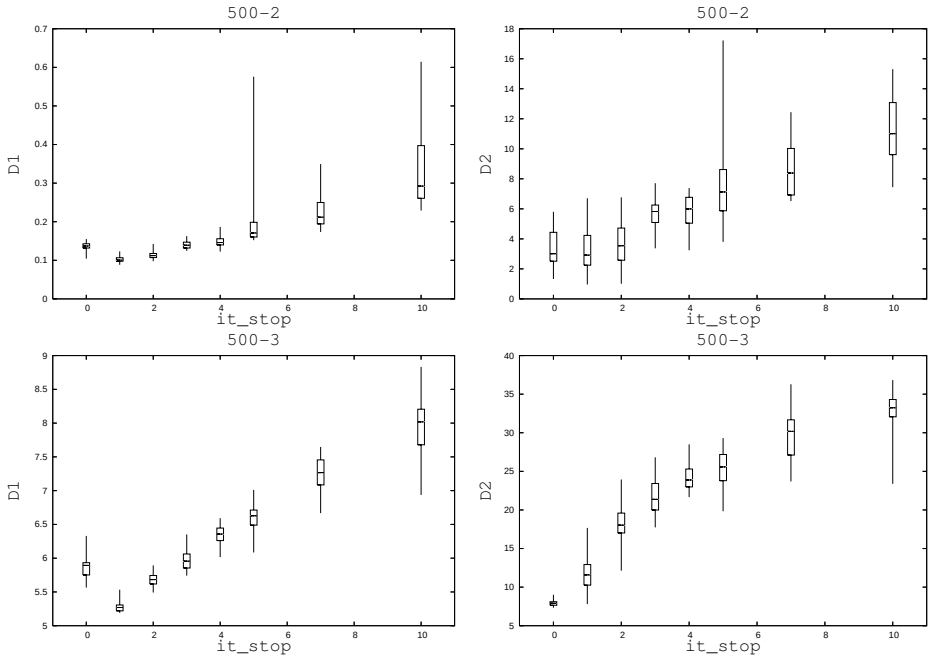

FIGURE 12. Influence of it_{stop}.

For the 500-2 instance, a value of it_{stop} equal to 1 seems to be an optimal value since this value makes it possible to minimize at the same time the D_1 and D_2 indicators.

On the other hand, for the 500-3 instance, to take it_{stop} equal to 1 in the place of zero (only one iteration for the tabu search) degrades the D_2 indicator, but it_{stop} equal to 1 remains the optimal value for the D_1 indicator.

We can interpret this difference between the two and three objectives instances by the fact that for the 3 objectives instances, the number of non-dominated solutions is more important, and that it is preferable to have a diversified research rather using intensively the local search. By consequence, for the 2 objectives instances, it_{stop} will be fixed at one and for the 3 objectives instances, only one iteration of the tabu search will be carried out.

6.2.4. *Influence of r*

We can see at figure 13 the influence of the r parameter, i.e. the number of solutions taken into account for the selection of the closest solution to the first parent. For the 750-2 instance, the results show that if r is higher than approximately

30, the D_2 indicator is degraded, which joined the conclusions of Hishibuchi on the fact that it is preferable to combine similar parents in multiobjective optimization [14]. For the 250-3 instance, the r value from which the D_2 indicator is degraded is higher, since the number of potentially efficient solutions obtained by the algorithm is more important.

On the other hand, there is a clear improvement of the D_1 indicator if the r value increases, but this improvement stops when the r value reaches the value from which D_2 is degraded.

