Journal of Computing and Information Technology - CIT 16, 2008, 1, 1–13
doi:10.2498/cit.1000809


# On the Sensitivity of Aggregative Multiobjective Optimization Methods

Yann Collette[1] and Patrick Siarry[2]

[1] Technocentre Renault, Guyancourt, France
[2] Université Paris 12, Créteil, France

In this paper, we present a study on the sensitivity of aggregation methods with respect to the weights associated with objective functions of a multiobjective optimization problem. To do this study, we have developped some measurements such as the speed metric or the distribution metric. We have performed this study on a set of biobjective optimization test problems: a convex, a non-convex, a continuous and a combinatorial test problems.

We show that some aggregation methods are more sensitive than others.

*Keywords:* multiobjective optimization, Chebyshev, weights, aggregation, metrics, performance, random, stochastic

## 1. Introduction

### 1.1. Introduction to Multiobjective Optimization

The multiobjective optimization has been a growing domain of interest since approximately 1990 [4]. A lot of methods have been developed so as to solve a multiobjective optimization problem. Many of these methods use a genetic algorithm to solve this problem without transforming it into an "equivalent" monobjective optimization problem, but most of the methods use a transformation to return to a monobjective optimization problem.

An example of the problem we are dealing with in multiobjective optimization is the following:

$$\left|\begin{array}{ll} \text{minimize} & f_1(\vec{x}) \\ \text{minimize} & f_2(\vec{x}) \\ \text{with} & \vec{x} \in S \subset \mathbf{R}^n \end{array}\right.$$

This example is a biobjective optimization problem.

When we perform an optimization, we must keep in mind the definition of an optimum solution. In monobjective optimization, this definition is easy to understand. But, in multiobjective optimization, we use a different form of definition for an optimum solution. And such a solution to a multiobjective optimization problem will be called a Pareto optimum solution. Let's consider a multiobjective optimization problem with $m$ objective functions to be minimized.

A solution $A$ is optimal with respect to a solution $B$ if:

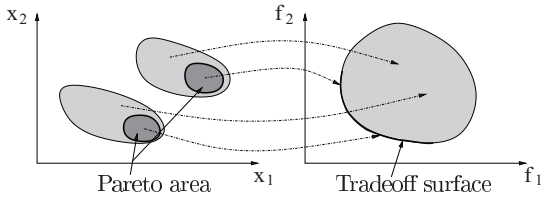



*Figure 1.* The tradeoff surface and the Pareto area.

- the objective function values of the solution $A$ are as good or better than the objective function values of the solution $B$ $(\forall i \in \{1, \cdots, m\} f_i(A) \leq f_i(B))$;
- the solution $A$ outperforms the solution $B$ for at least one objective $(\exists i \in \{1, \cdots, m\}$ $f_i(A) < f_i(B))$.

As we can notice when trying to solve a multiobjective optimization problem, such a problem hasn't got just one solution. Most of the time, we can find a huge number of solutions.

Indeed, all these solutions belong to a hypersurface and this locus of optimal solutions is called the tradeoff surface or the Pareto frontier (see Figure 1).

Not all these transformation methods (also called aggregation methods) are equivalent. Some of these transformations are sufficiently "efficient" so as to deal with a non convex multiobjective optimization problem and some of them aren't [3]. But, when one has chosen a method, one has to overcome the problem of choosing some weight coefficients so as to underline the tradeoff one is willing to do [1]. As we shall see, the problem of choosing a set of weights is not really critical because some methods are not really sensitive to a weight variation and so, one can give "raw" weights to the method to model a tradeoff.

We present a study on the sensitivity of aggregation methods with respect to the weights associated with objective functions of a multiobjective optimization problem. To do this study, we have developed some measurements such as the speed metric or the distribution metric. We have performed this study on a set of biobjective optimization test problems: a convex, a non-convex, a continuous and a combinatorial test problems. The results make us conclude that some aggregation methods are more sensitive than others.

In section 2, we present the various apparatus we used to perform our experiments: the optimization method, the aggregation methods, the metrics and the test problems. In section 2, we present the various steps followed during the experiment. In section 3, we present the results computed during the experiment and we conclude in section 4.

## 2. Description of the Experiment

Our aim is to study the influence of the aggregation method on the locus we reach on the Pareto surface, with respect to a certain amount of randomness in the optimization method.

We will try to quantify the amount of stochastic effect in the MOSA method [5]. We will also try to rank the main aggregation methods (the weighted sum of objective functions, the Chebyshev method and the hybrid method) with respect to their sensibilities to a variation of weights.

In order to test the random level of an optimization method, we have built a new optimization method: a local search method with a fixed probability of accepting a bad solution. This algorithm is described on algorithm 1.

By using this algorithm, we try to model the behavior of the simulated annealing with a constant temperature. With a real simulated annealing, the probability of accepting a bad solution depends on the level of the objective function associated with the solution. So, when the temperature is constant, the probability of accepting a bad solution is likely to change.

Nevertheless, the randomized local search in the main line models the simulated annealing with a constant temperature.

**Algorithm 1** The randomized local search

| | |
|---|---|
| $p_{accept}$ | probability of accepting a bad solution |
| $x_{curr}$ | the current solution |
| $Neigh(x)$ | returns a neighbor solution of solution $x$ |
| $f()$ | objective function to optimize |
| $N_{iter}$ | number of iterations to perform |
| | `for` $i = 1$ `to` $N_{iter}$ |
| | $x = Neigh(x_{curr})$ |
| | `if` $f(x) \leq f(x_{curr})$ `then` $x_{curr} = x$ |
| | `if` $f(x) > f(x_{curr})$ `and` $rand(0,1) < p_{accept}$ `then` $x_{curr} = x$ |
| | `end` |

## 2.1. The Aggregation Methods

### 2.1.1. The Weighted Sum of Objective Functions

The first method we used is a classical method in multiobjective optimization. We calculated the weighted sum of objective functions so as to aggregate objectives and have an equivalent single objective function to be optimized. This method (see in [4]) is defined as follows:

$$f_{eq}(\vec{x}) = \sum_{i=1}^{N_{obj}} \omega_i \cdot f_i(\vec{x})$$

where:

- $f_i$ is the objective function number i of the multiobjective optimization problem,
- $N_{obj}$ is the number of objectives in the multiobjective optimization problem,
- $\omega_i$ is the weight associated with the objective function number i,
- $\vec{x}$ is the decision vector,
- $f_{eq}$ is the single objective function.

