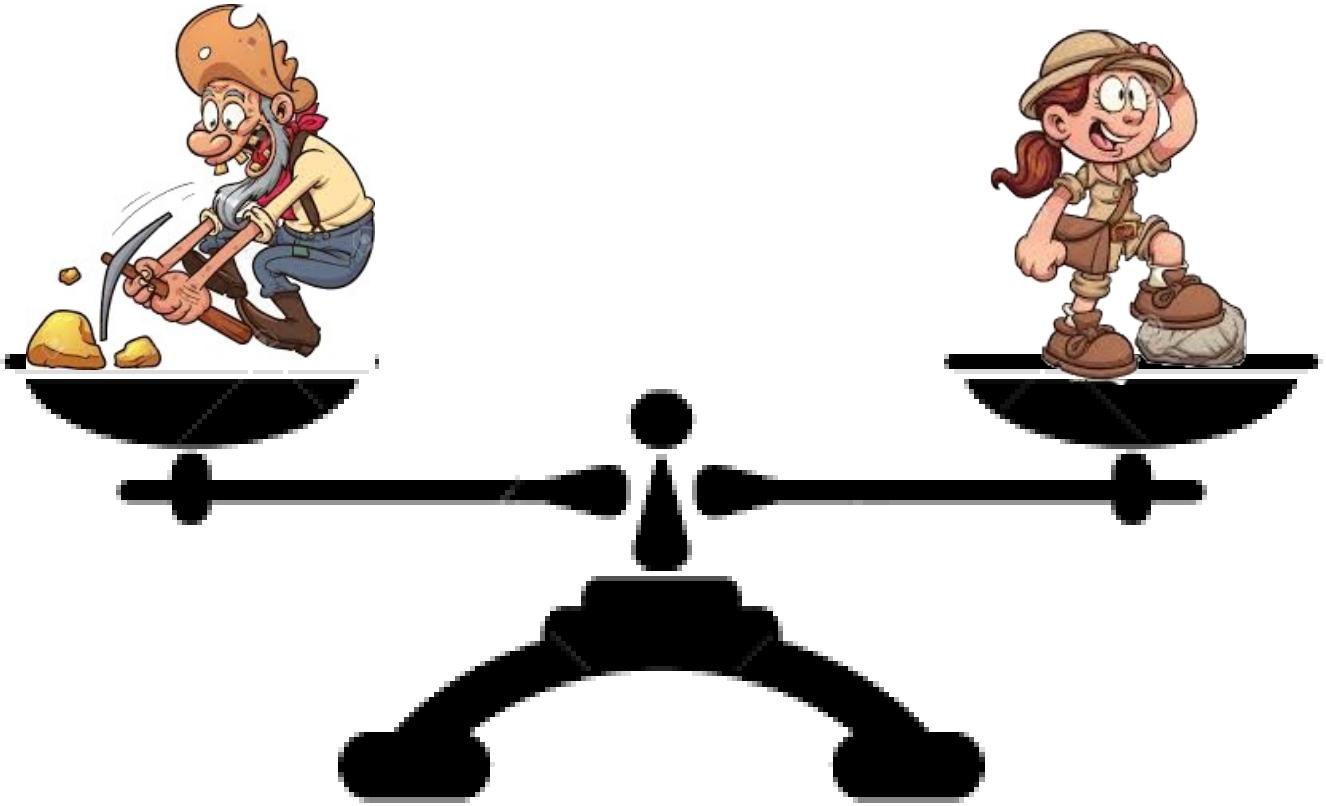
The questionable balance mantra



Maurice.Clerc@WriteMe.com

A classical claim to carefully examine

randomness

"This iterative optimiser is efficient for it ensures a good balance between exploitation and exploration"

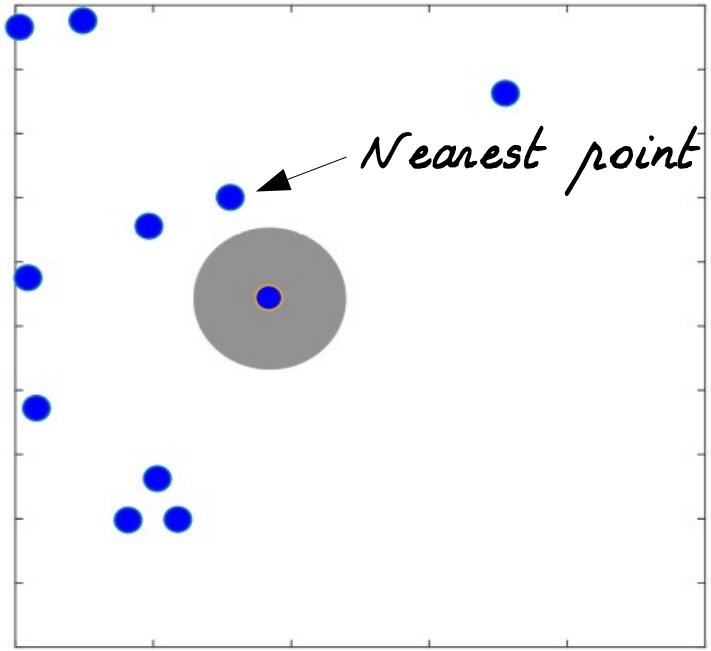
?

?

?

Maurice.Clerc@WriteMe.com

Exploitation
D-sphere based

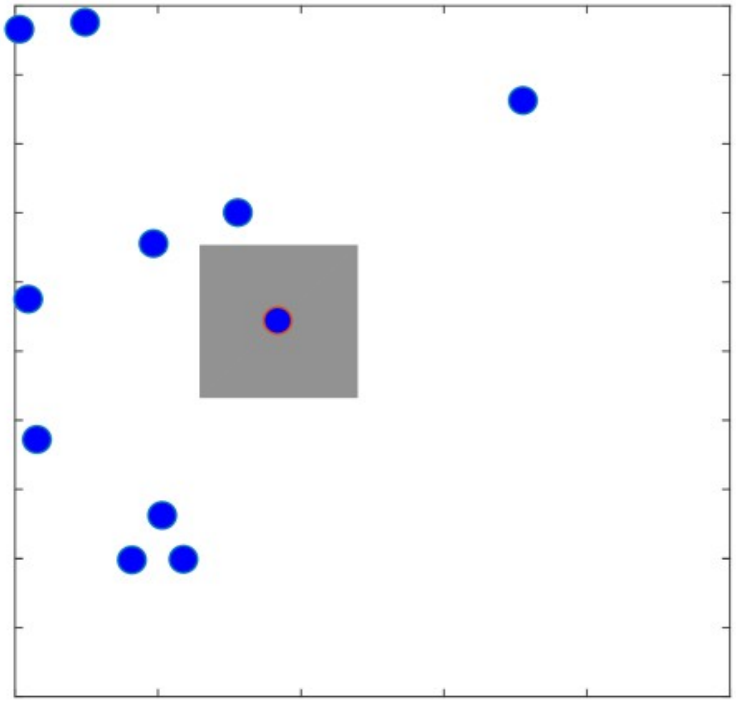


- Needs to save all sampled points.
- radius = k*(distance to the nearest one)
- Sampling: uniform, Gaussian, any local search.

to simplify
k=1

Maurice.Clerc@WriteMe.com

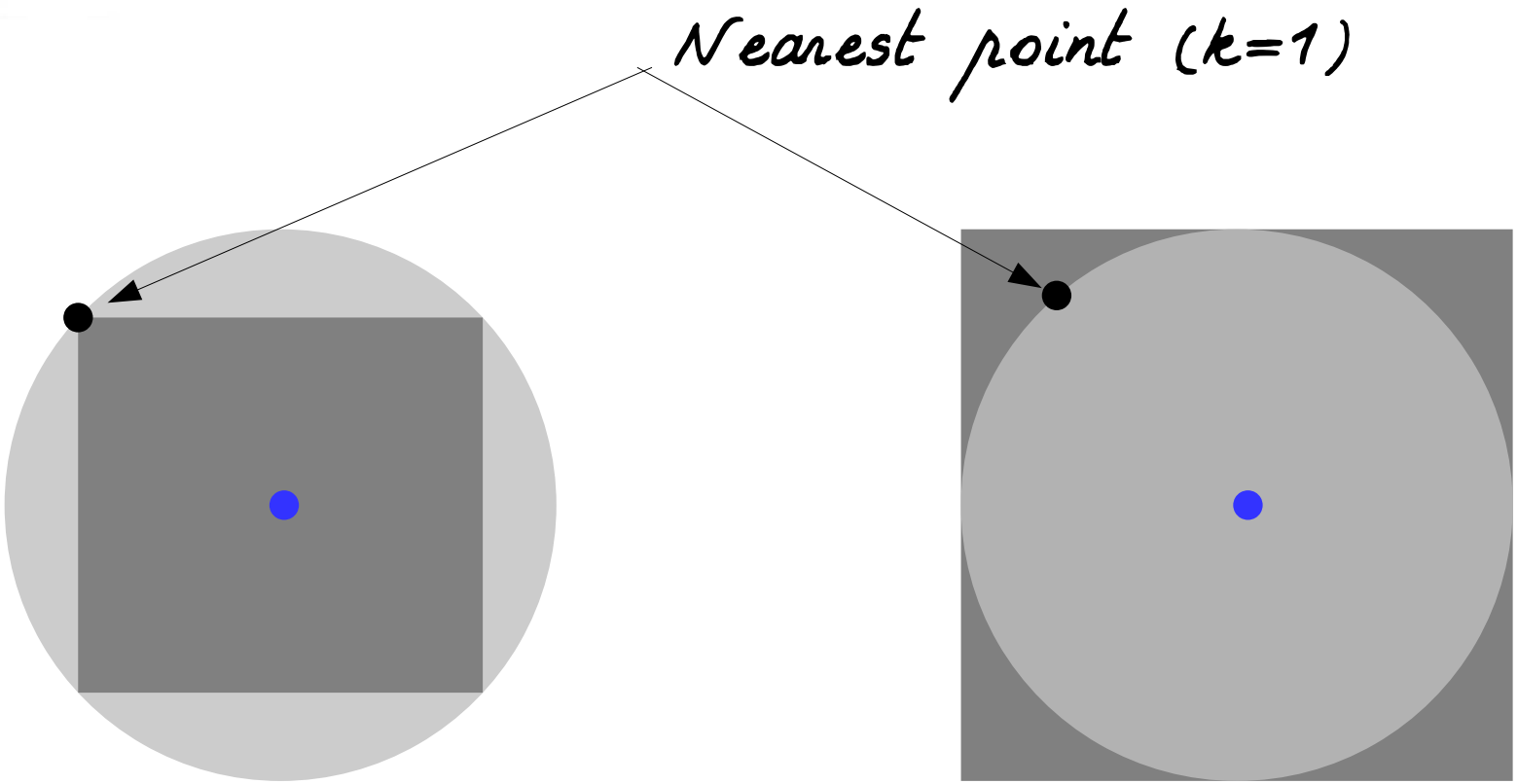
Exploitation
D-cube based



- Needs to save all sampled points
- 1/2 side or diagonal < distance to the nearest one
- Sampling: uniform, Gaussian, any local search

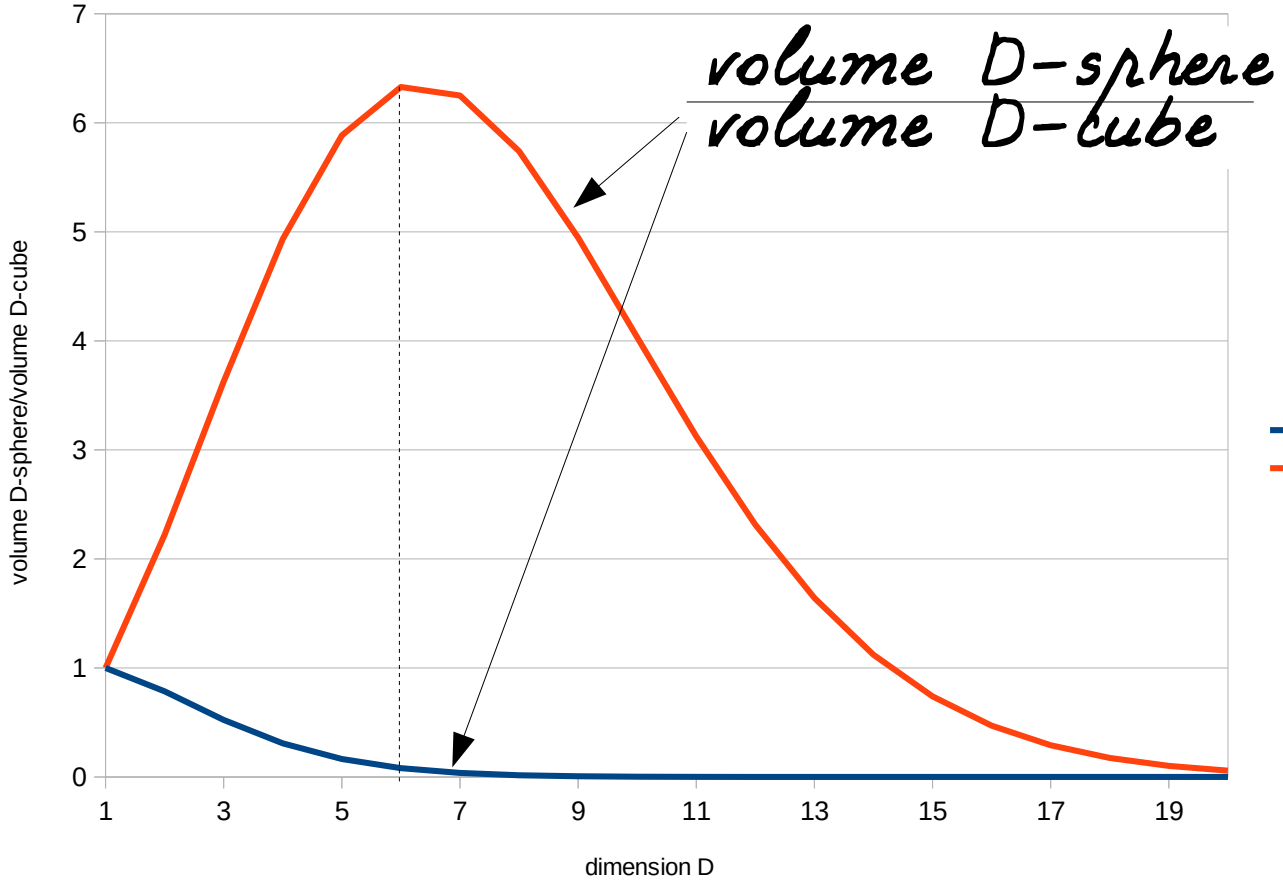
Maurice.Clerc@WriteMe.com

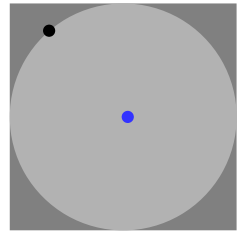
D-cube vs D-sphere (1)

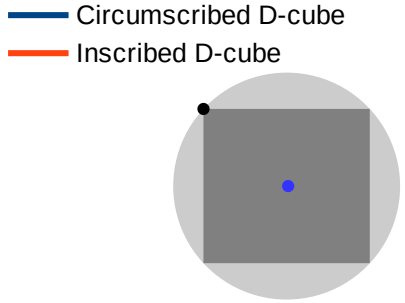


Maurice.Clerc@WriteMe.com

D-cube vs D-sphere (2)



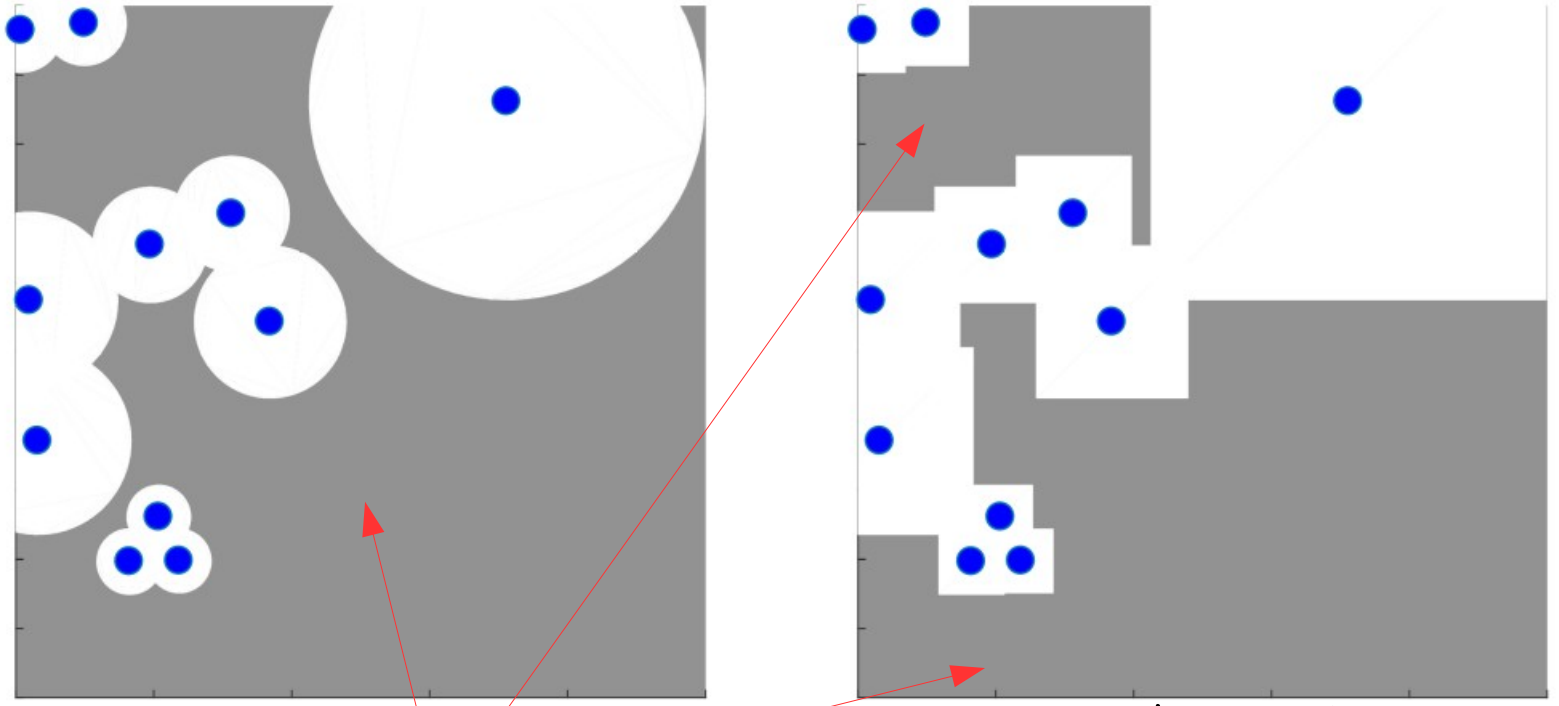




Maurice.Clerc@WriteMe.com

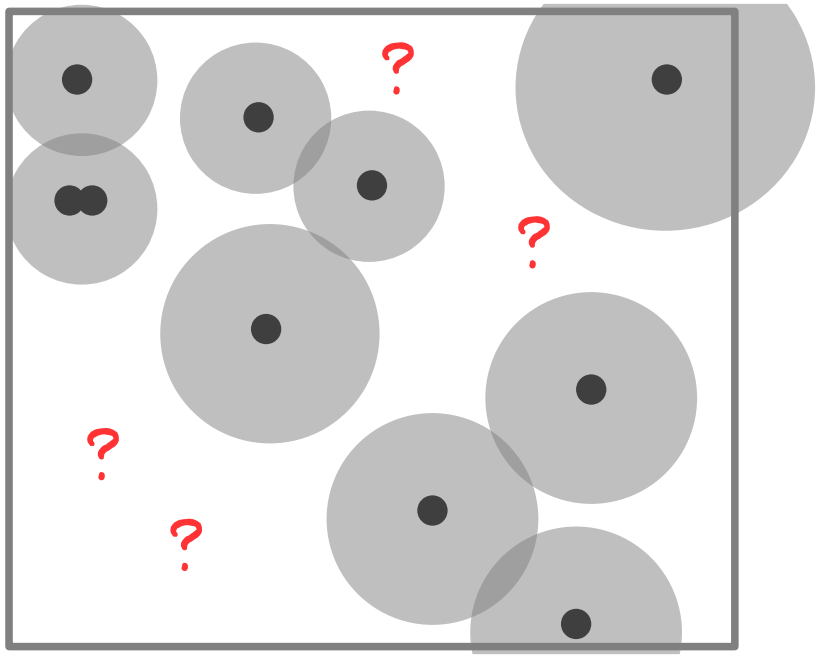
Exploration

Logical negation



Maurice.Clerc@WriteMe.com

How and where to explore?

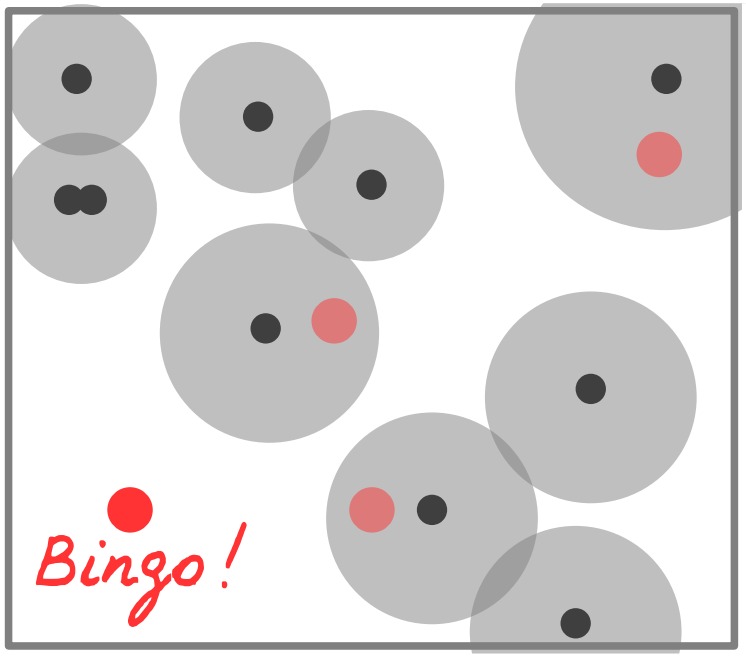


Ten exploitation domains

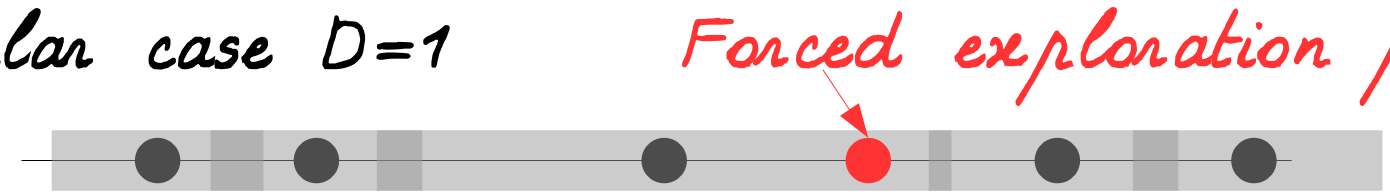
Maurice.Clerc@WriteMe.com

Random exploration

At random, until not in an exploitation domain



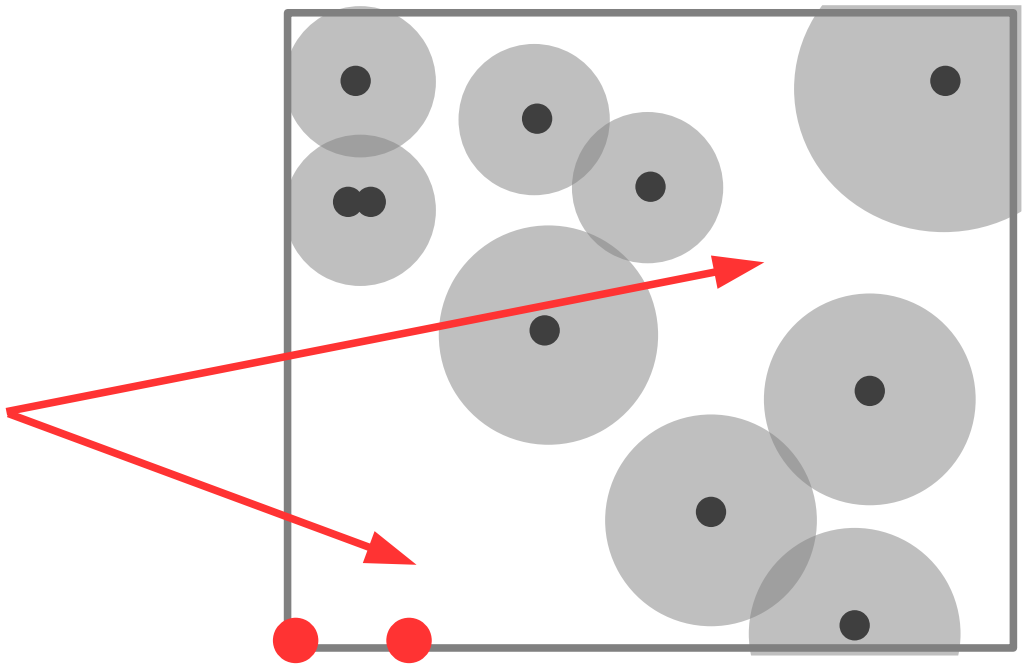
Particular case D=1



Maurice.Clerc@WriteMe.com

No Man's Land

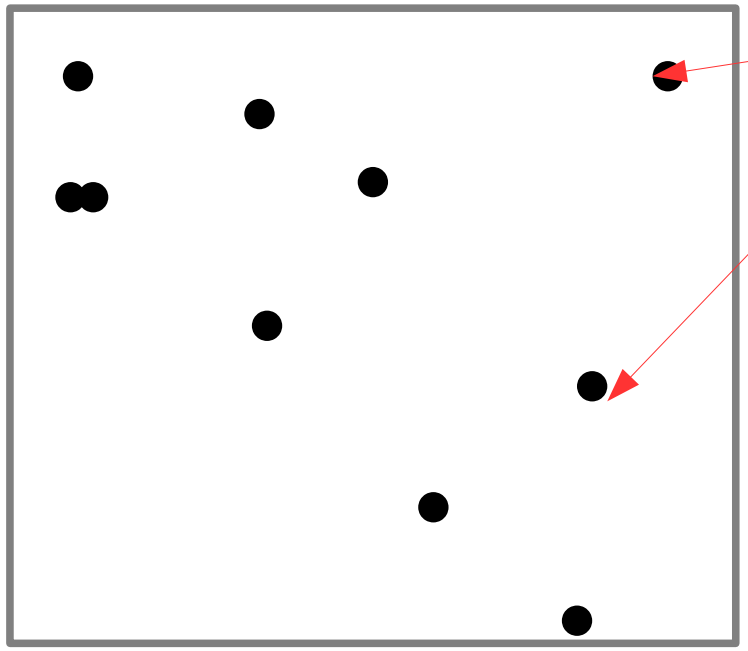
Far from already sampled points



Maurice.Clerc@WriteMe.com

SunnySpell technique

Normalised search space $[0,1]^D$



Sampled points:

$$\{X_1, X_2, \ldots, X_N\}$$

Looking for $X = (x_1, x_2, \ldots, x_D)$

minimising

arbitrary β

$$\frac{1}{min_{d=1}^{D} min\left(x_d^{\beta}, (1-x_d)^{\beta}\right)} + \frac{1}{min\|X^* - X_d\|}$$

Repulsion with respect to the "walls"