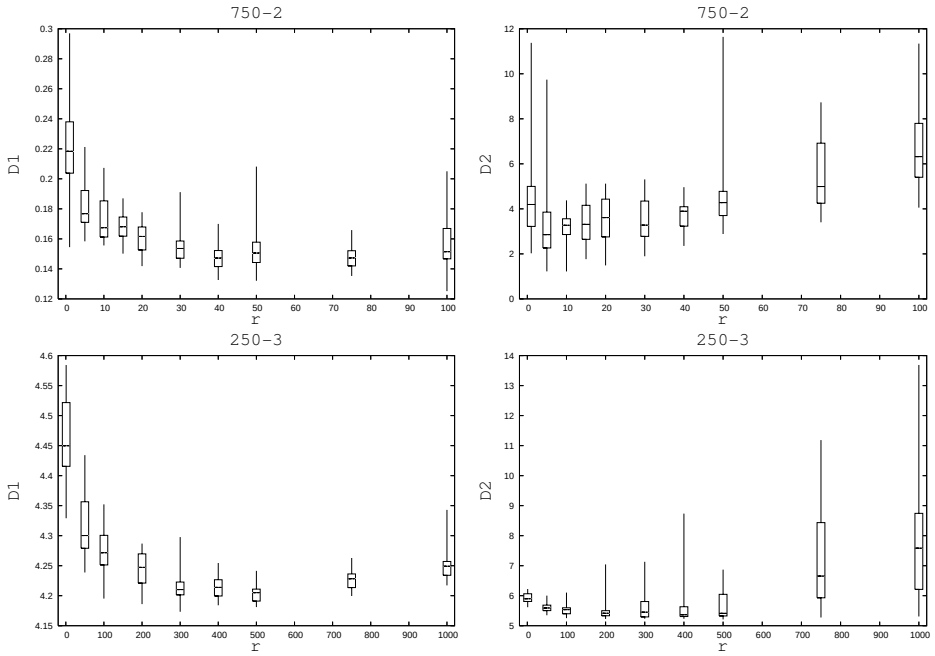

FIGURE 13. Influence of r.

6.3. PERFORMANCES COMPARISON

We compare the performances of the MEMOTS algorithm to three other multiobjective memetic algorithms: the MOGLS algorithm [19], the PMA algorithm [18] and the IMMOGLS algorithm [13]. The MOMHLib++ library, developed by Jaszkiewicz [16], and making it possible to launch various existing multiobjective metaheuristics on the multidimensional multiobjective knapsack problem, has been used to generate the results of these three algorithms (but has not been employed to generate the MEMOTS results). The values used for the various parameters of these algorithms correspond to the default values defined in the library.

We do not employ for the comparison the memetic algorithm M-PAES [22], because it has been shown by Ishibuchi [12] that the performances of MOGLS were better than M-PAES on the knapsack problem. Also, we do not use the classic multiobjective genetic algorithms SPEA [36] and NSGA-II [6] because several studies [1, 2, 17, 19] have shown than MOGLS outperforms these methods.

It should be noted that we do not employ the solutions of MOGLS published on the web site of Jaszkiewicz [16] because the specific procedures to the knapsack problem integrated in the MOGLS algorithm which made it possible to generate these solutions [15] do not correspond to the last procedures used by Jaszkiewicz [20] (and also employed in the MEMOTS method). Thus, we use the library of Jaszkiewicz (where the last MOGLS adaptation is implemented) to generate better solutions that those published.

The values of the parameters used by MEMOTS for the comparison are given at table 2. The iterations number has been fixed to the same number employed by the other memetic algorithms used for the comparison. The values of the other parameters were fixed by considering the results of the preceding section presenting the influence of the parameter values on the MEMOTS performances.

TABLE 2. Parameter values of MEMOTS used for the performances comparison.

Instance	n	β	m	$density$	$step$	t	r	it_{stop}
250-2	7500	10	1	3	5	3	30	1
500-2	10000	10	1	3	5	5	30	1
750-2	12500	10	1	3	5	7	30	1
250-3	10000	10	1	3	5	/	200	0
500-3	12500	5	2	3	5	/	300	0
750-3	15000	5	3	3	5	/	400	0

We represent at tables 3 and 4 the average value for the various indicators of quality obtained by the four memetic algorithms on all the knapsack instances studied in this paper (for each instance, the results correspond to the four first lines).

For the two objectives instances, MEMOTS outperforms the other algorithms for the D_1 indicator, the percentage of efficient solutions found, the spacing metric, the number of potentially efficient solutions found and the R measure. The D_2 indicator of the other memetic algorithms is slightly better than MEMOTS on the 500-2 and 750-2 instances. The hypervolume of MEMOTS is better than the other algorithms, except for the 750-2 instance.

For the 250-3 instance, MEMOTS is better than the other algorithms on all the quality indicators. For the 500-3 and 750-3 instances, MEMOTS is better than MOGLS and PMA only for the SM and $|PE|$ indicators, but remains better than IMMOGLS for the 500-3 instance.

TABLE 3. Comparison of the MEMOTS, MOGLS, IMMOGLS and PMA indicators for the 2 objectives instances.