This method shows poor performances on non-convex solution sets. It can't find the solutions hidden in non-convexities of the Pareto frontier (see in [3] or in [1]).

### 2.1.2. The Chebyshev Aggregation of Objective Functions

Another way to aggregate objective functions is to use the Chebyshev distance (see in [4]). This way of aggregating objective functions is a very efficient one, it can find solutions hidden in non-convexities of the Pareto frontier. This distance is defined as follows:

$$f_{eq}(\vec{x}) = \max_{i=1,\cdots,N_{obj}} \omega_i \cdot (f_i(\vec{x}) - F_i)$$

For this method, the $F_i$'s correspond to an ideal bound for the objective $f_i$. Because sometimes it's hard to find good bounds for this method, we have decided to change the expression of the Chebyshev method:

$$f_{eq}(\vec{x}) = \max_{i=1,\cdots,N_{obj}} (1 - \omega_i) \cdot (f_i(\vec{x}) - F_i)$$

where $F_i$ refers to a limit on the objective $f_i$ under which the values of $f_i$ are interesting for the decision maker.

The notations are the same as those defined above.

### 2.1.3. The Hybrid Aggregation of Objectives Functions

Another way to aggregate objective functions is to use the Hybrid distance (see in [4]). This way of aggregating objective functions is nearly as efficient as with the Chebyshev one. It can find solutions hidden in non convexities of the Pareto frontier if the $K$ parameter is well tuned. This distance is defined as follows:

$$f_{eq}(\vec{x}) = \max_{i=1,\cdots,N_{obj}} \omega_i \cdot (f_i(\vec{x}) - F_i) + K \cdot \sum_{i=1}^{n} \omega_i \cdot f_i(\vec{x})$$

As with the Chebyshev method, we will use a transformed expression of the hybrid method which is related to the transformed expression of the Chebyshev method:

$$f_{eq}(\vec{x}) = \max_{i=1,\cdots,N_{obj}} (1 - \omega_i) \cdot (f_i(\vec{x}) - F_i) + K \cdot \sum_{i=1}^{n} \omega_i \cdot f_i(\vec{x})$$

For this method, we performed some experiments so as to tune the value of the coefficient $K$. We used the non convex TRIGO test problem. We searched the value of $K$ for which the distribution of solutions along the Pareto surface was as near as possible to the distribution of solutions we have obtained with the Chebyshev method and 100 equally distributed weights.

## 2.2. The Metrics Used

### 2.2.1. The Distribution Metric

The goal of the distribution metric is to count the number of solutions along the tradeoff surface. To do so, one first has to find solutions equally spaced on the tradeoff surface. This

first requirement implies knowing the analytical expression of the tradeoff surface of the test problem. Having computed this set of solutions, one has to translate it using two vectors $\vec{v}$ and $-\vec{v}$. Then, one builds cells around an area of the tradeoff surface by linking 4 neighbor solutions (see Figure 2).

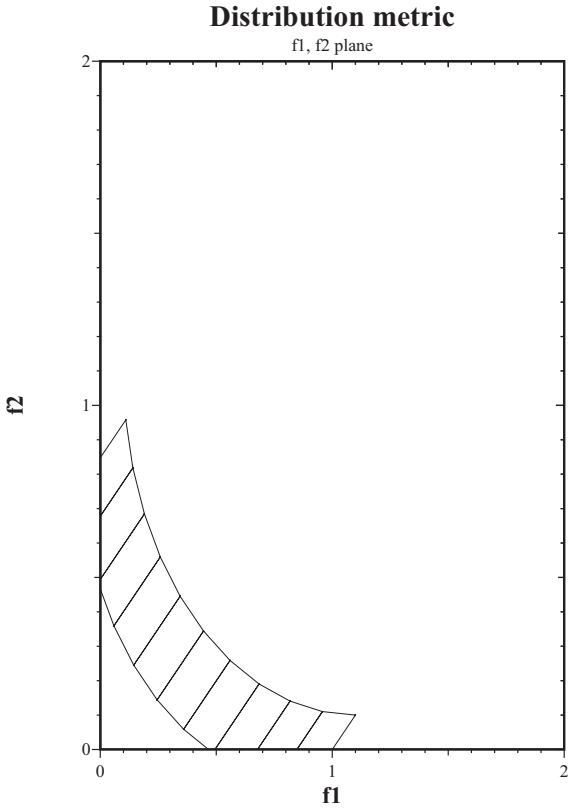



*Figure 2.* The shape of the distribution metric.

This metric is easy to use: one has to put a threshold frontier so as to stop the optimization when all the solutions are lying inside the cells, then one counts the solutions in the cells and reports these results on a bar chart.

### 2.2.2. The Speed Metric

One way to compute the speed of an algorithm in single objective optimization is to set a threshold (over the global optimum value) and wait for the algorithm to reach this threshold. To be able to use this kind of "speed measurement", we must use a test problem for which the local minima are well known. Using such a test problem, we will be able to size the value of the threshold correctly.

It is relatively easy to translate this idea for multiobjective optimization, with the same limitations as with the single objective optimization test problem: we must work with a biobjective test problem. To compute the threshold frontier, we follow the steps illustrated in Figure 3.

Step 1: compute the theoretical Pareto frontier (see Figure 3a)

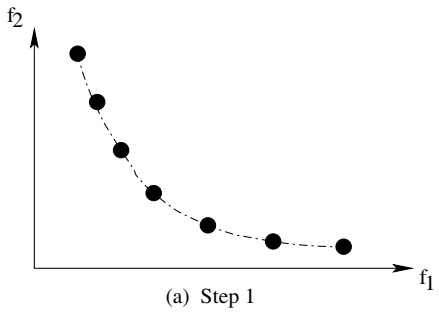
Step 2: move the theoretical Pareto frontier using a threshold vector $\vec{v}$ (see Figure 3b). The amplitude of $\vec{v}$ is not really important, but it must be sufficiently important so as to catch a set of Pareto efficient solutions. The translation of the Pareto frontier using vector $-\vec{v}$ is performed so as to be sure to surround all the points in a cell defined by 2 Pareto efficient points (between these 2 points, we can have some solutions which can be under the line joining these 2 points).

