



ITERATIVE OPTIMISATION: THE QUESTIONABLE BALANCE MANTRA

Maurice Clerc

► To cite this version:

Maurice Clerc. ITERATIVE OPTIMISATION: THE QUESTIONABLE BALANCE MANTRA. 2018. hal-01930529

HAL Id: hal-01930529

<https://hal.science/hal-01930529>

Preprint submitted on 22 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ITERATIVE OPTIMISATION : THE QUESTIONABLE BALANCE MANTRA

MAURICE CLERC

ABSTRACT. In iterative optimisation a classical belief is that for efficiency a fine balance between local intensive exploitation and global exploration should be achieved. However to date (November 2018), there is no rigorous approaches (theoretical or experimental) that support it. We present here how exploitation and exploration can be measured in order to really monitor the progress of their ratio. Preliminary results show that it may be far from what one would expect.

1. INTRODUCTION

The claim we are studying can be found in different forms in the literature. Here is a possible one:

Claim 1. To be efficient an iterative optimiser has to ensure a good balance between exploration and exploitation

Unfortunately, most authors do not give any precise definitions of exploitation and exploration (see for example [5]). In most paper one just can find vague ones like

- exploitation = sampling a point around a known good position
- exploration = sampling a point far away from all the known ones

With such definitions it is of course not possible to experimentally check whether the claim is pertinent or not.

So, a first step is to rigorously define the two concepts so that the number of exploitation points and the number of exploration points can be counted. Also we need to know what “efficiency” means.

2. DEFINING EFFICIENCY

This is not the purpose of this paper, so I just explain what criterion is used here, and why. A detailed discussion can be found in [2]. As an user I have a budget, given by a maximum number of evaluations for one run. What I want to know is for each possible result value the probability to find it or a smaller one, formally speaking the cumulative distribution of probability (CDF). However, when comparing two optimisers A1 and A2, as soon as the CDF curves intersect one or several times the interpretation may become difficult. Even comparing the best values found is quite arbitrary if their probabilities are different. For example, in the figure 2.1 the optimisers are, strictly speaking, in-comparable. The choice is depending on the user expectation. Does he prefer a very good result with a very small probability or a less good result with a higher probability?

In this paper this difficulty is avoided as follows:

- Let $f_{min,A1}$ (*budget*) and $f_{min,A2}$ (*budget*) are the best values found by the algorithms A1 and A2 after 100 runs. The common budget is chosen so that the CDFs do not intersect.
- So if $f_{min,A1} < f_{min,A2}$ A1 can indeed be said better than A2.

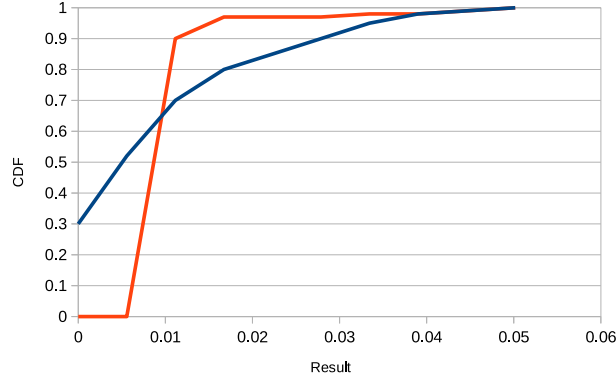


FIGURE 2.1. Comparing two stochastic optimisers after a given number of evaluations. The probability is supposed to be estimated over a sufficient number of runs to be a good approximation. Which one is “the best” is depending on what the user wants, because the CDFs intersect. If only values smaller than 0.01 are acceptable A1 is the best. If, for example, values smaller than 0.02 are accepted A2 is the best choice.

3. EXPLORATION AND EXPLOITATION

To define these concepts there are two prerequisites:

- because we shall use distances, the search space has to be normalised as a D -cube, for example $[0, 1]^D$;
- to prevent making exploitation while believing it is doing exploration, the list of all sampled positions has to be kept in memory.

Then a first possible definition of exploitation around the position i is the following (more details can be found in [3]):

Exploitation, method 1.

- Find in the list, the position j such that $r = \text{distance}(i, j)$ be minimal.
- Consider the D -sphere $S(i, \alpha r)$ of centre i and radius αr with $\alpha < 1$. This is the exploitation domain.
- Sample in S .
- If the point is outside the search space, bring back to the boundary (0 or 1) the erroneous coordinates.

Sampling can be done according to any probability distribution law. One that seems to give quite good results is a "bell" distribution, centred in i and of support S .

The advantage of this method is that the average radius of exploitation domains decreases as the search progresses because it is easy to see that it is of the order of $N^{1/D}$ where N is the number of sampled points.

The disadvantage is that when D increases, the volume of S decreases, relatively to that of the circumscribed D -cube, the ratio of both quickly tending towards zero. Hence the idea of a definition directly from D -cubes:

Exploitation, method 2.

- Start from the D -sphere $S(i, \alpha r)$ of the previous method.
- Consider its circumscribed D -cube C_i .
- Sample in C_i . Here again, sampling can be done following several methods.
- If the point is outside the search space, bring to the boundary (0 or 1) the erroneous coordinates.