Instance	Algorithm	$\mathcal{H}(10^7)$	SM	R	D_1	D_2	$\|EP\|(\%)$	$\|PE\|$	Time(S)
250-2	MEMOTS	**9.87**	**3.00**	**246.56**	**0.083**	**1.91**	**21.51**	**399.25**	**4**
	MOGLS	9.86	4.22	248.97	0.23	2.69	3.96	202.35	7
	IMMOGLS	9.86	7.03	253.44	0.55	3.88	0.84	82.80	6
	PMA	9.86	4.59	249.69	0.28	2.51	2.80	169.30	5
	GREEDY	9.85	9.69	256.78	0.72	3.88	0.75	42.50	2
	MEMOTS+	9.87	3.07	246.49	0.073	1.08	25.01	393.20	6
500-2	MEMOTS	**40.77**	**2.62**	**432.53**	**0.10**	3.12	**2.72**	**828.30**	**15**
	MOGLS	40.76	4.48	435.88	0.25	2.41	0.32	281.85	22
	IMMOGLS	40.74	7.79	441.06	0.51	**2.29**	0.067	96.40	23
	PMA	40.76	4.68	436.41	0.27	2.86	0.30	260.35	20
	GREEDY	40.74	8.88	442.24	0.58	2.49	0.081	73.75	13
	MEMOTS+	40.78	2.61	431.89	0.074	2.24	6.19	844.30	28
750-2	MEMOTS	89.30	**2.57**	**744.63**	**0.11**	3.83	**0.16**	**1455.80**	**40**
	MOGLS	**89.33**	5.41	750.01	0.29	**1.49**	0.081	329.25	53
	IMMOGLS	89.31	8.73	755.82	0.52	2.13	0.066	121.60	57
	PMA	**89.33**	5.44	750.13	0.30	1.51	0.094	323.45	50
	GREEDY	89.31	9.03	756.84	0.57	2.52	0.076	109.90	37
	MEMOTS+	89.35	2.53	743.47	0.070	1.52	1.15	1557.85	77

To take into account the variations in the results of the algorithms, we also carried out the non-parametric statistical test of Mann-Whitney [5], allowing to compare the MEMOTS distributions to the distributions of the other algorithms.

The results of the comparison of MEMOTS with the three other memetic algorithms are given to the table 5 and 6. We used only the D_1 and D_2 indicators, particularly revealing. We test the following assumption: 'an algorithm is better than an other' for the D_1 or D_2 indicator on a given instance. When the assumption is checked, the result '$>$' (MEMOTS is better) or '$<$' (MEMOTS is worse) is given as well as the value of the corresponding risk (the 'p-value', given between brackets). When the assumption is not checked, the sign '=' is indicated.

We can thus affirm with a very low risk (equal to 6.3e-08) that, for the D_1 indicator, MEMOTS is better than the three other algorithms for the 250-2, 500-2, 750-2 and 250-3 instances. For the 500-3 instance, MEMOTS is of less quality than PMA and MOGLS but remains better than IMMOGLS. For the 750-3 instance, MEMOTS is worse than the three other algorithms.

For the D_2 indicator, MEMOTS is better than the three other memetic algorithms for the 250-2 and 250-3 instances. For the 500-3 instance, MEMOTS is incomparable with MOGLS and PMA but is better than MOGLS. For the the other instances, MEMOTS is either worse, or incomparable with the other algorithms.

TABLE 4. Comparison of the MEMOTS, MOGLS, IMMOGLS and PMA indicators for the 3 objectives instances.

