A method to improve Standard PSO

Maurice Clerc

A method to improve Standard PSO

Technical report. DRAFT MC2009-03-13

Maurice Clerc

Abstract

In this report, I present my personal method to design a more accurate version of PSO, assuming we know what kind of problems we will have to solve. To illustrate the method, I consider a class of problems that are moderately multimodal and not of very high dimensionality (typically smaller than 30). The starting version is Standard PSO 2007 (SPSO 07). I use a modified velocity update equation is used, specific to the considered class of problems. Then, in order to improve the robustness of the algorithm, more general modifications are added (in particular, better initialisations, and the use of two kind of particles). On the whole, the resulting algorithm is indeed an improvement of SPSO 07, though the improvement is not always very impressive. In passing, I give a precise definition of "better than", and explain why the classical "mean best" performance criterion may easily be completely meaningless.

1   Why this paper?

A question I often face is this: "You are always referring to Standard PSO 2007 (SPSO 07, [1]). Isn't it possible to design a better version?". The answer is of course "yes". It is even written in the source code available on line:

> "This PSO version does not intend to be the best one on the market"

On the one hand, it is indeed not very difficult to write a better PSO. On the other hand, as a reviewer I read a lot of papers in which the authors claim they have succeeded in doing that, though it is not really true., mainly for two reasons:

- they start from a bad version of PSO (usually a "global best" one). Although they may really improve it, the resulting algorithm is still not as good as SPSO 07.

- or they start from a reasonably good version (or even from SPSO 07 itself), but they update it having apparently no clear idea of what the modification is for. A typical example is the usual claim "we want to improve the exploitation/exploration balance" without giving any rigorous definition of what this balance is. As a result, the modified algorithm may or may not be better, more or less in a random way, and we do no event know why.

Also, of course, there is the well known problem of comparison of two algorithms. Often, we find something like "our new algorithm B is therefore better than algorithm A" ... without any clear definition of "better than". Let us first say a few words about this point.

2   My algorithm is better than yours

What does such a claim mean? First, such a claim is valid only if we agree on a common benchmark. Let us say, for example, we consider a subset of the CEC 2005 benchmark [2]. Then, we have to agree on a given criterion. For example for each function, the criterion may be Cr = "success rate over N runs, for a given accuracy ε, after at most F fitness evaluations for each run, or, in case of a tile, mean of the best results". Note that the criterion may be a probabilistic one (for example Cr' = "the probability that A is better than B on this problem

according to the criterion Cr is greater than 0.5"). What is important is to clearly define the meaning of "A is better than B on *one* problem". A lot of researchers indeed use such an approach, and then perform nice statistical analyses (null hypothesis, p-test, and so on), in order to decide "in probability" whether an algorithm A is better than an algorithm B, on the whole benchmark.

However, in the process, they miss a simple and frustrating fact: there is no complete order in R^D, for all $D > 1$. Why is that important? It may be useful to express it more formally. Let us say the benchmark contains P problems. We build two comparison vectors. First $C_{A,B} = (c_{A,B,1}, \ldots, c_{A,B,P})$ with $c_{A,B,i} = 1$ if A is better than B on problem i (according to the unique criterion defined), $c_{A,B,i} = 0$ otherwise. Second $C_{B,A} = (c_{B,A,1}, \ldots, c_{B,A,P})$ with $c_{B,A,i} = 1$ if B is better than A on the problem i, and $c_{B,A,i} = 0$ otherwise. We have to compare the two numerical vectors $C_{A,B}$ and $C_{B,A}$. Now, precisely because there is no complete order in R^D, we can say that A is better than B if and only if for any i we have $c_{A,B,i} \geq c_{B,A,i}$ for all i, and if there exists j so that $c_{A,B,j} > c_{B,A,j}$.

This is similar to the definition of the classical Pareto dominance. As we have P values of one criterion, the process of comparing A and B can be seen as a multicriterion (or multiobjective) problem. It implies that most of the time no comparison is possible, except by using an aggregation method. For example, here, we could count the number of 1s in each vector, and say that the one with the larger sum "wins". But the point is that any aggregation method is arbitrary, i.e. for each method there is another one that leads to a different conclusion[1].

Let us consider an example:

- the benchmark contains 5 unimodal functions f_1 to f_5, and 5 multimodal ones f_6 to f_{10}

- the algorithm A is extremely good on unimodal functions (very easy, say for a gradient method)

- the algorithm B is quite good for multimodal functions, but not for unimodal ones.

You find $c_{A,B,i} = 1$ for $i = 1, 2, 3, 4, 5$, and also for 6 (just because, for example, the attraction basin of the global optimum is very large, compared to the ones of the local optima), and $c_{B,A,i} = 1$ for $i = 7, 8, 9, 10$. You say then "A is better than B". An user trusts you, and chooses A for his problems. And as most of interesting real problems are multimodal, he will be very disappointed.

So, we have to be both more modest and more rigorous. That is why the first step in our method of designing an "improved PSO" is the choice of a small benchmark. But we will say that A is better than B only if it is true for all the problems of this benchmark.

3 Step 1: a small "representative" benchmark

This is the most specific part of the method, for it is depends on the kind or problems we do want to solve later with our "improved PSO". Let us consider the following class of problems:

- moderately multimodal, or even unimodal (but of course we are not supposed to know it in advance)

- not of too high dimensionality (say no more than 30)

For this class, to which a lot of real problems belong, I have found that a good small benchmark may be the following one (see the table 1):

- CEC 2005 Sphere (unimodal)

- CEC 2005 Rosenbrock (one global optimum, at least one local optimum as soon as the dimension is greater than 3)

- Tripod (two local optima, one global optimum, very deceptive [3])

These three functions are supposed to be "representative" of our class of problem. If we have an algorithm that is good on them, then it is very probably also good for a lot of other problems of the same class. Our aim is then to design a PSO variant that is better than SPSO 07 for these three functions. Our hope is that this PSO variant will indeed be also better than SPSO 07 on more problems of the same kind. And if it is true even for some highly multimodal problems, and/or for higher dimensionality, well, we can consider that as a nice bonus!

---

[1] For example, it is possible to assign a "weight" to each problem (which represents how "important" is this kind of problem for the user) and to linearly combine the $c_{A,B,i}$ and $c_{B,A,i}$. But if, for a set of (non identical) weights, A is better than B, then it always exists another one for which B is better than A.

Tab. 1: The benchmark. More details are given in 9.1

	Search space	Required accuracy	Maximum number of fitness evaluations
CEC 2005 Sphere	$[-100, 100]^{30}$	0.000001	15000
CEC 2005 Rosenbrock	$[-100, 100]^{10}$	0.01	50000
Tripod	$[-100, 100]^{2}$	0.0001	10000

4   Step 2: a highly flexible PSO

My main tool is a PSO version (C code), which is based on SPSO 07. However I have added a lot of options, in order to have a very flexible research algorithm. Actually, I often modify it, but you always can find the latest version (named Balanced PSO) on my technical site [4]. When I used it for this paper, the main options were:

- two kind of randomness (KISS [5], and the standard randomness provided in LINUX C compiler). In what follows, I always use KISS, so that the results can be more reproducible

- seven initialisation methods for the positions (in particular a variant of the Hammersley's one [6])

- six initialisation methods for the velocities (zero, completely random, random "around" a position, etc.)