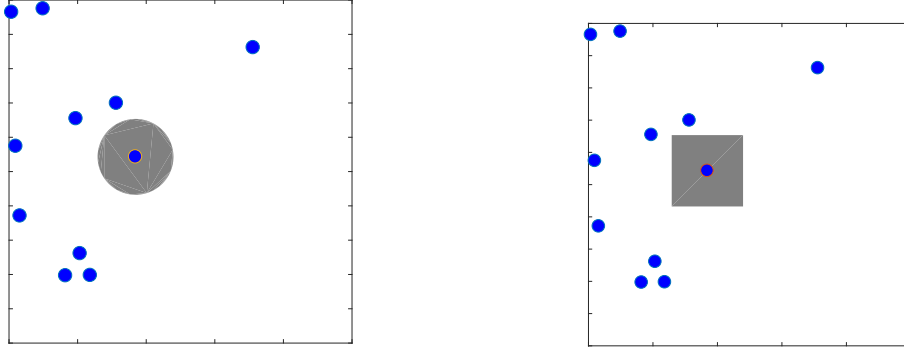


FIGURE 3.1. Two definitions of exploitation, by D -sphere and by D -cube.

At every instant, there are N possible exploitation domains. Which one should be used? Here are some possible strategies:

- (1) the one around the best point;
- (2) the one around one of the k best points, chosen randomly in a uniform manner;
- (3) the one around one of the k best points, chosen randomly according to a decreasing probability with the value of the point.

Actually all these strategies are particular cases of this one:

- Sort the known positions by increasing order of their values (in case of minimisation).
- Select the rank of a position in this list according to a probability distribution. For example for the strategy 1 the distribution is the Dirac on value 1, for the strategy 2 it is a step function, etc.

Intuition may suggest the strategy 3, but experimentally it seems that strategy 2 is more robust in a sufficiently diversified test case.

Once the operation is well defined and because the list of all sampled points has been set aside, the definition of exploration is immediate: it is merely the logical negation of exploitation. That is to say, considering the union of all the domains of exploitation (D -spheres for method 1 or D -cubes for method 2) centred on the points in the list, then sampling outside this set.

A possible implementation is more precisely as follows:

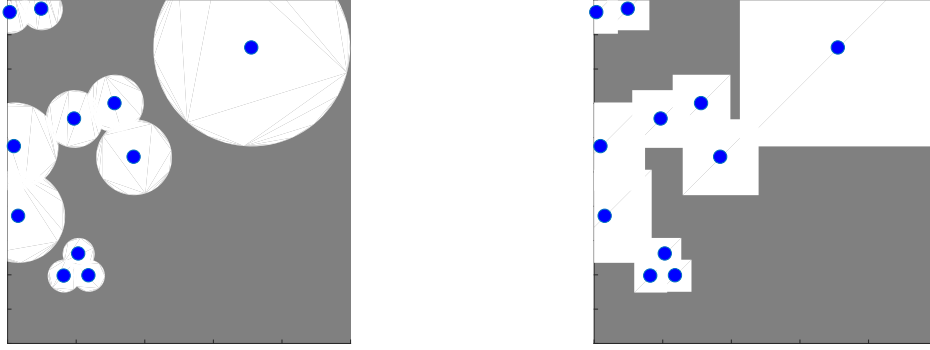


FIGURE 3.2. Two definitions of exploration, by the logical negation of operation.

Random exploration, methods 1 and 2, using loops of attempts.

- Choose a point randomly in $[0, 1]^D$
- Consider the closest known point and test if, by chance, the chosen point belongs to its domain of exploitation, either the D -sphere (method 1) or the D -cube (method 2).
- if it does, repeat.

This process is not quite the logical negation of exploitation. As a matter of fact, we should consider not only the closest point, but every point. The computational time then quickly becomes prohibitive for a non-significant gain of efficiency as soon as its dimension is greater than 1.

In dimension 1, in addition, the probability of finding by chance one point that is not in any domain of exploitation decreases very quickly with this number of points. As a result, the number of attempts increases so much that the process becomes particularly slow. This second reason justifies, in practice, the need for a specific method, for example this one:

Deterministic exploration in dimension 1.

- sort by ascending order the list of known positions;
- if necessary, add 0 before the first element and 1 after the last;
- in this new list, take the middle of the larger interval.

An interesting question is the following: is it still possible to find an exploration point? Or, expressed differently, is it possible that the set of exploitation domains completely covers the search space. The answer is positive, particularly if these domains are D -squares and D is small. The risk is however low and, primarily, temporary. Indeed, adding a new point (obviously of exploitation) destroys some D -squares and replaces them by others which, for their part, are no longer a covering (see Figure 3.3).

4. BALANCE PROFILE AND EFFICIENCY

We are supposed to use an iterative optimiser in which exploration points and exploitation points are perfectly defined. For this study I have written a simple one, Explo2, and more precisely the results have been found by running the version 2 (see 7.1 in the Appendix).

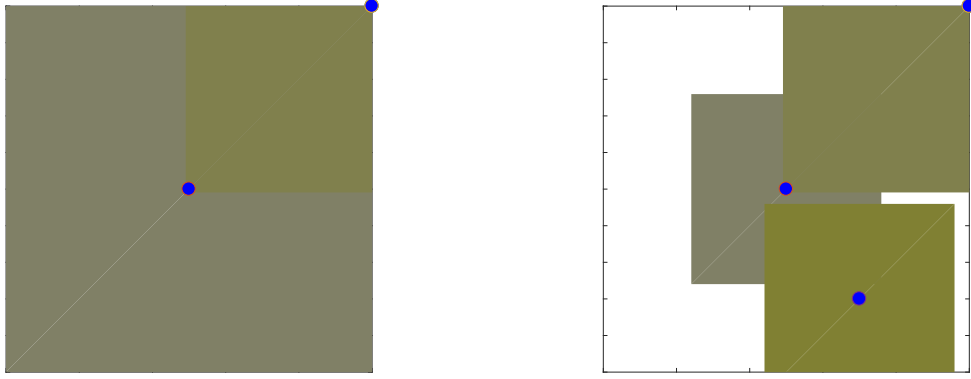


FIGURE 3.3. At the visible point on the left-hand side figure, no exploration is possible, the D -squares cover the search space. Nonetheless, adding an exploitation point recreates "free space".

