# Comparing two stochastic algorithms on a benchmark set

14th May 2007

Maurice.Clerc@WriteMe.com, 2006-07-04

Last update: 2007-04-01

Note: I have written a small spreadsheet to compute formula 1 for Gaussian distributions. You may download it with this paper from my Maths site

http://clerc.maurice.free.fr/Maths/index.htm#Comparing_algorithms

## 1 Comparing on a given problem

### 1.1 What "better" means

**Let us suppose we have two stochastic optimisation algorithms to compare, A and B. On a given problem and from the user point of view, A is better than B if it gives "better results". But what does it mean? This can be defined only on a probabilistic way. We run A and we obtain a result $x$. We run B and we obtain a result $y$. We say that** *A is better than B if the probability that x is better than y is greater than* 0.5

**For a minimisation problem we have then to estimate $proba\left(x<y\right)$.**

### 1.2 Estimation formula

**For a test problem, in order to perform this estimation we can run A and B $T$ times on it, and collect the results. Note that as A and B are stochastic algorithms it is important to not have the same sequence of pseudo-random numbers for each run. There are mainly two ways to do that:**

1. performing the $T$ runs inside a loop
2. performing each run by restarting the program, but by modifying the seed of the pseudo-random numbers generator

**The first method is the best one if the generator is a good enough so that there is no cycle during the whole process.**

**The first step is to "guess", for each algorithm, what kind of probability distribution represents results at the best, and to estimate the parameters of this distribution. Usually a normal distribution is acceptable, and we then just have to compute the mean and the standard deviation (or more precisely *estimations* of these parameters).**

**Let $D_A$ and $D_B$ be the two distributions. So, for a given $u$, we have**

$$proba\,(u < y) = \int_u^{+\infty} D_B\,(v)\,dv$$

**for running B is equivalent to draw at random $y$ according to distribution $D_B$. However running A is similarly equivalent to draw at random $x$ according to $D_A$. So we have to integrate this probability over all possible $x$, weighted by their probability density. The final formula is then**

$$proba\,(x < y) = \int_{-\infty}^{+\infty} \left( \int_u^{+\infty} D_B\,(v)\,dv \right) D_A\,(u)\,du \tag{1}$$

**Important: Probability density estimations are depending on the number of runs $T$. In particular the standard deviation may decrease to zero when $T$ increases (however, contrarily to one can think, it is not always the case). So, when deciding that A is better than B according to formula 1, we should add "to the best of our knowledge".**

## 1.3 Example

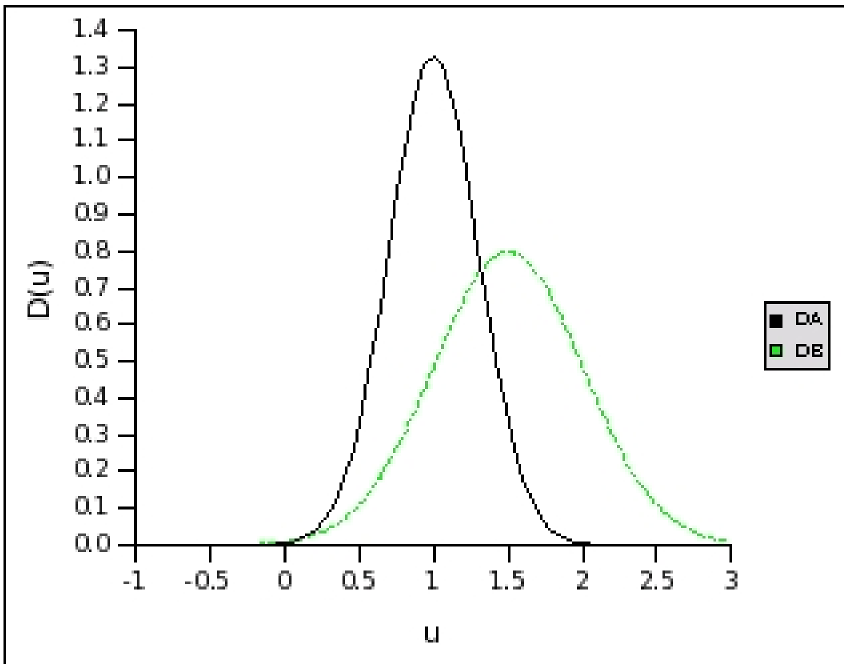

Figure 1: Two Gaussian distributions of results

**Let us note $N(\mu,\sigma)$ the normal distribution with a mean equal to $\mu$and a standard deviation $\sigma$. We run A on the problem several times and find $D_A \simeq N(1, 0.3)$. Similarly we find $D_B \simeq N(1.5, 0.5)$ (see figure 1). Formula 1 gives us 0.8. It means that if we pick up $x$ at random according to $D_A$ and $y$ at random according to $D_B$, it is perfectly possible (probability 0.2) that we find $y < x$. Or, in other words, if we run A and B, there is a probability of 0.2 that we find a better result with B than with A. That is why we have to be very careful when comparing the two algorithms over a whole benchmark set of problems.**

## 1.4 A particular case: the success rate

**When the performance criterion is not a "direct" value (like say the best value after $N$ fitness evaluations) but a calculated one, like the success rate, we need more runs: we need $T$ sequences of $R$ runs. For each sequence the success rate is computed, and for the $T$ success rates, we compute the mean and the standard deviation.**

## 2 Comparing on a benchmark set

### 2.1 What “better” means

**Let us note first that there is an assumption almost never explicitly said: a benchmark set is supposed to be representative of the whole set of problems that the algorithm will have to solve. This assumption is very strong for it does suppose the following:**

- we are able to define equivalence classes and to assign any possible problem to one class. For example mono-modal, bimodal, etc.
- we know the relative sizes of these classes. For example we know that the algorithm will have to solve two times more often bimodal functions than uni-modal ones.

**Unfortunately we almost never have such information. So the best that we can do is to experimentally build a “reasonable” benchmark set and to assign the same weight to each function of this benchmark. Then, saying that A is better than B from the user point of view can be detailed as follows:** *If I take at random (uniform distribution) a function of the benchmark, and if I run A and B, the probability that the result given by A is better than the one given by B is greater than 0.5*

### 2.2 Example

| | A | B | A better than B |
|---|---|---|---|
| $f_1$ | $N(10,2)$ | $N(11,2)$ | 63.8% |
| $f_2$ | $N(0,1)$ | $N(0.5,1)$ | 63.8% |
| $f_3$ | $N(15,1)$ | $N(10,1)$ | 0.0002% |
| $\{f_1,f_2,f_3\}$ | | | 42.5% |

Table 1: Benchmark of three functions. Probability density for A and B, probability for A to be better than B, for each function, and for the whole benchmark

**Let** $F = \{f_1, f_2, f_3\}$**a benchmark of three functions. As we can see on table 1, A is better than B for two functions. So it is tempting to say it is better on the whole for the benchmark. However, according to the above definition it would be wrong. Algorithm A is indeed slightly better two times but B is far better one time, and the final conclusion is that B “wins”.**