- two clamping options for the position (actually, just "clamping like in SPSO 07", or "no clamping and no evaluation")

- possibility to define a search space greater than the feasible space. Of course, if a particle flies outside the feasible space, its fitness is not evaluated

- six local search options (no local search as in SPSO 07, uniform in the best local area, etc.). Note that it implies a rigorous definition of what a "local area" is. See [7]

- two options for the loop over particles (sequential or at random)

- six strategies

The strategies are related to the classical velocity update formula

$$v(t+1) = wv(t) + R(c_1)(p(t) - x(t)) + R(c_2)(g(t) - x(t)) \tag{1}$$

One can use different coefficients w, and different random distributions R. The most interesting point is that different particles may have different strategies.

In the C source code, each option has an identifier to easily describe the options used. For example, PSO P1 V2 means: SPSO 07, in which the initialisation of the positions is done by method 1, and the initialisation of the velocities by the method 2. Please refer to the on line code for more details. In our case, we will now see now how an interesting PSO variant can be designed by using just three options.

5   Step 3: selecting the right options

First of all, we simulate SPSO 07, by setting the parameters and options to the corresponding ones. The results over 500 runs are given in the table 2. In passing, it is worth noting that the usual practice of launching only 25 or even 100 runs is not enough, for really bad runs may occur quite rarely. This is obvious for Rosenbrock, as we can see from the table 3. Any conclusion that is drawn after just 100 runs is risky, particularly if you consider the mean best value. The success rate is more stable. More details about this particular function are given in 9.5.

Tab. 2: Standard PSO 2007. Results over 500 runs

	Success rate	Mean best
CEC 2005 Sphere	84.8%	10^{-6}
CEC 2005 Rosenbrock	15%	12.36
Tripod	46%	0.65

Tab. 3: For Rosenbrock, the mean best value is highly depending on the number of runs (50000 fitness evaluations for each run). The success rate is more stable

Runs	Success rate	Mean best value
100	16%	10.12
500	15%	12.36
1000	14,7%	15579.3
2000	14%	50885.18

5.1 Applying a specific improvement method

When we consider the surfaces of the attraction basins, the result for Tripod is not satisfying (the success rate should be greater then 50%). What options/parameters could we modify in order to improve the algorithm? Let us call the three attraction basins as B_1, B_2, and B_3 . The problem is deceptive because two of them, say B_2 and B_3, lead to only local optima. If, for a position x in B_1 (i.e. in the basin of the global optimum) the neighbourhood best g is either in B_2 or in B_3, then, according to the equation 1, even if the distance between x and g is high, the position x may be easily modified such that it is not in B_1 any more. This is because in SPSO 07 the term $R(c_2)(g(t) - x(t))$ is simply $U(0, c_2)(g(t) - x(t))$, where U is the uniform distribution.

However, we are interested on functions with a small number of local optima, and therefore we may suppose that the distance between two optima is usually not very small. So, in order to avoid the above behaviour, we use the idea is that the further an informer is, the smaller is its influence(this can be seen as a kind of niching). We may then try a $R(c_2)$ that is in fact a $R(c_2, |g - x|)$, and decreasing with $|g - x|$. The optional formula I use to do that in my "flexible" PSO is

$$R(c_{2,} |g - x|) = U(0, c_2)\left(1 - \frac{|g - x|}{x_{max} - x_{min}}\right)^{\lambda} \tag{2}$$

Experiments suggest that λ should not be too high, because in that case, although the algorithm becomes almost perfect for Tripod, the result for Sphere becomes quite bad. In practice, $\lambda = 2$ seems to be a good compromise. With this value the result for Sphere is also improved as we can see from the table 4. According to our nomenclature, this PSO is called PSO R2. The result for Rosenbrock may be now slightly worse, but we have seen that we do not need to worry too much about the mean best, if the success rate seems correct. Anyway, we may now also apply some "general" improvement options.

Tab. 4: Results with PSO R2 ("distance decreasing" distribution, according to the equation 2

	Success rate	Mean best
CEC 2005 Sphere	98.6%	0.14×10^{-6}
CEC 2005 Rosenbrock	13.4%	10.48
Tripod	47.6%	0.225

5.2 Applying some general improvement options (initialisations)

The above option was specifically chosen in order to improve what seemed to be the worst result, i.e. the one for the Tripod function. Now, we can trigger some other options that are often beneficial, at least for moderately multimodal problems:

- modified Hammersley method for the initialisation of the positions x

- One-rand method for the initialisation of the velocity of the particle whose initial position is x, i.e. $v = U\left(x_{min}, x_{max}\right) - x$. Note that in SPSO 07, the method is the "Half-diff" one, i.e.

$$v = 0.5\left(U\left(x_{min}, x_{max}\right) - U\left(x_{min}, x_{max}\right)\right)$$

This modified algorithm is PSO R2 P2 V1. The results are given in the table 5, and are clearly better than the ones of SPSO 07. They are still not completely satisfying (cf. Rosenbrock), though. So, we can try yet another option, which can be called "bi-strategy".

Tab. 5: Results when applying also different initialisations, for positions and velocities (PSO R2 P2 V1)

	Success rate	Mean best
CEC 2005 Sphere	98.2%	0.15×10^{-6}
CEC 2005 Rosenbrock	18.6%	31132.29
Tripod	63.8%	0.259

5.3 Bi-strategy

The basic idea is very simple: we use two kinds of particles. In practice, during the initialisation phase, we assign one of two possible behaviours, with a probability equal to 0.5. These two behaviours are simply:

- the one of SPSO 07. In particular, $R\left(c_2\right) = U\left(0, c_2\right)$

- or the one of PSO R2 (i.e. by using equation 2)

The resulting algorithm is PSO R3 P2 V1. As we can see from the table 6, for all the three functions now we obtain results that are also clearly better than the ones of SPSO 07. Success rates are slightly worse for Sphere and Rosenbrock, slightly better for Tripod, so no clear comparison is possible. However more tests (not detailed here) show that this variant is more robust, as we can guess by looking at the mean best values, so we keep it. Two questions, though. Is it still valid for different maximum number of fitness evaluations ("search effort"). And is it true for more problems, even if they are not really in the same "class", in particular if they are highly multimodal? Both answers are affirmative, as tackled in next sections.

Tab. 6: Results by adding the bi-strategy option (PSO R3 P2 V1)

	Success rate	Mean best
CEC 2005 Sphere	96.6%	$< 10^{-10}$
CEC 2005 Rosenbrock	18.2%	6.08
Tripod	65.4%	0.286

6 Now, let's try

6.1 Success rate vs "Search effort"

Here, on the same three problems, we simply consider different maximum numbers of fitness evaluations (FE_{max}), and we evaluate the success rate over 500 runs. As we can see from the figure 1, for any FE_{max} the

success rate of our variant is greater than the one of SPSO 07. SO, we can safely say that it is really better, at least on this small benchmark. Of course, it is not always so obvious. Giving a long list of results is out of the scope of this paper, which is just about a design method, but we can nevertheless have an idea of the performance on a few more problems.