Instance	Algorithm	SM	R	D_1	D_2	$\|PE\|$	Time(S)
250-3	MEMOTS	**4.84**	**311.41**	**4.20**	**5.86**	**10829.85**	155
	MOGLS	7.29	315.64	4.38	6.14	1928.95	16
	IMMOGLS	10.34	324.90	4.84	7.41	483.65	**11**
	PMA	7.51	316.62	4.43	6.46	1635.75	**11**
	GREEDY	13.22	327.41	4.94	8.26	172.00	4
	MEMOTS+	4.75	309.64	4.08	5.20	12070.20	157
500-3	MEMOTS	**5.80**	602.18	5.23	7.36	**19717.45**	595
	MOGLS	9.37	**599.21**	4.94	7.36	2827.20	49
	IMMOGLS	15.11	613.90	5.53	9.57	450.45	40
	PMA	9.38	599.33	**4.93**	**7.20**	2755.95	**38**
	GREEDY	16.89	615.46	5.56	10.12	240.15	23
	MEMOTS+	5.65	589.09	4.57	5.97	24489.55	559
750-3	MEMOTS	**6.09**	878.31	6.22	10.70	**29921.65**	1921
	MOGLS	10.92	868.80	5.61	8.19	2930.15	104
	IMMOGLS	17.61	884.60	6.03	10.04	465.30	98
	PMA	10.93	**868.53**	**5.60**	**8.10**	2900.95	**95**
	GREEDY	18.86	886.11	6.10	11.32	291.25	63
	MEMOTS+	5.86	855.62	5.24	7.12	39489.65	1623

TABLE 5. Results of the Mann-Whitney test for the D_1 indicator.

	Instance	MOGLS	IMMOGLS	PMA
MEMOTS	250-2	> (6.3e-08)	> (6.3e-08)	> (6.3e-08)
	500-2	> (6.3e-08)	> (6.3e-08)	> (6.3e-08)
	750-2	> (6.3e-08)	> (6.3e-08)	> (6.3e-08)
	250-3	> (6.3e-08)	> (6.3e-08)	> (6.3e-08)
	500-3	< (6.3e-08)	> (6.3e-08)	< (6.3e-08)
	750-3	< (6.3e-08)	< (6.3e-08)	< (6.3e-08)

TABLE 6. Results of the Mann-Whitney test for the D_2 indicator.

	Instance	MOGLS	IMMOGLS	PMA
MEMOTS	250-2	> (5.4e-03)	> (1.7e-07)	> (1.8e-02)
	500-2	< (1.2e-02)	< (5.1e-04)	=
	750-2	< (7.3e-09)	< (8.5e-07)	< (2.3e-08)
	250-3	> (2.0e-02)	> (8.5e-07)	> (9.7e-04)
	500-3	=	> (3.7e-07)	=
	750-3	< (4.3e-06)	=	< (6.5e-07)

We can explain that MEMOTS does not reach the performances of the other memetic algorithms on certain instances and for certain indicators by the following fact. The other memetic algorithms need two initialization phases. The aim of the first one is to provide an approximation of an ideal point, necessary to the evaluation functions used in the various procedures of the algorithms. The second one is employed to generate an initial population of solutions. A greedy algorithm, particularly effective for the knapsack problem, is used. The MEMOTS algorithm does not use this initialization phase, and starts from a simple solution randomly generated, to show that there is not a too great dependence between the performances of the algorithm and the quality of the initialization phase.

To prove the importance of the initialization phase in the MOGLS, IMMOGLS and PMA algorithms, we add in tables 3 and 4, the indicators values of the solution set obtained after the initialization phase (called GREEDY) of the three other memetic algorithms (which use the same initialization phase). We can see that these results are already quite good, which means that the main iterations of MOGLS, IMMOGLS and PMA do not bring so many improvements in the solution set.

Moreover, we also add in the table 3 and 4 the results of MEMOTS, which uses as initial solutions, the non-dominated solutions generated by the initialization phase of MOGLS, IMMOGLS and PMA (the GREEDY algorithm). These results are given under the name of MEMOTS+. In this way, although the execution time of the algorithm increases for the 2 objectives instances, the application of this initialization phase makes it possible to improve the MEMOTS performances and to make MEMOTS+ better than the other algorithms on all the instances and for all the quality indicators used, except for the D_2 indicator of the 750-2 instance. Thus, when for a given problem, an effective algorithm allowing to generate good initial solutions is available, it is interesting to use it in order to improve the MEMOTS performances, but it's not essential since the MEMOTS performances starting from a simple solution randomly generated remain very good.

For the execution time, we can see that MEMOTS is faster for the 2 objectives instances but for the 3 objectives instances, the MEMOTS time is higher than the others algorithms. It is simply because the number of solutions generated by MEMOTS is very high in comparison to the other algorithms, and the updating of the potentially efficient solution set as well as the calculation of the DPR ranks take more time in this case. On the other hand, the number of evaluations of the other algorithms is more important that MEMOTS, since the parents selection is based on the evaluation of the individuals according to a scalarizing function, which is a technique not used in MEMOTS.