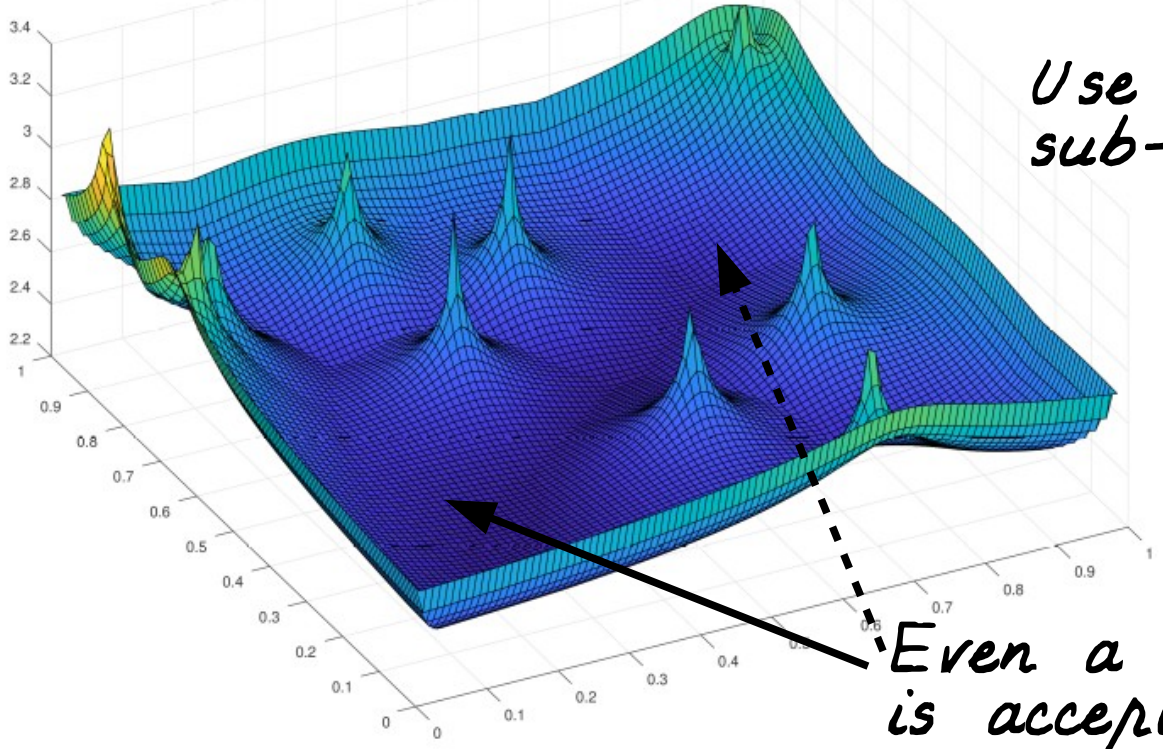
Repulsion with respect to sampled points

Maurice.Clerc@WriteMe.com

SunnySpell landscape

Potential function over 10 sampled points

$\beta = 0.1$

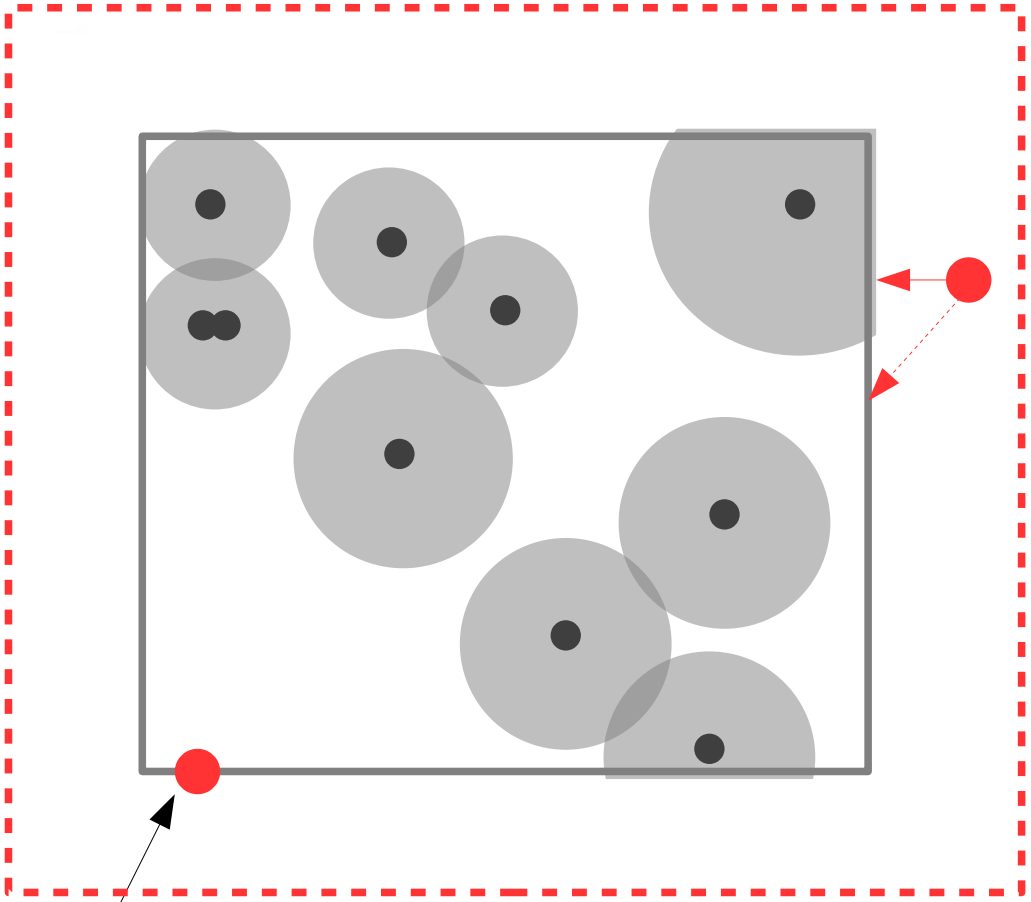


Use a simple sub-optimisation

Even a local minimum is acceptable

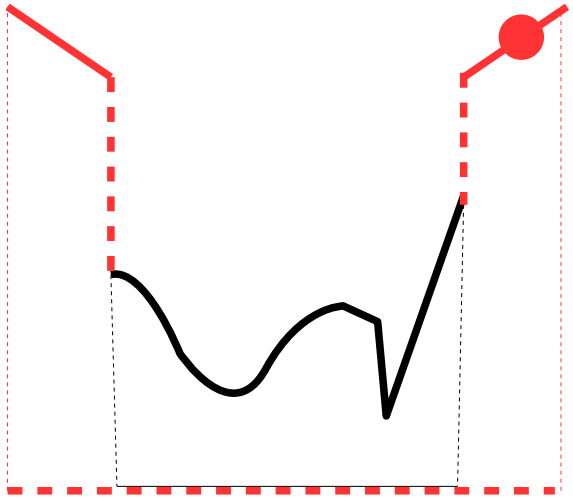
Maurice.Clerc@WriteMe.com

Extended search space



Back to the real search space. Different moves may be needed.

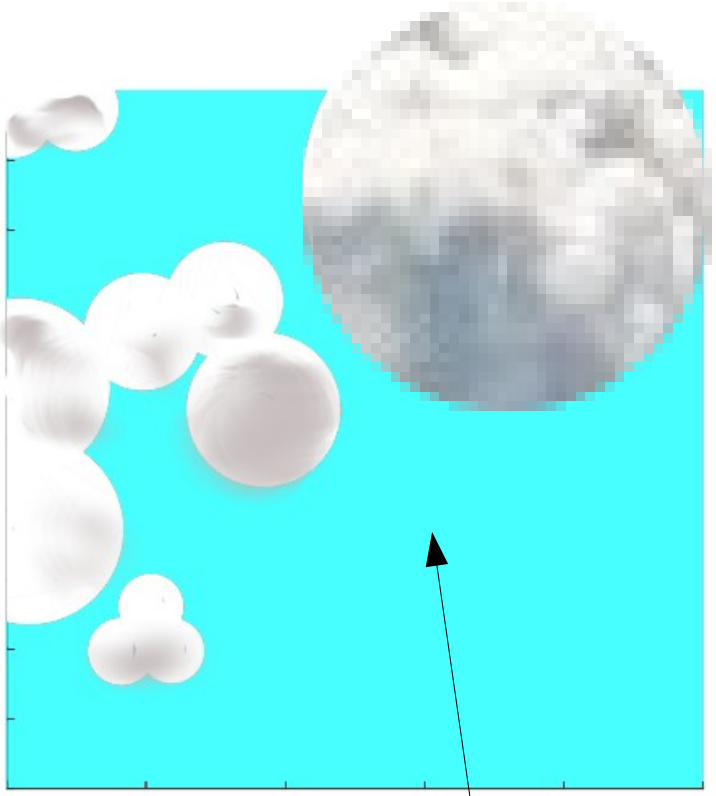
Or, better: assign a high value, increasing function of the distance to the centre of the search space.



Can be on the frontier

Maurice.Clerc@WriteMe.com

Why "SunnySpell" ?



Exploration domain

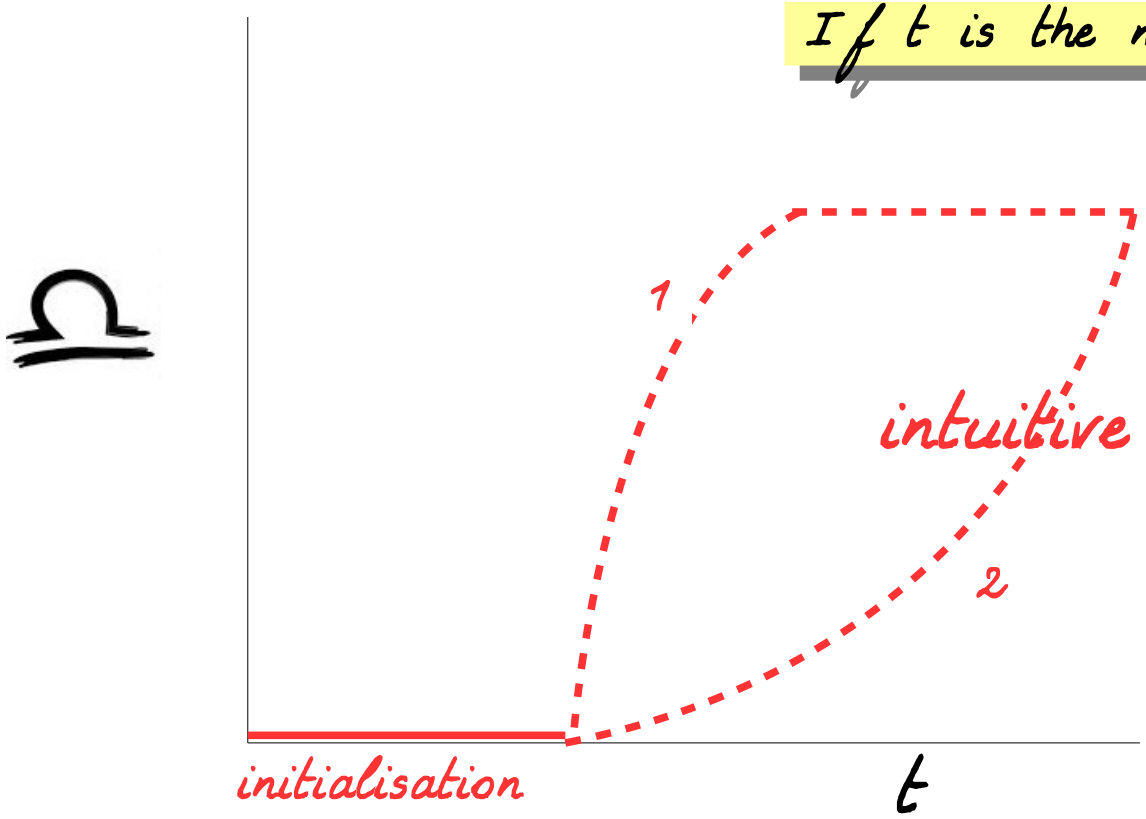
Maurice.Clerc@WriteMe.com

Balance profile

$$\Omega(t) = \frac{\text{number of exploitation points } (t)}{\text{number of exploration points } (t)}$$

If t is the number of evaluations

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



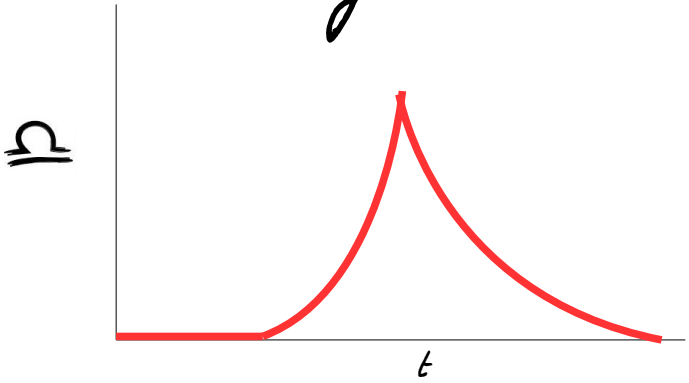
Maurice.Clerc@WriteMe.com

Two logics

The more you find good positions, the more it is worth spending time . . .

1) . . . looking close to them, for there is probably an even better not too far away.

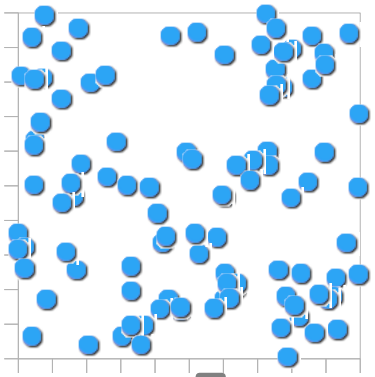
2) . . . looking elsewhere, now you already have a good solution.

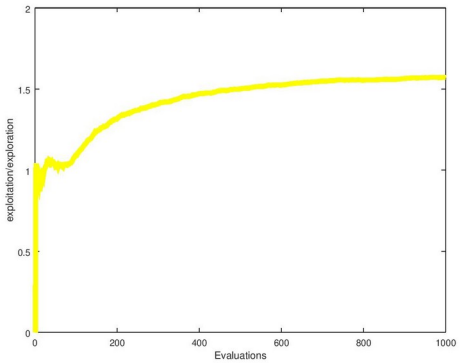


Maurice.Clerc@WriteMe.com

Three experiments

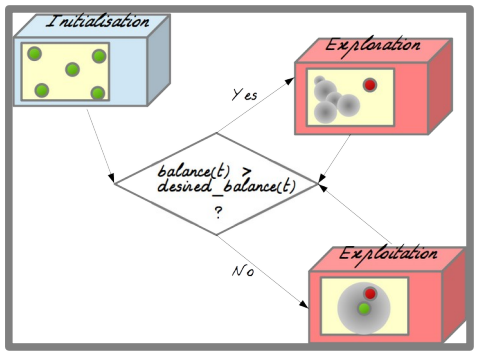
Random search

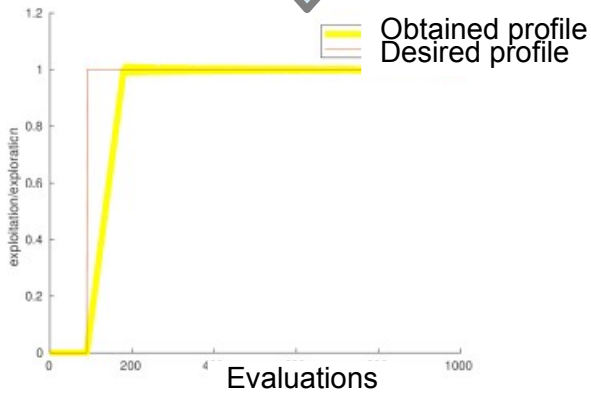




Generated profiles

Explo2

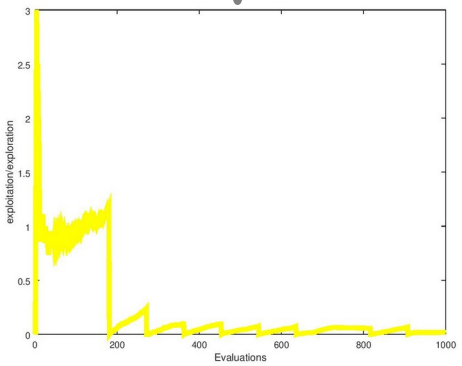




Two good algorithms

APS

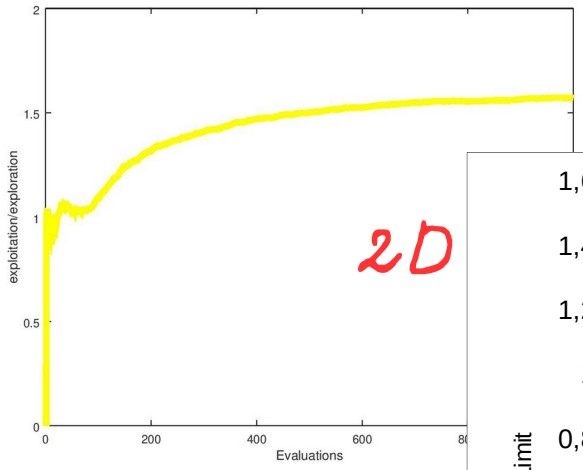
SPSO 2011

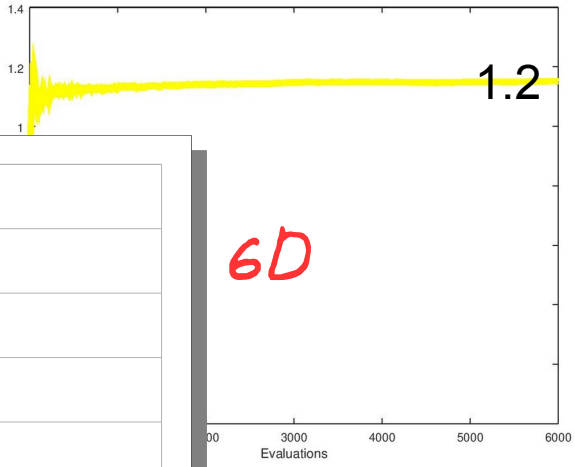


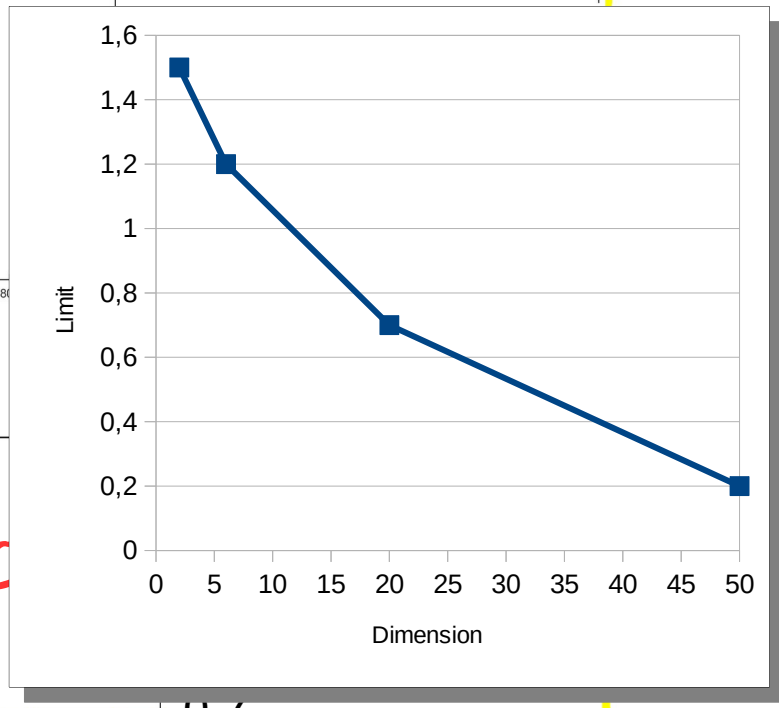
Generated profiles

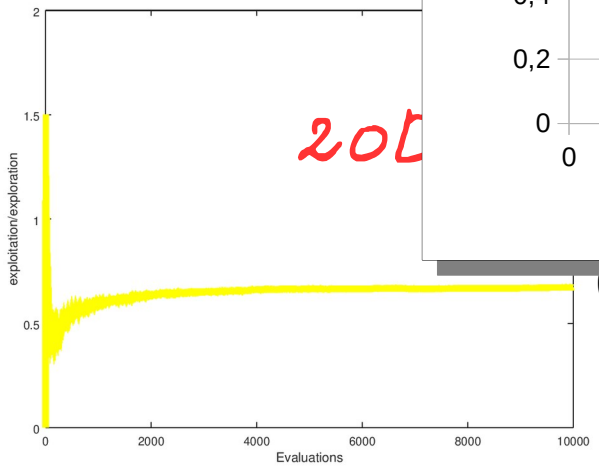
Maurice.Clerc@WriteMe.com

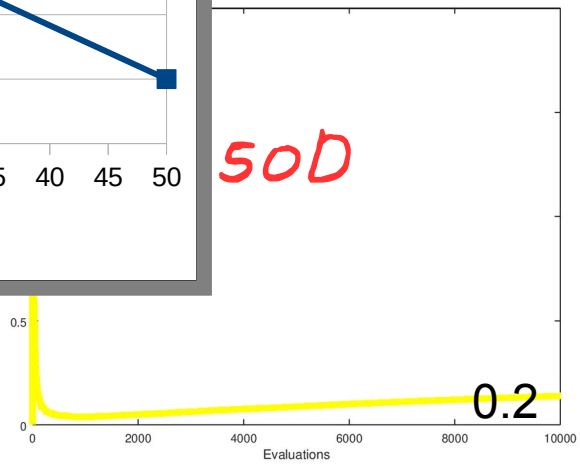
Random search profiles





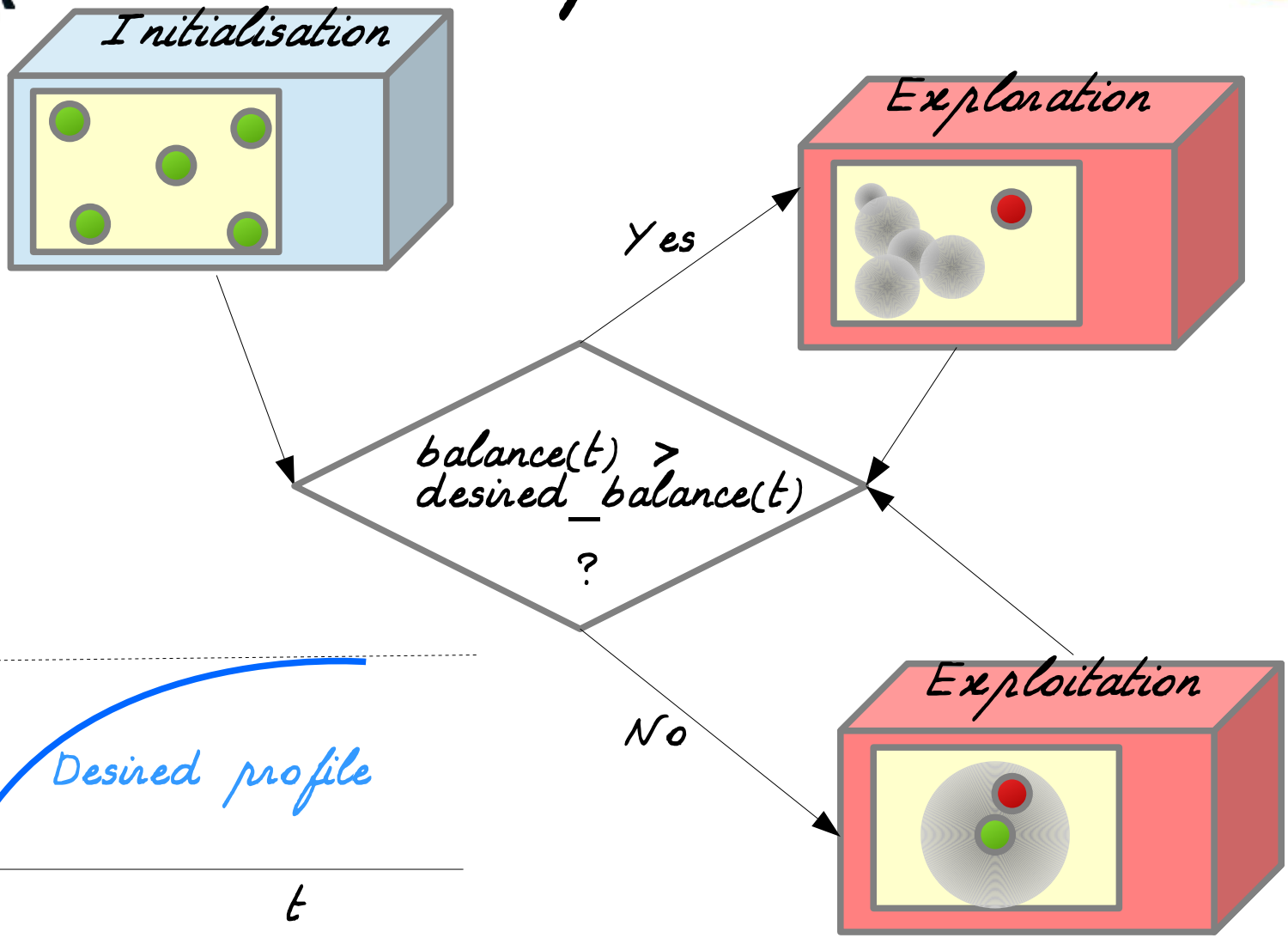






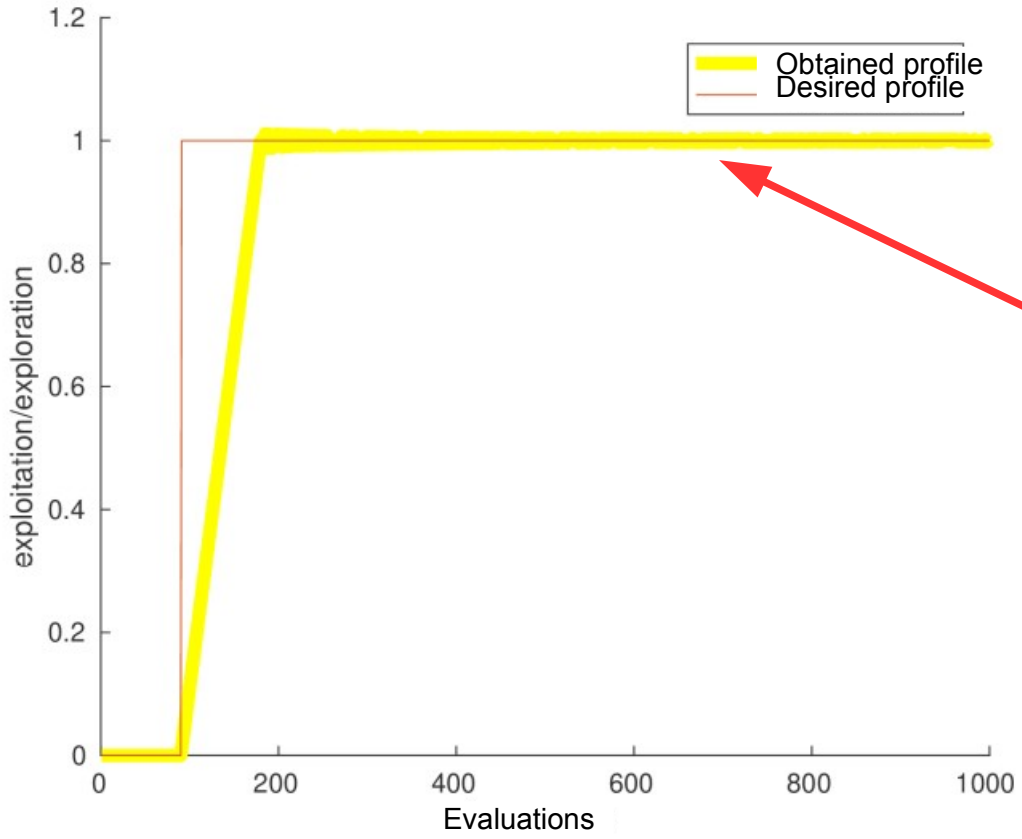
Maurice.Clerc@WriteMe.com

Explo2



Maurice.Clerc@WriteMe.com

Explo2 with predefined profile

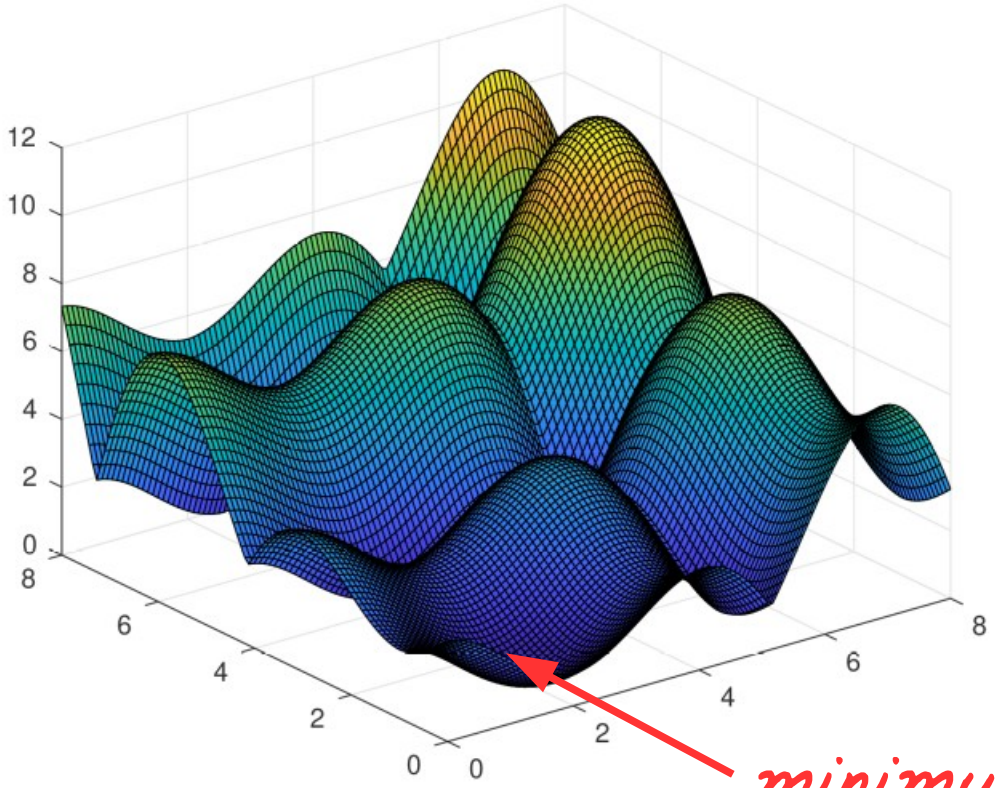


Perfect balance

Obtained profile= mean on 100 runs

Maurice.Clerc@WriteMe.com

A simple example
Alpine 2D



minimum 0 on (1,2)