6.2 Moderately multimodal problems

Table 7 and figure 2 are about moderately multimodal problems. This is a small selection, to illustrate different cases:

- clear improvement, i.e. no matter what the number of fitness evaluations is, but the improvement is small (Schwefel, Pressure Vessel). Actually SPSO 07 is already pretty good on these problems (for example, for Pressure Vessel, SOMA needs more than 50000 fitness evaluations to solve it [8]), so our small modifications can not improve it a lot.

- questionable improvement, i.e. depending on the number of fitness evaluations (Compression spring)

- clear big improvement (Gear train). For this problem, and after 20000 fitness evaluations, not only the success rate of PSO R3 P2 V1 is 92.6%, but it finds the very good solution $(19, 16, 43, 49)$ (or an equivalent permutation), 85 times over 500 runs. The fitness of this solution is 2.7×10^{-12} (SOMA needs about 200,000 evaluations to find it).

Even when the improvement is not very important, the robustness is increased. For example, for Pressure Vessel, with 11000 fitness evaluations, the mean best is 28.23 (standard dev. 133.35) with SPSO 07, as it is 18.78 (standard dev. 56.97) with PSO R3 P2 V1.

Tab. 7: More moderately multimodal problems. See 9.2 for details

	Search space	Required accuracy
CEC 2005 Schwefel	$[-100, 100]^{10}$	0.00001
Pressure vessel (discrete form)	4 variables objective 7197.72893	0.00001
Compression spring	3 variables objective 2.625421 (granularity 0.001 for x_3)	0.000001
Gear train	4 variables	10^{-9}

## 6.3 Highly multimodal problems

Table 8 and figure 3 are for highly multimodal problems. The good news is that our modified PSO is also better even for some highly multimodal problems. It is not true all the time (see Griewank or Cellular phone), but it was not its aim, anyway.

7 Claims and suspicion

We have seen that it is possible to improve Standard PSO 2007 by modifying the velocity update equation and the initialisation schemes. However, this improvement is not valid across all kinds of problems, and not valid across all criterions (in particular, it may be depending on the number of fitness evaluations). Also, the improvement is not always very impressive. Thus, this study incites us to be suspicious when reading an assertion like "My PSO variant is far better than Standard PSO". Such a claim has to be very carefully supported, by a rigorous definition of what "better" means, and by significant results on a good "representative" benchmark, on a large range of maximum number of fitness evaluations. Also, we have to be very careful when using the mean best criterion for comparison, for it may be meaningless. And, of course, the proposed PSO variant should be compared to the *current* Standard PSO, and not to an old bad version.
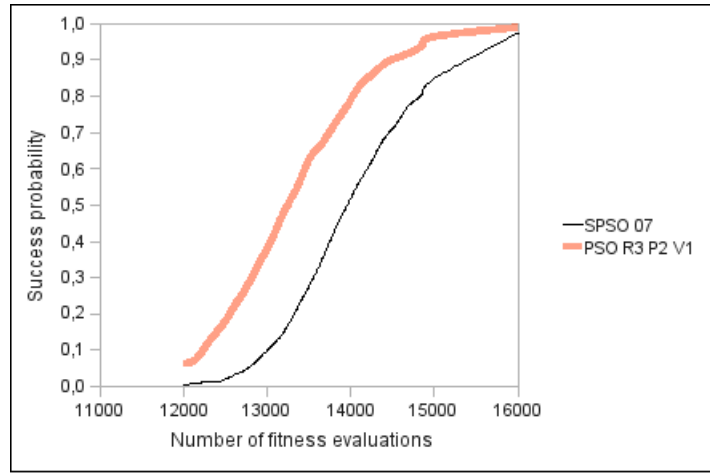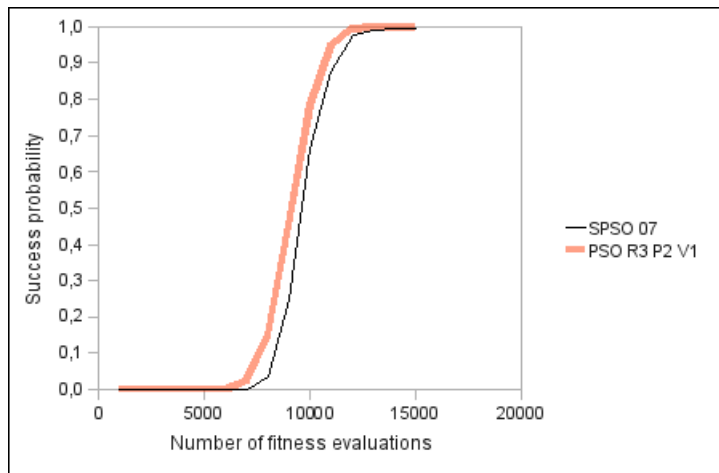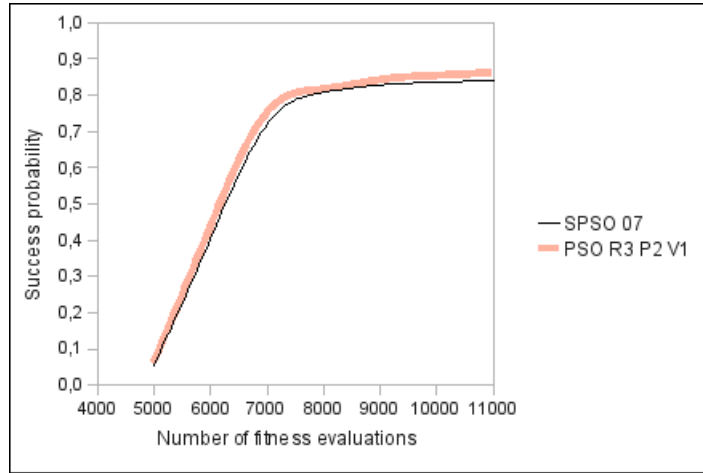
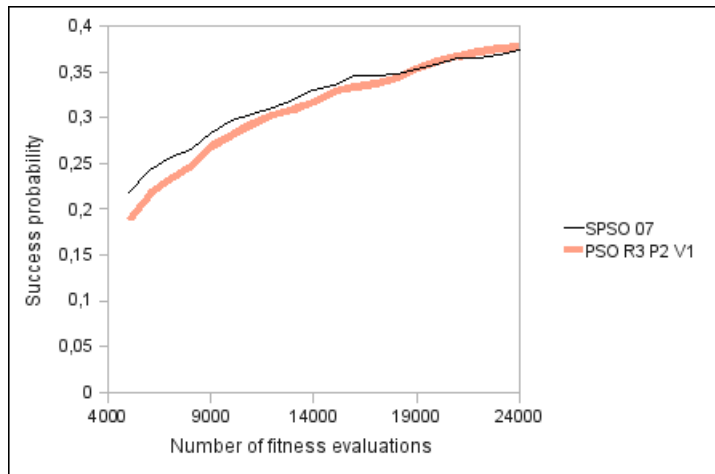(a) Sphere
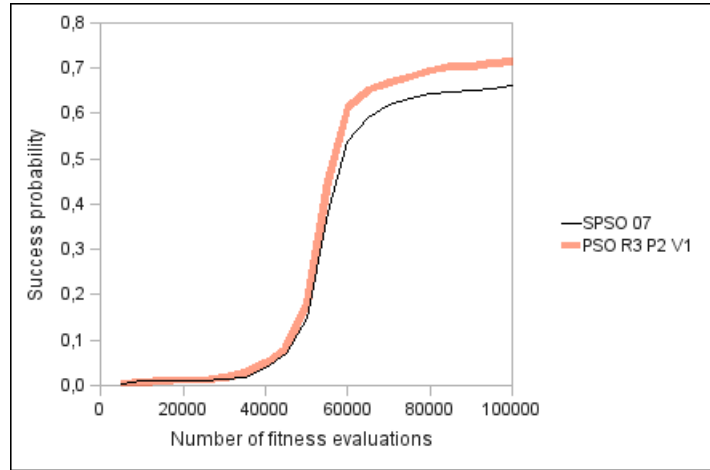
(b) Rosenbrock
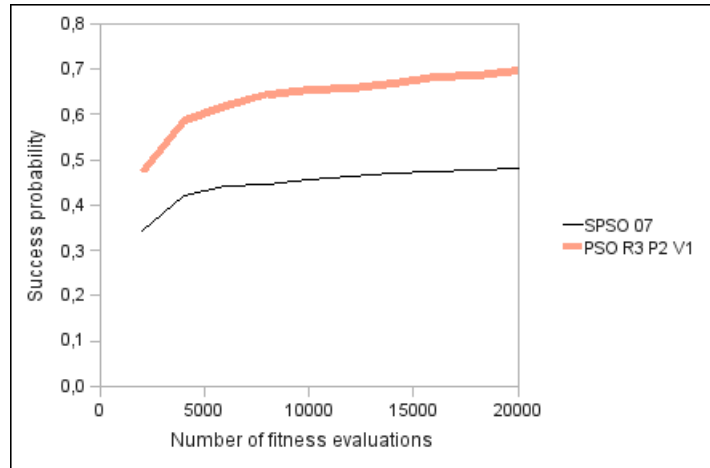
(c) Tripod