Step 3: remove points above $\arg\max f_2$ and below $\arg\min f_2$

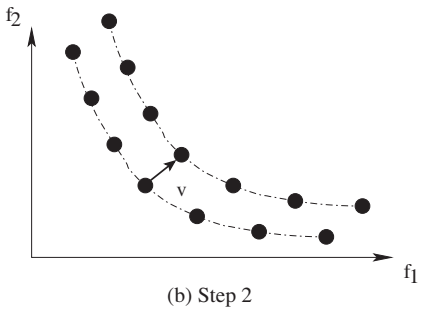
Step 4: add two points $(f_1(A), \max f_2)$ and $(\max f_1, f_2(B))$ (see Figure 3c)

Step 5: add two points $(\min f_1, \max f_2)$ and $(\max f_1, \min f_2)$ (see Figure 3d)

Step 6: sort points in an increasing order, considering the first objective $f_1$ and the threshold frontier are defined by the path linking all these points (see Figure 3e)




(a) Step 1




(b) Step 2

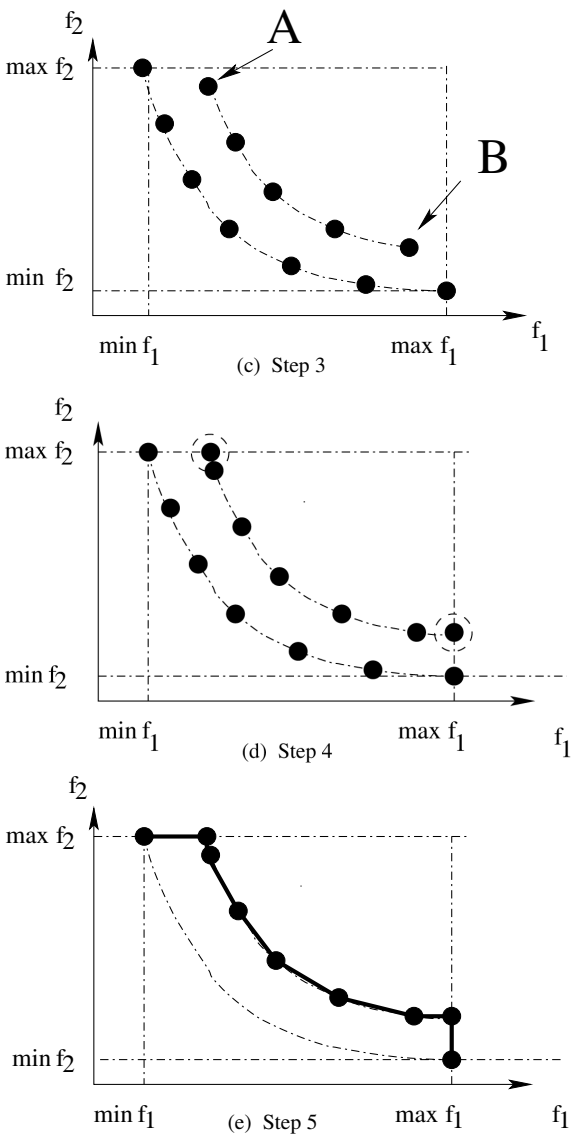



*Figure 3.* How to build a treshold frontier in the speed metric.

Let us use the following notation:

- $x_i^{TF}$ the $i^{\text{th}}$ point of the threshold frontier
- $x_j^S$ the $j^{\text{th}}$ point of the solution set

To perform the test, we follow these steps for each $x_i^S \in S$:

Step 1: find $i$ such as $f_1\left(x_i^{TF}\right) \leq f_1(x_j^S) \leq f_1\left(x_{i+1}^{TF}\right)$

Step 2: we denote $f_2(x) = A \cdot f_1(x) + B$ the line between the points $\left(f_1\left(x_i^{TF}\right), f_2\left(x_i^{TF}\right)\right)$ and $\left(f_1\left(x_{i+1}^{TF}\right), f_2\left(x_{i+1}^{TF}\right)\right)$: if $\left(f_2\left(x_j^S\right) - A \cdot f_1\left(x_j^S\right) - B \leq 0\right)$ then the point $\left(f_1\left(x_i^S\right), f_2\left(x_i^S\right)\right)$ is under the threshold frontier

To use the threshold frontier as a test to stop the running of a multiobjective optimization method, we can count the number of points that fall under the threshold frontier. If all the points (or a fraction like 90% for example) are under the threshold frontier, we stop the optimization method and check how many iterations were necessary to move all the points under the threshold. The theoretical Pareto frontier doesn't need to verify any convexity hypothesis. The speed metric can be applied to any kind of multiobjective optimization problem since we are able to compute the theoretical Pareto frontier.

This stopping test for multiobjective optimization methods seems a good one to measure the speed of a method.

## 2.3. The Test Problems

### 2.3.1. The Continuous Test Problems

To be able to perform our experiments, we need a set of test problems so that we can easily move the starting point of the optimization and start closer to or farther from the Pareto set. Such a test problem doesn't exist in the published set of multiobjective test problems. So we have developed a simple set of test problems for which we can tune easily the position of the starting point with respect to the Pareto set. This set of test problems covers the classical difficulties we meet in multiobjective optimization: non-convexity, discontinuities, etc.