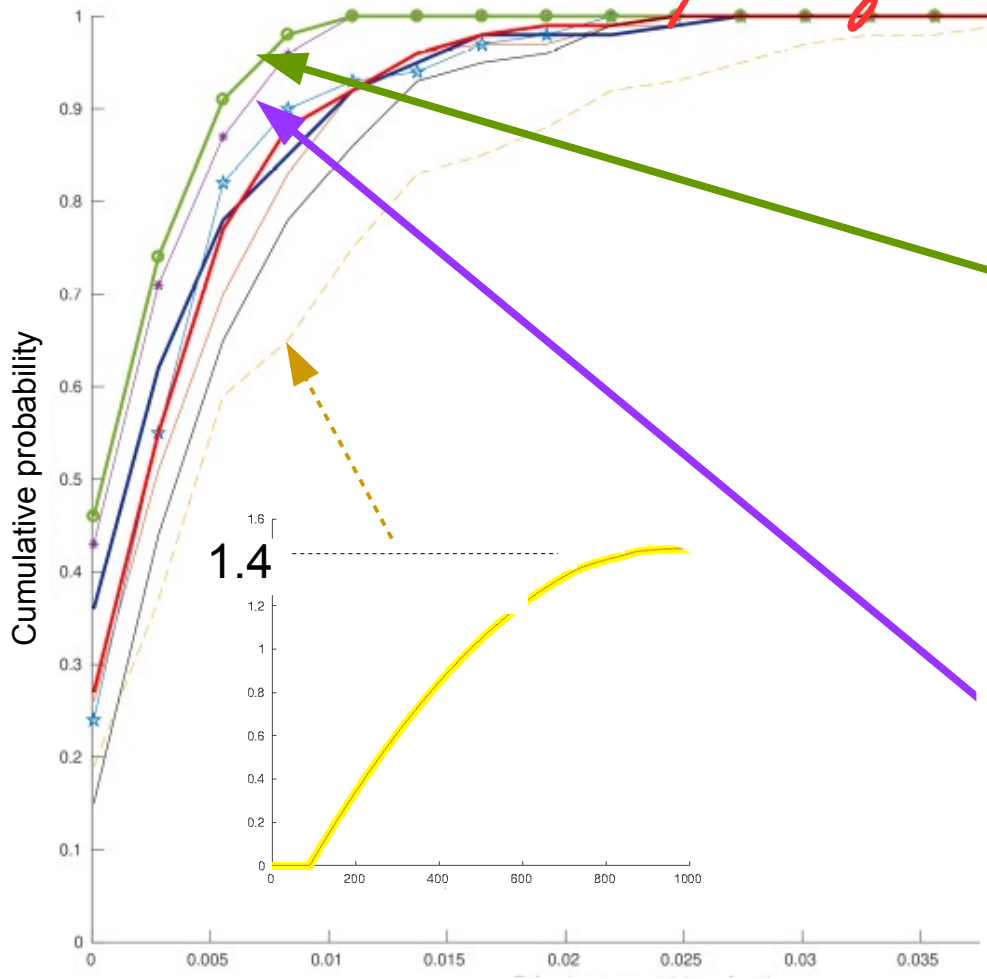
$$f(x)=\sum_{d=1}^{D}|(x_d-d)\sin(x_d-d)|+0.1|x_d-d|$$

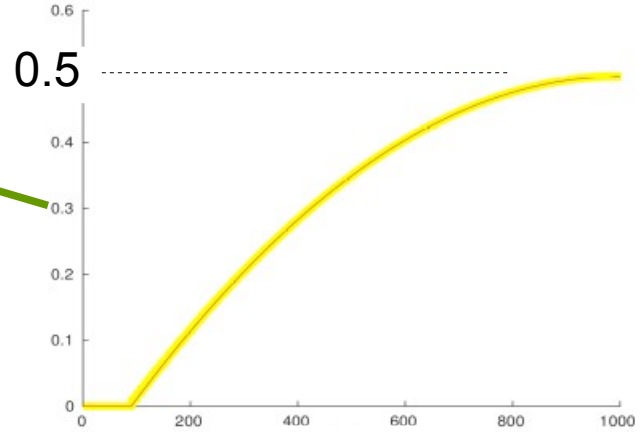
See Appendix

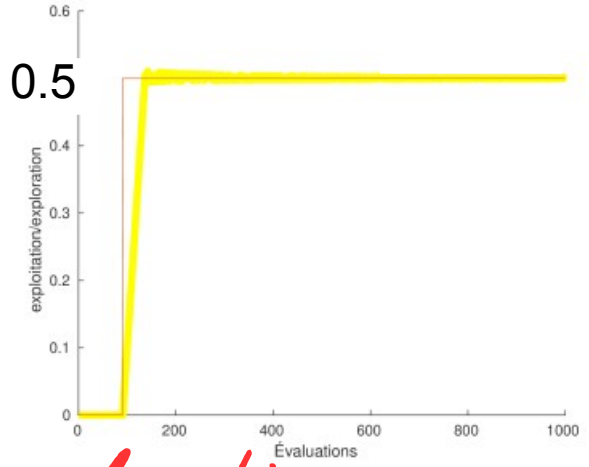
Maurice.Clerc@WriteMe.com

Best profile ?

Comparing CDFs





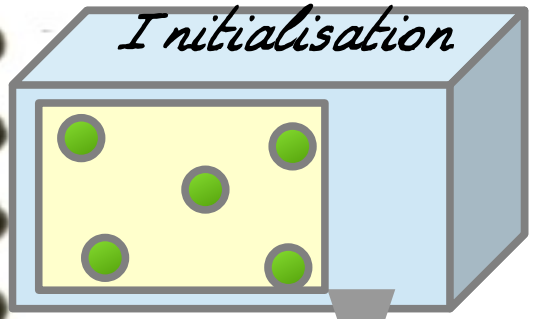


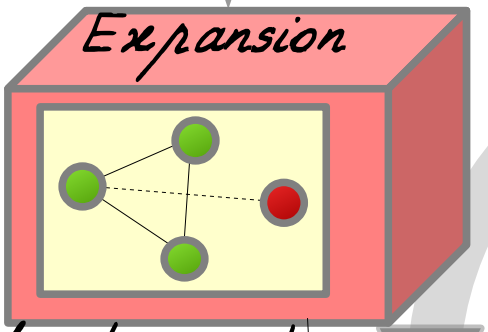
Better less exploitation than exploration

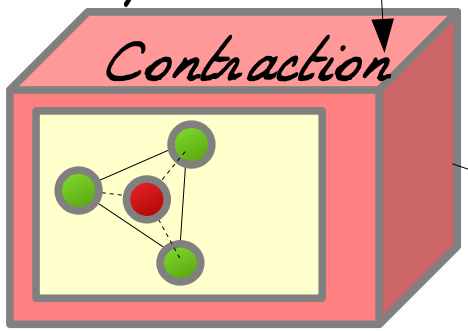
Maurice.Clerc@WriteMe.com

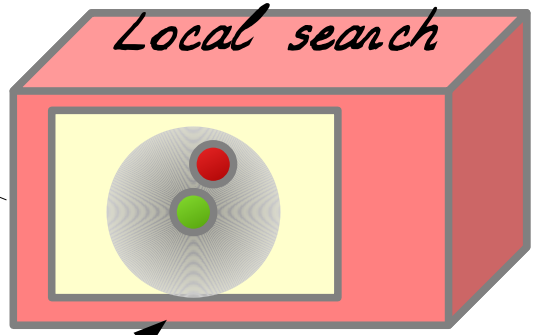
Adaptive Population-based Simplex

http://aps-optim.info/









if not enough improvement

if not enough improvement

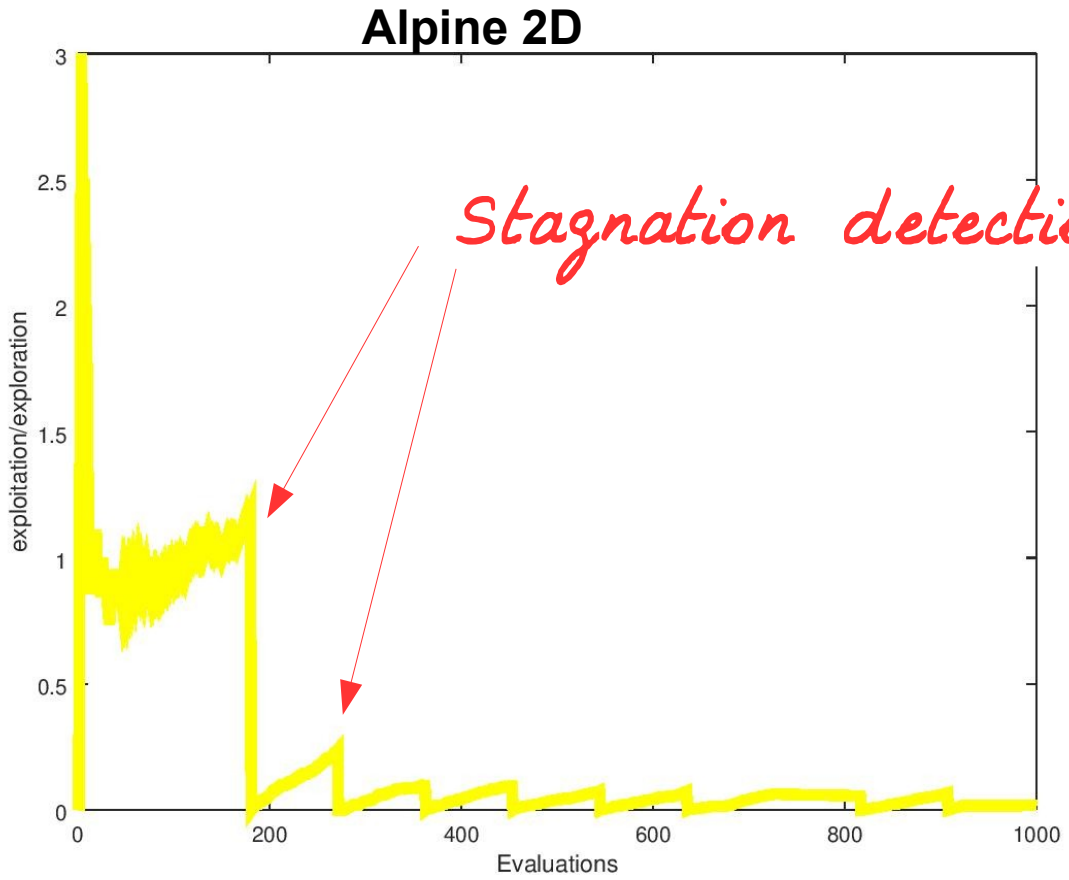
Stagnation detection
Selection
Partial restart

Optional (not used h

Maurice.Clerc@WriteMe.com

APS
+ balance observer

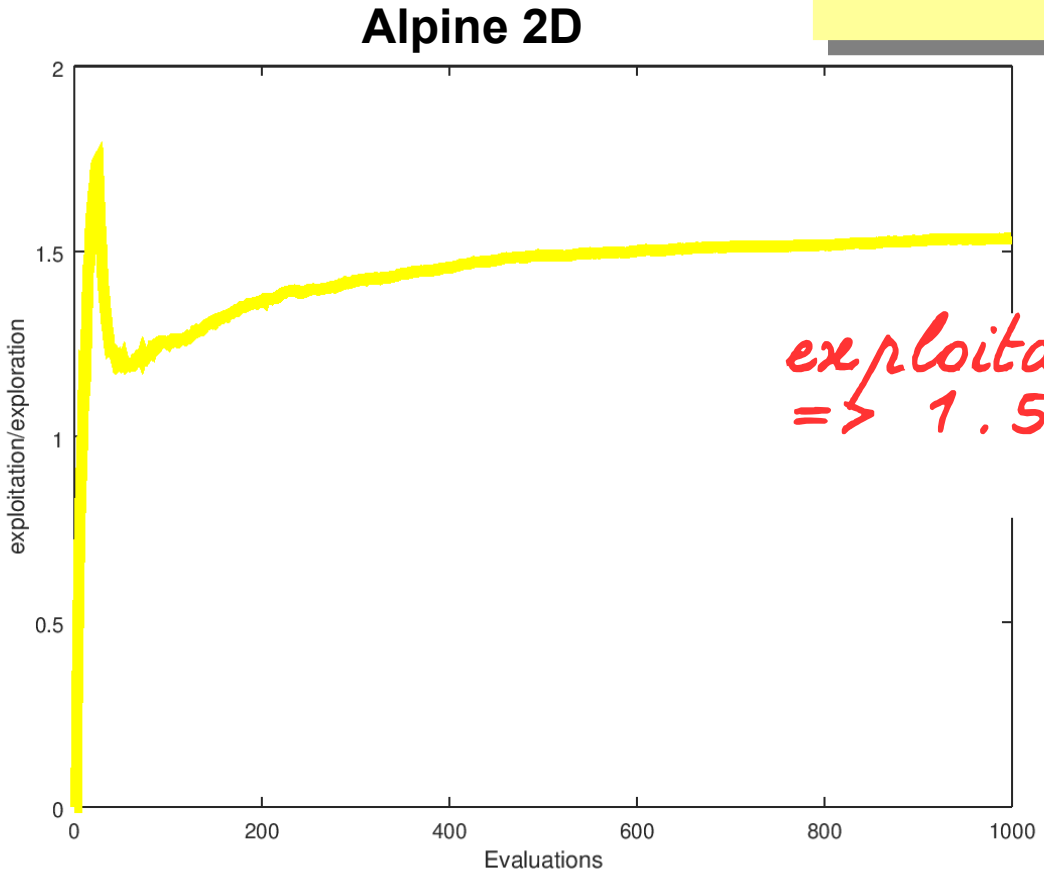
Adaptive Population-Based Simplex



Maurice.Clerc@WriteMe.com

SPSO 2011 + balance observer

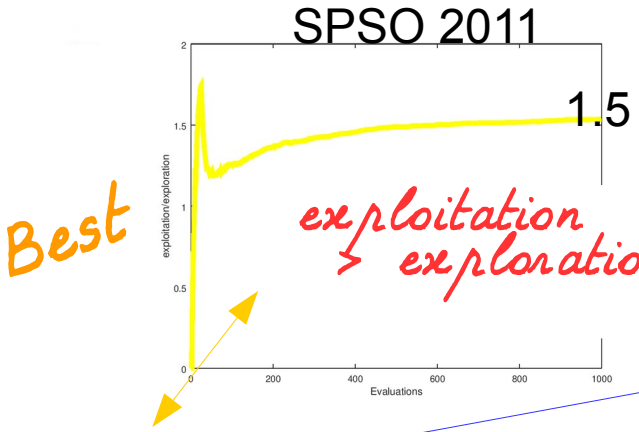
http://particleswarm.info/



exploitation/exploration
=> 1.5

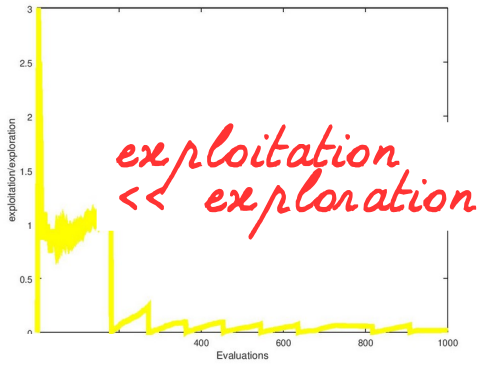
Maurice.Clerc@WriteMe.com

CDF vs Profile (Alpine 2D)

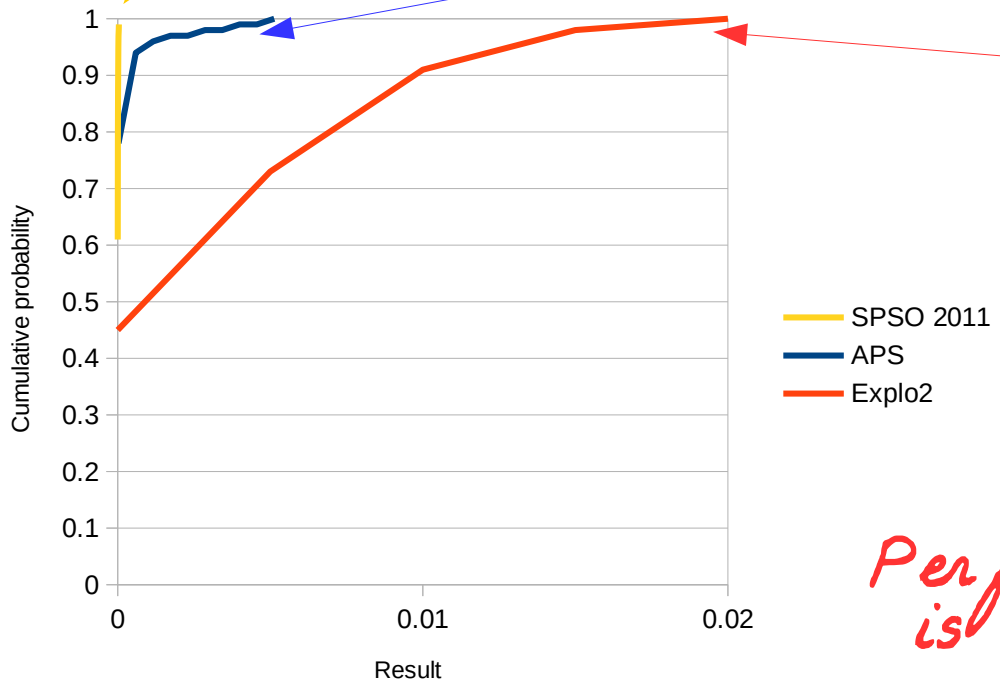


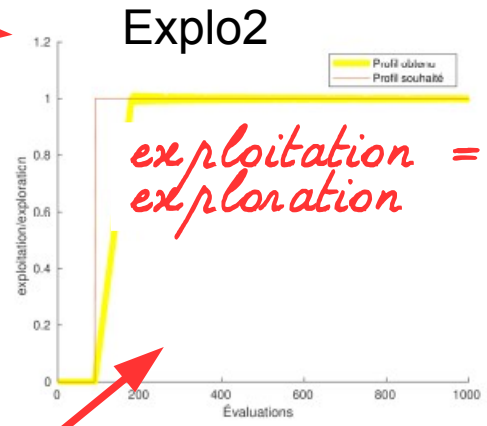
Best

exploitation > exploration



exploitation << exploration





1

exploitation = exploration

Perfect balance is the worst strategy

Maurice.Clerc@WriteMe.com

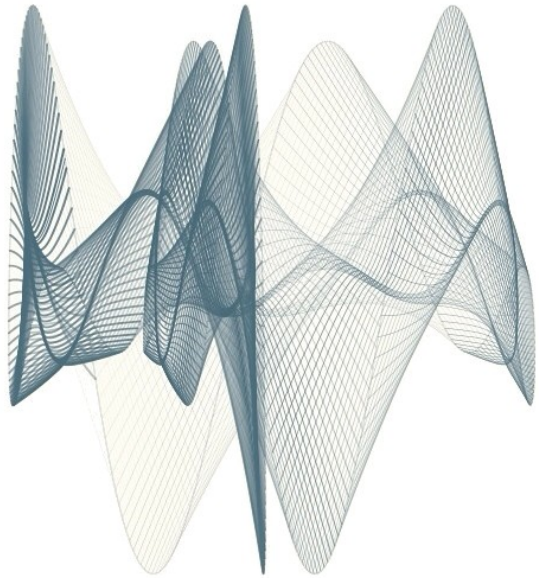
Another example (FM 6D)

CEC 2011 Function 1: Frequency-Modulated Sound Waves

$$y(t) = a_1 \sin(\omega_1) t\,\theta + a_2 \sin(\omega_2) t\,\theta + a_3 \sin(\omega_3) t\,\theta$$
$$y_0(t) = \sin(5) t\,\theta + 1.5 \sin(4.8) t\,\theta + 2 \sin(4.9) t\,\theta$$

$$\theta = \frac{\pi}{50}$$
$$x = (a_1, \omega_1, a_2, \omega_2, a_2, \omega_3) \in [-6.4, 6.35]^6$$

$$f(x) = \sum_{t=0}^{100} (y(t) - y_0(t))^2$$



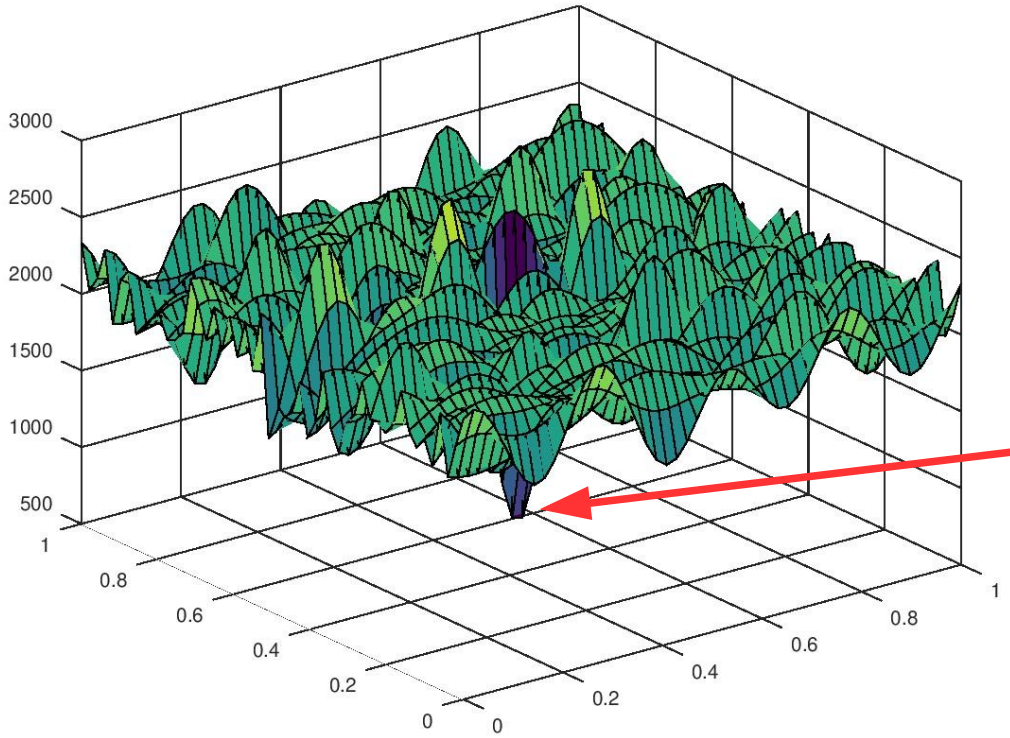
solution=(1, 5, 1.5, 4.8, 2, 4.9)

Maurice.Clerc@WriteMe.com

Difficulty

“This problem is a highly complex multimodal one having strong epistasis”

Swagatam Das and Ponnuthurai N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition



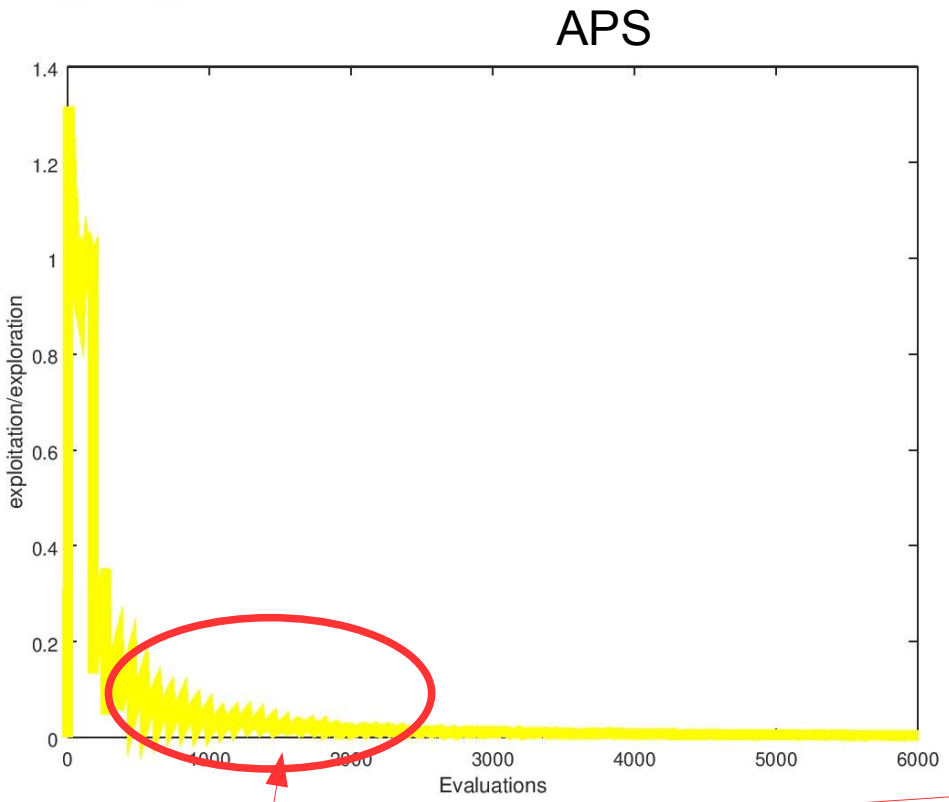
narrow attraction basin

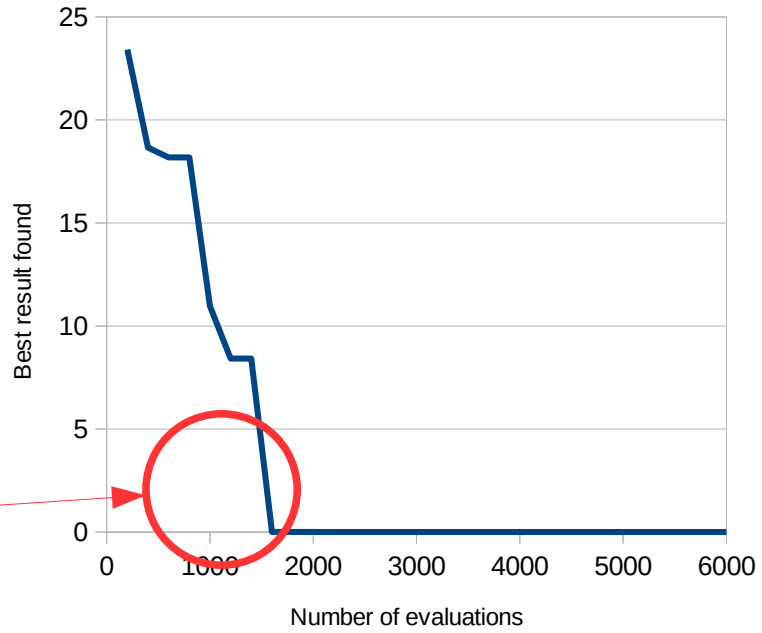
A typical 3D (normalised) cross section

See Appendix

Maurice.Clerc@WriteMe.com

Profile and evolution





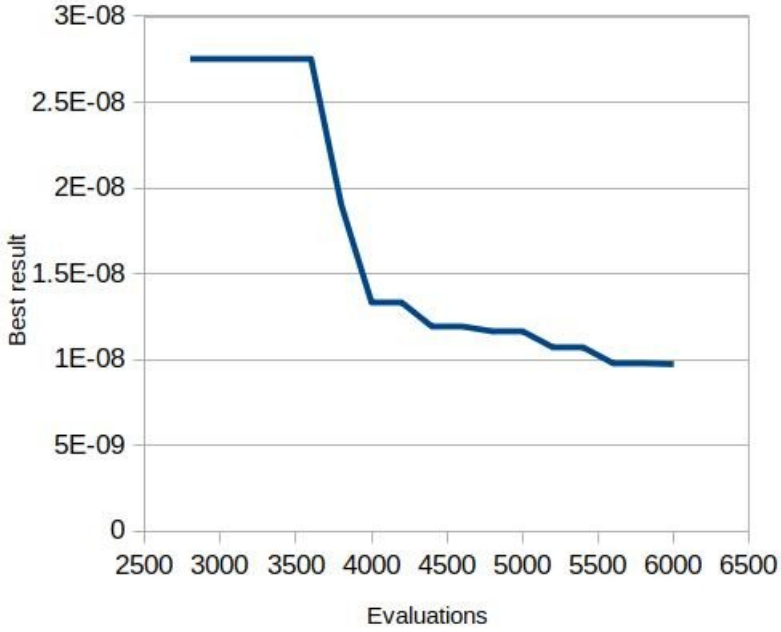
Far more explorations than exploitations still improve the search