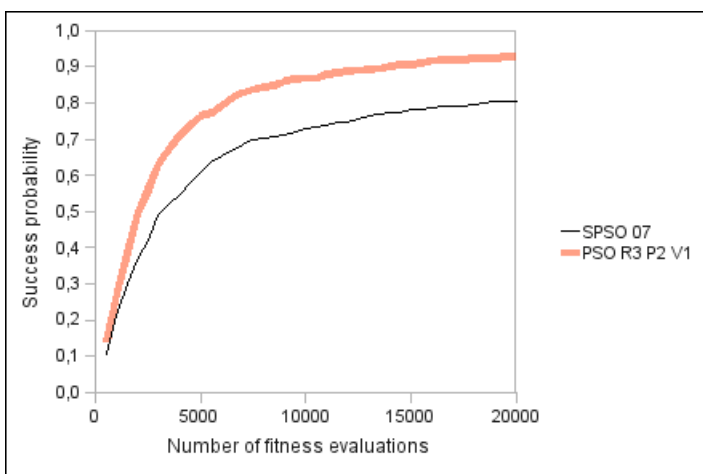Fig. 1: Success probability vs Search effort. For any FE_{max} the variant is better

(a) Schwefel

(b) Pressure vessel

(c) Compression spring

(d) Gear train

Fig. 2: On the Schwefel and Pressure vessel problems PSO R3 P2 V1 is slightly better than SPSO 07 for any number of fitness evaluations. On the Compression spring problem, it is true only when the number of fitness evaluations is greater than a given value (about 19000). So, on this problem, either claim "SPSO 07 is better" or "PSO R3 P2 V1 is better" is wrong
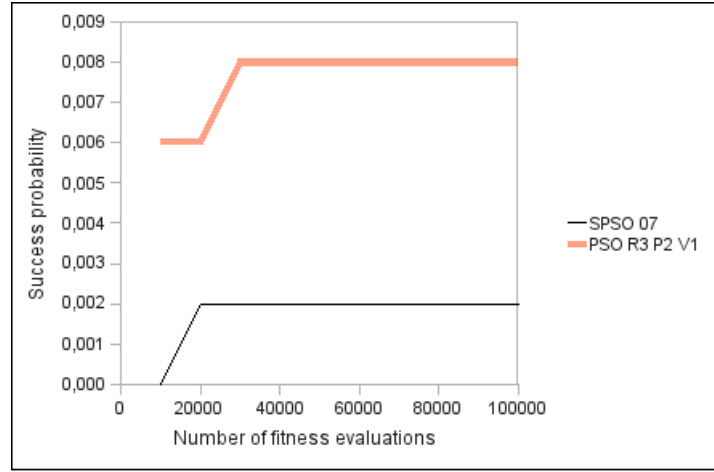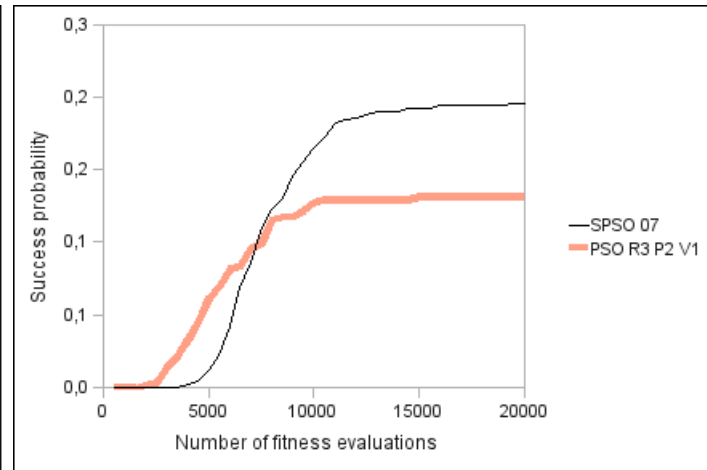
(a) Rastrigin
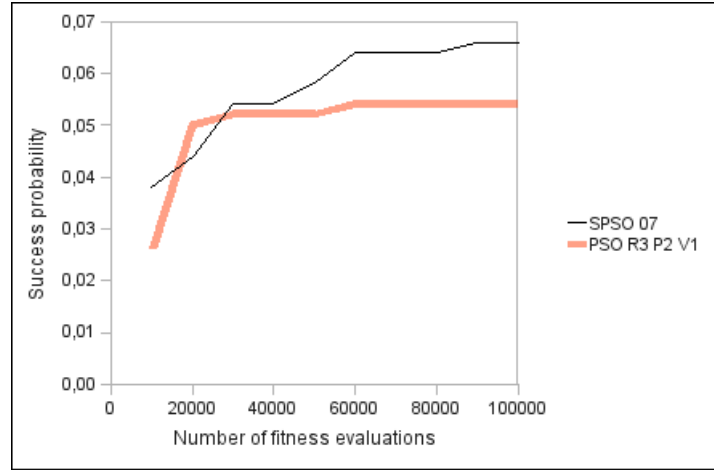
(b) Griewank
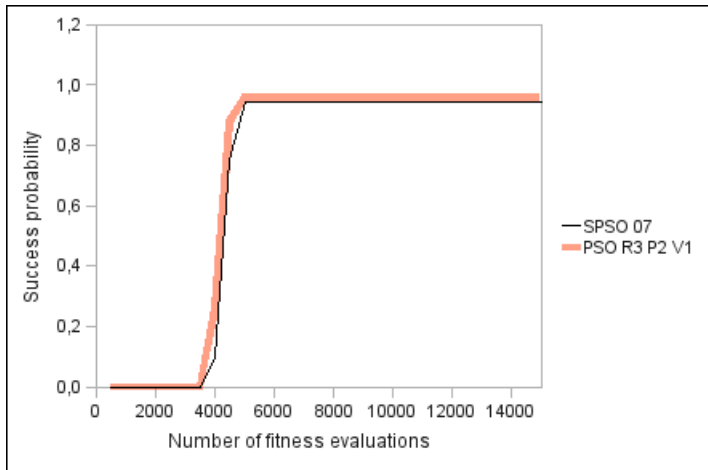
(c) Ackley