At each time step t we count the number of exploitation points, $n_{exploit}(t)$, and the number of exploration points, $n_{explor}(t)$, which in fact simply $t - n_{exploit}(t)$. Of course both are either increasing or constant. The balance at time t is defined by

$$(4.1) \quad \underline{\Omega}(t) = \frac{n_{exploit}(t)}{n_{explor}(t)}$$

and $\underline{\Omega}$ is called the *balance profile*. It means that the "good balance" invoked in many papers should be a profile more or less equal to 1, or at least more less constant (not clear in the literature).

In Explo2 V2 we can either use an user-predefined profile or let it free to decide at each time step if the next sample point should be exploitation or exploration. This is done through a probabilistic stagnation detection. Let us see what happens in both cases on some simple test problems. Details are in the Appendix 7.2.

4.1. Adaptive balance profile. We begin with profiles automatically generated step by step by the algorithm, and with different β values, where β is the parameter for stagnation detection (see 7.1). The exploitation domains are D -cubes.

For a given problem the profiles of several runs may be quite different, as on the figure 4.1. However, when averaged on 100 runs, they have a clear structure.

As we can see on the table 1 none of them ensures "a good balance" if it means "equal to 1", and from the point of view of the algorithm, if I dare say, it is better to have less exploitation than exploration. For the best β values (around 0.6, 0.7) the balance tends to a constant smaller than 1 for Alpine, and to 1 for CEC 2011.

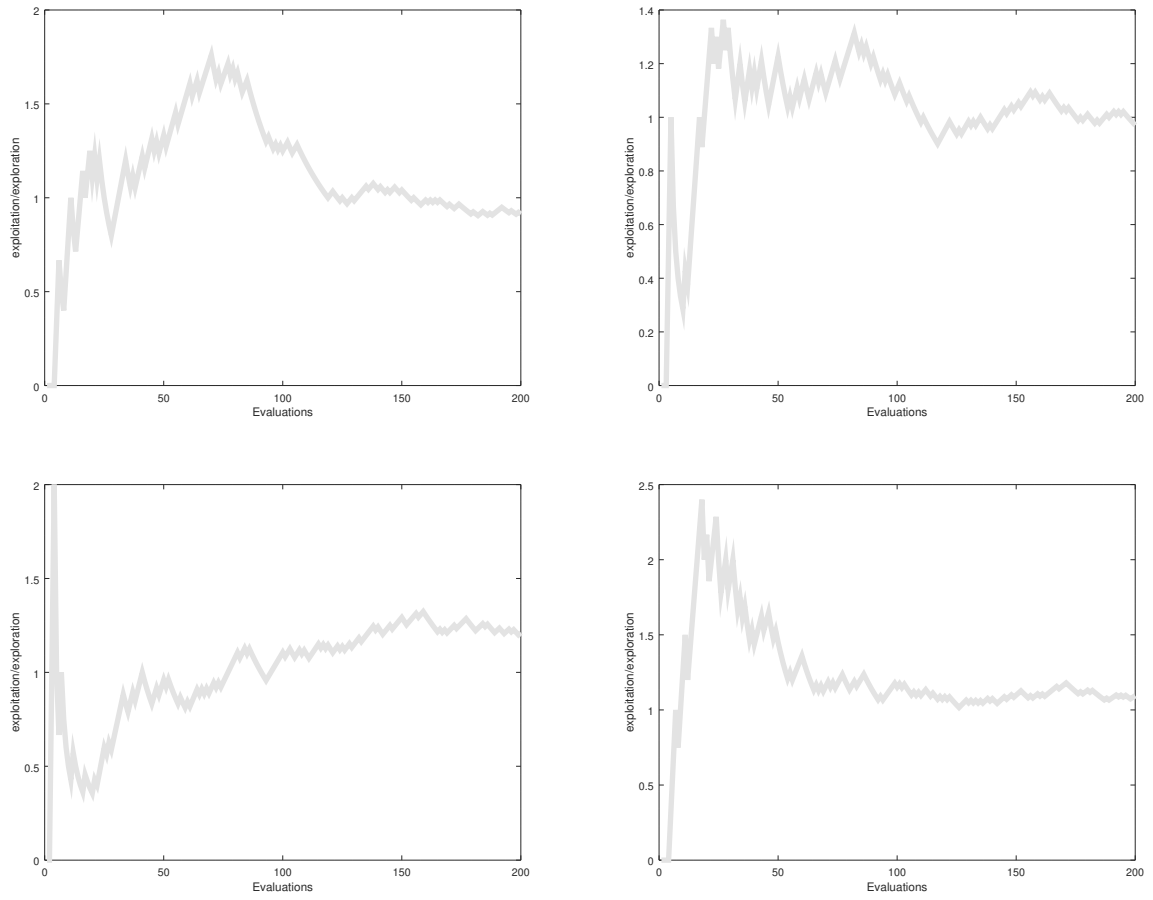
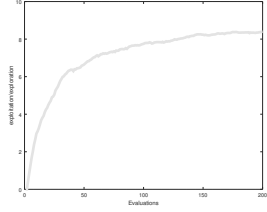
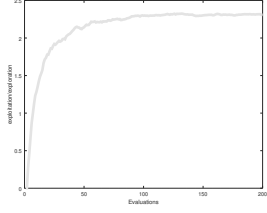
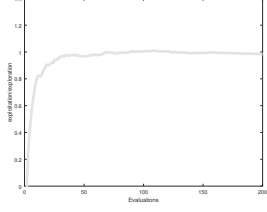
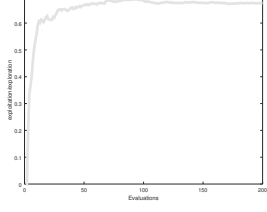
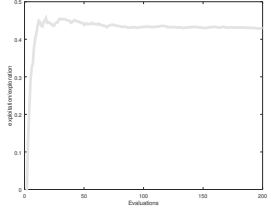
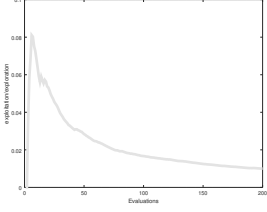
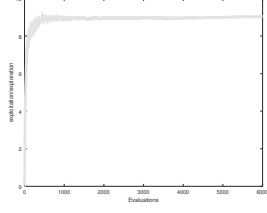
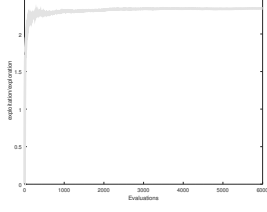
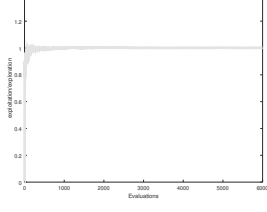
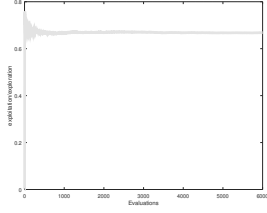
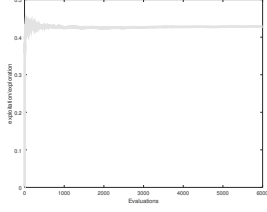
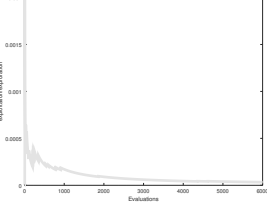