Maurice.Clerc@WriteMe.com

Evaluations	Best value
2600	1.382402E-05
2800	2.750586E-08
3000	2.750586E-08
3200	2.750586E-08
3400	2.750586E-08
3600	2.750586E-08
3800	1.90227E-08
4000	1.331773E-08
4200	1.331773E-08
4400	1.189554E-08
4600	1.189554E-08
4800	1.168904E-08
5000	1.168904E-08
5200	1.075004E-08
5400	1.075004E-08
5600	9.80309E-09
5800	9.80309E-09
6000	9.740409E-09

Zoom

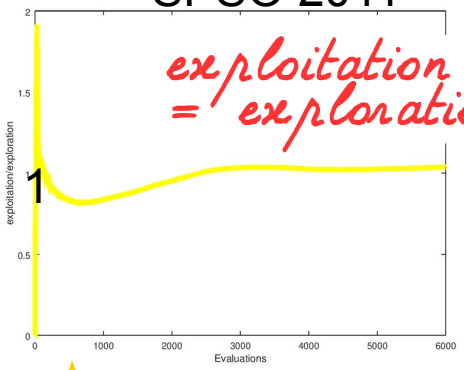
almost no more exploitation, but still improving



Maurice.Clerc@WriteMe.com

CDF vs Profile (FM 6D)

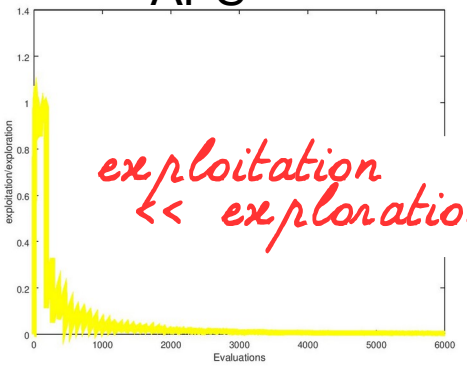
SPSO 2011



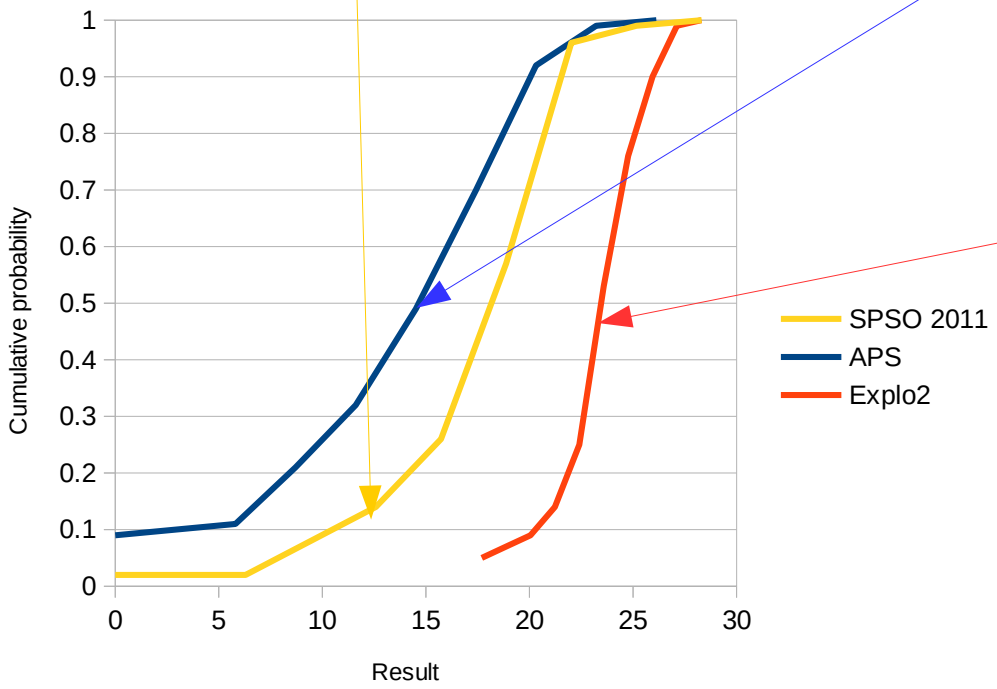
exploitation = exploration

Best

APS

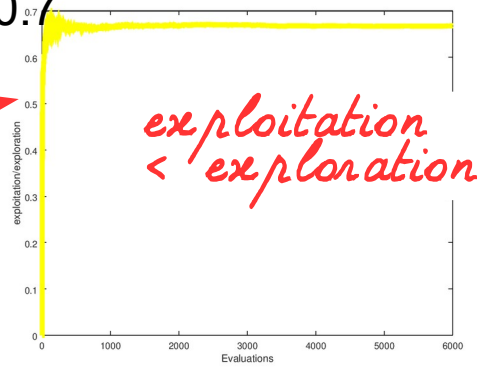


exploitation << exploration



Explo2

0.7

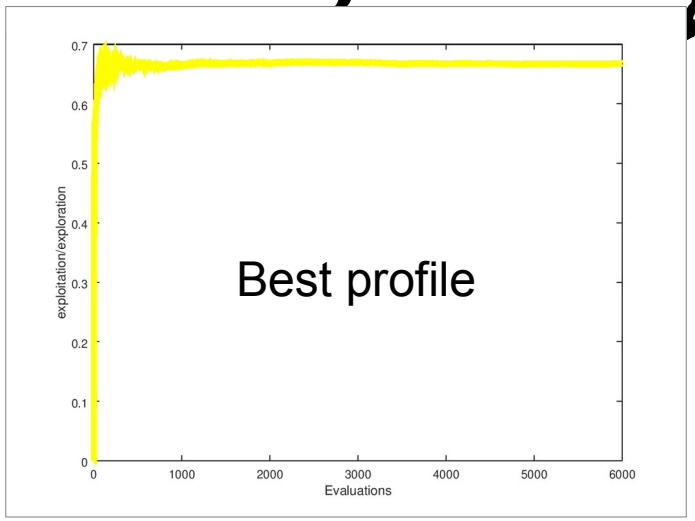


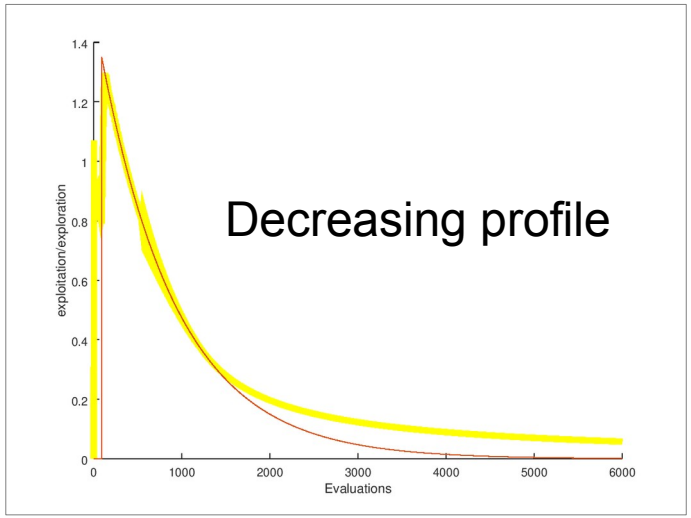
exploitation < exploration

(best profile)

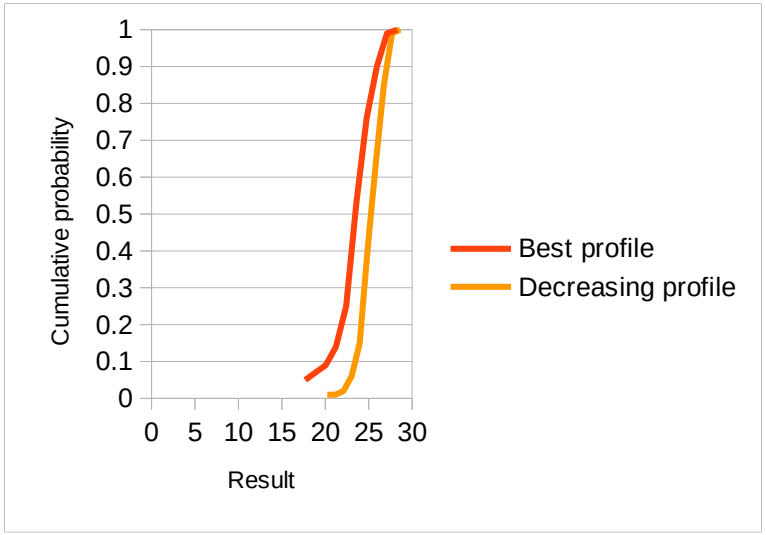
Maurice.Clerc@WriteMe.com

Two profiles with Explo2 for FM 6D





CFDs



Using a profile similar to the one generated by APS is not a good idea

Maurice.Clerc@WriteMe.com

So, please

Prove

Prove

"This iterative optimiser is efficient for it ensures a good balance between exploitation and exploration"

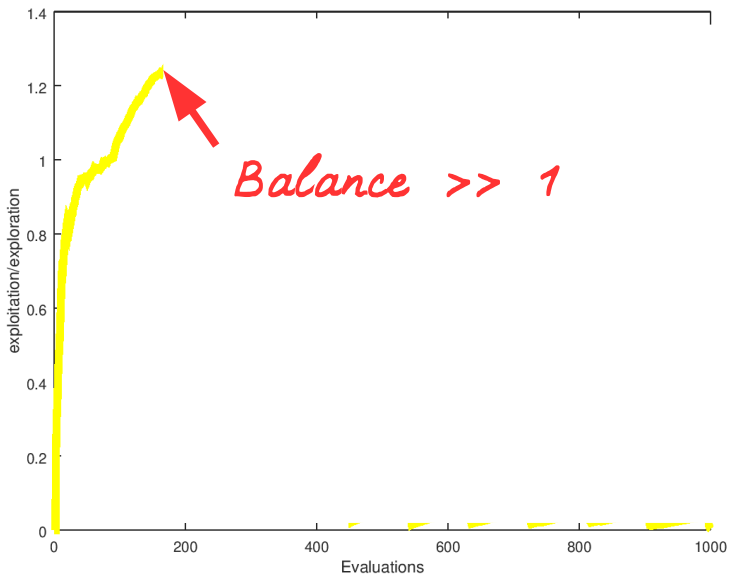
Define

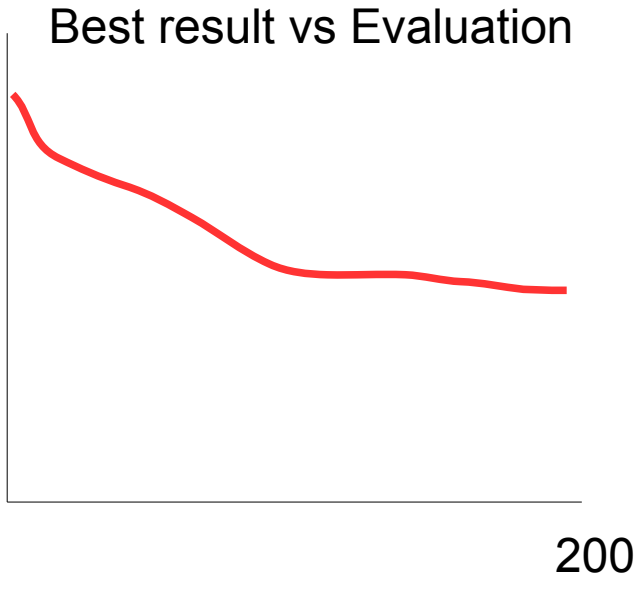
Define and prove

Define

Maurice.Clerc@WriteMe.com

Profile coach?





Adaptation

"You want to exploit but:
- your ratio is already high,
and
-your efficiency is low.
So you should explore instead".

Maurice.Clerc@WriteMe.com

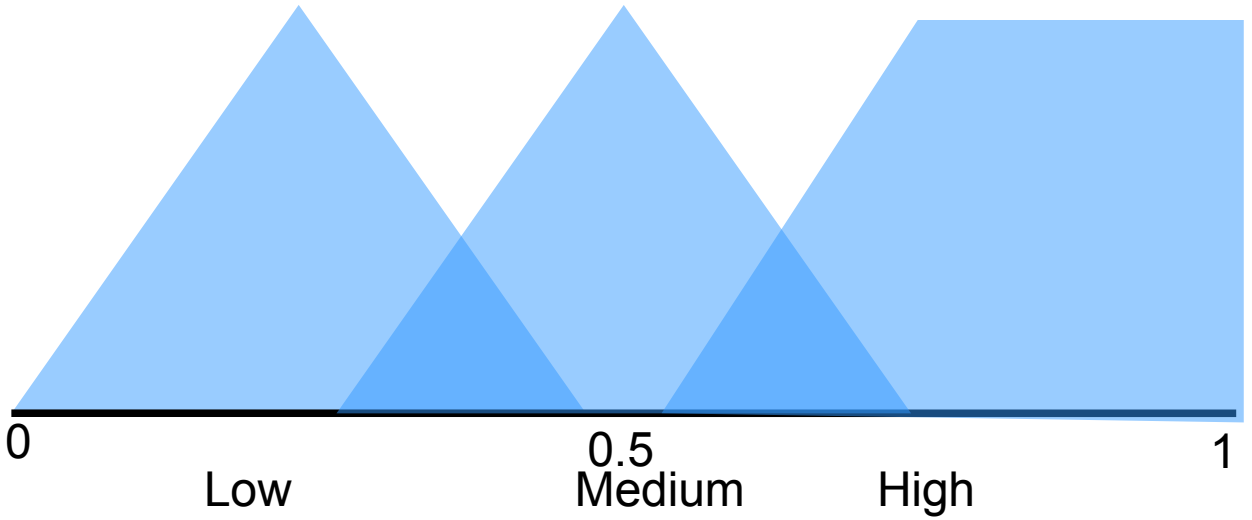
Driving balance by fuzzy logic?

Examples of possible rules:

If efficiency is Low AND ratio is High then Explore

If efficiency is Medium AND ratio is Medium then Explore or Exploit (flip a coin)

If efficiency is High AND ratio is Medium then keep the same strategy



It supposes a numerical evaluation of the efficiency, or of its evolution, on [0,1]. Not that edsy

Maurice.Clerc@WriteMe.com

A table of possible fuzzy rules

Efficiency

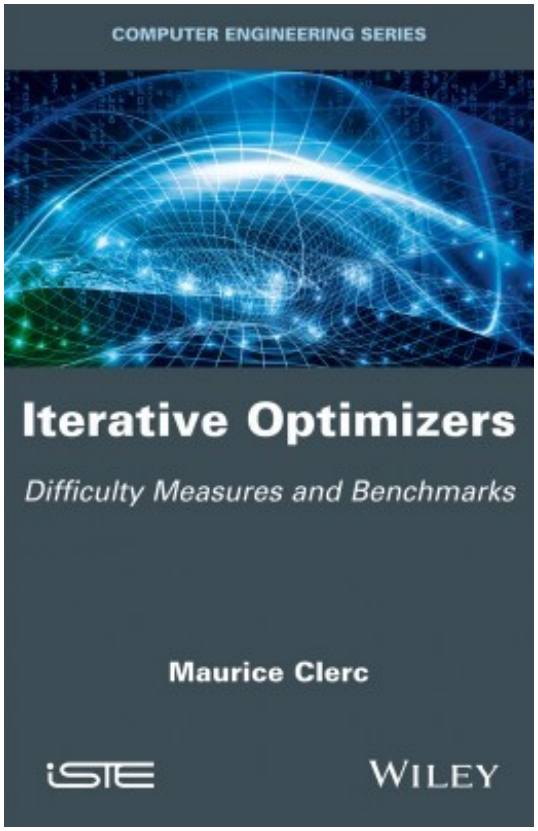
exploitation / *exploration*

	Low	**Medium**	**High**
High	explore	explore or exploit (flip a coin)	exploit
Medium	switch to the other strategy	flip a coin	keep the same strategy
Low	exploit	flip a coin	explore

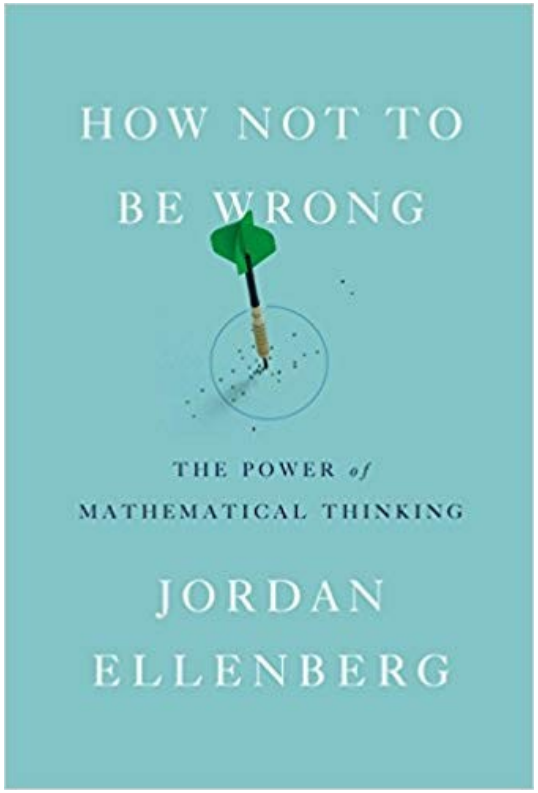
Logic: I already have good solutions, so I can look elsewhere

Maurice.Clerc@WriteMe.com

A little advertising



Mainly chapters 7 and 8



At least the discussion about p-values

Maurice.Clerc@WriteMe.com

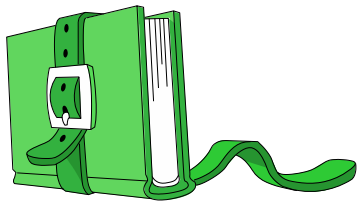
Thank you for your attention!

Maurice.Clerc@WriteMe.com

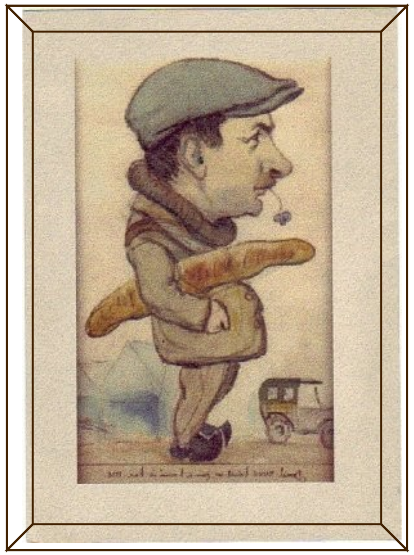
Any question ?

Maurice.Clerc@WriteMe.com

Appendix

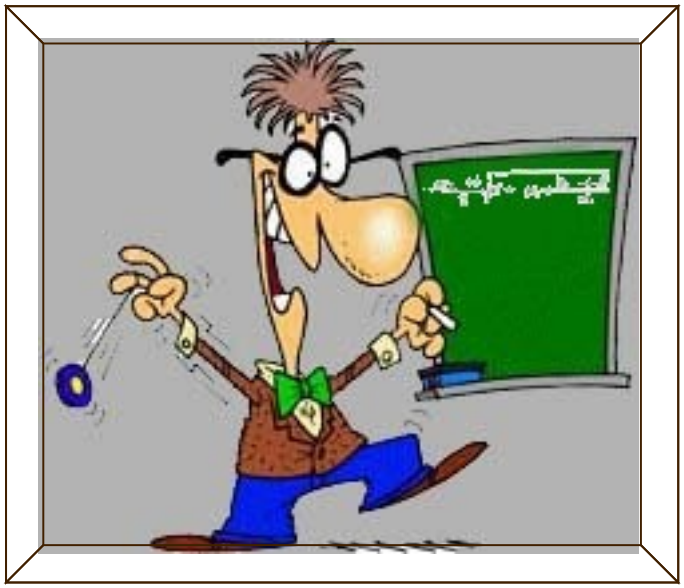


For normal guys



* Counting exploitation points
* Is Exploration always possible?
* Nearer is better

For maths addict



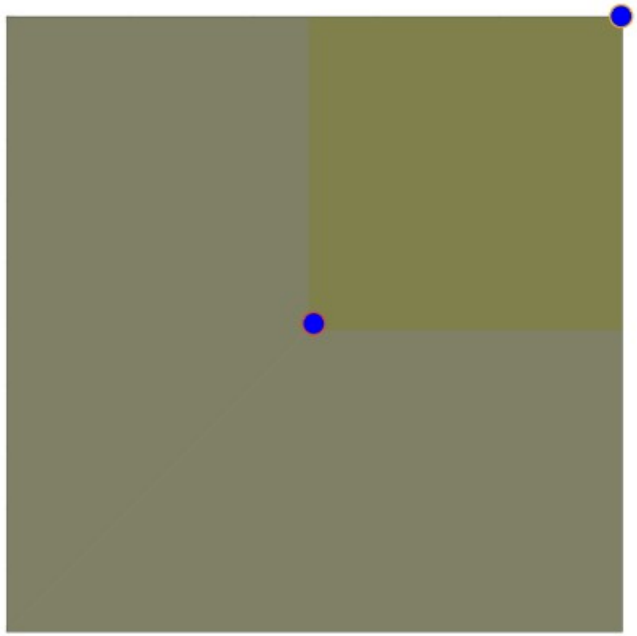
* CDF, Qu'es acò?
* Success proba vs numberb of runs
* Stochastic geometry, a simple example

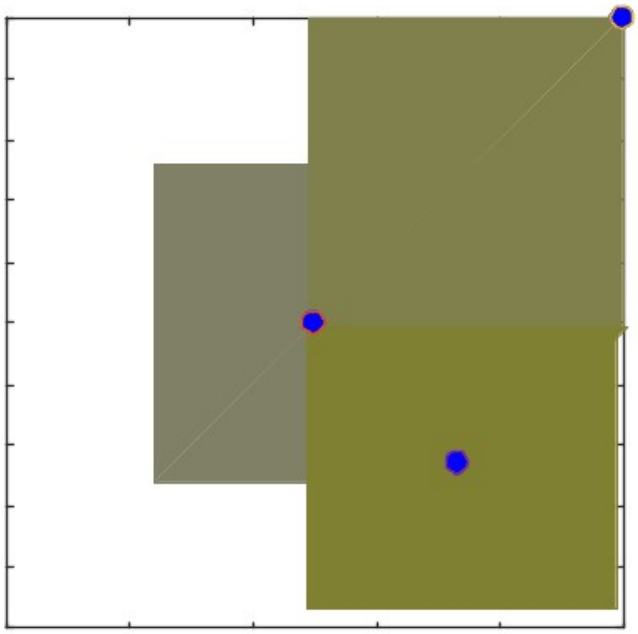
* Difficulty measure
* Difficulty vs Dimension

Maurice.Clerc@WriteMe.com

Is exploration always possible?

D-cubes



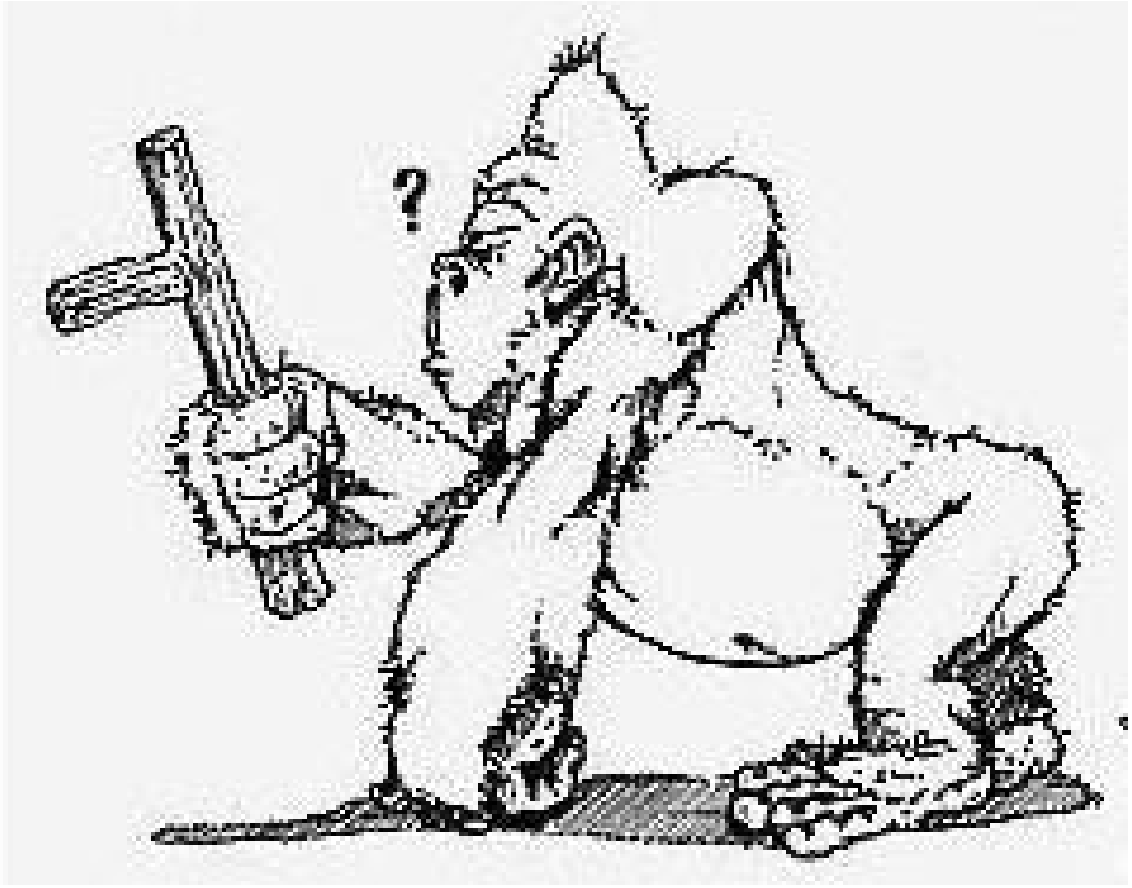


No way

Adding an exploitation point recreates "free space"

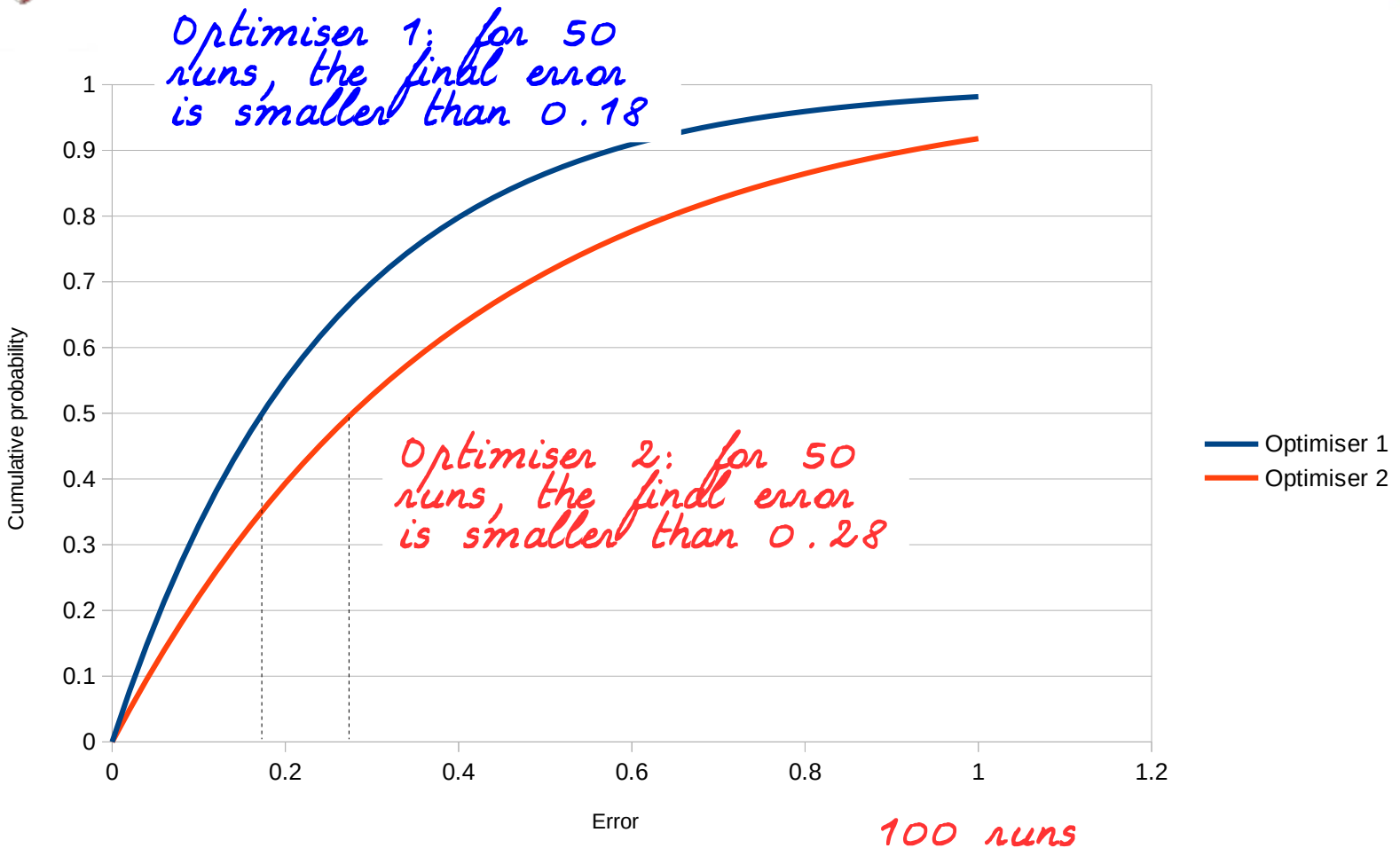
Maurice.Clerc@WriteMe.com

CDF, Qu'es acò ?