(d) Cellular phone

Fig. 3: Success probability for some highly multimodal problems. Although designed for moderately multimodal problems, PSO R3 P2 V1 is even sometimes good for these problems. But not always

Tab. 8: Highly multimodal problems. See 9.3 for details

	Search space	Required accuracy
CEC 2005 Rastrigin	$[-5,5]^{10}$	0.01
CEC 2005 Griewank (not rotated)	$[-600,600]^{10}$	0.01
CEC 2005 Ackley (not rotated)	$[-32,32]^{10}$	0.0001
Cellular phone	$[0,100]^{20}$	10^{-8}

8   Home work

The specific improvement modification of SPSO 07 used here was for moderately multimodal problems, in low dimension. Let us call them M-problems. Now, what could be an effective specific modification for another class of problems? Take, for example the class of highly multimodal problems, but still in low dimension (smaller than 30). Let us call them H-problems.

First, we have to define a small representative benchmark. Hint: include Griewank 10D, from the CEC 2005 benchmark (no need to use the rotated function). Second, we have to understand in which way the difficulty of an H-problem is different from that of an M-problem. Hint: on an H-problem, SPSO 07 is usually less easily trapped into a local minimum, just because the attraction basins are small. On the contrary, if a particle is inside the "good" attraction basin (the one of the global optimum), it may even leave it prematurely. And third, we have to find what options are needed to cope with the found specific difficulty(ies). Hint: just make sure the current attraction basin is well exploited, a quick local search may be useful. A simple way is to define a local area around the best known position, and to sample its middle (PSO L4)[2]. With just this option, an improvement seems possible, as we can see from figure 4 for the Griewank function. However, it does not work very well for Rastrigin.

All this will probably be the topic of a future paper, but for the moment, you can think at it yourself. Good luck!

---

[2] Let $g = (g_1, g_2, \ldots, g_D)$ be the best known position. On each dimension i, let p_i and p'_i are the nearest coordinates of known points, "on the left", and "on the right" of g_i. The local area H is the D-rectangle (hyperparallelepid) cartesian product $\otimes_i \left[g_i - \alpha(g_i - p_i), g_i + \alpha(p'_i - g_i) \right]$ with, in practice, $\alpha = 1/3$. Then its center is sampled. Usually, it is not g.
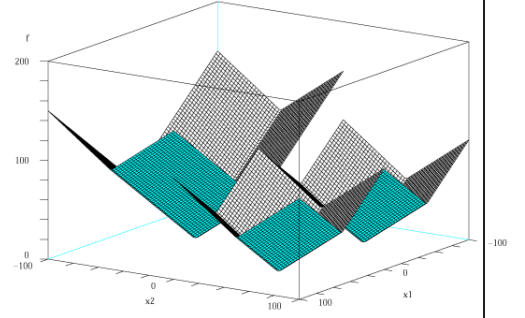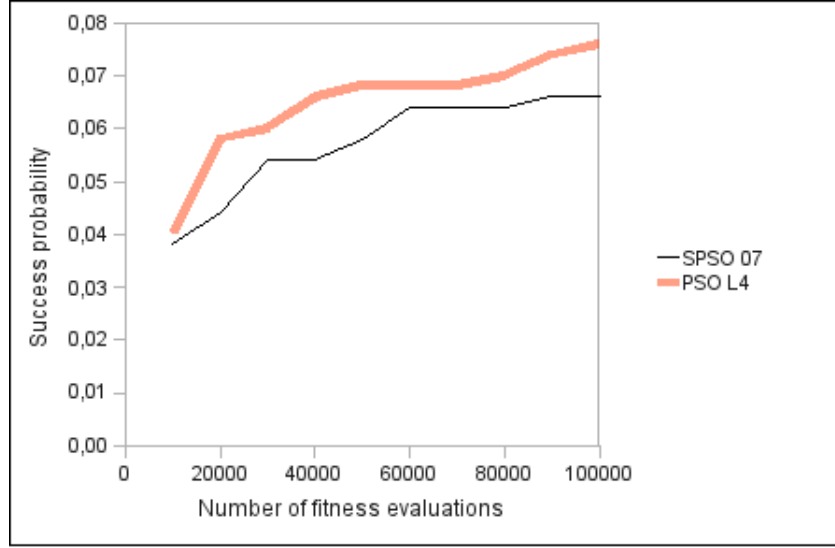
Fig. 4: Griewank, comparison between SPSO 07 and PSO L4. For a highly multimodal problem, a very simple local search may improve the performance.

9   Appendix

9.1   Formulae for the benchmark

Tab. 9: Benchmark details

	Formula	
Sphere	$-450 + \sum\limits_{d=1}^{30} (x_d - o_d)^2$	The random offset vector $O = (o_1, \cdots, o_{30})$ is defined by its C code. This is the solution point.
Rosenbrock	$390 + \sum\limits_{d=2}^{10} \left(100 \left(z_{d-1}^2 - z_d \right)^2 + (z_{d-1} - 1)^2 \right)$ with $z_d = x_d - o_d + 1$	The random offset vector $O = (o_1, \cdots, o_{10})$ is defined by its C code. This is the solution point There is also a local minimum on $(o_1 - 2, \cdots, o_{30})$. The fitness value is then 394.
Tripod	$\frac{1-sign(x_2)}{2} \left(|x_1| + |x_2 + 50| \right)$ $+ \frac{1+sign(x_2)}{2} \frac{1-sign(x_1)}{2} \left(1 + |x_1 + 50| + |x_2 - 50| \right)$ $+ \frac{1+sign(x_1)}{2} \left(2 + |x_1 - 50| + |x_2 - 50| \right)$ with $\begin{cases} sign(x) &= -1 \quad \text{if} \quad x \le 0 \\ &= \quad 1 \quad \text{else} \end{cases}$ 11	The solution point is $(0, -50)$

Offset for Sphere/Parabola (C source code)

static double offset_0[30] = { -3.9311900e+001, 5.8899900e+001, -4.6322400e+001, -7.4651500e+001, -1.6799700e+001, -8.0544100e+001, -1.0593500e+001, 2.4969400e+001, 8.9838400e+001, 9.1119000e+000, -1.0744300e+001, -2.7855800e+001, -1.2580600e+001, 7.5930000e+000, 7.4812700e+001, 6.8495900e+001, -5.3429300e+001, 7.8854400e+001, -6.8595700e+001, 6.3743200e+001, 3.1347000e+001, -3.7501600e+001, 3.3892900e+001, -8.8804500e+001, -7.8771900e+001, -6.6494400e+001, 4.4197200e+001, 1.8383600e+001, 2.6521200e+001, 8.4472300e+001 };

Offset for Rosenbrock (C source code)

static double offset_2[10] = { 8.1023200e+001, -4.8395000e+001, 1.9231600e+001, -2.5231000e+000, 7.0433800e+001, 4.7177400e+001, -7.8358000e+000, -8.6669300e+001, 5.7853200e+001};

9.2 Formulae for the other moderately multimodal problems

9.2.1 Schwefel

The function to minimise is

$$f\left(x\right) = -450 + \sum_{d=1}^{10}\left(\sum_{k=1}^{d} x_k - o_k\right)^2$$

The search space is $[-100, 100]^{10}$. The solution point is the offset $O = (o_1, \ldots, o_{10})$, where $f = -450$.