FIGURE 4.1. Alpine 2D. Profiles of three runs of 200 evaluations (FEmax), for $\beta = 0.5$

Table 1: Explo2 V2. Adaptive profiles. Results over 100 runs. The β parameter is used to tune the stagnation detection (see 7.1). For the best results the profile tends to a constant significantly smaller than 1 for Alpine, and equal to 1 for CEC 2011 F1. As expected, when there is almost no exploitation (with $\beta = 1$) the results are pretty bad.

Problem	β	Balance profile	Median	Minimum
Alpine Dimension=2 FEmax=200	0.1	 limit $\simeq 2.3$	3.18×10^{-1}	1.97×10^{-4}
Alpine 2D	0.3	 limit $\simeq 2.3$	7.73×10^{-2}	2.76×10^{-3}
Alpine 2D	0.5	 limit $\simeq 1$	4.52×10^{-2}	8.59×10^{-4}
Alpine 2D	0.6 (best)	 limit $\simeq 0.68$	4.16×10^{-2}	3.10×10^{-6}

Problem	β	Balance profile	Median	Minimum
Alpine 2D	0.7	 <p>limit $\simeq 0.43$</p>	4.15×10^{-2}	3.19×10^{-4}
Alpine 2D	1.0	 <p>limit $\simeq 0.01$</p>	6.56×10^{-2}	2.99×10^{-3}
CEC 2011 F1 Dimension=6 FEmax=6000	0.1	 <p>plateau $\simeq 9$</p>	27.29	18.00
CEC 2011 F1	0.3 (best)	 <p>plateau $\simeq 1$</p>	25.13	12.30
CEC 2011 F1	0.5	 <p>plateau $\simeq 1$</p>	25.62	12.85

Problem	β	Balance profile	Median	Minimum
CEC 2011 F1	0.6	 plateau $\simeq 0.65$	24.73	17.69
CEC 2011 F1	0.7	 plateau $\simeq 0.42$	24.84	16.81
CEC 2011 F1	1.0	 limit $\simeq 0.0$	26.20	20.15

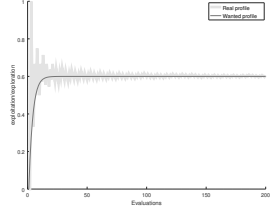
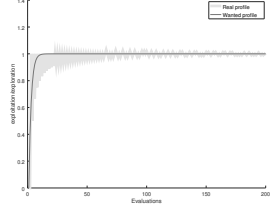
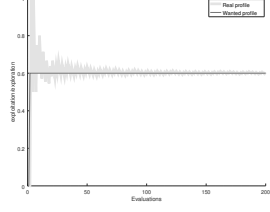
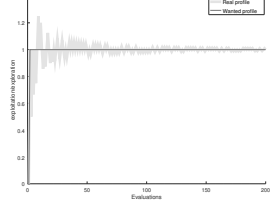
4.2. Predefined balance profiles. Now, what happens if we force the balance profile? At each time step the algorithm computes the current rate $\frac{n_{exploit}(t)}{n_{explor}(t)}$ and compare it to the user-predefined $\underline{\Omega}(t)$. Then it decides if the next sample point must be exploitation or exploration, in order to maintain the current rate as near as possible to the wanted one. For each problem I tried four profiles:

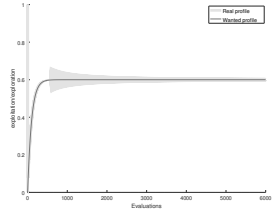
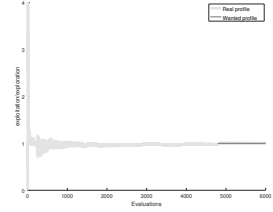
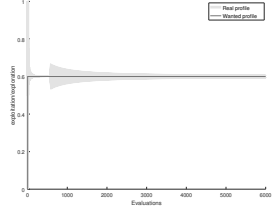
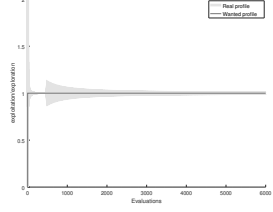
- (1) increasing to a constant smaller than 1 (0.6 here, according to the results with adaptive profiles);
- (2) increasing to 1;
- (3) constant, smaller than 1 (again, here, 0.6);
- (4) constant equal to 1.