Maurice.Clerc@WriteMe.com

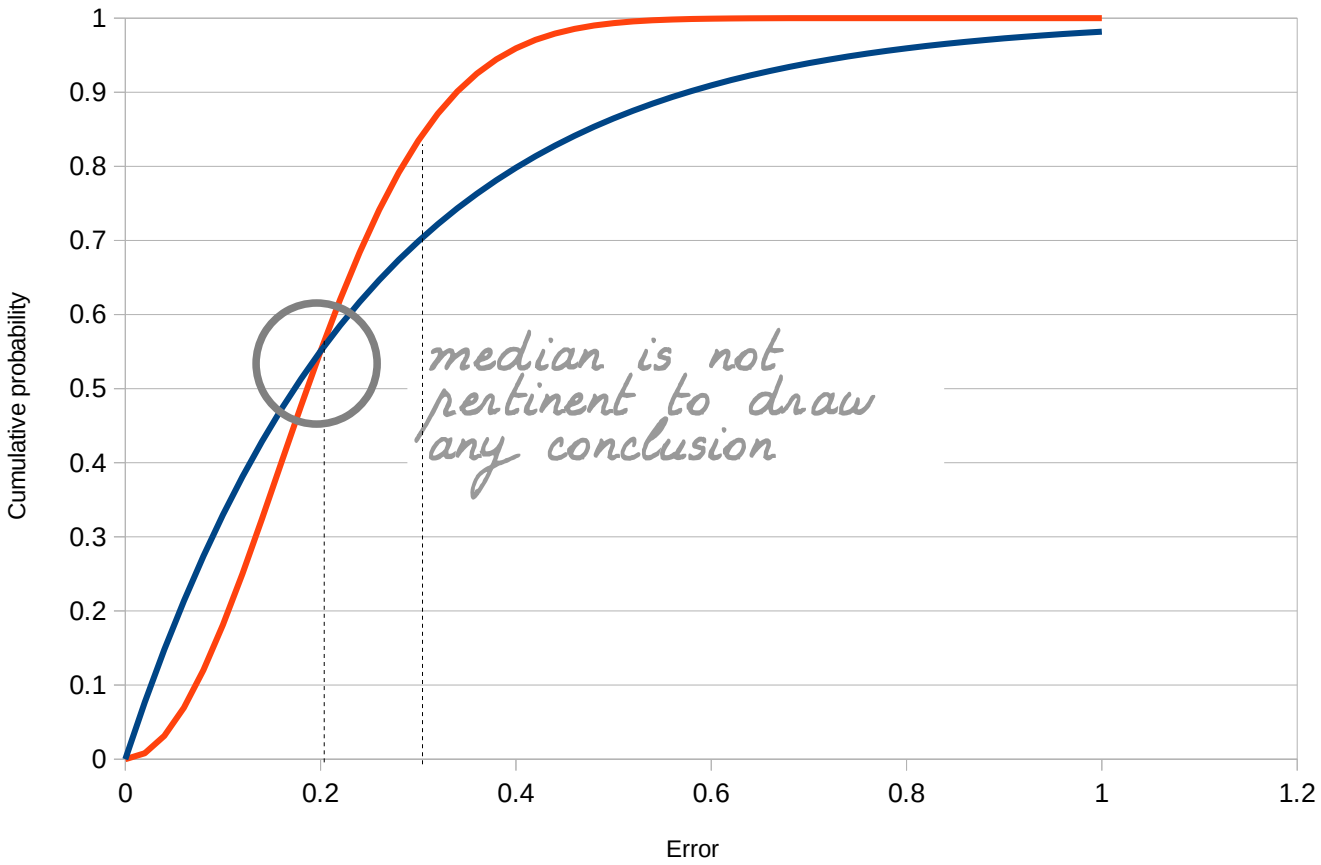
Cumulative Distribution Functions (1)



No doubt, Optimiser 1 is better

Maurice.Clerc@WriteMe.com

Cumulative Distribution Functions (2)

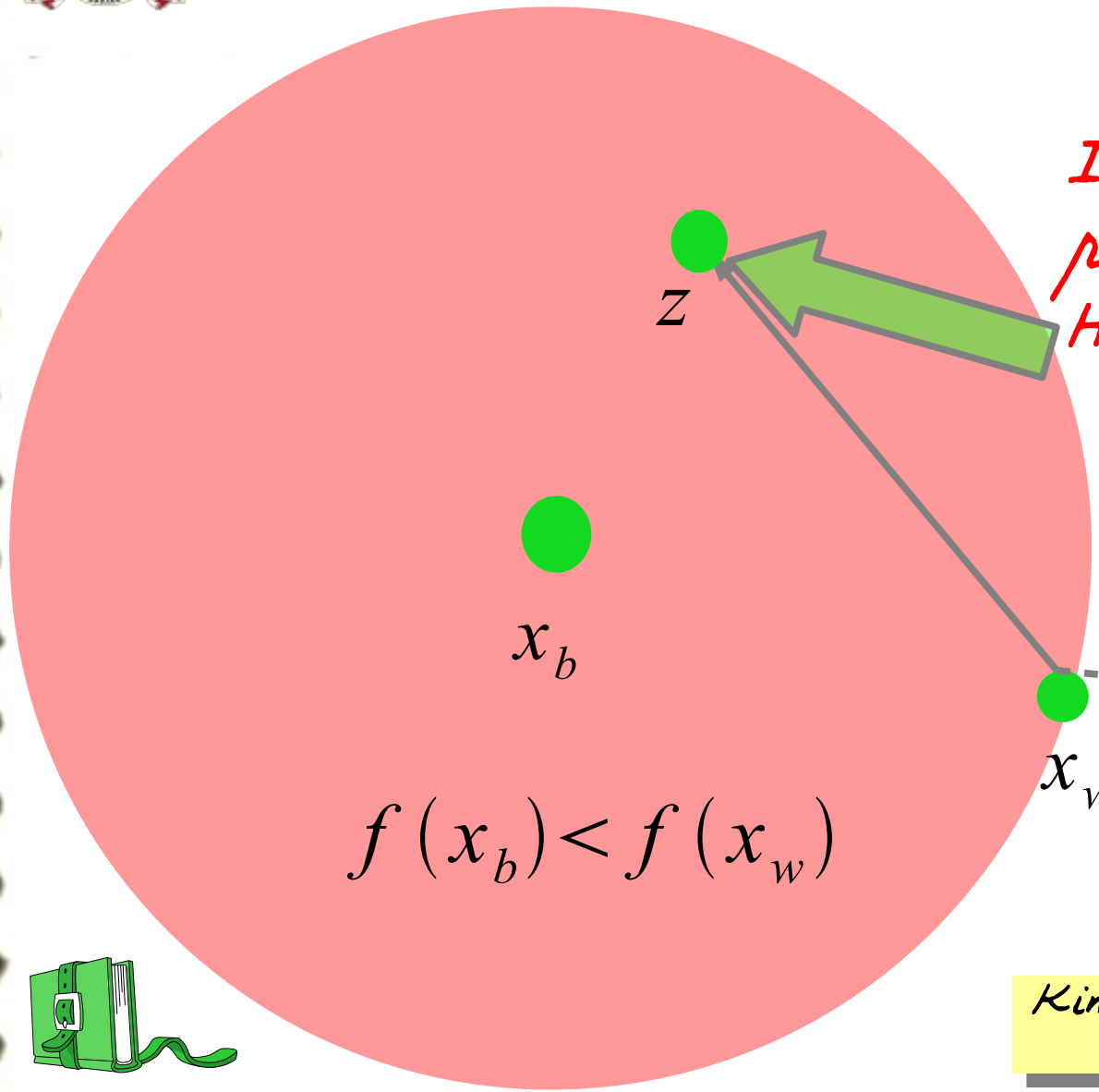


If you want results smaller than 0.2 Optimiser 1 is better (probability 0.55). The contrary if you are less demanding.

back to main slide

Maurice.Clerc@WriteMe.com

Nearer is Better



Improvement is more probable HERE than HERE

Kind of spatial autocorrelation

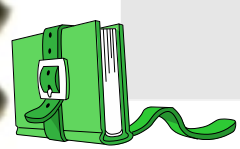
Maurice.Clerc@WriteMe.com

Difficulty measures

CEC 2011

Code	Name	Dimension	Difficulty $\delta^*_{\neg NisB}$
1	Parameter Estimation for Frequency Modulated Sound Waves	6	0.458
2	Lennard-Jones Potential Problem	30	0.973
3	The Bifunctional Catalyst Blend Optimal Control Problem	1	0.046
7	Spread Spectrum Radar Polyphase Code Design	20	0.545

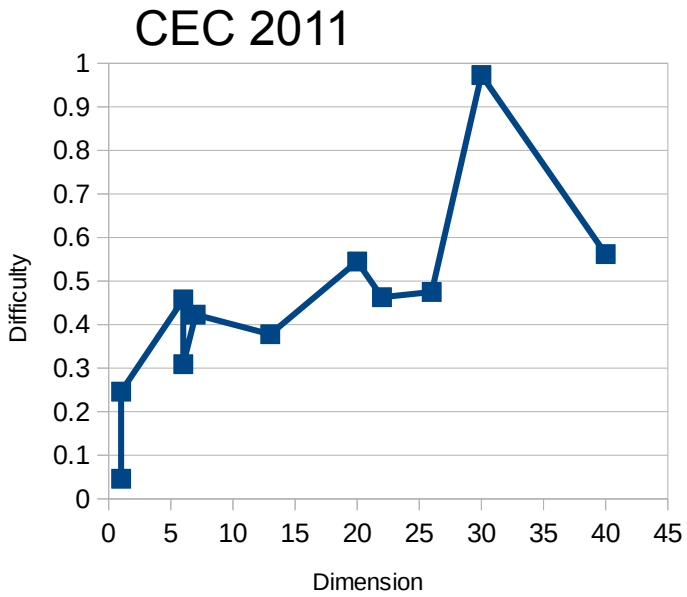
depending on the "Nebrer is Better" probability



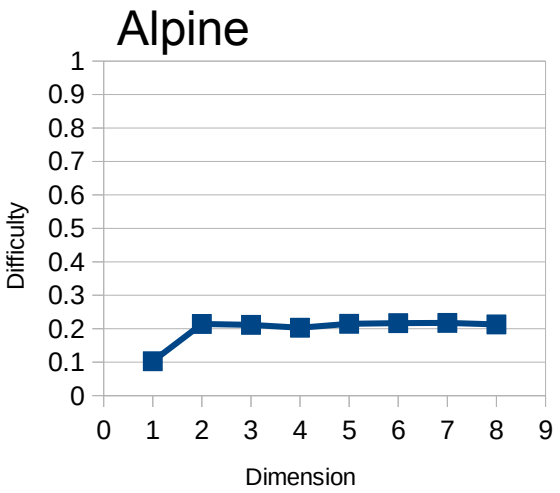
back to main slide

Maurice.Clerc@WriteMe.com

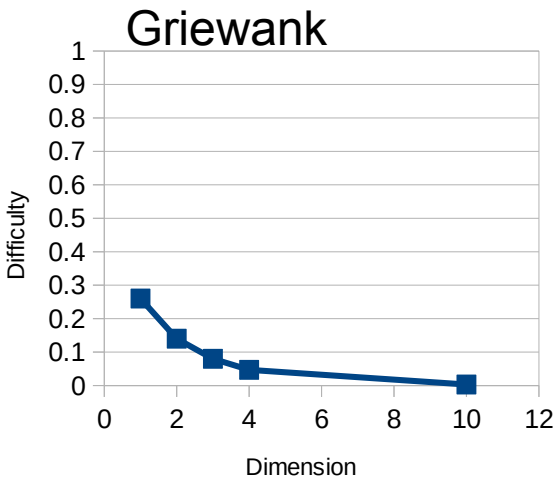
Difficulty vs Dimension



Not linear



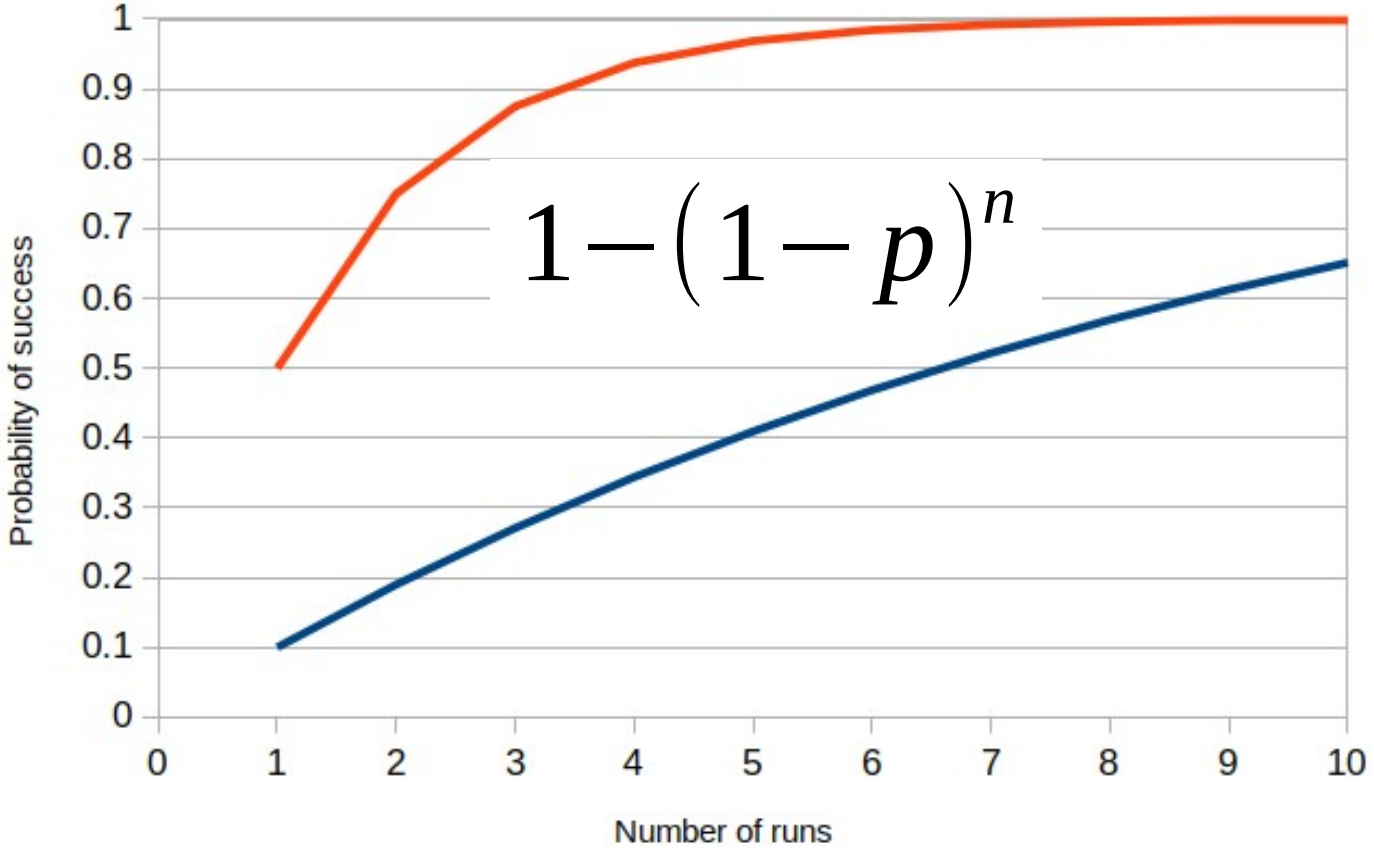
Not increasing



Decreasing

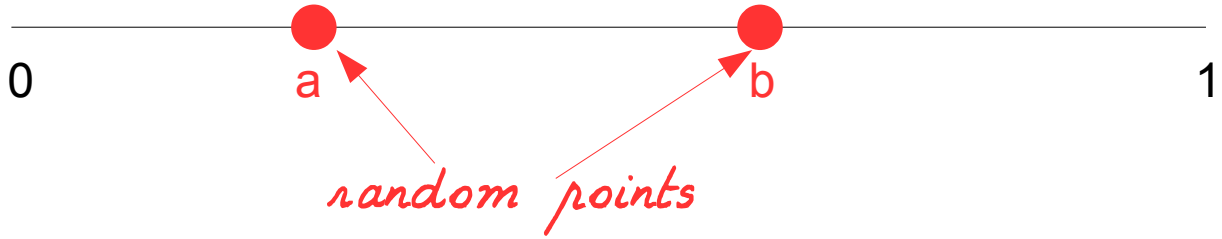
Maurice.Clerc@WriteMe.com

Probability of success vs Number of runs



Maurice.Clerc@WriteMe.com

Stochastic geometry (Warming up) 1/2



$$proba\left(|b-a| \geq \frac{1}{2}\right) = \frac{2\int_{a=0}^{\frac{1}{2}} \int_{b=a+\frac{1}{2}}^{1} (b-a)}{2\int_{a=0}^{1} \int_{b=a}^{1} (b-a)} = \frac{1}{4}$$

← acceptable intervals

← all intervals

Maurice.Clerc@WriteMe.com

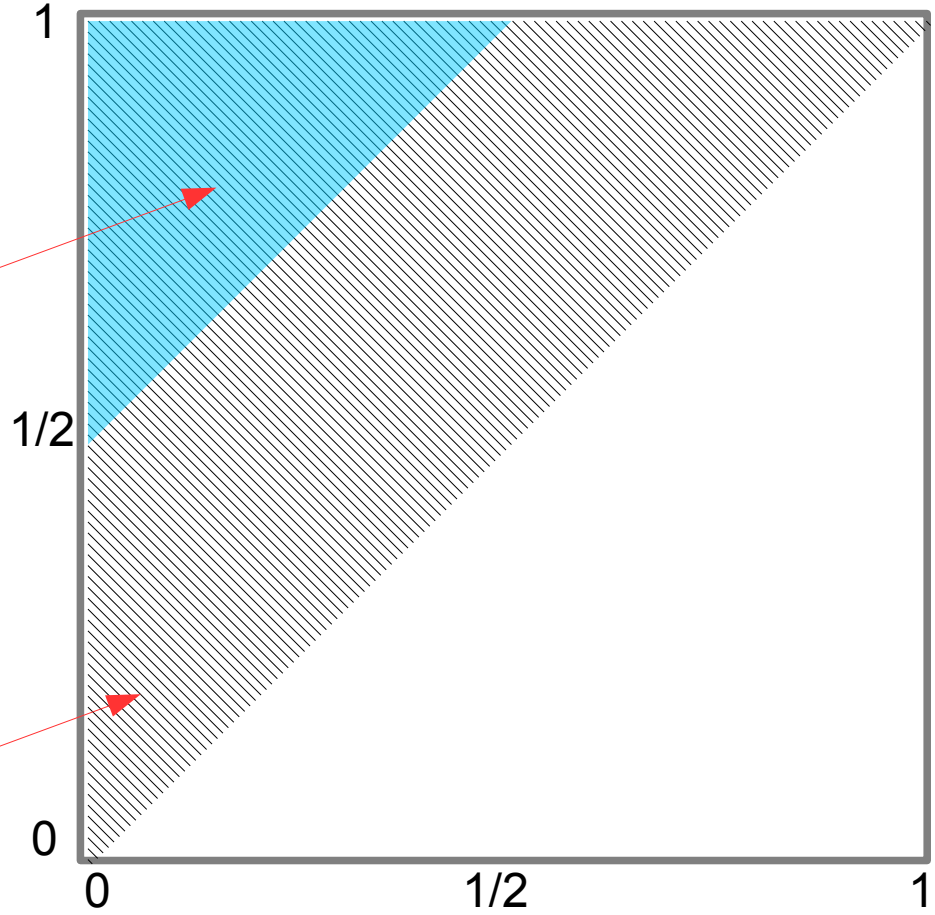
Stochastic geometry (Warming up) 2/2

$$\int_{a=0}^{\frac{1}{2}} \int_{b=a+\frac{1}{2}}^{1} (b-a)$$

acceptable intervals

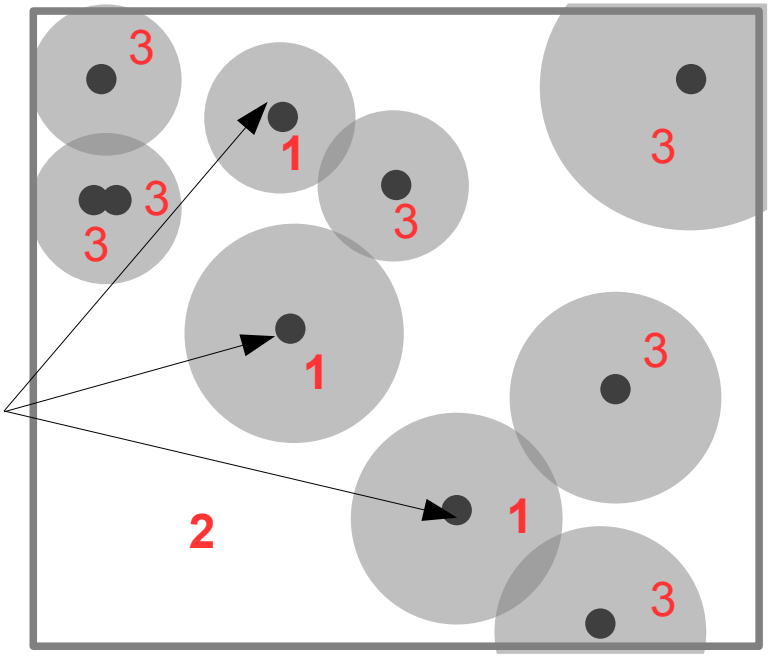
$$\int_{a=0}^{1} \int_{b=a}^{1} (b-a)$$

all intervals



Maurice.Clerc@WriteMe.com

Counting exploitation points (1/2)



Good points

1 = exploitation

2 = exploration

3 = questionable

In practice, many optimisers exploit only around one of the *k* best positions found.

With explicit exploitation, different *k* values modify neither the number of exploitation points nor the balance, but do modify the efficiency.

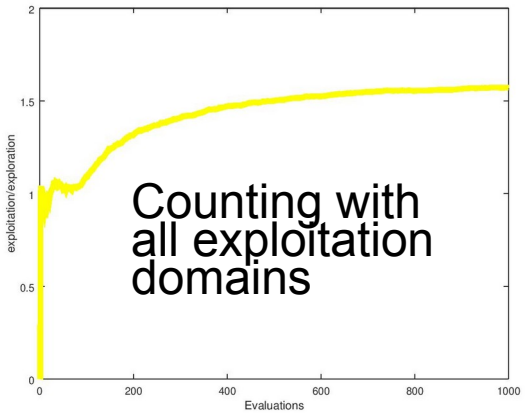
With implicit exploitation, the balance may be modified but not the efficiency (for the number of exploitation points is not used to define the strategy).