Offset (C source code)

static double offset_4[30] =
    { 3.5626700e+001, -8.2912300e+001, -1.0642300e+001, -8.3581500e+001, 8.3155200e+001, 4.7048000e+001, -8.9435900e+001, -2.7421900e+001, 7.6144800e+001, -3.9059500e+001};

9.2.2 Pressure vessel

Just in short. For more details, see[9, 10, 11]. There are four variables

$$\begin{aligned}
x_1 &\in [1.125, 12.5] &\text{granularity} &\quad 0.0625 \\
x_2 &\in [0.625, 12.5] &\text{granularity} &\quad 0.0625 \\
x_3 &\in]0, 240] \\
x_4 &\in]0, 240]
\end{aligned}$$

and three constraints

$$\begin{aligned}
g_1 &:= 0.0193x_3 - x_1 \leq 0 \\
g_2 &:= 0;00954x_3 - x_2 \leq 0 \\
g_3 &:= 750 \times 1728 - \pi x_3^2\left(x_4 + \tfrac{4}{3}x_3\right) \leq 0
\end{aligned}$$

The function to minimise is

$$f = 0.06224x_1x_3x_4 + 1.7781x_2x_3^2 + x_1^2\left(3.1611x + 19.84x_3\right)$$

The analytical solution is $(1.125, 0.625, 58.2901554, 43.6926562)$which gives the fitness value 7,197.72893. To take the constraints into account, a penalty method is used.

9.2.3 Compression spring

For more details, see[9, 10, 11]. There are three variables

$$
\begin{aligned}
x_1 &\in \{1, \ldots, 70\} && \text{granularity} && 1 \\
x_2 &\in [0.6, 3] && \\
x_3 &\in [0.207, 0.5] && \text{granularity} && 0.001
\end{aligned}
$$

and five constraints

$$
\begin{aligned}
g_1 &:= \frac{8 C_f F_{max} x_2}{\pi x_3^3} - S \leq 0 \\
g_2 &:= l_f - l_{max} \leq 0 \\
g_3 &:= \sigma_p - \sigma_{pm} \leq 0 \\
g_4 &:= \sigma_p - \frac{F_p}{K} \leq 0 \\
g_5 &:= \sigma_w - \frac{F_{max} - F_p}{K} \leq 0
\end{aligned}
$$

with

$$
\begin{aligned}
C_f &= 1 + 0.75 \frac{x_3}{x_2 - x_3} + 0.615 \frac{x_3}{x_2} \\
F_{max} &= 1000 \\
S &= 189000 \\
l_f &= \frac{F_{max}}{K} + 1.05 (x_1 + 2) x_3 \\
l_{max} &= 14 \\
\sigma_p &= \frac{F_p}{K} \\
\sigma_{pm} &= 6 \\
F_p &= 300 \\
K &= 11.5 \times 10^6 \frac{x_3^4}{8 x_1 x_2^3} \\
\sigma_w &= 1.25
\end{aligned}
$$

and the function to minimise is

$$
f = \pi^2 \frac{x_2 x_3^2 (x_1 + 1)}{4}
$$

The best known solution is $(7, 1.386599591, 0.292)$which gives the fitness value 2.6254214578. To take the constraints into account, a penalty method is used.

9.2.4 Gear train

For more details, see[9, 11]. The function to minimise is

$$
f(x) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4} \right)^2
$$

The search space is $\{12, 13, \ldots, 60\}^4$. There are several solutions, depending on the required precision. For example $f(19, 16, 43, 49) = 2.7 \times 10^{-12}$

9.3 Formulae for the highly multimodal problems

9.3.1 Rastrigin

The function to minimise is

$$
f = -230 + \sum_{d=1}^{10} \left((x_d - o_d)^2 - 10 \cos (2\pi (x_d - o_d)) \right)
$$

The search space is $[-5, 5]^{10}$. The solution point is the offset $O = (o_1, \ldots, o_{10})$, where $f = -330$.

Offset (C source code)

static double offset_3[30] =
{ 1.9005000e+000, -1.5644000e+000, -9.7880000e-001, -2.2536000e+000, 2.4990000e+000, -3.2853000e+000, 9.7590000e-001, -3.6661000e+000, 9.8500000e-002, -3.2465000e+000};

9.3.2 Griewank

The function to minimise is

$$f = -179 + \frac{\sum_{d=1}^{10} (x_d - o_d)^2}{4000} - \prod_{d=1}^{10} \cos\left(\frac{x_d - o_d}{\sqrt{d}}\right)$$

The search space is $[-600, 600]^{10}$ .The solution point is the offset $O = (o_1, \ldots, o_{10})$, where $f = -180$.

Offset (C source code)

static double offset_5[30] =
{ -2.7626840e+002, -1.1911000e+001, -5.7878840e+002, -2.8764860e+002, -8.4385800e+001, -2.2867530e+002, -4.5815160e+002, -2.0221450e+002, -1.0586420e+002, -9.6489800e+001};

9.3.3 Ackley

The function to minimise is

$$f = -120 + e + 20e^{-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{10}(x_d - o_d)^2}} - e^{\frac{1}{D}\sum_{d=1}^{10}\cos(2\pi(x_d - o_d))}$$

The search space is $[-32, 32]^{10}$ .The solution point is the offset $O = (o_1, \ldots, o_{10})$, where $f = -140$.