Let us rapidly comment the table 2. First, as expected, the algorithm indeed follows the predefined profile. Second it is clear that the “perfect balance” (profile 4) does not give good results. Third, the best results are with a balance profile increasing to a value smaller than 1. However this constant is problem depending, so it is interesting to see such profiles are precisely the ones generated by the adaptive method, at least if the β value is about 0.6, 0.7.

This is a good new for in the context of black box optimisation we can in fact not know in advance the best profile.

Table 2: Explo2 V2. Predefined profiles. The best is a profile constantly smaller than 1. However the value of this constant is problem depending.

Problem	Profiles	Median	Minimum	Probability of the minimum
Alpine Dimension=2 FEmax=200	<p>1</p>  <p>limit = 0.6</p>	4.00×10^{-2}	4.29×10^{-4}	1%
Alpine 2D	<p>2</p>  <p>limit = 1</p>	6.09×10^{-2}	2.27×10^{-4}	1%
Alpine 2D	<p>3</p>  <p>constant = 0.6</p>	5.85×10^{-2}	1.11×10^{-4}	1%
Alpine 2D	<p>4</p>  <p>constant = 1</p>	6.05×10^{-2}	8.13×10^{-4}	1%

Problem	Profiles	Median	Minimum	Probability of the minimum
CEC 2011 F1 Dimension=6 FEmax=6000	<p>1</p>  <p>limit = 0.6</p>	24.92	15.13	1%
CEC 2011 F1	<p>2</p>  <p>limit = 1</p>	25.11	16.35	1%
CEC 2011 F1	<p>3</p>  <p>constant = 0.6</p>	25.10	14.15	1%
CEC 2011 F1	<p>4</p>  <p>constant = 1</p>	25.19	17.03	2%

5. EXPLOITCHECK

A small module can be added to any iterative optimiser, in order to count the number of exploitation points $n_{exploit}(t)$ after t samplings and evaluations. Note that some optimisers may sample a point without evaluating it (for example by computing the “gravity centre” of a set of points, or simply because the new point is outside the search space). ExploitCheck ignore them but could be easily modified to take them into account.

Algorithm 1 ExploitCheck. Check if a new point is in an exploitation domain.

The search space of dimension D is supposed to be a hypercube.

Inputs:

$x(t)$ list of the t already evaluated points, supposed sorted by increasing values of the x_{new} , new point to check

$n_{exploit}(t)$, number of exploitation points after t evaluations

κ , rate of points considered as “good”. If $\kappa = 0$ only the best points is taken into account.

Output:

$n_{exploit}(t+1)$

$K = \max(1, \lfloor \kappa t \rfloor)$;

In $x(1:K)$ find the point $x(s)$ that is the nearest to x_{new} .

In x find the point $x(s')$, $s \neq s'$ that is the nearest to $x(s)$

If $\|x_{new} - x(s)\| \leq \|x(s') - x(s)\|$ then $n_{exploit}(t+1) = n_{exploit}(t) + 1$

We may add the balance as output:

$$\underline{\Omega}(t+1) = \frac{n_{exploit}(t+1)}{t+1-n_{exploit}(t+1)}$$

Let us add this tool to APS12 (Adaptive Population-based Simplex, [6, 1]), here in version 12, and plot the averaged profile over 100 runs. APS can adaptively use four strategies:

- (1) expansion;
- (2) contraction;
- (3) external local search (fmincon if the program is launched under Matlab, or `nonlin_min` under Octave);
- (4) like 3, but adding diversity (based on a bell-like probability distribution).

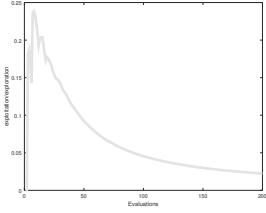
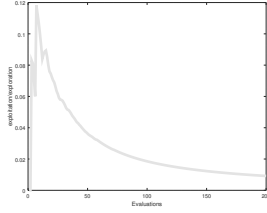
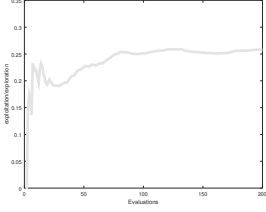
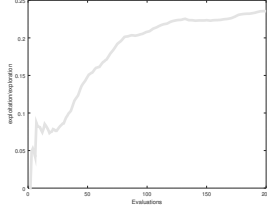
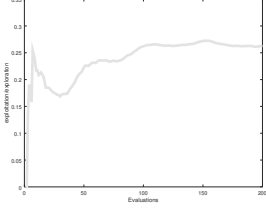
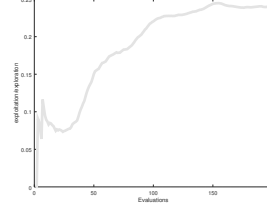
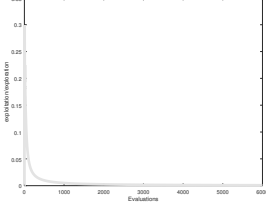
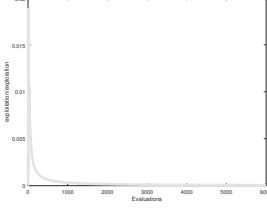
Note that the stagnation detection mechanism of Explo2 V2 has in fact been borrowed from the one of APS12. So, for easier comparison, we use the β values that were the best for Explo2 V2.