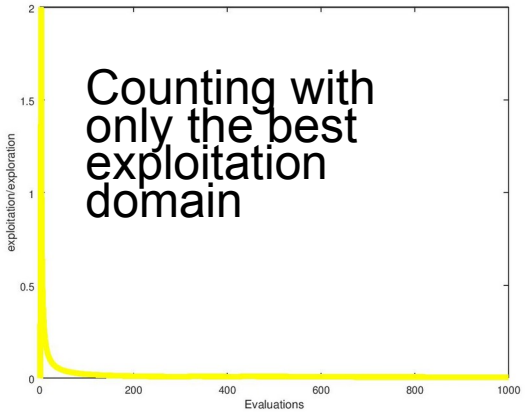
Maurice.Clerc@WriteMe.com

Counting exploitation points (2/2)

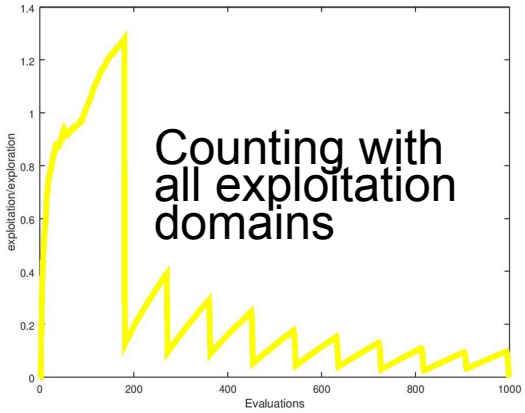
Alpine 2D

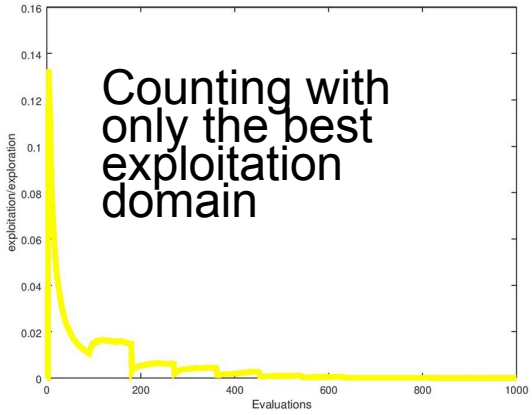
Random search



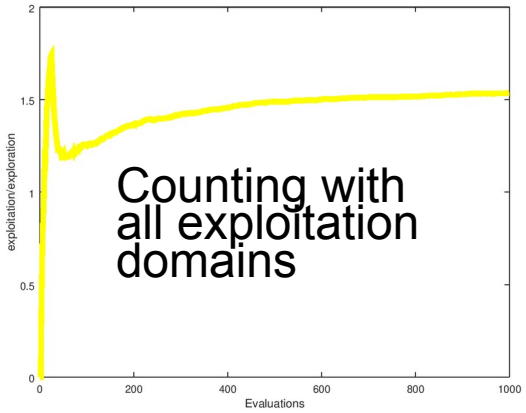


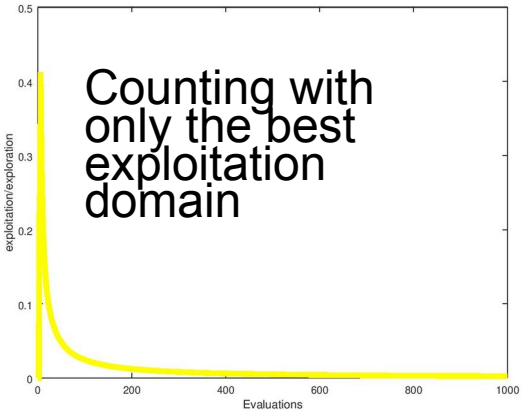
APS



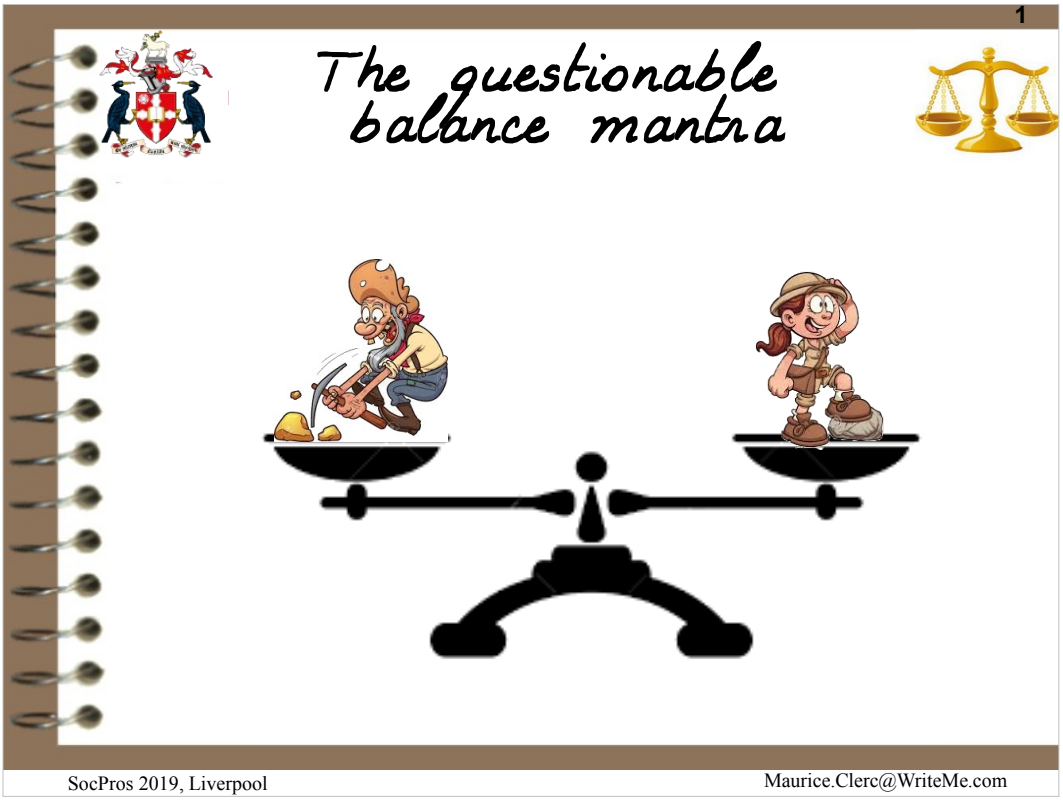


SPSO 2011





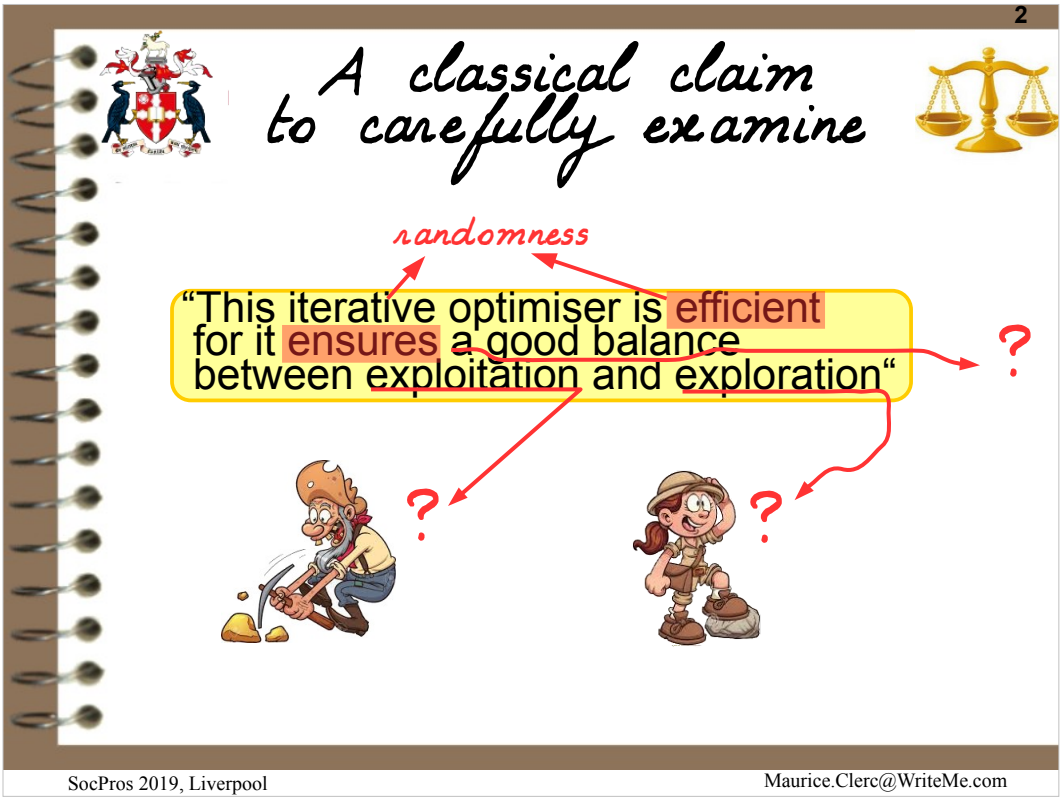
Maurice.Clerc@WriteMe.com



Good morning!
We are here in the context of iterative stochastic optimisers. Such algorithms have to sample points inside the search space, but where?

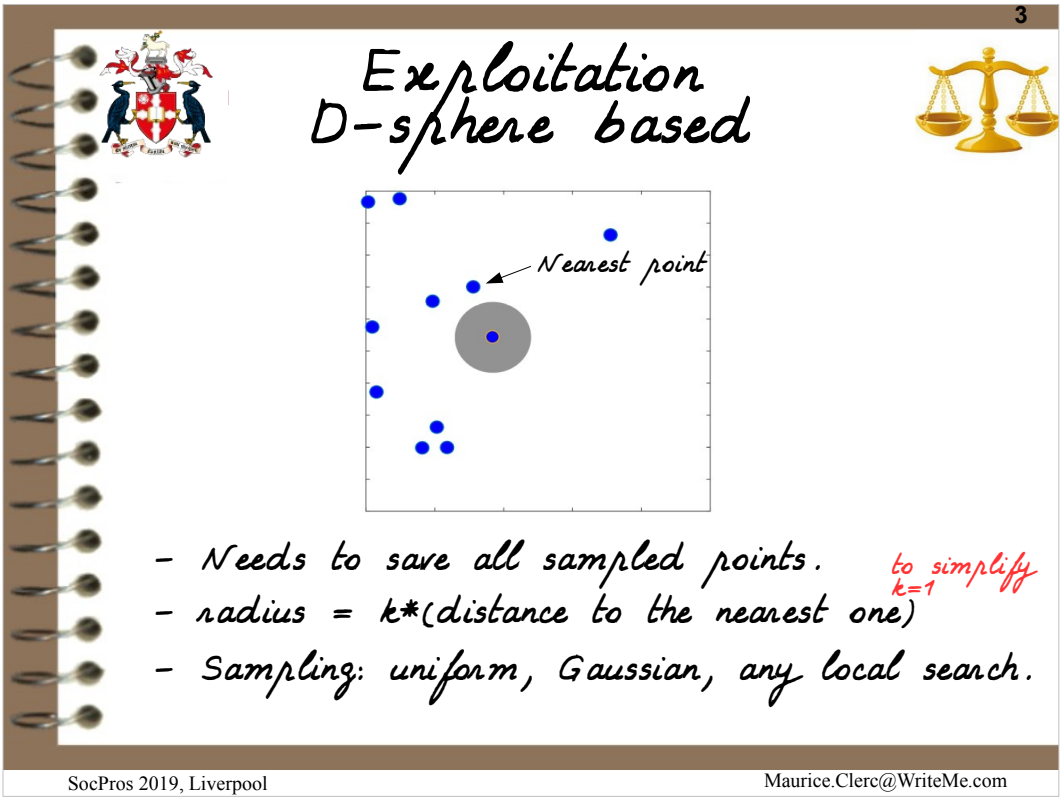
Actually the exploration exploitation trade-off is a dilemma we frequently face in choosing between options. Should you choose what you know and get something close to what you expect (‘exploit’) or choose something you aren’t sure about and possibly learn more (‘explore’)?

This trade-off is sometimes called intensification-diversification, but no matter the name: in many papers about optimisation you can read a claim like THIS ONE.



I call it a mantra for it is almost never supported by clear definitions and convincing proofs, at least experimental ones.
Ensuring efficiency is not the topic of this talk, so I will say just a few words about it.
The point is that comparing two stochastic optimisers can be very tricky, precisely because the use of randomness. Classical tools like medians or p-values can easily lead to wrong conclusions.

Today I would insist on what could be measurable definitions of exploitation, exploration and balance. Actually even the word "good" has to be clarified.

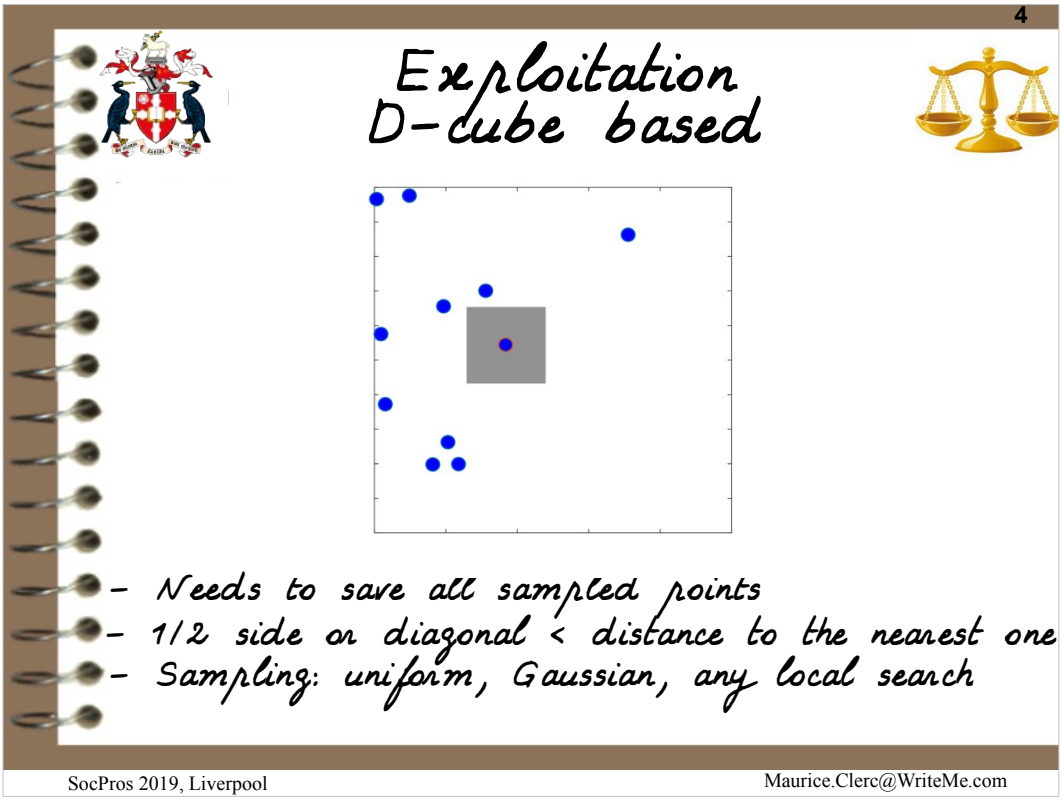


Here we have a two dimensions search space in which 11 points are already sampled.
Now we want to exploit around THIS point.
To precisely define what “around” means, I suggest two methods: by using a DISC (more generally a hypersphere) or a square (more generally a hypercube).
Note that the search space has to be normalised.

The RADIUS of the sphere is given by the distance to the nearest other point. Note that to do that you may have to keep all sampled points. Many optimisers do not.

To avoid any arbitrary parameter, THIS coefficient may be simply set to 1. it doesn’t make much difference, except at the very beginning.

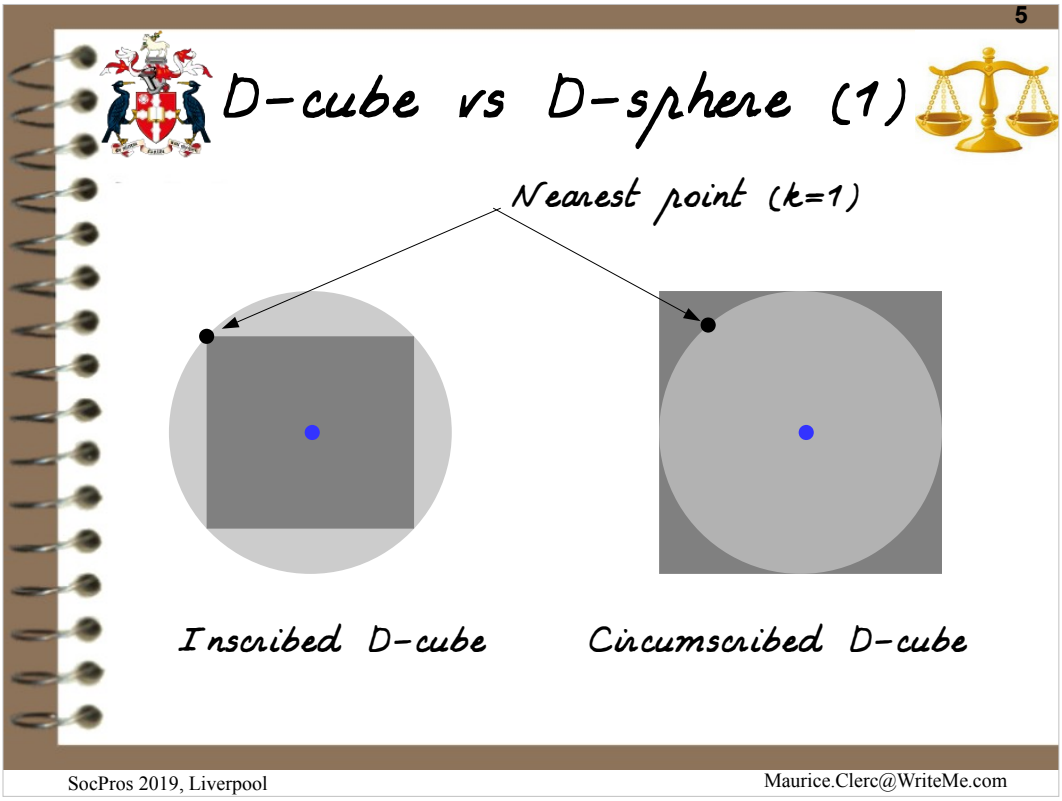
And then a new point is sampled inside this exploitation domain, by using any method.



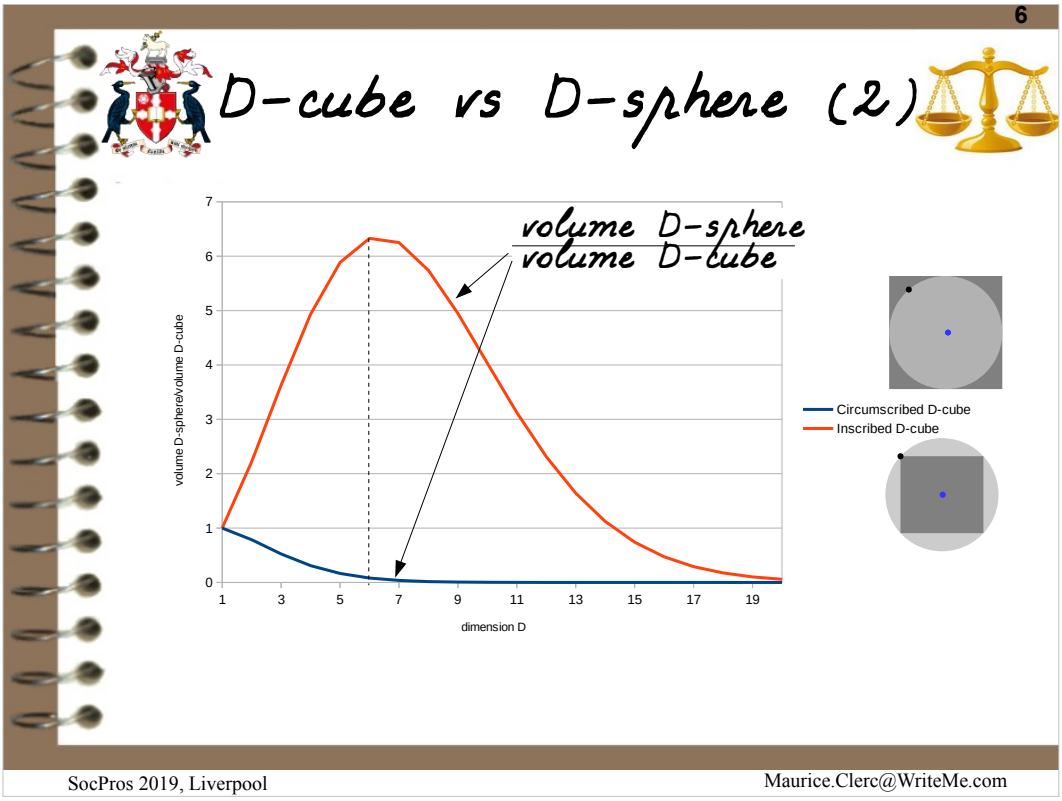
We can of course easily do something similar by using hypercubes.
Again you can use any local search method inside the exploitation domain, for example a greedy one.

There are two main ways to use the distance to the nearest point to define the size of the cube: by using the diagonal or by using the side.
The two approaches are very similar if the dimension is low.

But not any more in high dimension. Let's quickly see why.



Obviously, in dimension 1 there is no difference at all.
In dimension 2, as on this figure, the difference is not that important.
By using the diagonal, the spherical exploitation domain is slightly bigger than the cubical one.
By using the side, this is the contrary.
Or, said differently:
- if you use the distance to the nearest point to define the diagonal, then the sphere contains the cube
- if you use it to define the side, the cube contains the sphere.
But the point is that the evolution of the ratio between the two volumes is not the same when the dimension increases.

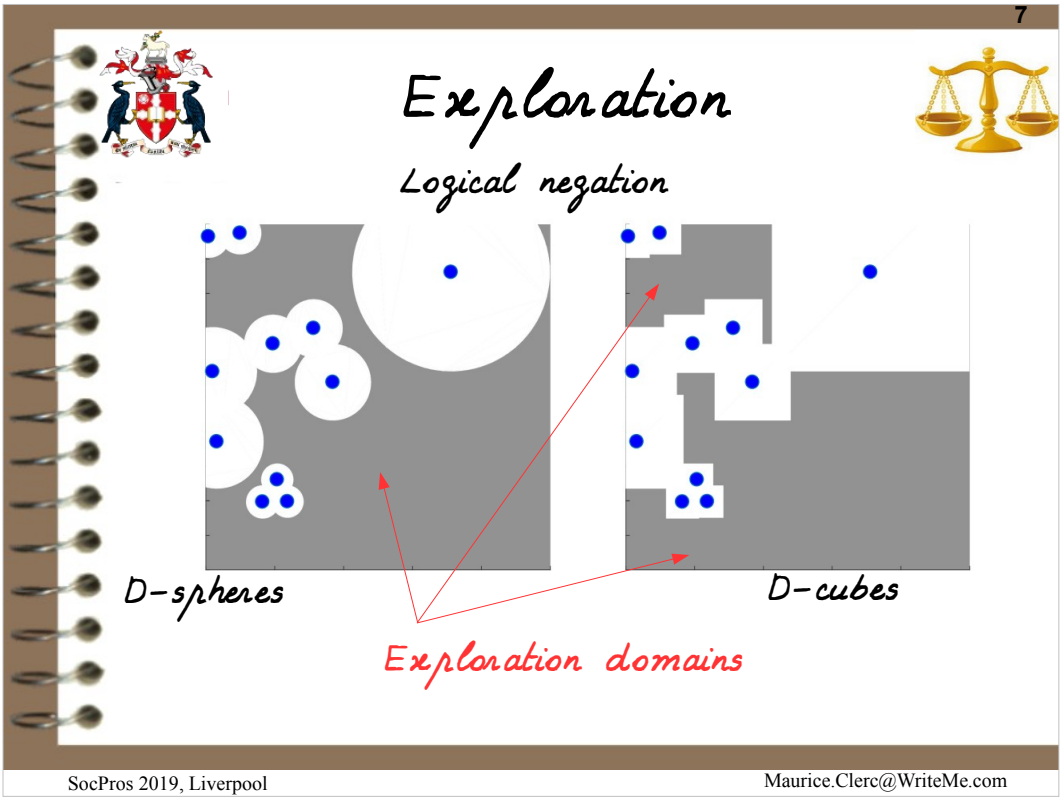


This is of course a quite boring technical point, and I do not insist on it.
In one case the evolution is not monotonic, and dimensions around 5 are critical.

Therefore, if you define a brand new algorithm with explicit exploitation just be careful. What kind of exploitation you use, and for what kind of problems.

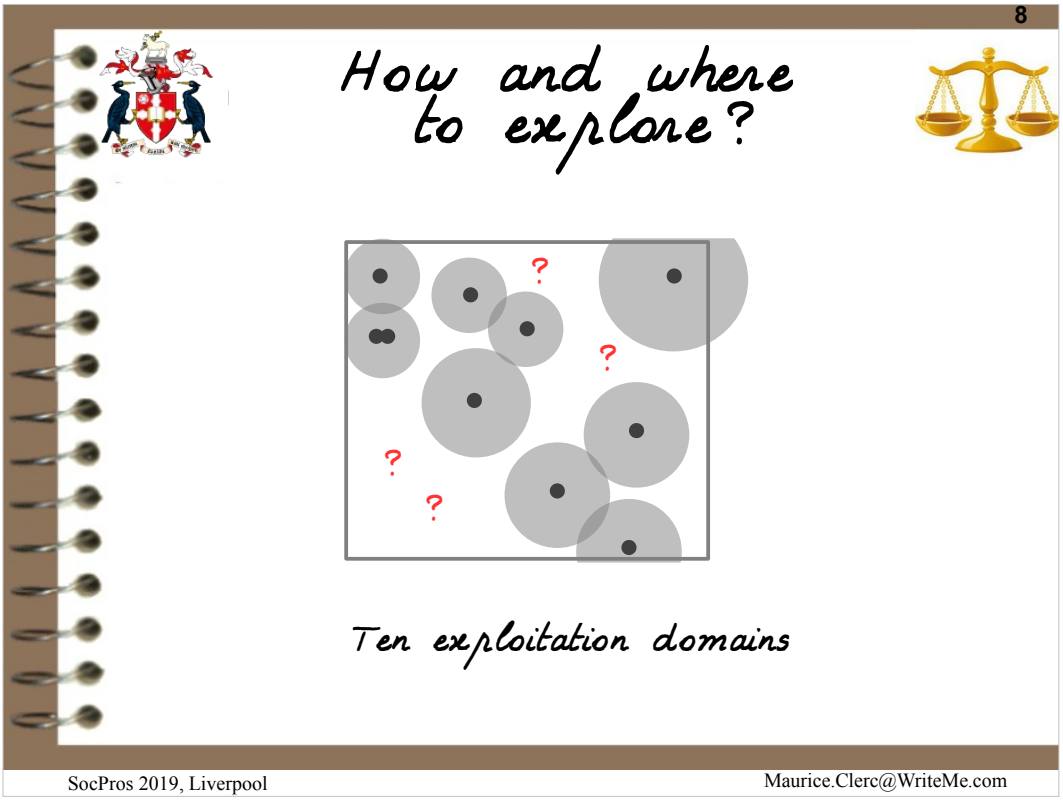
Note that in practice one doesn't consider all possible exploitation domains. It doesn't modify the conclusions we will see, but in case you are interested, I added a few extra-slides at the end of this presentation.

Now, what about exploration?



As soon as exploitation is defined, defining then exploration is theoretically very easy: just the logical negation.
Either by using spherical exploitation domains, or cubical ones.

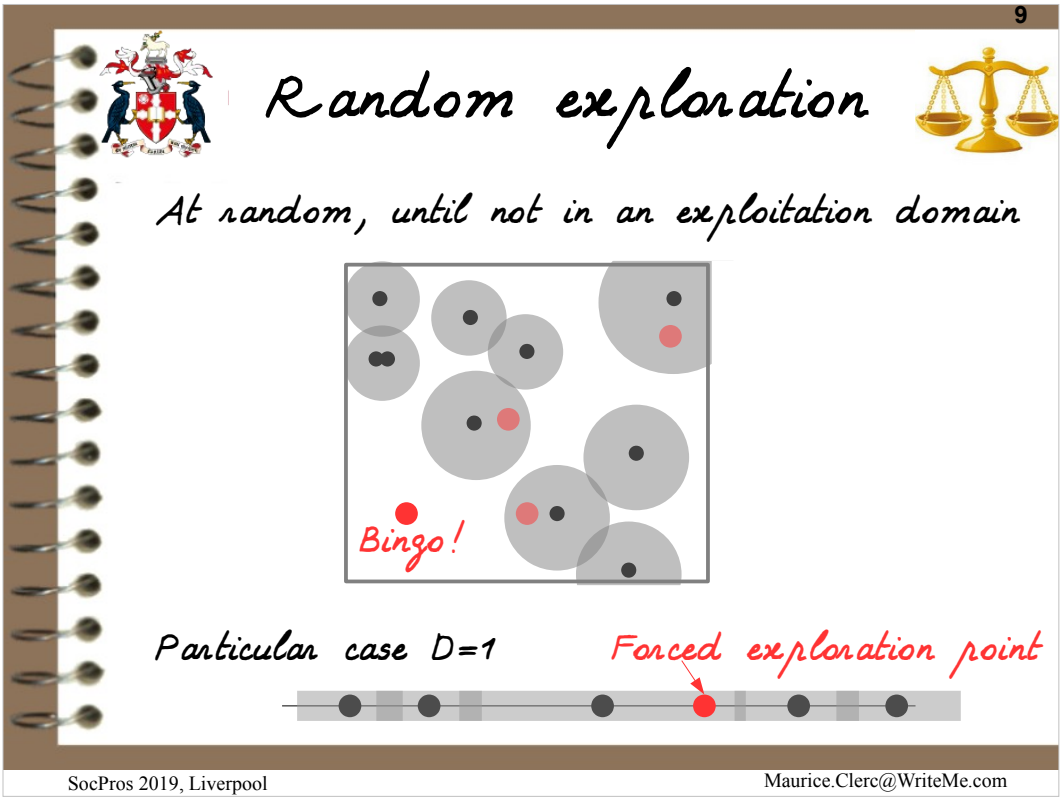
I said “theoretically very easy”, but what about in practice?



Here you have some exploitation domains. How to sample a point that is not in any of them?

Think about it for a second. How would YOU do that?