Offset (C source code)

static double offset_6[30] =
{ -1.6823000e+001, 1.4976900e+001, 6.1690000e+000, 9.5566000e+000, 1.9541700e+001, -1.7190000e+001, -1.8824800e+001, 8.5110000e-001, -1.5116200e+001, 1.0793400e+001};

9.3.4 Cellular phone

This problem arises in a real application, on which I have worked in the telecommunications domain. However, here, all constraints has been removed, except of course the ones given by the search space itself. We have a square flat domain $[0, 100]^2$, in which we want to put M stations. Each station m_k has two coordinates $(m_{k,1}, m_{k,2})$. These are the $2M$ variables of the problem. We consider each "integer" point of the domain, i.e. $(i, j), i \in \{0, 1, \ldots, 100\}, j \in \{0, 1, \ldots, 100\}$. On each integer point, the field induced by the station m_k is given by

$$f_{i,j,m_k} = \frac{1}{(i - m_{k,1})^2 + (j - m_{k,2})^2 + 1}$$

and we want to have at least one field that is not too weak. Finally, the function to minimise is

$$f = \frac{1}{\sum_{i=1}^{100} \sum_{j=1}^{100} \max_k (f_{i,j,m_k})}$$

In this paper, we set $M = 10$. Therefore the dimension of the problem is 20. The objective value is 0.005530517. This is not the true minimum, but enough from an engineering point of view. Of course, in reality we do not know the objective value. We just run the algorithm several times for a given number of fitness evaluations, and keep the best solution. From the figure 5 we can see a solution found by SPSO 07 after 20000 fitness evaluations. Actually, for this simplified problem, more efficient methods do exist (Delaunay's tessellation, for example), but those can not be used as soon as we introduce a third dimension and more constraints, so that the field is not spherical any more.
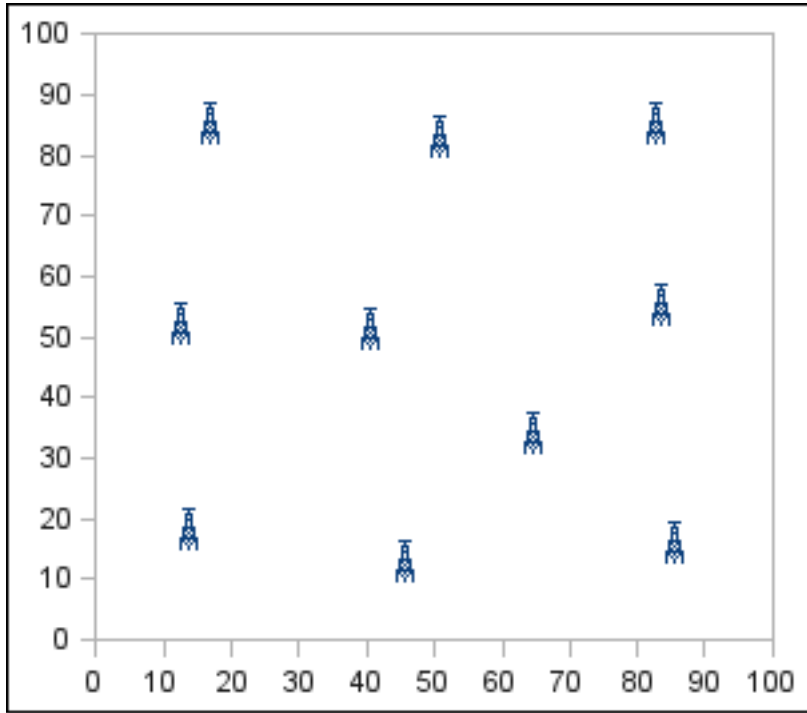
Fig. 5: Cellular phone problem. A possible (approximate) solution for 10 stations, found by SPSO 07 after 20000 fitness evaluations

9.4 A possible simplification

We may wonder whether the two initialisation methods used in 5.2are really useful or not. Let us try just the bi-strategy option, by keeping the initialisations of SPSO 07. Results are in the table 10. When we compare the results with those given in the table 6, we can see that for the three functions, the results are not as good. However, they are not bad at all. So, for simplicity, it may be perfectly acceptable to use just PSO R3.

Tab. 10: Results with just the bi-strategy option (PSO R3)

	Success rate	Mean best
CEC 2005 Sphere	%	
CEC 2005 Rosenbrock	%	
Tripod	60.6%	0.3556

9.5 When the mean best may be meaningless

On the Rosenbrock function, we have quickly seen that the mean best depends heavily on the number of runs (see table 3), and therefore is not an acceptable performance criterion. Here is a more detailed explanation of this phenomenon. First we show experimentally that the distribution of the errors for this function is not Gaussian, and, more precisely, that the probability of a very bad run (i.e. a very high fitness value) is not negligible. Then, and more generally, assuming that for a given problem this property is true, a simple probabilistic analysis explains why the success rate is a more reliable criterion.

9.5.1 Distribution of the errors for Rosenbrock

We run the algorithm 5000 times, with 5000 fitness evaluations for each run, i.e. just enough to have a non zero success rate. Each time, we save the best value found. We can then estimate the shape of the distribution

of these 5000 values, seen as occurrences of a random variable. Contrary to what is sometimes said, this distribution is far from normal (Gaussian) one. Indeed, the main peak is very acute, and there are some very high values. Even if these are rare, it implies that the mean value is not really representative of the performance of the algorithm. It would be better to consider the value on which the highest peak (the mode) lies. For SPSO 07, it is about 7 (the right value is 0), and the mean is 25101.4 (there are a few very bad runs). As we can see from figure 6, we have a quite nice model by using the union of a power law (on the left of the main peak), and a Cauchy law (on the right).

$$
\begin{aligned}
frequency \quad &= \quad \alpha \frac{m^k}{class^{k+1}} \qquad \text{if} \quad class \le m \\
&= \quad \frac{1}{\pi} \frac{\gamma}{(class-m)^2 + \gamma^2} \quad \text{else}
\end{aligned}
$$

with $\gamma = 1.294$, $m = 7$, and $k = 6.5$. Note that a second power law for the right part of the curve (instead of the Cauchy one) would not be suitable: although it could be better for class values smaller than say 15, it would "forget" the important fact that the probability of high values is far from zero. Actually, even the Cauchy model is overly optimistic, as we can see from the magnified version (classes 40-70) of the figure 6, but at least the probability is not virtually equal to zero, as with the power law model.

For PSO R3 P2 V1, the mode is about 6, i. e. just slightly better. However, the mean is only 3962.1. It shows that this version is a bit more robust (very bad runs do not exist). For both algorithms, the small peak (around 10, as the right value is 4) corresponds to a local optimum . The small "valley" (around 3) is also due to the local optimum: sometimes (but very rarely) the swarm is quickly trapped into it. It shows that as soon as there are local optima the distribution has necessarily some peaks, at least for a small number of fitness evaluations.

9.5.2   Mean best vs success rate as criterion

A run is said to be *successful* if the final value is smaller than a "small" ε , and *bad* if the final value is greater than a "big" M. For one run, let p_M be the probability of that run being bad. Then, the probability, over N runs, that at least one of the runs is bad is

$$
p_{M,N} = 1 - (1 - p_M)^N
$$

This probability increases quickly with the number of runs. Now, let f_i be the final value of the run i. The estimate of the mean best value is usually given by

$$
\mu_N = \frac{\sum_{i=1}^{N} f_i}{N}
$$

Let us say the success rate is ς . It means we have ςN successful runs. Let us consider another sequence of N runs, exactly the same, except that k runs are "replaced" by bad ones. Let m be the maximum of the corresponding f_i in the first sequence of N runs. The probability of this event is