The comparison, in the objective space, between the efficient solution set $E(P)$ and the solutions of MEMOTS, MOGLS and PMA for the 750-2 instance is represented at figure 14. The solution sets of the three memetic algorithms are very close to the efficient solution set, and are practically similar, apart that MEMOTS generates more solutions than the other metaheuristics.

It comes out from these comparisons that the resolution scheme MEMOX, which made it possible to develop the MEMOTS algorithm for the resolution of the

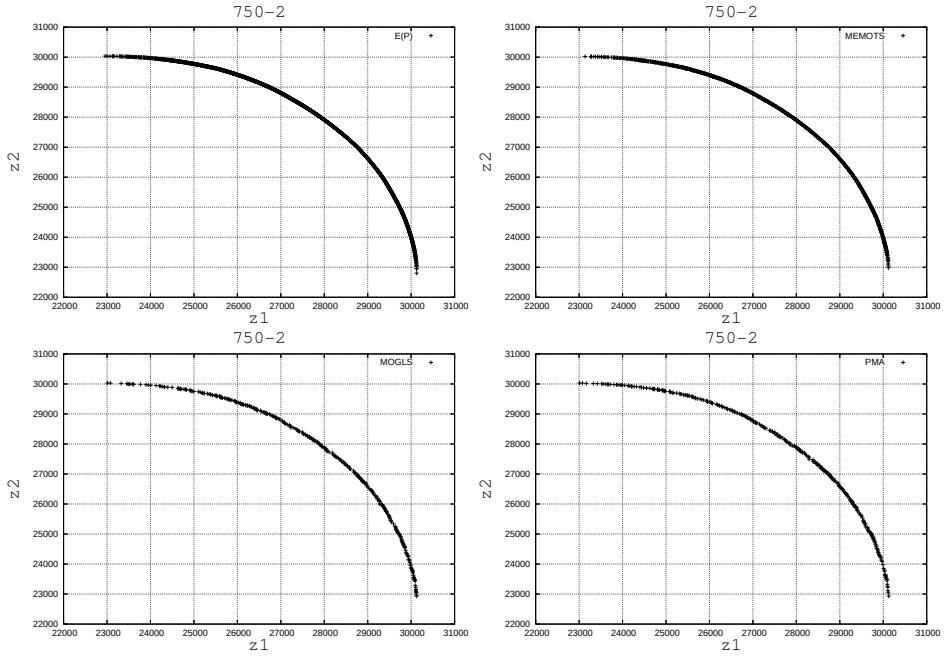

FIGURE 14. Comparison between the efficient solution set $E(P)$, and the solutions of MEMOTS, MOGLS and PMA in the objective space for the 750-2 instance.

knapsack problem, seems really competitive and essentially for the 2 objectives instances, since MEMOTS manages to benefit from the tabu search for these instances. Also, MEMOTS, at least on basis of this knapsack problem, seems well adapted to problems where the evaluation of a solution could take time, since MEMOTS never rejects a solution if the evaluation of the solution is not satisfactory according to the scalarizing function considered.

7. Conclusions and perspectives

We have presented in this paper MEMOX, a new memetic algorithm scheme for multiobjective optimization, employed with an original multiobjective tabu search PRTS, what gave rise to the memetic algorithm MEMOTS. However, the MEMOX scheme can be used with any local search.

The experiments carried out on this algorithm made it possible to highlight the following points:

- The problem of the parameters determination is even less obvious in multiobjective optimization since various properties in the solution set are required.

- The neighborhood must be adapted according to the instance. A question arises then: why not use several neighborhoods, and using that which is best adapted to the zone of the objective space where the current solution is located. For example, if the solution is in a zone of low density, small neighborhood step, and if the solution is in a zone of high density, great neighborhood step.
- The first parent selection based on the density measure improves the diversity of the solutions and the convergence speed.
- The conclusions of Ishibuchi were confirmed: it is better to select two parents quite close in the objective space.
- The parameters of the dynamic hypergrid, making it possible to measure the density of the individuals, are relatively easy to set, and this technique could be easily integrated in other multiobjective algorithms.
- The use of tabu search did not appear necessary for the 3 objectives instances, where it is preferable to use a diversified exploration of the research space.