Let's consider here just two methods:
- at random
- or, more sophisticated, by using a No Man's Land search

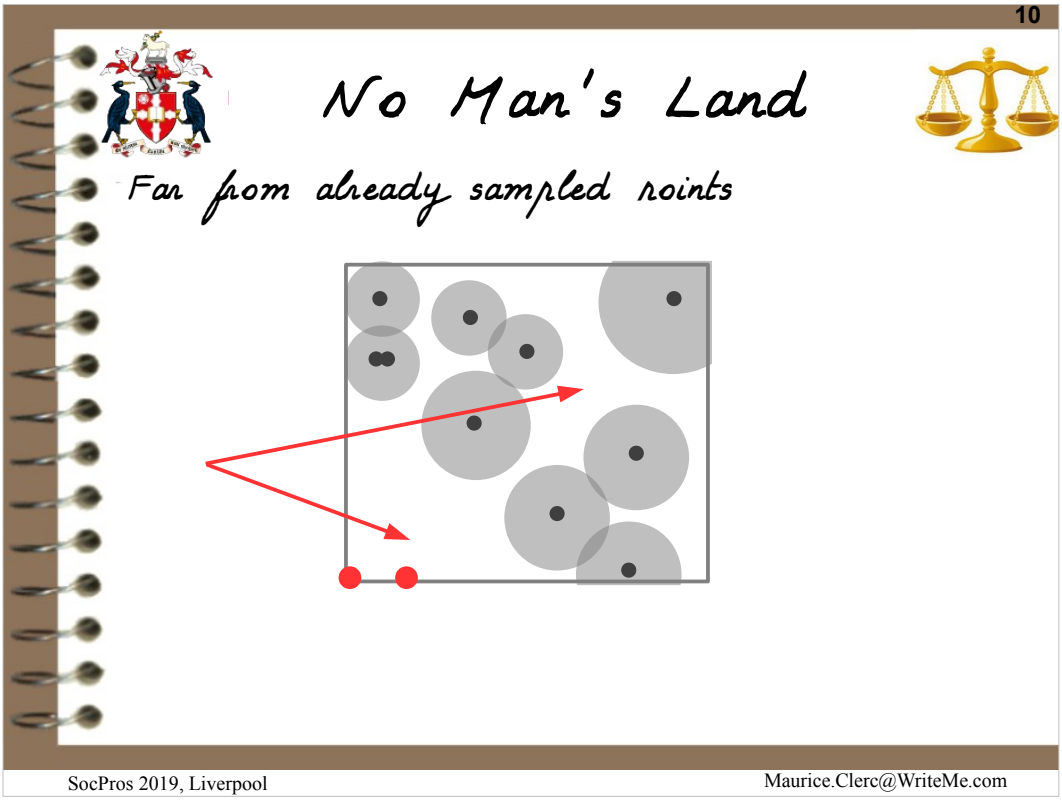


The random method is simply a loop:
- sample at random, according to an uniform distribution
- check if the point is in an exploitation domain
- if so, repeat.

In low dimension it can take a long time, for the exploitation domains can cover a large part of the search space.

Actually, it may be impossible to find a real exploration point, like HERE in dimension 1.

In that case you have to cheat a bit, for example by selecting the middle of the largest interval between two sampled points.

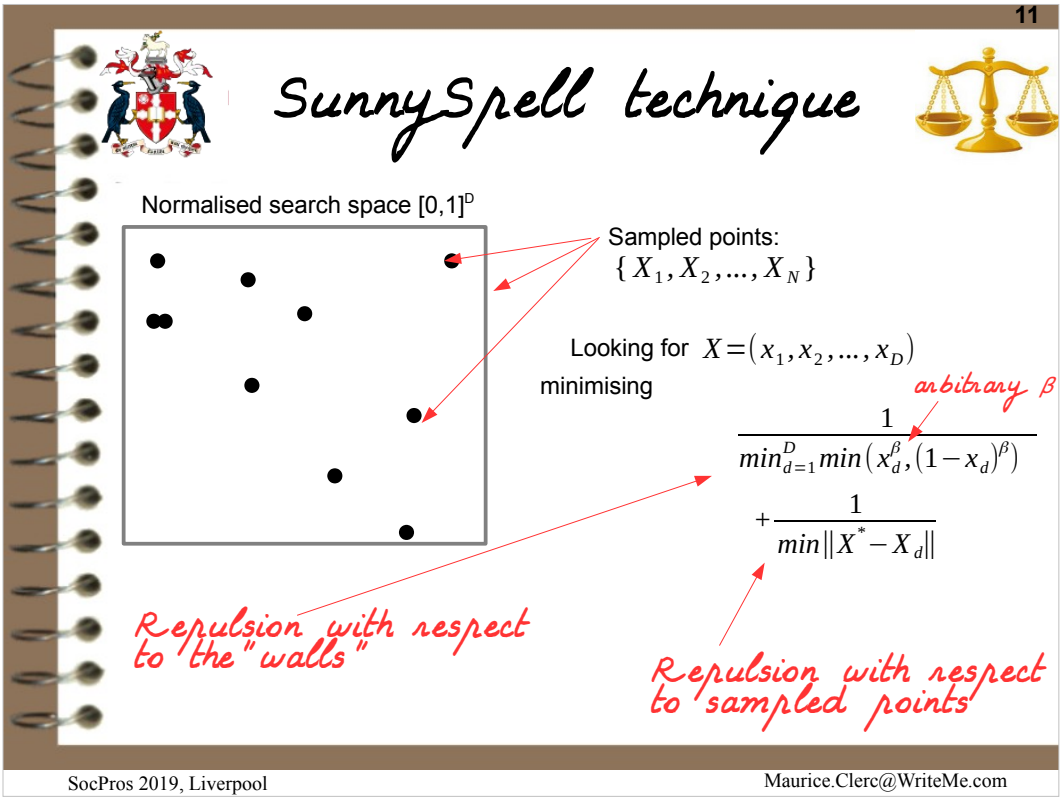


Intuitively it seems interesting to sample as far as possible of already sampled points.

However, if you do that, the new point may be too often just a corner of the search space or on a side.

So you have to use a trick to avoid this phenomenon.

But let's see first more precisely how such a method can be formalised.

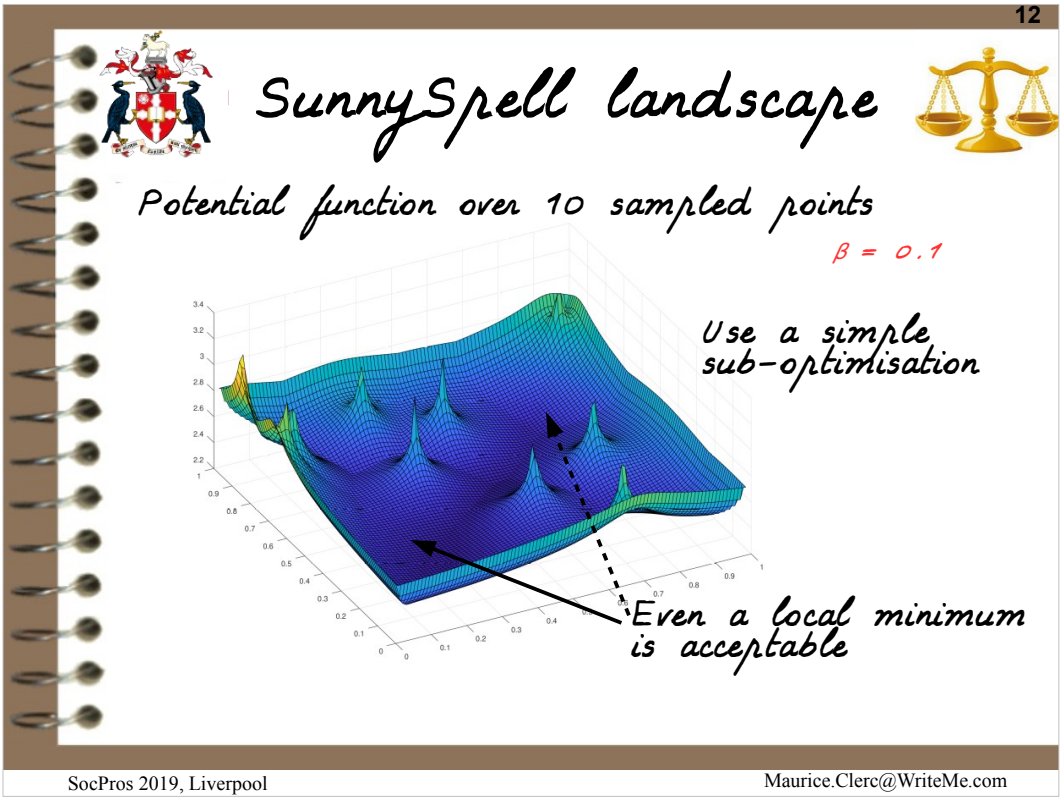


Of course the formulae given here are quite arbitrary. You can easily find other ones, for example by analogy with electrical potentials.

But the point is that using more “realistic” potentials, if I dare say, is not necessarily a good idea, for it can be very difficult to find the minimum. Think, for example, at the famous Lennard-Jones problem, in which you have to minimise the total energy of a cluster of atoms.

Note that the parameter beta is here just to increase the height of the peaks in the landscape of the sub-problem.

Let’s see how this landscape looks like.

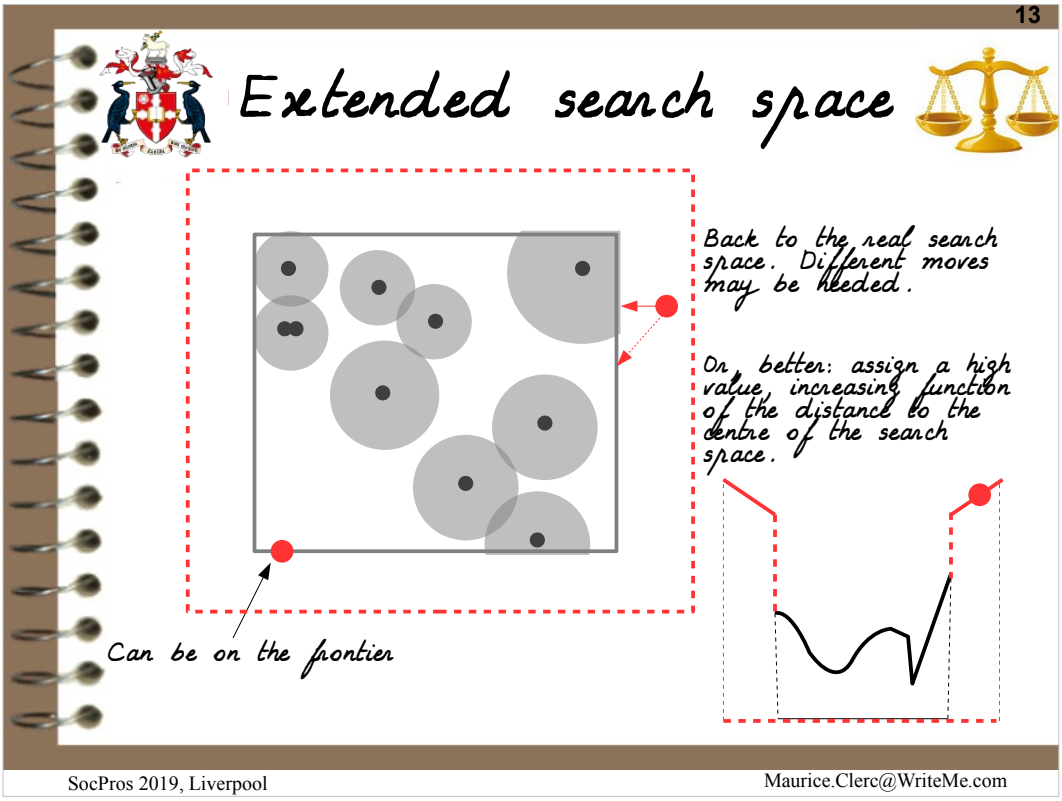


Here is a typical landscape for a two dimensions problem. By construction, the peaks are located on the known positions, for they are the places to avoid.

A point near to the global minimum or even near to a local minimum will be an acceptable exploration point.

So the sub-optimisation can perfectly be a simple one, with just a few iterations.
Actually you even can use a recursive method.

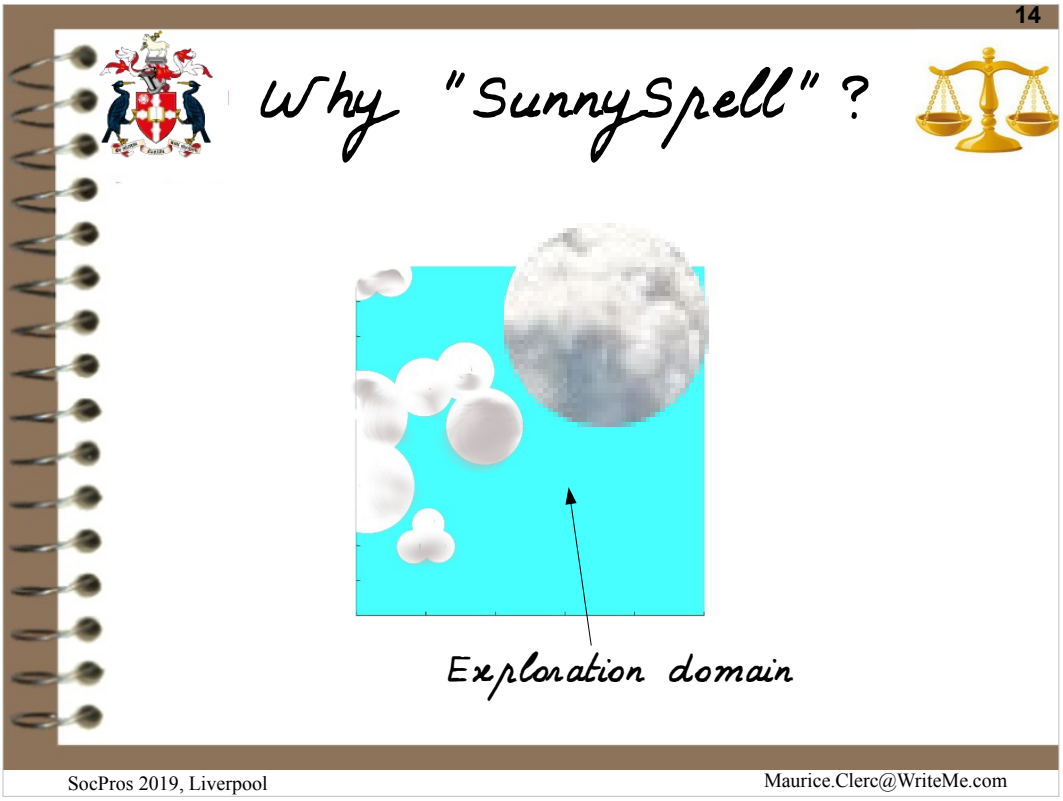
Now, what about the trick I mentioned a few minutes ago?



If you apply the SunnySpell method, or a similar one based on repulsive potentials on the real search space, you may never sample a point on the frontier .
For some problems it may be a drawback for, precisely, the solution point may be on this frontier.

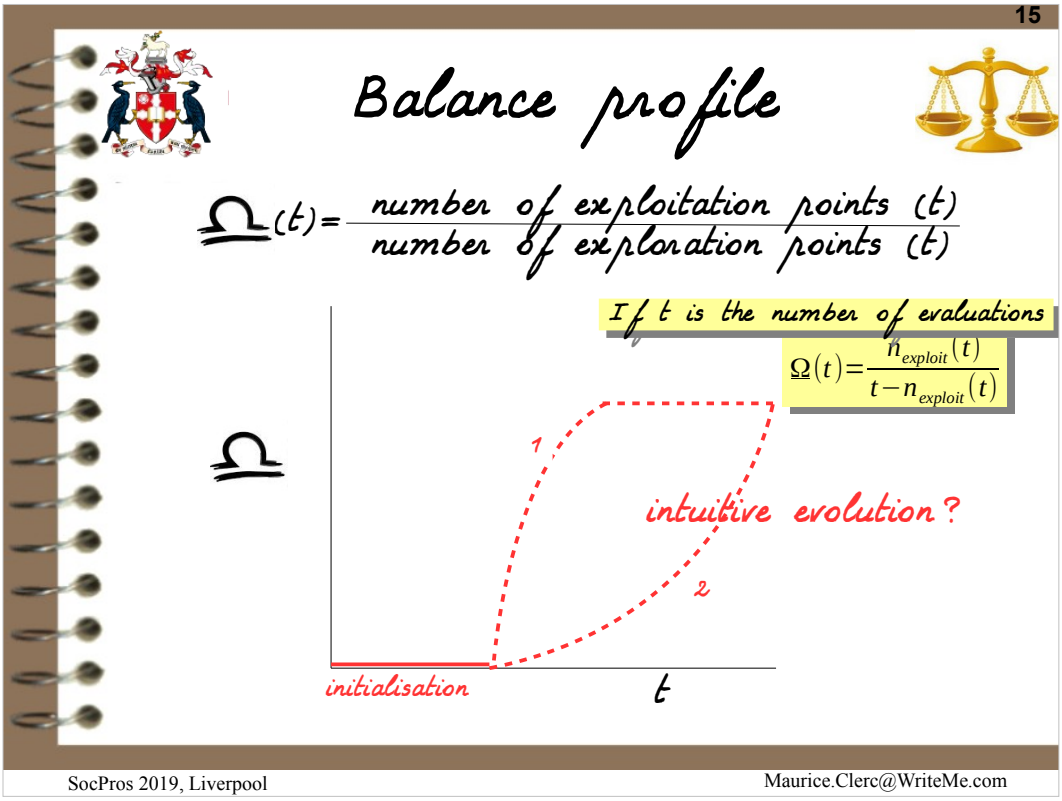
A simple workaround is to use a slightly extended search space. And if a new position is sampled outside the real one, you can assign an artificial high value, or move the point.

If you move the point, you have to check if the modified position is not in an exploitation domain. Because of the shape of the potential landscape, it very rarely happens, probably never, but, if so, you have to try another moving method, in practice by using a bit of randomness.



Just for fun. Why this name “SunnySpell”?
If you imagine the exploitation domains are clouds, then the exploration domain is the remaining clear sky.

OK, now we have precise definitions of exploitation and exploration, we can easily define the concept of balance.



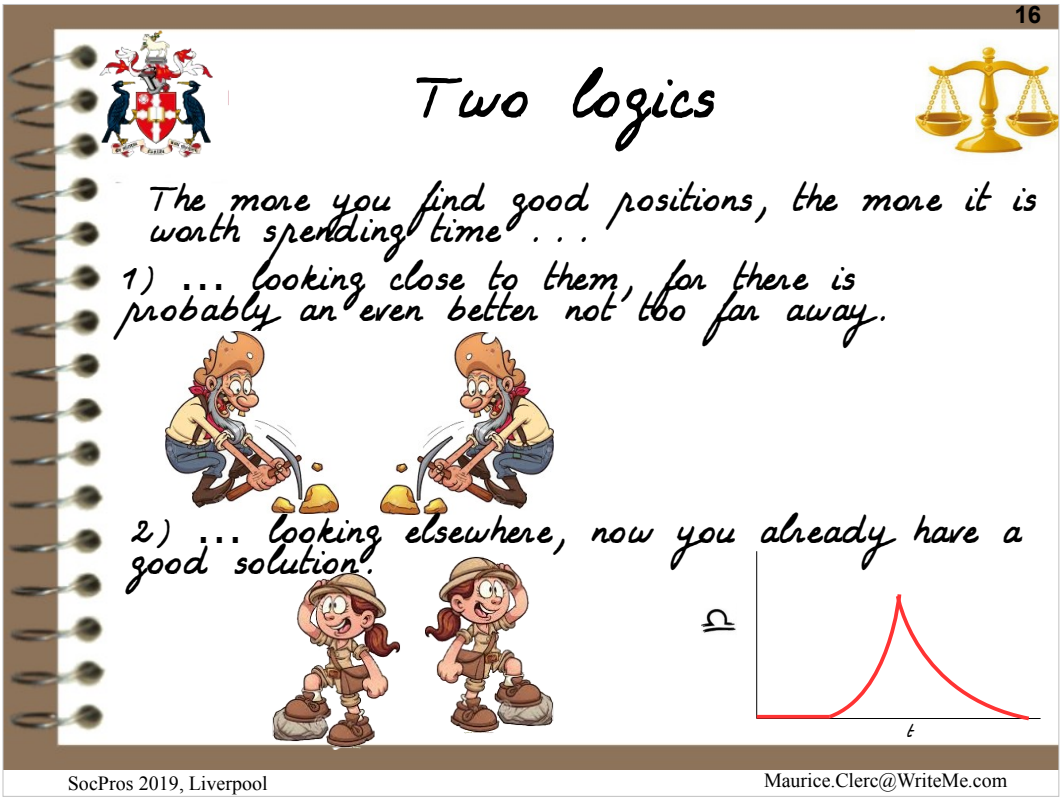
We can look at the evolution of THIS ratio during a run. Intuitively, during the initialisation phase of a population-based algorithm, there is no exploitation point.

Note that, if the initialisation is at random, this may be not completely true. Particularly in low dimension, a random point can be inside the exploitation domain of a previous sampled one.

Let’s try to guess what could be a typical profile.

A first reasoning is to say that the number of exploitation points is more or less the same at each iteration. Hence the curve 1, if the population size is constant.

Another reasoning is to say that there is more and more exploitation at each iteration. Hence the curve 2.

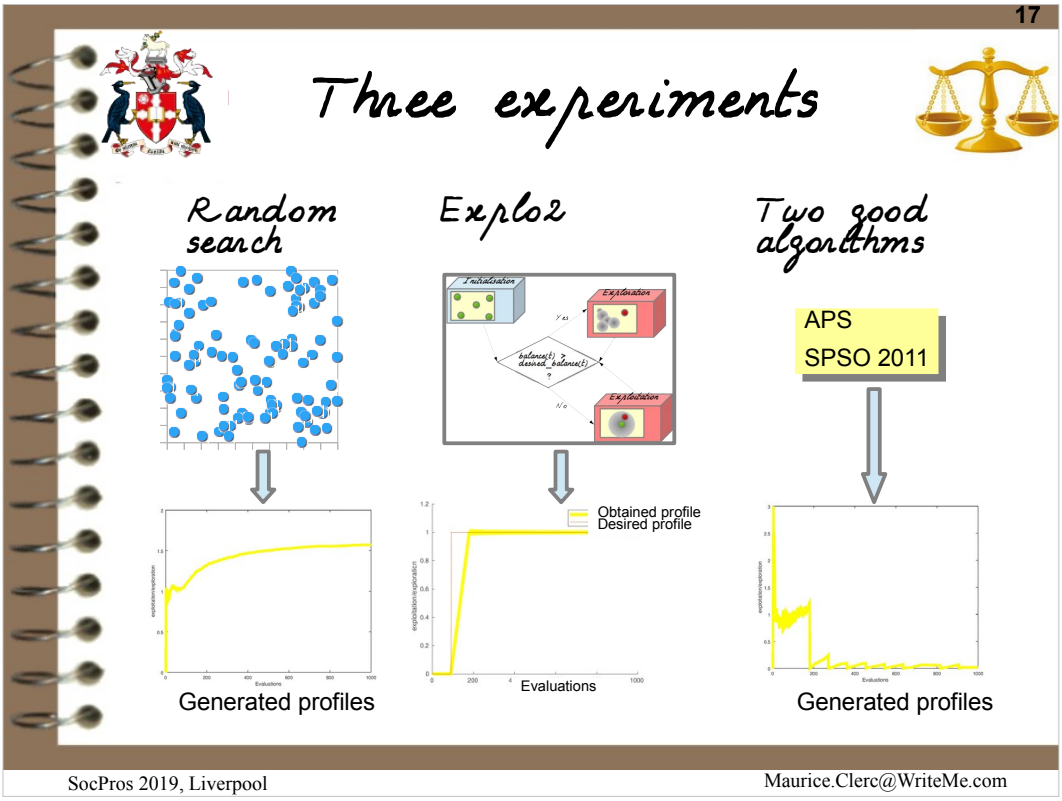


A variant of this second reasoning can be reworded like THIS.

However, under the same condition, that is more and more exploitation points, another logic can be considered.

After all, if you already found good points, you may say “OK, now I have something acceptable, why not looking completely elsewhere? “

Of course, it depends on your budget, in practice the maximum number of evaluations. If you did not spend it entirely, the second logic is tempting.
In that case the profile may be something like THAT.

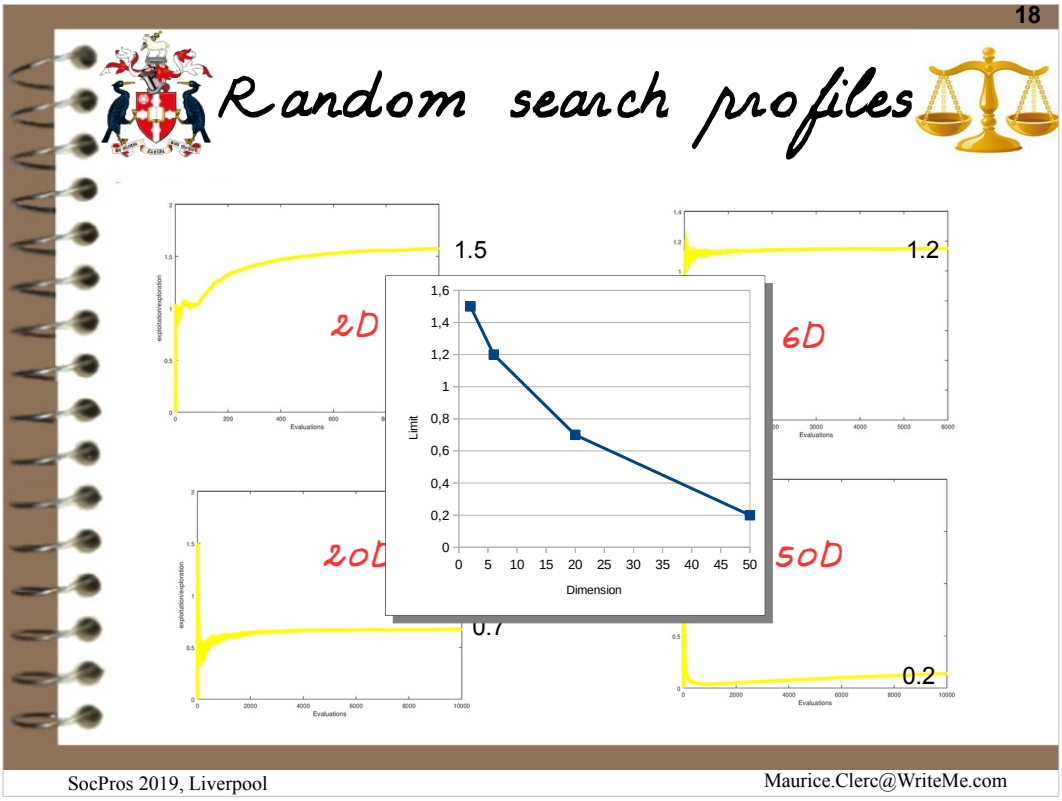


To investigate the relationship between profile and efficiency I performed three kinds of experiments.

The first one is to consider the pure random search. Actually it was just to test my code, for in that case the curve can be mathematically found.

The second one is to design a simple algorithm that has to follow a predefined profile. And to try different profiles, of course.

The third one is to add a profile observer to some good algorithms, to plot the generated profiles, and to try to understand them.

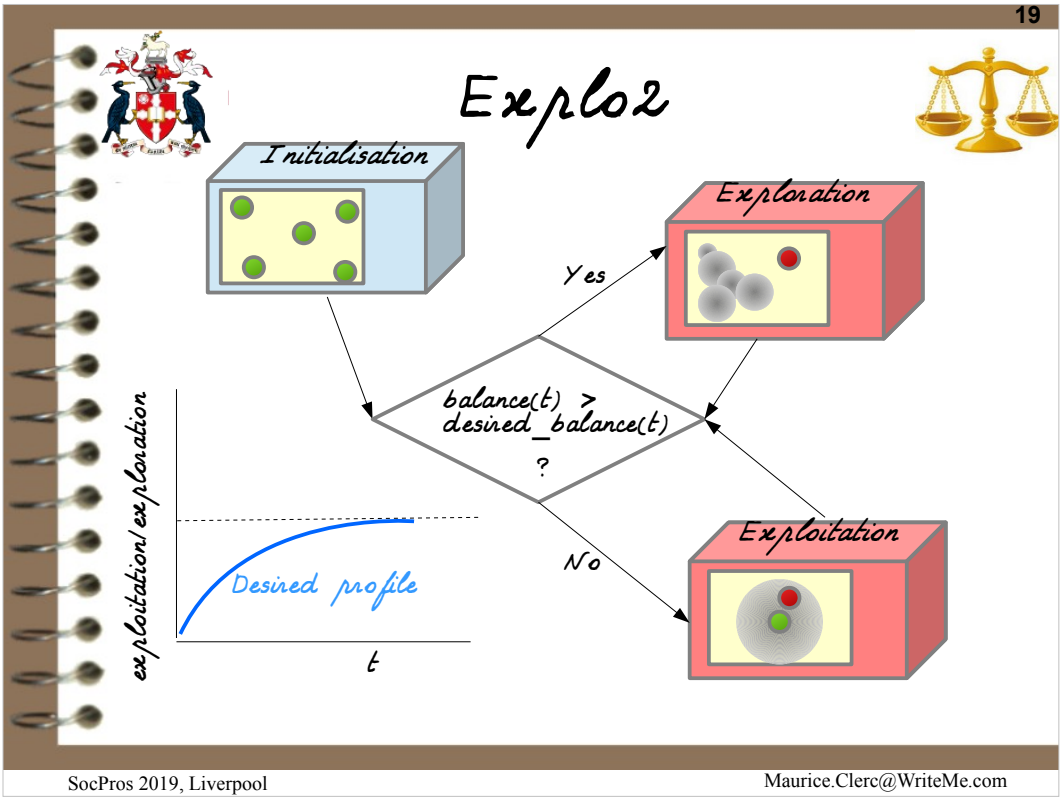


For the random search, the generated profile is of course not depending on the landscape of the problem, only on its dimension.