$$
p_{M,N,1} = p_M^k (1 - p_M)^{N-k}
$$

For the new success rate ς' , we have

$$
\varsigma \ge \varsigma' \ge \varsigma - \frac{k}{N}
$$

For the new estimate μ_N' of the mean best, we have

$$
\mu_N' > \mu_N + k \frac{M-m}{N}
$$

We immediately see that there is a problem when a big value M is possible with a non negligible probability: when the number of runs N increases the success rate may slightly decrease, but then the mean dramatically increases. Let us suppose that, for a given problem and a given algorithm, the distribution of the errors follows a Cauchy law. Then we have
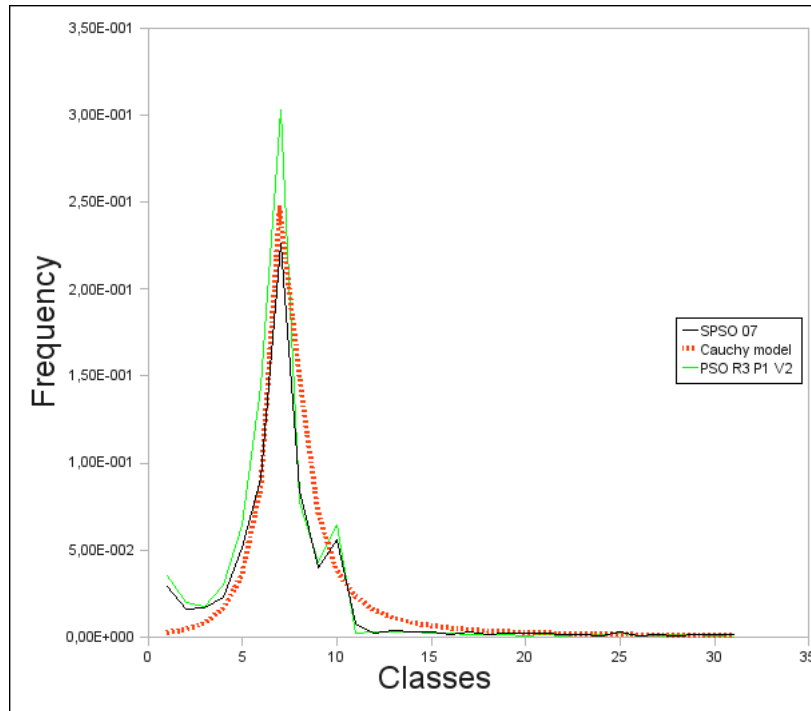
(a) Global shape
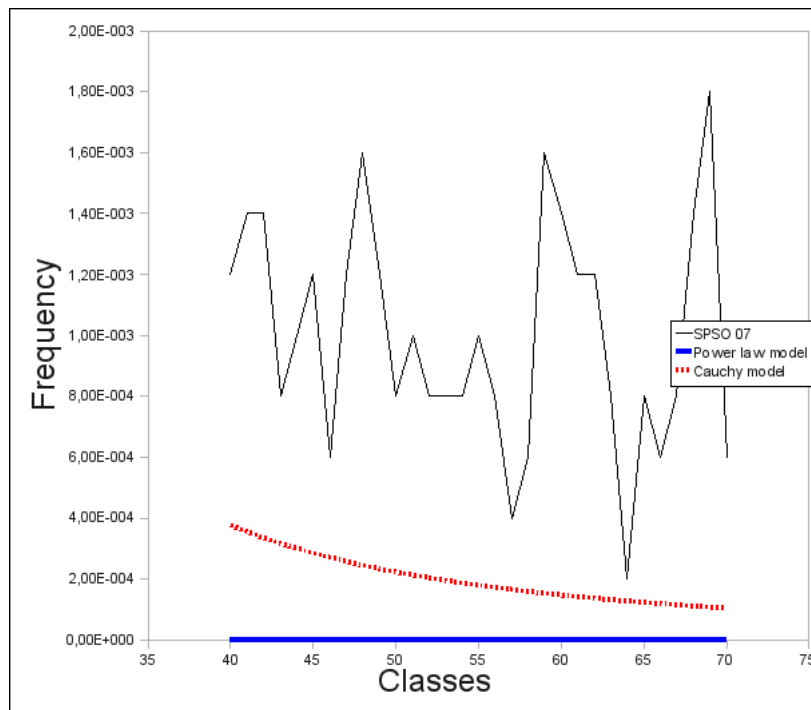
(b) Zoom" on classes 40 to 70

Fig. 6: Rosenbrock. Distribution of the best value over 5000 runs. On the "zoom", we can see that the Cauchy model, although optimistic, gives a better idea of the distribution than the power law model for class values greater than 40

$$p_M = 0.5 - \frac{1}{\pi} \arctan\left(\frac{M}{\gamma}\right)$$

With the parameters of the model of the figure 6, we have for example $p_{5000} = 8.3 \times 10^{-5}$. Over $N = 30$ runs, the probability to have at least one bad run (fitness value greater than $M = 5000$) is low, just 2.5×10^{-3}. Let us say we find an estimate of the mean to be m. Over $N = 1000$ runs, the probability is 0.08, which is quite high. It may easily happen. In such a case, even if for all the other runs the best value is about m, the new estimate is about $(4999m + 5000)/1000$, which may be very different from m. In passing, and if we look at the table 3, this simplified explanation shows that for Rosenbrock a Cauchy law based model is indeed optimistic.

In other words, if the number of runs is too small, you may never have a bad one, and therefore, wrongly estimate the mean best, even when it exists. Note that in certain cases the mean may not even exist at all (for example, in case of a Cauchy law), and therefore any estimate of a mean best is wrong. That is why it is important to estimate the mean for different N values (but of course with the same number of fitness evaluations). If it seems not stable, forget this criterion, and just consider the success rate, or, as seen above, the mode. As there are a lot of papers in which the probable existence of the mean is not checked, it is worth insisting on it: if there is no mean, giving an "estimate" of it is not technically correct. Worse, comparing two algorithms based on such an "estimate" is simply wrong.

References

[1] PSC, "Particle Swarm Central, http://www.particleswarm.info."

[2] CEC, "Congress on Evolutionary Computation Benchmarks,http://www3.ntu.edu.sg/home/epnsugan/," 2005.

[3] L. Gacôgne, "Steady state evolutionary algorithm with an operator family," in *EISCI*, (Kosice, Slovaquie), pp. 373–379, 2002.

[4] M. Clerc, "Math Stuff about PSO, http://clerc.maurice.free.fr/pso/."

[5] G. Marsaglia and A. Zaman, "The kiss generator," tech. rep., Dept. of Statistics, U. of Florida, 1993.

[6] T.-T. Wong, W.-S. Luk, and P.-A. Heng, "Sampling with Hammersley and Halton points," *Journal of Graphics Tools*, vol. 2 (2), pp. 9–24, 1997.

[7] M. Clerc, "The mythical balance, or when PSO doe not exploit," Tech. Rep. MC2008-10-31, 2008.

[8] I. Zelinka, "SOMA - Self-Organizing Migrating Algorithm," in *New Optimization Techniques in Engineering*, pp. 168–217, Heidelberg, Germany: Springer, 2004.

[9] E. Sandgren, "Non linear integer and discrete programming in mechanical design optimization," 1990. ISSN 0305-2154.

[10] M. Clerc, *Particle Swarm Optimization*. ISTE (International Scientific and Technical Encyclopedia), 2006.

[11] G. C. Onwubolu and B. V. Babu, *New Optimization Techniques in Engineering*. Berlin, Germany: Springer, 2004.