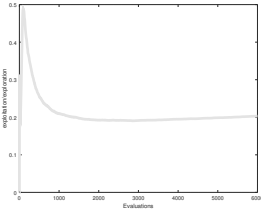
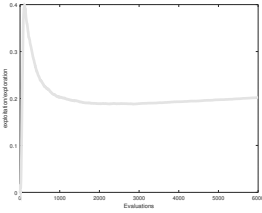
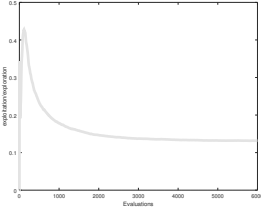
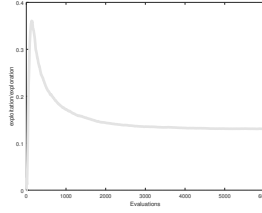
In the table 3 we can see the profiles in three cases: using only strategies 1 and 2, using 1, 2, 3, and using 1, 2, 4. As expected, for an easy problem like Alpine 2D when local search is triggered the profile is globally increasing. For a difficult problem like CEC 2011 F1, it rapidly tends to a small constant

It is worth pointing out that although APS is a pretty good optimiser “good balances” have never a value near to 1.

One may also note that using a local search does not always improve the efficiency because it increases the risk to be trapped into a local minimum. That is why the results may be better by using the strategy 4 instead of the strategy 3.

Table 3: Real profiles with APS12 (population 6). To compare to Explo2 V2 a point is counted as “exploitation” only if it is sampled in the exploitation domain of the current best position. The exploitation domains are either D -spheres or D -cubes. The profiles are of course different, but the global shapes are similar.

Problem	Profiles (D -spheres)	Profiles (D -cubes)	Median	Minimum
Alpine 2D FEmax 200 $\beta = 0.6$	 strategies 1, 2 limit $\simeq 0.02$	 strategies 1, 2 limit $\simeq 0.01$	4.28×10^{-2}	9.17×10^{-5}
(best)	 strategies 1, 2, 3 plateau $\simeq 0.26$	 strategies 1, 2, 3 maximum $\simeq 0.24$	6.68×10^{-2}	4.73×10^{-5}
	 strategies 1, 2, 4 plateau $\simeq 0.26$	 ? strategies 1, 2, 4 plateau $\simeq 0.28$	3.72×10^{-2}	8.96×10^{-5}
CEC 2011 F1 FEmax 6000 $\beta = 0.3$	 strategies 1, 2 limit $\simeq 0.0$	 strategies 1, 2 limit $\simeq 0.0$	24.92	16.14

Problem	Profiles (<i>D</i> -spheres)	Profiles (<i>D</i> -cubes)	Median	Minimum
(best)	<div></div> <div>strategies 1, 2, 3 plateau $\simeq 0.2$</div>	<div></div> <div>strategies 1, 2, 3 plateau $\simeq 0.2$</div>	25.25	8.06×10^{-5}
	<div></div> <div>strategies 1, 2, 4 plateau $\simeq 0.13$</div>	<div></div> <div>? strategies 1, 2, 4 plateau $\simeq 0.13$</div>	24.82	8.423

6. DISCUSSION

Let us consider another classic form of the claim 1:

Claim 2. This algorithm is efficient because it is able to maintain a good balance between exploration and exploitation.

This short study suggests a few things:

- If “good balance” means “as many exploitations as explorations” then the claim is more a erroneous mantra than anything else, and not supported by experiments as soon as we rigorously define this balance and measure it.
- If “good balance” means “an exploitation/exploration ratio that is more or less constant”, then the claim may be correct, but only if this ratio is adaptively found by the algorithm, for the value of the best constant is problem dependent. Note that it is a bit counter intuitive. One would think that the best strategy is first mainly explorations, then less and less, and mainly exploitations, but the above results suggest that it is in fact not always the case (see the Appendix 7.3.2 for details).

To confirm (or not!) these conclusions for some other algorithms and on more problems is of course necessary. It would be easy for algorithms that explicitly “know” if a sampled point is of exploration or exploitation: we just have to count them. Unfortunately it is not directly possible for many algorithms. Let us consider one example.

Some algorithms, from time to time, sample a new point at random to supposedly “improve the diversity”. Intuitively one could think it is then an exploration point. However the algorithm almost never checks whether this new point is really far from know ones. If not it means it should be in fact counted as exploitation. Worse it may even be an exploitation point around a bad position, which is often a waste of time.

Fortunately it is possible to add to any algorithm a tool that automatically checks whether a new point is in an exploitation domain (as defined above) or not, so that the balance can be computed, as it has been done here for APS. But remember that in order to do that we have to keep all the sampled positions.

So, from now on, each time you read or write a “balance claim”, you should ask yourself “How is measured this balance?”.

7. APPENDIX

7.1. Explo2 V2. This rudimentary optimiser explicitly uses a balance profile, either adaptive or user-defined, to sample points in the search space.

7.2. Test problems. For all problems 100 runs are executed.

7.2.1. Alpine. For dimension D , the search space is $[0, 4D]^D$. The function f is defined as:

$$(7.1) \quad f(x_1, \dots, x_D) = \sum_{d=1}^D |x_{d,\delta} \sin(x_{d,\delta})| + 0.1x_{d,\delta}$$

with $x_{d,\delta} = x_d - \delta d$. here, we have simply chosen $\delta = 1$. This parameter serves to ensure that the minimum is not at the centre of the search space or on a diagonal. The problem is multimodal and non-separable.

In this paper $D = 2$ (see the figure 7.1).