We will consider the following test problems:

**The convex TRIGO1 test problem**

$$\begin{array}{ll} \text{minimize} & f_1(\theta, x) = 1 - \cos(\theta) + x \\ \text{minimize} & f_2(\theta, x) = 1 - \sin(\theta) + x \\ \text{with} & 0 \leq \theta \leq \frac{\pi}{2} \text{ and } 0 \leq x \leq 1 \end{array}$$

**The convex TRIGO2 test problem**

$$\left|\begin{array}{ll} \text{minimize} & \begin{cases} f_1(\theta) = 1-\cos(\theta)+x & \text{if } 0 \le x \le 0.25 \\ f_1(\theta) = 1-\cos(\theta)+0.25\cdot x+\frac{3}{16} & \text{if } 0.25 \le x \le 0.75 \\ f_1(\theta) = 1-\cos(\theta)+2.5\cdot x-1.5 & \text{if } 0.75 \le x \le 1 \end{cases} \\ \text{minimize} & \begin{cases} f_2(\theta) = 1-\sin(\theta)+x & \text{if } 0 \le x \le 0.25 \\ f_2(\theta) = 1-\sin(\theta)+0.25\cdot x+\frac{3}{16} & \text{if } 0.25 \le x \le 0.75 \\ f_2(\theta) = 1-\sin(\theta)+2.5\cdot x-1.5 & \text{if } 0.75 \le x \le 1 \end{cases} \\ \text{with} & 0 \le \theta \le \frac{\pi}{2} \text{ and } 0 \le x \le 1 \end{array}\right.$$

**The convex TRIGO3 test problem**

$$\left|\begin{array}{ll} \text{minimize} & \begin{cases} f_1(\theta) = 1-\cos(1.5\cdot\theta) & \text{if } 0 \le \theta \le \frac{\pi}{8} \\ f_1(\theta) = 1-\cos(0.5\cdot\theta+\frac{\pi}{8}) & \text{if } \frac{\pi}{8} \le \theta \le \frac{3\cdot\pi}{8} \\ f_1(\theta) = 1-\cos(1.5\cdot\theta-\frac{\pi}{4}) & \text{if } \frac{3\cdot\pi}{8} \le \theta \le 1 \end{cases} \\ \text{minimize} & \begin{cases} f_2(\theta) = 1-\sin(1.5\cdot\theta) & \text{if } 0 \le \theta \le \frac{\pi}{8} \\ f_2(\theta) = 1-\sin(0.5\cdot\theta+\frac{\pi}{8}) & \text{if } \frac{\pi}{8} \le \theta \le \frac{3\cdot\pi}{8} \\ f_2(\theta) = 1-\sin(1.5\cdot\theta-\frac{\pi}{4}) & \text{if } \frac{3\cdot\pi}{8} \le \theta \le 1 \end{cases} \\ \text{with} & 0 \le \theta \le \frac{\pi}{2} \text{ and } 0 \le x \le 1 \end{array}\right.$$

We find the analytical expression of the tradeoff surface with $x = 0$. The expression of this surface is the following:

$$\left|\begin{array}{l} f_1(\theta) = 1-\cos(\theta) \\ f_2(\theta) = 1-\sin(\theta) \\ \text{with } 0 \le \theta \le \frac{\pi}{2} \end{array}\right.$$

We have the same expression for the TRIGO3 test problem if we don't consider the point density variation with respect to $\theta$.

We will also use a non convex form of the preceding problem. The analytical expression of this problem is similar to the convex one. We just have to change $1-\cos()$ and $1-\sin()$ into $\cos()$ and $\sin()$ respectively.

### 2.3.2. The Combinatorial Test Problems

We will apply a continuous to combinatorial (binary, to be more precise) transformation, so as to use the same kind of test problem in the continuous space and in the combinatorial space.

To obtain the combinatorial test problem:

- we define a binary size for the contiuous variables $x$ and $\theta$ (for example 8 bits);
- the variation interval of the binary variable is included between 0 and $2^N - 1$;
- this variation interval is normalized to the variation interval $[x_{min}, x_{max}]$ and $[\theta_{min}, \theta_{max}]$.

So, we find the typical behavior of the combinatorial problems (huge variation of the objective function when we change the value of one bit on the combinatorial variable) while keeping the ability to visualize the behavior of the optimization method. For example, we can:

- visualize the movements of the current point in two dimensions during the optimization;
- choose accurately the position of the initial point (we choose the initial value by using the continuous form of the test problem, then we convert this value into a binary one by following the steps we have described previously);
- build a parallel between the behavior of a combinatorial optimization method and a continuous optmization method by comparing the trajectories of both methods.

## 2.4. The Experiment

For each problem, we proceed as follows:

1. We choose an initial point which allows us to "see" the whole tradeoff surface. This point has the coordinates $(\pi/4, 1)$.

2. We compute the weights so as to find solutions equally spaced along the tradeoff surface. To do so, we take the extreme points of the tradeoff surface ($(1,0)$ and $(0,1)$), we draw a line between these two points, then we divide this line into as many pieces as we need couples of weights. We then compute the line which goes through the initial point and through one of the points of the preceding line, then we identify the leading coefficient of the line with the desired couple of weights.
3. We place a threshold frontier close to the tradeoff surface. So, when the generated points are sufficiently close to the tradeoff surface, the optimization stops.
4. We apply the distribution metric which has the shape described in Figure 2.

The experiment follows these steps:

1. Choice of the initial point $(\pi/4, 1)$ and initialization of the couple of weights to $(0.5, 0.5)$.
2. Initialization of the optimization method.
3. Initialization of the random number generator with a given random seed.
4. Execution of the optimization method, then returning to step 3. We reproduce this step 100 times.
5. We apply the distribution metric to the computed solutions to compute the distribution of solutions along the tradeoff surface.
6. $x = x - \Delta x$ then returning to step 2. We reproduce this step until $x = 0$.

We will execute two kinds of experiments using this test problem:

- An experiment with the continuous test problem.
- An experiment with the continuous test problem converted into a combinatorial one.

## 3. Results

### 3.1. The Graphs

We will present two kinds of graphs:

- graphs related to the distribution of points along the Pareto surface, with respect to the probability of accepting a bad solution and with respect to the position of the initial point.
- graphs related to the number of iterations required to reach the Pareto surface, with respect to the position of the initial point and to the probability of accepting a bad solution.

These graphs have special scales:

**Graphs showing distribution of the points**

- The dimension entitled "Initial point position" has a scale which spreads from 0 to 100 where 0 corresponds to the position of the initial point on the Pareto surface and 100 corresponds to the position of the initial point at a distance 1 from the Pareto surface.
- The dimension entitled "Surface point position" has a scale which spreads from 0 to 100, where 0 corresponds to the position $\theta = 0, x = 0$ on the Pareto surface and 100 corresponds to the position $\theta = \pi/2, x = 0$ on the Pareto surface.
- The dimension entitled "Density" has a scale which spreads from 0 to 100 where 0 corresponds to no points counted at this position on the Pareto surface and 100 corresponds to all points counted at this position on the Pareto surface.