Also, the comparisons of the MEMOTS performances with other multiobjective memetic algorithms showed the excellent properties of the algorithm. To reduce the computational time of MEMOTS on the instances with more than two objectives, one point of view would be to reduce the number of solutions generated (if only a small set of diversified solutions is required), while being based on the (1+1)-PAES method [21], where new solutions are rejected if they do not bring diversity, measured by the hypergrid.

The diversity in the solution sets obtained could also be improved by using a more diversified population of solutions, i.e. which is not only composed of non-dominated solutions.

Lastly, the adaptation of MEMOTS to other multiobjective combinatorial optimization problems will be interesting, to see if the performances of MEMOTS are confirmed and if the facility of determination of the parameters of the hypergrid is always present.

References

[1] V. Barichard and J-K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Journal of Tsinghua Science and Technology*, 8(1):8–13, 2003.

[2] V. Barichard and J.K. Hao. An empirical study of tabu search for the mokp. In Series of Information & Management Sciences, editor, *Proceedings of the First International Workshop on Heuristics*, volume 4, pages 47–56, China, 2002.

[3] Y. Collette and P. Siarry. *Optimisation multiobjectif*. Eyrolles, 2002.

[4] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

[5] P. Dagnelie. *Statistique théorique et appliquée*. De Boeck-Université, Bruxelles, 1998.

[6] K. Deb, S. Agrawal, A. Pratab, 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*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[7] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50, Washington, 1989. Springer. Lecture Notes in Computer Science No. 1917.

[8] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Kluwer Academic Publishers, Boston, 2002.

[9] S. Elaoud, T. Loukil, and J. Teghem. Pareto fitness genetic algorithm. To appear in European Journal of Operational Research, 2005.

[10] X. Gandibleux, M. Sevaux, K. Srensen, and V. T'Kindt. *Metaheuristics for Multiobjective Optimisation*. Springer, 2004.

[11] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[12] H. Ishibuchi and S. Kaige. Comparison of Multiobjective Memetic Algorithms on 0/1 Knapsack Problems. In Alwyn Barry, editor, *2003 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 222–227, Chicago, Illinois, USA, July 2003. AAAI.

[13] H. Ishibuchi and T. Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[14] H. Ishibuchi and K. Narukawa. Recombination of Similar Parents in EMO Algorithms. In C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 265–279, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[15] A. Jaszkiewicz. Genetic Local Search for Multiple Objective Combinatorial Optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.

[16] A. Jaszkiewicz. Experiments done with the momhlib: http://www-idss.cs.put.poznan.pl/jaszkiewicz/momhlib/. Technical report, 2000.

[17] A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, July 2000.

[18] A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. Technical Report RA-003/01, Institute of Computing Science, Poznan University of Technology, Poznan, Poland, 2001.

[19] A. Jaszkiewicz. Genetic Local Search for Multiple Objective Combinatorial 0ptimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[20] A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, August 2002.

[21] J. Knowles and D. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.

[22] J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[23] J. Knowles and D. Corne. Memetic algorithms for multiobjective optimization: issues, methods and prospects. In N. Krasnogor, J.E. Smith, and W.E. Hart, editors, *Recent Advances in Memetic Algorithms*, pages 313–352. Springer, 2004.

[24] T. Lust and J. Teghem. PRTS+D et MEMOTS : Nouvelles Mtaheuristiques pour l'Optimisation Combinatoire Multicritre. In *Actes des articles longs slectionns lors du 7me congrs de la roadef*, pages 137–151, Lille, February 2006. Presses Universitaires de Valenciennes.

[25] Z. Michalewicz and J. Arabas. Genetic algorithms for the 0/1 knapsack problem. In Z.W Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems Conference (ISMIS)*, pages 134–143, Berlin, 1994.

[26] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report C3P 826, Caltech Concurrent Computation Program, 1989.

[27] J.R. Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Institute of Technology, Department of Aeronautics and Astronautics, Massachusetts, 1995.

[28] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[29] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Applications*. John Wiley & Sons, New-York, 1985.

[30] E-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(2):541–564, 2002.

[31] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[32] L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, 2005.

[33] E. Zitzler. http://www.tik.ee.ethz.ch/∼zitzler/testdata.html.

[34] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[35] E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. Grunert da Fonseca. Why Quality Assessment of Multiobjective Optimizers Is Difficult. In W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 666–673, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[36] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.