Algorithm 2 Explo2, version 2, main algorithm.

```

adaptive_profile=<true or false>
beta=<any value in ]0,1]>

                                stagn=true;

if ¬adaptive_profile
define the desired profile  $\underline{\Omega}_0(2:t_{max})$ 
First point  $x(1)$  on the centre of the search space
 $n_{exploit}(1) = 0$ ; % It is of exploration ...
 $\underline{\Omega}(1) = 0$ ; % ... and therefore the balance is zero.
Evaluate  $F(1) = f(x(1))$ 
 $F_{min} = F(1)$ 
 $t = 2$ ;
while  $t < t_{max}$ 
if adaptive_profile
exploit=¬stagn;
else
exploit=  $\underline{\Omega}(t) < \underline{\Omega}_0(t)$ 
if exploit % Around the best known point either by  $D$ -cubes or  $D$ -spheres
exploitation  $\Rightarrow x(t)$ 
 $n_{exploit}(t) = n_{exploit}(t-1) + 1$ ;
else
exploration  $\Rightarrow x(t)$ 
 $n_{exploit}(t) = n_{exploit}(t-1)$ ;
end
% Compute the balance
 $\underline{\Omega}(t) = \frac{n_{exploit}(t)}{t - n_{exploit}(t)}$ ;
% Save the best value
 $F_{min,prev} = F_{min}$ ;
 $F_{min} = \min(F_{min}, F(t))$ ;
if adaptive_profile
stagn=stagnation( $F_{min,prev}, F_{min}, 0, 1, \beta$ );
 $t = t + 1$ ;

```

Algorithm 3 Stagnation detection (Octave/Matlab[®]code)

The stagnation detection could be done only from time to time. You would just have to save not only $F_{min,prev}$ and F_{min} but also after what numbers of evaluations they have been evaluated, i.e. FEprev and FEs.

```

function stagn=stagnation(fMinPrev,fMin, FEprev, FEs, beta)
% Normalised pseudo-gradient of the fitness evolution
delta=2*(fMinPrev-fMin)/abs(fMinPrev+fMin+eps); % + eps just to avoid 0
delta=delta/(FEs-FEprev);
stagProba=beta*exp(-delta);
% Probabilistic decision
stagn= rand<stagProba;
end

```

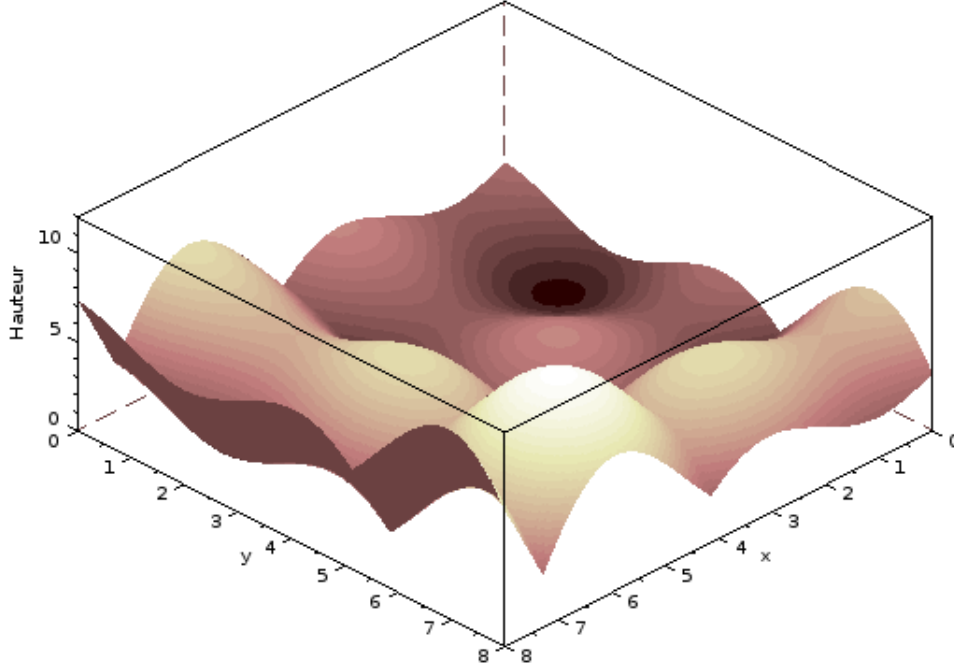


FIGURE 7.1. Alpine 2D

7.2.2. *CEC 2011 F1*. This problem (Parameter Estimation for Frequency-Modulated (FM) Sound Waves) is the first one of the CEC 2011 competition. Its detailed description is in [4]. We have to minimise

$$f(t) = \sum_{t=1}^{100} (y(t) - y_0(t))^2$$

where

$$\begin{cases} y(t) &= x_1 \sin(x_2 t \theta + x_3 \sin(x_4 t \theta + x_5 \sin(x_5 t \theta))) \\ y_0(t) &= \sin(5t\theta - 1.5 \sin(4.8t\theta + 2 \sin(4.9t\theta))) \end{cases}$$

for the position $x = (x_1, x_2, x_3, x_4, x_5, x_6)$, and with $\theta = \frac{2\pi}{100}$. The search space is $[-6.4, 6.35]^6$.

7.3. **About profiles.** Let t be the number of sampled points. Let $n_{exploit}(t)$ be the number of them that are exploitation points and $n_{explor}(t)$ the number of them that are exploration points. Note that $n_{explor}(1) = 1$ because the first sampling is necessary exploration.