**Speed graph**

- The dimension entitled "Initial position" has a scale which spreads from 0 to 100. The meaning of these values is the same as in the "Initial point position" in the distribution graph mentioned above.
- The dimension entitled "Probability" has a scale which spreads from 0 to 10 where 0 corresponds to the probability of accepting a bad solution of 1 and 10 corresponds to the probability of accepting a bad solution of 0.
- The dimension entitled "Iterations" has a scale which varies depending on the maximum number of iterations needed to reach the Pareto surface. The scale of this dimension is represented in the legend on the left of the Figures.

## 3.2. Some Results

**The Chebyshev method and the continuous test problem**

First, we tested the Chebyshev method on the convex and continuous test problems. The results we have obtained with this test problem are represented in Figures 4a, 4b, 5a. The speed diagram is represented in Figure 5b.

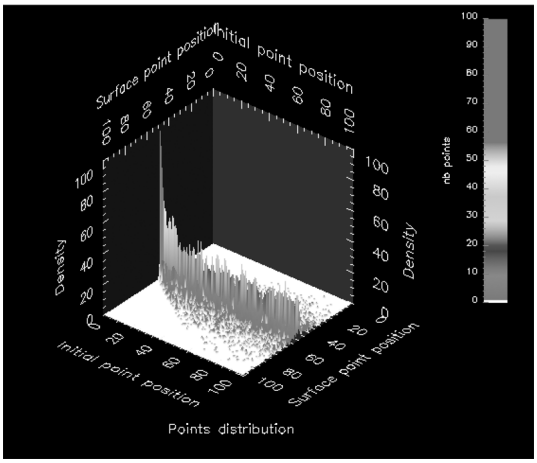



*Figure 4a.* Probability 0.9.

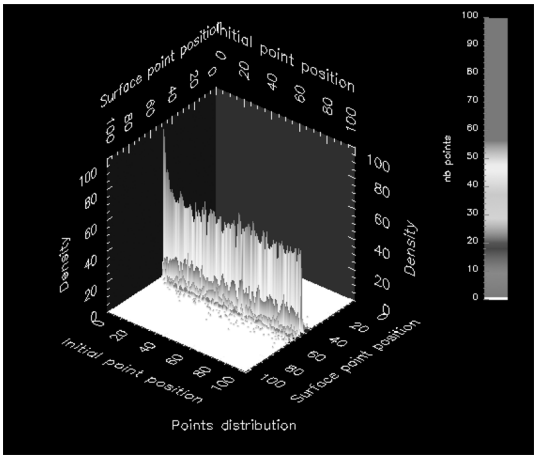



*Figure 4b.* Probability 0.5.

*Figure 4.* Results for the continuous test problem: the Chebyshev method for probabilities 0.9 and 0.5.

The conclusions we can draw about this experiment are the following:

- When the probability of accepting a bad solution reaches 0, the distribution of points along the Pareto surface is concentrated on the locus designated by the direction given by the couple of weights. This is a normal behavior of a multiobjective optimization method.
- When considering the speed diagram, the closer the initial point to the Pareto surface, the less iterations we need to reach the Pareto surface. The closer to 0 the probability of accepting a bad solution, the less iterations we need to reach the Pareto surface. Again, this is a normal behavior of a multiobjective optimization method.

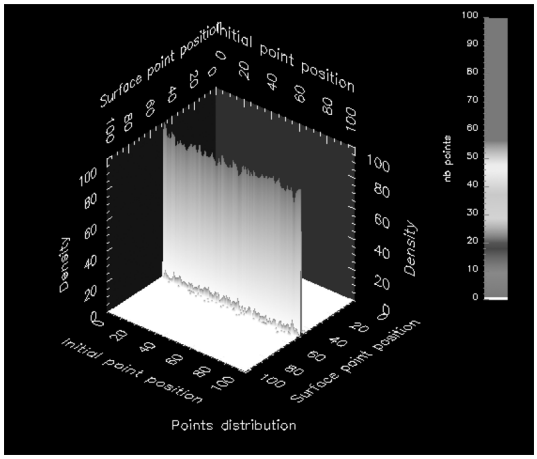



*Figure 5a.* Probability 0.1.

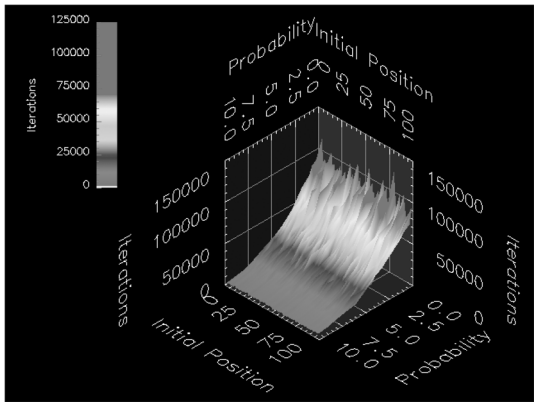



*Figure 5b.* The speed diagram 0.5.

*Figure 5.* Results for the continuous test problem: the Chebyshev method for probability 0.1 and the speed diagram.

**The Chebyshev method and the combinatorial test problem**

Then, we have tested the Chebyshev method on the convex and combinatorial test problem with a 64 bits variable. The results we have obtained with this test problem are represented in Figures 6a, 6b, 7a. The speed diagram is represented in Figure 7b.

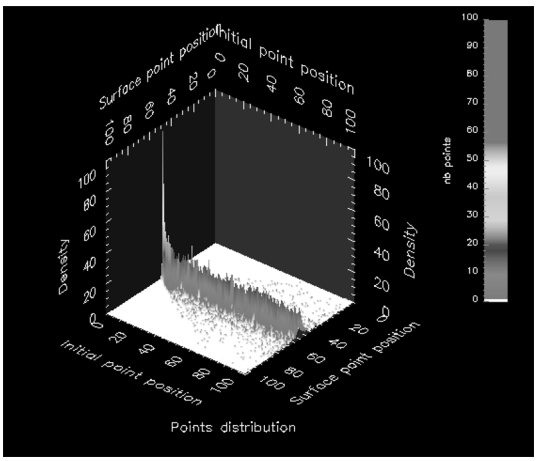



*Figure 6a.* Probability 0.1.

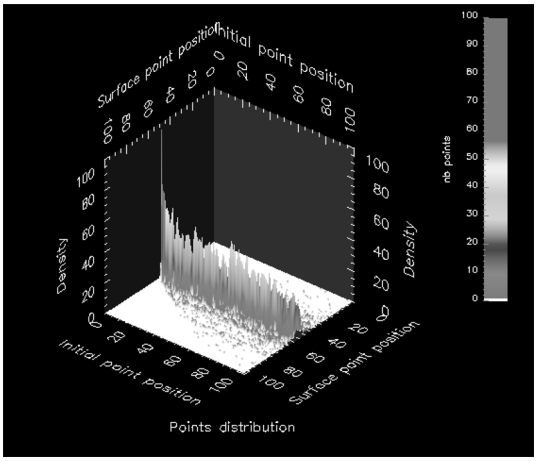



*Figure 6b.* Probability 0.2.

*Figure 6.* Results for the combinatorial test problem with a 64 bits variable: the Chebyshev method for probabilities 1.0 and 0.2.

The conclusions we can draw are rather interesting:

- We have nearly the same influence of the probability of accepting a bad solution on the distribution of points as with the continuous case. The main difference lies in the fact that the distribution evolves rather abruptly with respect to the probability of accepting a bad solution. Between 1.0 and 0.3, 0.4, the distribution is still the same as in Figure 6a. Then, between 0.3, 0.4 and 0.0, the distribution evolves quickly to look like the distribution of the continuous case.
- For the speed diagram, it is completely different from the continuous case. The initial position has no influence on the number of iterations needed to reach the Pareto surface. This is due to the combinatorial aspect of the problem. A small change in the value of a bit can induce a huge change in the value of the objective function. The meaning of "being closer to the Pareto surface" seems to lose its meaning.

We can notice a change in the shape of the speed diagram when we are closer to the Pareto surface: an overcost in terms of iterations number appears. This is certainly due to the neighborhood used: we have a mean change of 3 bits between a point and its neighbor. When we are really close to the Pareto surface, less than 3 bits need to be changed, so, to reach the Pareto surface, we first need to go back and then go forth. This movement induces an overcost in term of the number of iterations. This behavior has been observed in [2].

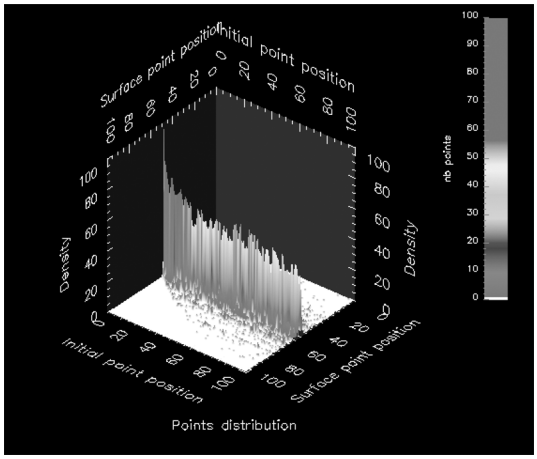



*Figure 7a.* Probability 0.1.

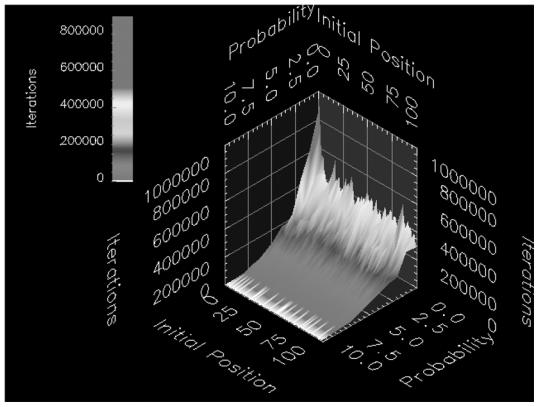



*Figure 7b.* The speed diagram.

*Figure 7.* Results for the combinatorial test problem with a 64 bits variable: the Chebyshev method for probability and the speed diagram.

### The weighted sum of objective functions and the combinatorial test problem

We will now perform the same experiments on the same combinatorial test problem, but with the weighted sum of objective functions. The points distribution obtained are represented in Figures 8a, 8b and 9a. The speed diagram is represented in Figure 9b.

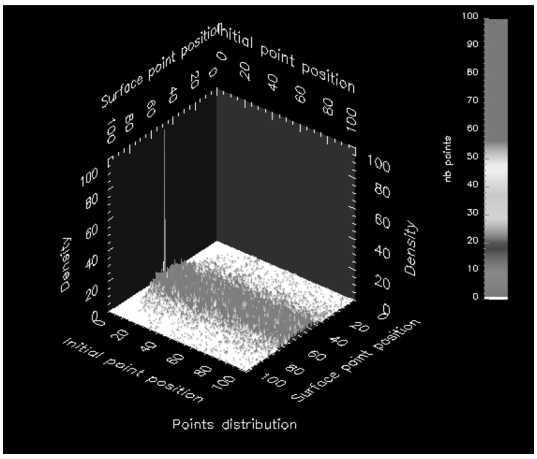



*Figure 8a.* Probability 1.0.

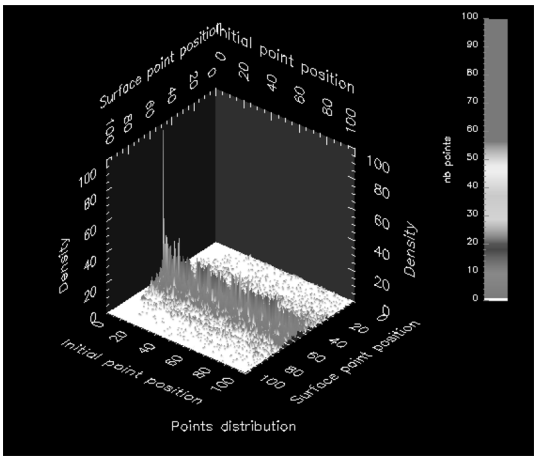



*Figure 8b.* Probability 0.2.

*Figure 8.* Results for the combinatorial test problem with a 64 bits variable: the weighted sum of objective functions for probabilities 1.0 and 0.2.