So we have

$$(7.2) \quad \begin{cases} n_{exploit}(t) + n_{explor}(t) &= t \\ \underline{\Omega}(t) &= \frac{n_{exploit}(t)}{n_{explor}(t)} \end{cases}$$

It means the general form of the balance profile is

$$(7.3) \quad \underline{\Omega}(t) = \frac{a(t)}{t - a(t)}$$

where a is an increasing function of t , with $0 \leq a(t) < t$. So not all shapes of profiles are possible. Let us consider a few of them.

7.3.1. *Constant profile.* We have

$$(7.4) \quad \frac{a(t)}{t - a(t)} = \omega < 1, \forall t$$

hence

$$(7.5) \quad \begin{cases} n_{exploit}(t) = a(t) &= t \frac{\omega}{1+\omega} \\ n_{explor}(t) = t - a(t) &= t \frac{1}{1+\omega} \end{cases}$$

So, for such a profile the number of exploration points increases quicker than the number of exploitation points. On the one hand a constant profile seems experimentally quite good, but, on the other hand, this strategy is a counter intuitive. One would expect more and more exploitation and less and less exploration during the process.

7.3.2. *"Intuitive" profile.* Let us define a profile that is conform to the intuition: at the beginning more exploration than exploitation, and then the contrary. For example we may want that it increases like t^μ to a maximum value h (i.e. after t_{max} sampled points) with $\mu > 1$. It implies that we should have

$$(7.6) \quad a(t) = \lambda \frac{t^{\mu+1}}{1 + \lambda t^\mu}$$

where $\lambda = \frac{h}{t_{max}^\mu}$.

Note that no algorithm can follow such a predefined profile after a while, more precisely as soon as the slope of the curve is greater than 1 (see the figure 7.3). This is because even if after a given t all points are for exploitation the curve can not increase more quicker than linearly. This t value is given by

$$(7.7) \quad t = \left(\frac{t_{max}}{h\mu} \right)^{\frac{1}{\mu-1}}$$

The point is that the results with such profiles are not significantly better than the ones with constant profiles, as we can see on the table 4.

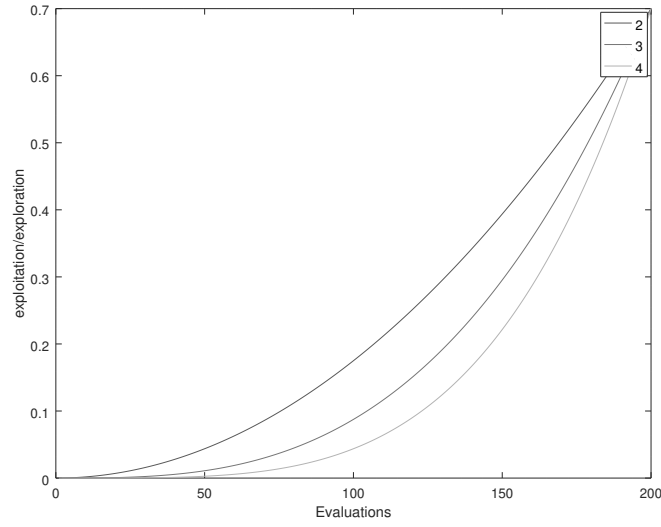


FIGURE 7.2. Some “intuitive” profiles for $h = 0.7$ and different values of μ . More explorations at the beginning and then more and more exploitations.

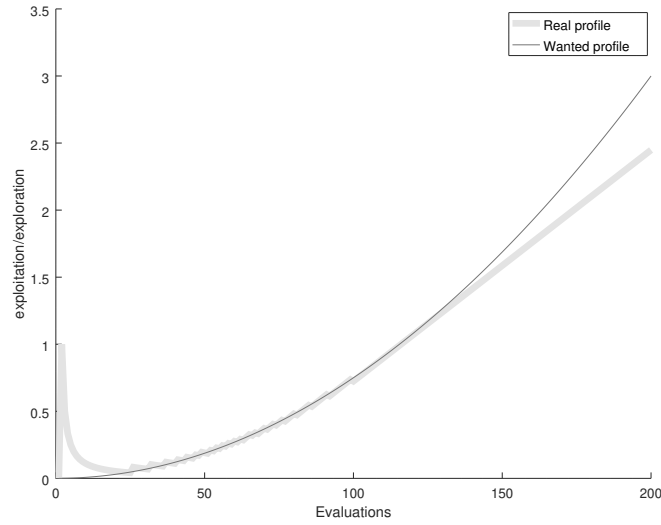


FIGURE 7.3. No algorithm can follow a profile that increases quicker than linearly. For $h = 0.7$ and $\mu = 2$ the real profile becomes linear after $t = 143$ because there is only exploitation.

TABLE 4. Results with “intuitive” profiles ($h = 0.7$).

	μ	Median	Minimum
Alpine 2D	2	1.02×10^{-1}	4.32×10^{-4}
	3	1.02×10^{-1}	2.86×10^{-4}
	4	8.47×10^{-2}	6.76×10^{-4}
CEC 2011 F1	2	25.76	17.20
	3	25.35	17.99
	4	25.00	10.22

REFERENCES

- [1] APS. Adaptive Population-based Simplex (<http://aps-optim.info/>).
- [2] Maurice Clerc. *Guided Randomness in Optimization*. ISTE (International Scientific and Technical Encyclopedia), Wiley, 2015.
- [3] Maurice Clerc. *Difficulty Measures and Benchmarks for Iterative Optimisers*. ISTE (International Scientific and Technical Encyclopedia), Wiley, 2019.
- [4] Swagatam Das and P. N. Suganthan. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Technical report, 2011.
- [5] O. Olorunda and A. P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1128–1134, June 2008.
- [6] Mahamed G. H. Omran and Maurice Clerc. APS 9: an improved adaptive population-based simplex method for real-world engineering optimization problems. *Applied Intelligence*, pages 1–13, August 2017.