The conclusions we can draw are the following:

- The points are distributed on a wider area than with the Chebyshev method. We still have the "brutal" change in the shape of the distribution of points with respect to the probability of accepting a bad solution. Between a probability of 1 and 0.3, 0.4, the shape of the distribution of points is still "the same". But between a probability of 0.3, 0.4 and 0.0, the shape of the distribution of points changes quickly.
- We have a smaller overcost in terms of the number of iterations needed to reach the Pareto surface. This overcost doesn't appear clearly in Figure 9b, but it appears in other figures not represented here.
- We can also notice that we need less iterations to reach the Pareto surface than with the Chebyshev method. This difference can be explained theoretically and is due to the way an aggregation method "follows" a direction given by a couple of weights.
- Let us recall that the main theoretical difference between the weighted sum of

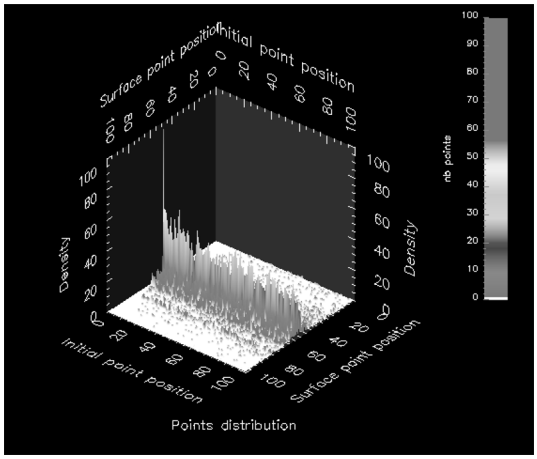



*Figure 9a.* Probability 0.1.

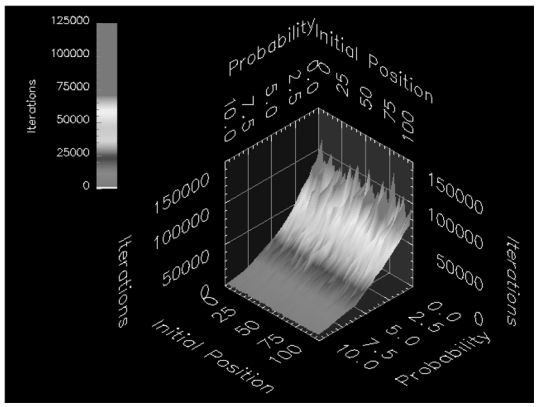



*Figure 9b.* The speed diagram.

*Figure 9.* Results for the combinatorial test problem with a 64 bits variable: the weighted sum of objective functions for probability 0.1 and the speed diagram.

objective functions and the Chebyshev method is that the Chebyshev method is efficient on multiobjective problems with a Pareto surface with a non convex shape, whereas the weighted sum of objective functions isn't.

## The hybrid method and the combinatorial test problem

Finally, we perform the same experiments on the hybrid aggregation method. The results are represented in Figures 10a, 10b and 11a. The speed diagram is represented in Figure 11b.

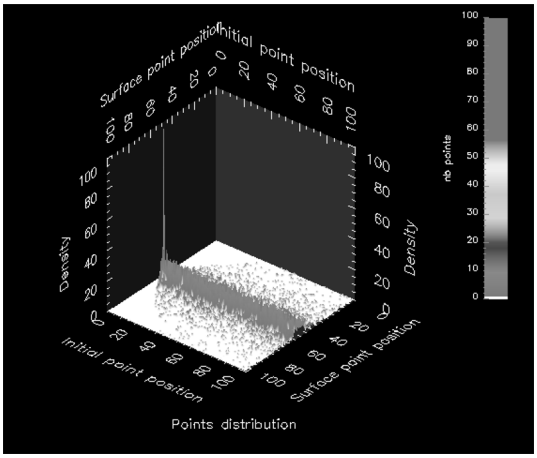



*Figure 10a.* Probability 0.1.

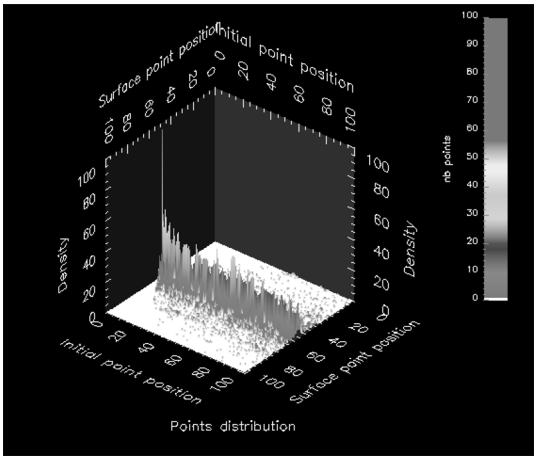



*Figure 10b.* Probability 0.2.

*Figure 10.* Results for the combinatorial test problem with a 64 bits variable: the hybrid method for probabilities 1.0 and 0.2.

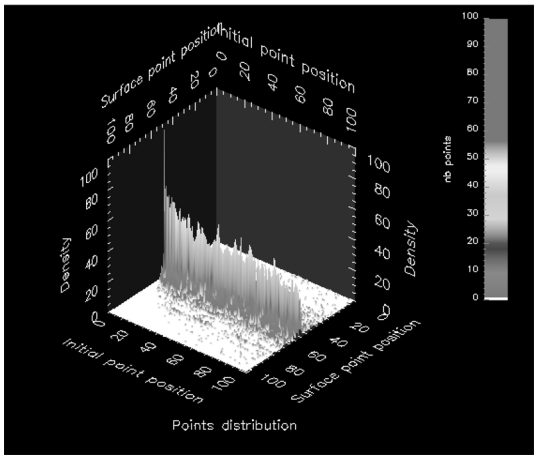



*Figure 11a.* Probability 0.1.

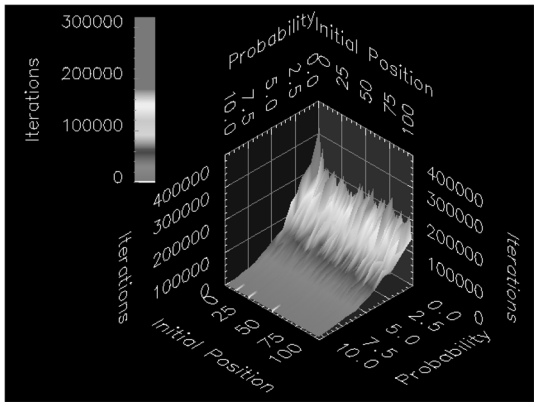



*Figure 11b.* The speed diagram.

*Figure 11.* Results for the combinatorial test problem with a 64 bits variable: the hybrid method for probability 0.1 and the speed diagram.

The conclusions we can draw are the following:

- First, let us recall that the hybrid method is efficient on multiobjective problem with Pareto surfaces of a non convex shape (it depends on the value of the coefficient $K$).
- All the results are "between" the results we obtained with the Chebyshev method and the results we obtained with the weighted sum of objective functions.
  - The width of the distributions of points is narrower than the width of the distribution of points we obtained with the weighted sum of objective functions but it is wider than the width of the distribution of points we obtained with the Chebyshev method.

– The number of iterations required to reach the Pareto surface is more important than the number of iterations required to reach the Pareto surface using the weighted sum of objective functions, but it is less important than the number of iterations required to reach the Pareto surface with the Chebyshev method.

The same kind of experiments were performed on various pairs of weights. The results were the same as with the pair $(0.5, 0.5)$.

## 4. Conclusions

These various experiments show a change in the shape of the distribution of points along the Pareto surface, with respect to the probability of accepting a bad solution.

We can observe two types of behavior:

- A behavior insensitive to the value of the probability of accepting a bad solution. In that case, the distribution of points is centered around the direction given by the couple of weights $(0.5, 0.5)$, but the distribution is really spread around this point. This induces that we can't reach accurately this point on the Pareto surface.
- A behavior sensitive to the value of the probability of accepting a bad solution. In that case, for some values of probability, we can reach accurately a locus on the Pareto surface.

We can conclude that, when using the weighted sum of objective functions with a stochastic optimization method, the choice of the weights is not really important because, due to the stochastic effect, we will not reach accurately a locus on the Pareto surface, but rather roughly.

We also notice the changes which intervene on the speed diagram. When no perturbation on the distribution of points appears, the resulting speed diagram is independent of the probability of accepting a bad solution.

On the first speed diagram, we can notice the relative independence in terms of performances, with respect to the position of the initial point. This can be explained by the "complexity" of the neighborhood we have used. When a large number of bits are modified on the binary variable, at each iteration the distance we walk is very important. This implies that the initial point position is either close or far from the Pareto surface, after the first iteration the current point reaches more or less the same position.

We also observe an interesting phenomenon: the more the initial point is close to the Pareto surface, the more it appears an overcost in terms of iterations number. This can be explained by the complexity of the neighborhood we used. The more we have bits changing, the more important is the distance walked in one iteration. So, when we are close to the Pareto surface (at a distance of 1 bit to change for example), we must first go back before being able to reach the optimal point on the Pareto surface.

Once a certain probability threshold is passed, the behavior of the distribution of points along the Pareto surface tends to come close to the behavior we found with the continuous problem.

Another phenomenon can be observed when considering the speed diagram and a neighborhood where 75% of the bits are changed (experiment not represented here). When the probability of accepting a bad solution is zero, we observe a very high overcost in terms of the number of iterations needed to reach the Pareto surface. There is a factor 10 between the part found using a probability of accepting a bad solution of 0.1 and the part found using a probability of accepting a bad solution of 0.0. This phenomenon is highlighted in the thesis of B. Krishnamachari [2]. We can imagine that for neighborhoods of high complexity, a minimum probability of accepting a bad solution is needed to allow the optimization method to adapt its movements to avoid having to go back too often when the current solution is too close to the Pareto surface. This tells us that we must be really careful when choosing the initial solution of an optimization method: this solution must be sufficiently close to the Pareto surface, but not too close.

This last phenomenon should be studied thoroughly such as the quick change in the shape of the distribution of solutions along the Pareto surface with respect to the probability of accepting a bad solution.

## References

[1] I. DAS, J. DENNIS, A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problem. *Structural Optimization*, **19**, (1997), Issue 1, 63–69.

[2] B. KRISHNAMACHARI, X. XIE, B. SELMAN, S. WICKER, Performance Analysis of Neighborhood Search Algorithms. Preprint, Janvier, 2000. `http://citeseer.nj.nec.com/293033.html`

[3] A. MESSAC, G. J. SUNDARARAJ, R. V. TAPPETA, J. E. RENAUD, Ability of Objective Functions to Generate Points on Nonconvex Pareto Frontiers. *AIAA Journal*, **38**, (2000), Issue 6, 1084–1091

[4] K. MIETTINEN, *Nonlinear Multiobjective Optimization*. Kluwer Academic Publisher, 1999.

[5] E. L. ULUNGU, J. TEGHEM, P. H. FORTEMPS, D. TUYTENTS, MOSSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multicriteria Decision Analysis*, **8** (4) (1999), 221–236.

*Received:* February, 2006
*Revised:* May, 2007
*Accepted:* May, 2007

*Contact addresses:*
Yann Collette
Renault – 68240
1 Av. du Golf
78288 Guyancourt, France
e-mail: `yann.collette@renault.com`

Patrick Siarry
LISSI
Université Paris 12
61 Av. du Général de Gaulle
94010 Créteil, France
e-mail: `siarry@univ-paris12.fr`

YANN COLLETTE graduated from Ecole Normale Supérieure de Cachan, prepared his PhD thesis on multiobjective optimization at Electricité de France. He received the PhD degree from the University Paris 12, in 2002. He is now a research engineer at Renault SA and is working on the optimization of various industrial problems, such as the tuning of diesel engines, structural optimization, etc.

PATRICK SIARRY was born in France in 1952. He received the PhD degree from the University Paris 6, in 1986 and the Doctorate of Science (Habilitation) from the University Paris 11, in 1994. He was first involved in the development of models of nuclear power plants at Electricité de France. Since 1995 he is a professor of automatics and informatics. His main research interests are the applications of new stochastic global optimization heuristics to various engineering fields. He is also interested in the fitting of process models to experimental data, learning of fuzzy rule bases and neural networks.