As you can see, the profile tends to a constant ratio. Actually, this could be proved in the context of stochastic geometry. And the limit is a DECREASING function of the dimension.

More generally, algorithms that make intensive use of randomness usually generate similar profiles.

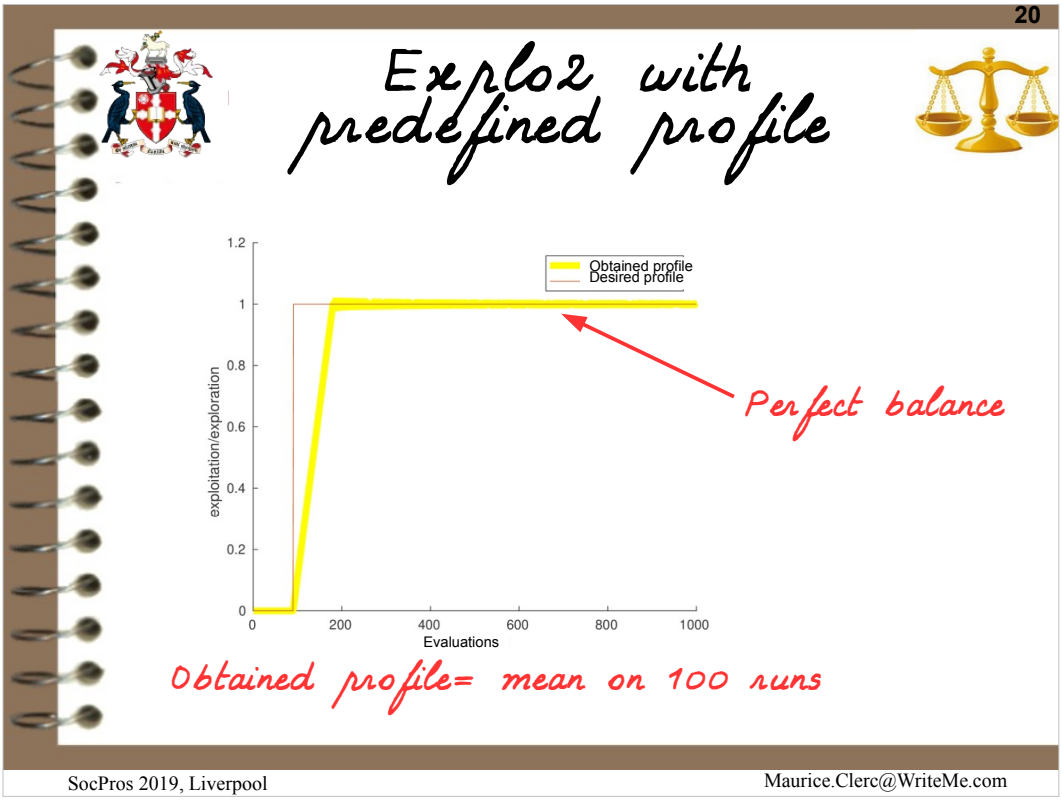
Please, keep in mind this kind of shape, for further comparisons with some sophisticated algorithms.



As said, the second set of experiments is based on an ad hoc optimiser, called Explo 2.

You give it a profile, and it just tries to conform to it.
Each time it has to sample a new point, the question is “Should I exploit, or should I explore”.
The decision is only depending on the comparison between the current generated profile and the predefined one.

Note that in this algorithm “to exploit” means “sampling inside the exploitation domain of the best current position”.

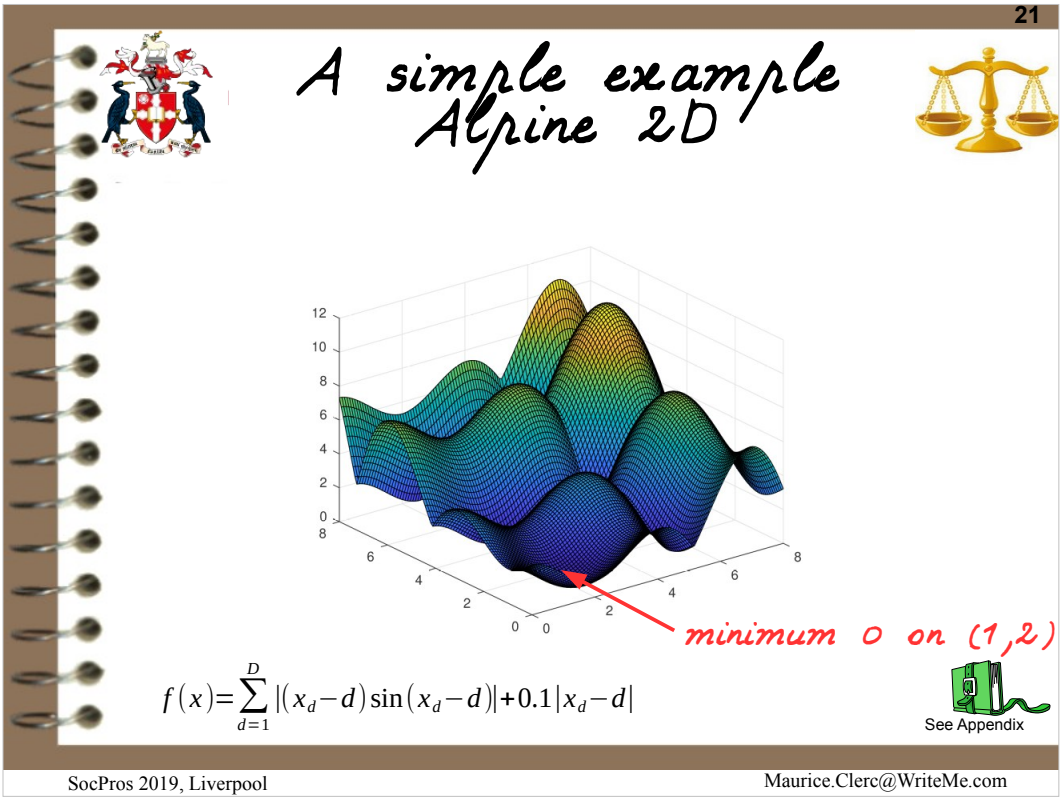


So, an output of the algorithm is of course the best position found and its value, but another one is a plot like THIS one.

Here the user wanted a perfect balance equal to 1 after initialisation.

The algorithm could not exactly do that, because of the big DISCONTINUITY, but it very quickly succeeds to reach the value 1 and to maintain it.

Now we can try to compare the influence of several predefined profiles, first on a simple example.

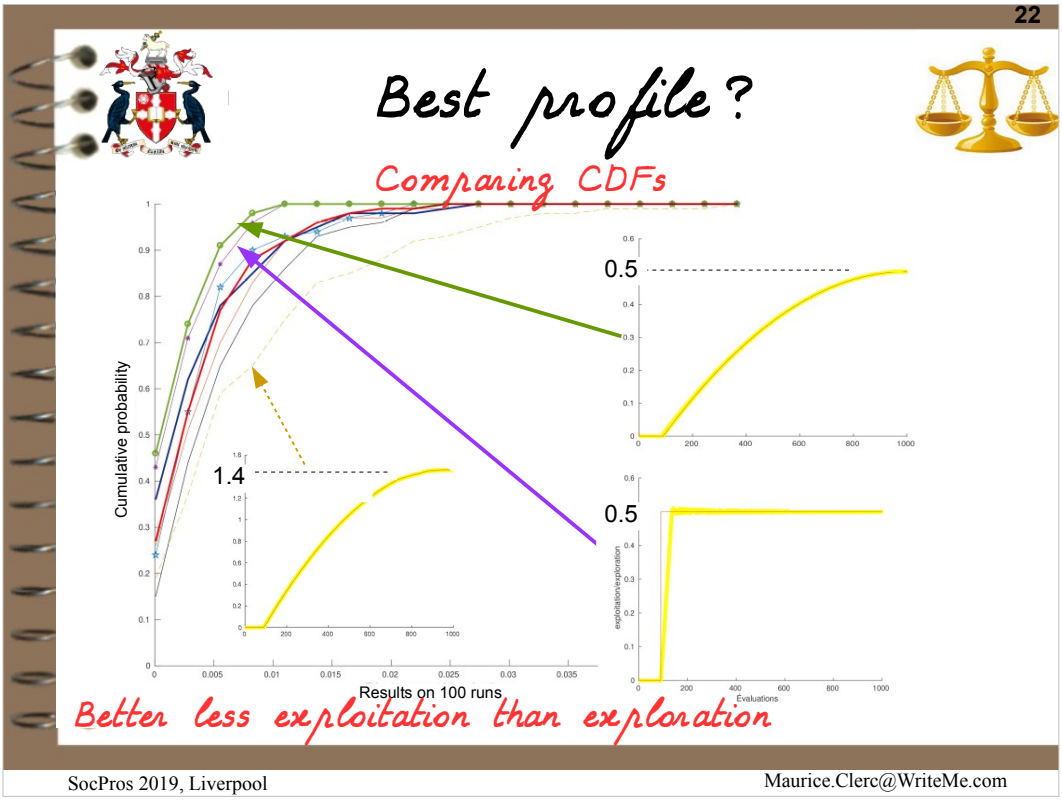


This problem is multimodal but separable. At least, the solution point is not on a particular position, contrarily to some other test problems for which it is on the centre of the search space, which is not acceptable for fair comparisons.

I tried many predefined profiles and to compare the results I used the CDF method, as the classical ones with mean, median, and p-values may be not discriminant.

CDF, hmm, do I see a little perplexity on some faces?

I think we have time to quickly look at the appendix, and then back.



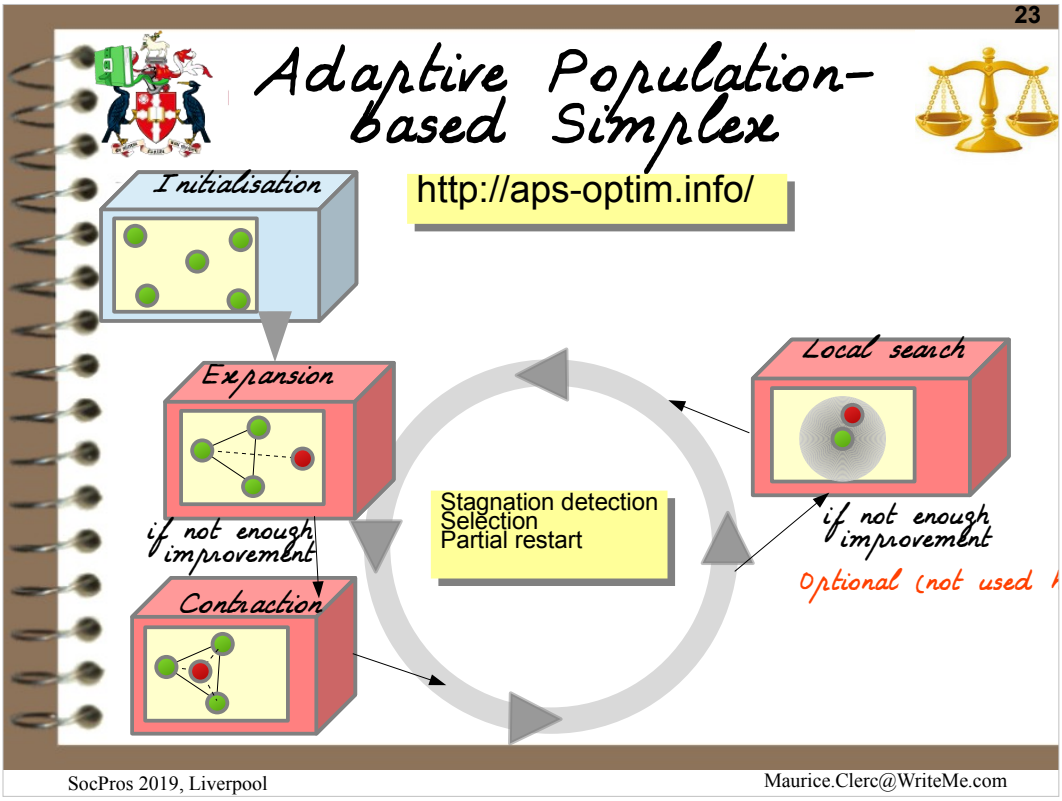
So, on our Alpine problem with Explo 2, I tried some profiles, and plotted the resulting CDFs for 1000 evaluations.

All profiles tend to a limit, like for the random search, but for the two best ones the limit is 0.5, as for the worst the limit is greater than 1.

One of the CDF is for a perfect balance, with a limit equal to 1, and it is not specially good.

So it is tempting to conclude that a ratio around 0.5 is a good choice.

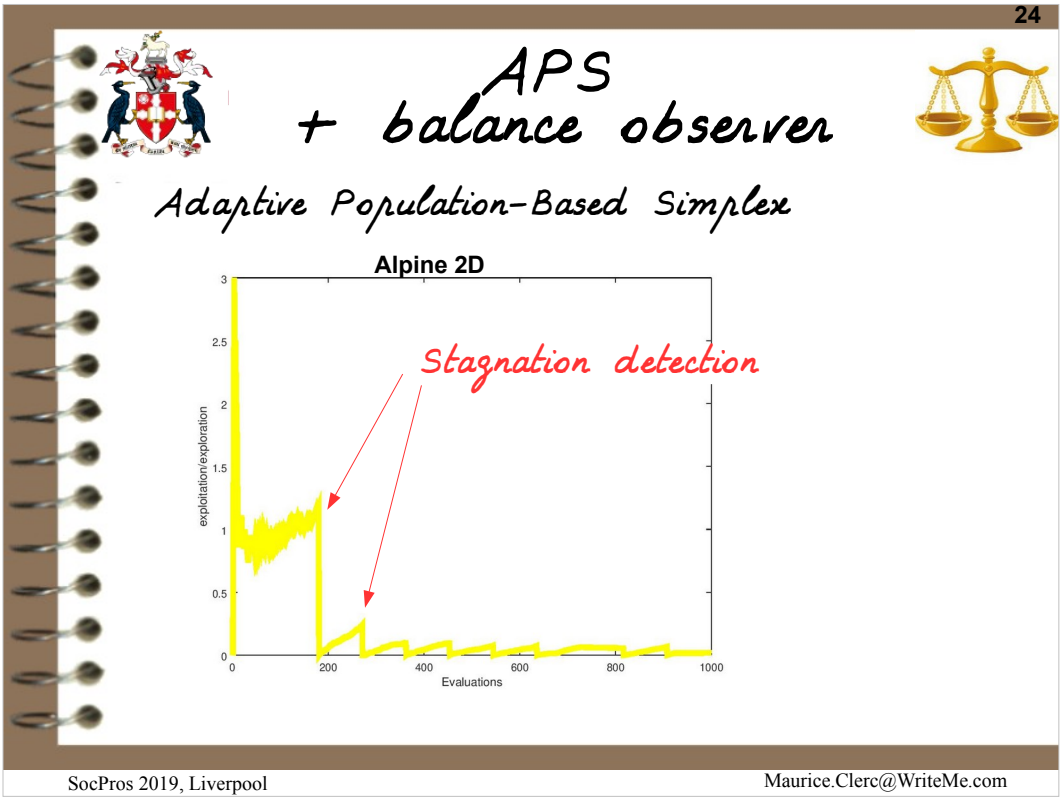
However the third set of experiments with good algorithms shows that is not always true. Let's see that.



Now I run a published algorithm, to which I just added a few instructions that count the number of points that are sampled inside exploitation domains of previous ones, at each time step.

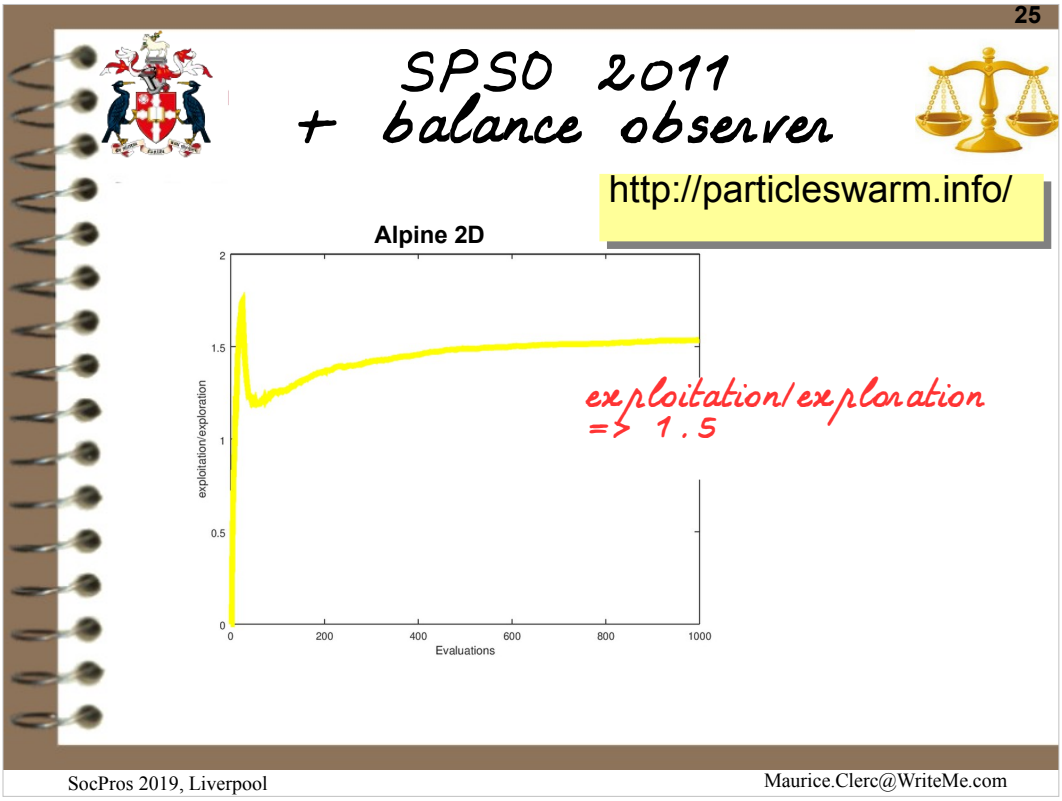
This optimiser is of course more sophisticated than Explo 2 and, in particular, it can detect stagnation in a probabilistic way.

As a result the observed profile on the same Alpine problem is completely different.



Here it is.
Just after each stagnation detection, the algorithm mainly explores and, therefore, the profile is decreasing.

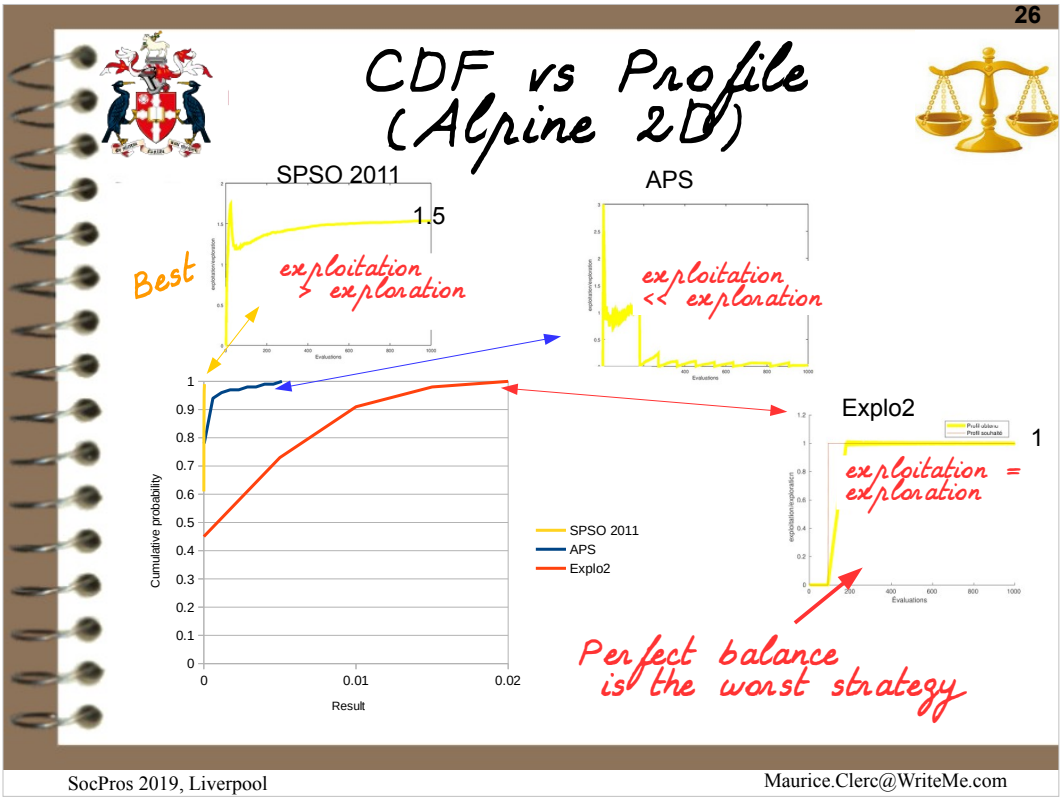
This is not really the case with an older optimiser, a PSO based one.



Particle Swarm Optimisation is well known, so I do not give here its flow chart.

Let's look at the generated profile. Now the profile is increasing to a limit, except just after initialisation.
Do you remember the profile with random search? This one is very similar.

Does it mean that SPSO is not better than random search? Not at all, fortunately, as we can see, again, by plotting the CDFs.



On our simple problem, the PSO based algorithm is better than APS. And of course both are better than our rudimentary Explo 2.

So, does it mean now that good algorithms generate increasing profiles tending to a limit greater than 1?

Unfortunately, it would be too good to be true.

To see that, we can now consider a more difficult problem.

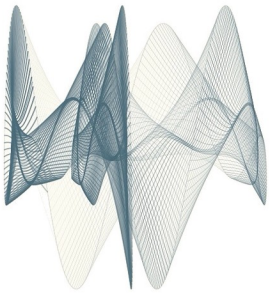
Another example (FM 6D)

CEC 2011 Function 1: Frequency-Modulated Sound Waves

$$y(t)=a_1\sin(\omega_1)t\,\theta+a_2\sin(\omega_2)t\,\theta+a_3\sin(\omega_3)t\,\theta$$
$$y_0(t)=\sin(5)t\,\theta+1.5\sin(4.8)t\,\theta+2\sin(4.9)t\,\theta$$

$$\theta=\frac{\pi}{50}$$
$$x=(a_1,\omega_1,a_2,\omega_2,a_2,\omega_3)\in[-6.4,6.35]^6$$

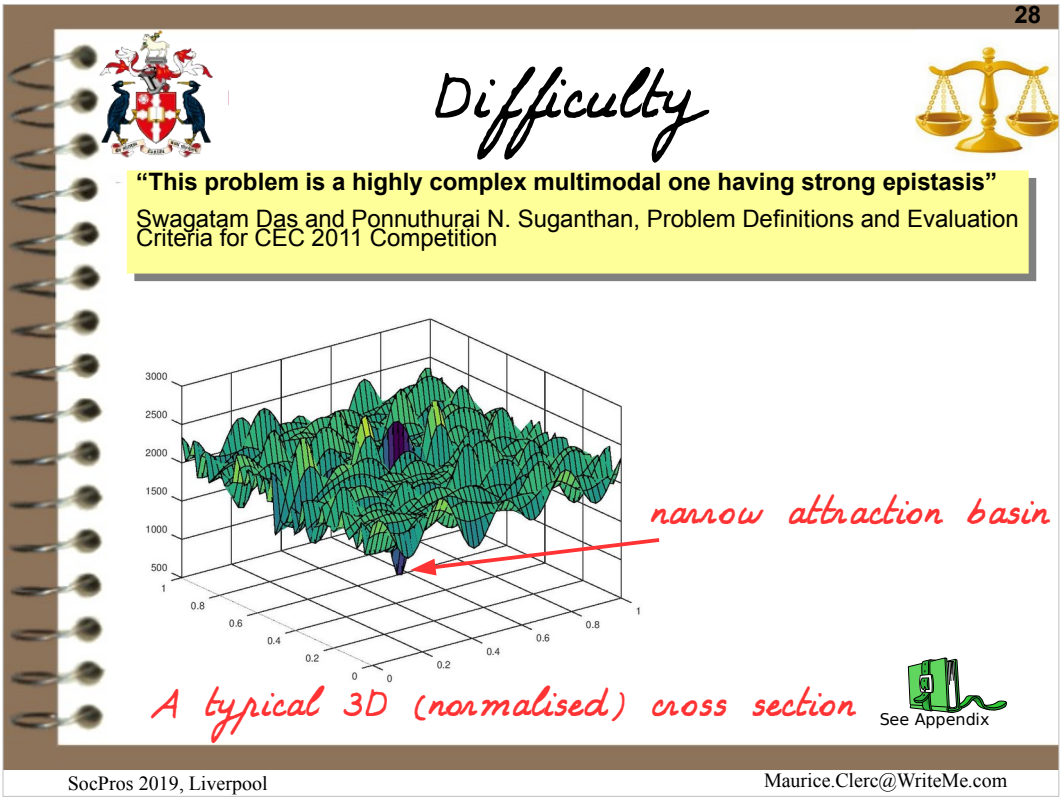
$$f(x)=\sum_{t=0}^{100}(y(t)-y_0(t))^2$$



solution=(1, 5, 1.5, 4.8, 2, 4.9)

This is a test problem largely used. Just of dimension 6, but nevertheless quite difficult.

Note that by looking at the formulae the solution is obvious, but an iterative algorithm is too stupid to see it.

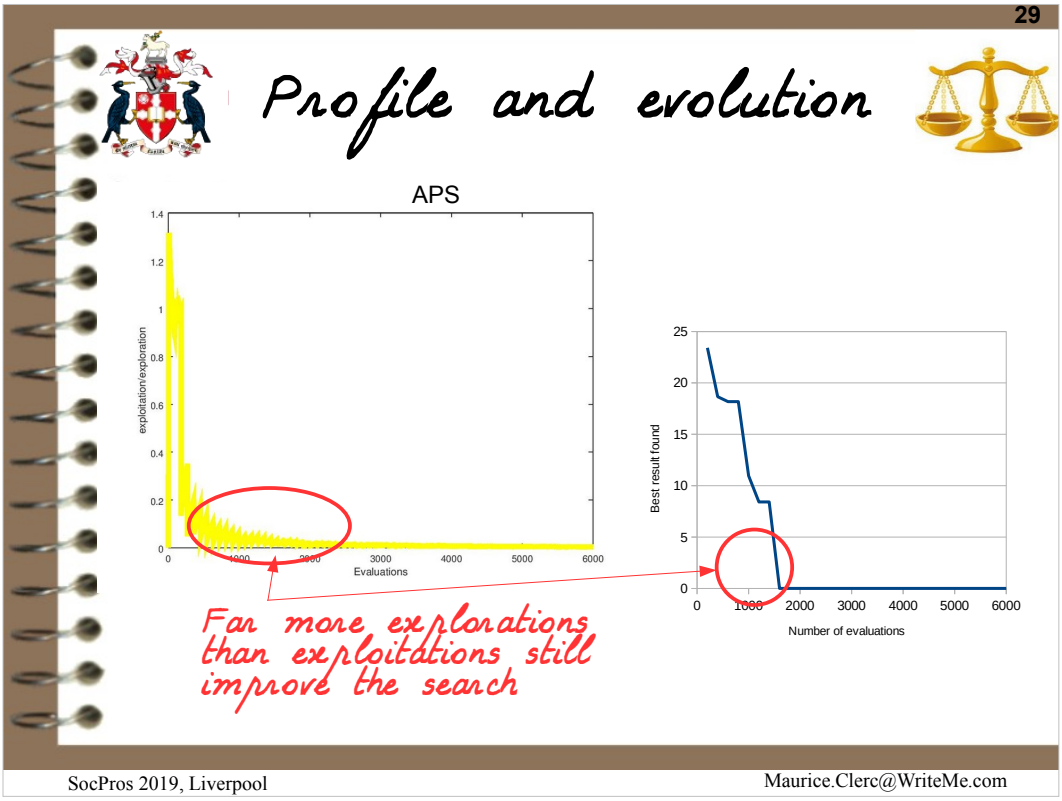


I do not know if you can imagine a 7 dimensions space, but I can not.
However it is possible to plot cross sections. They suggest that the solution is on the floor of a narrow basin.

I ran 100 times APS and SPSO, with a budget of 6000 evaluations for each run.

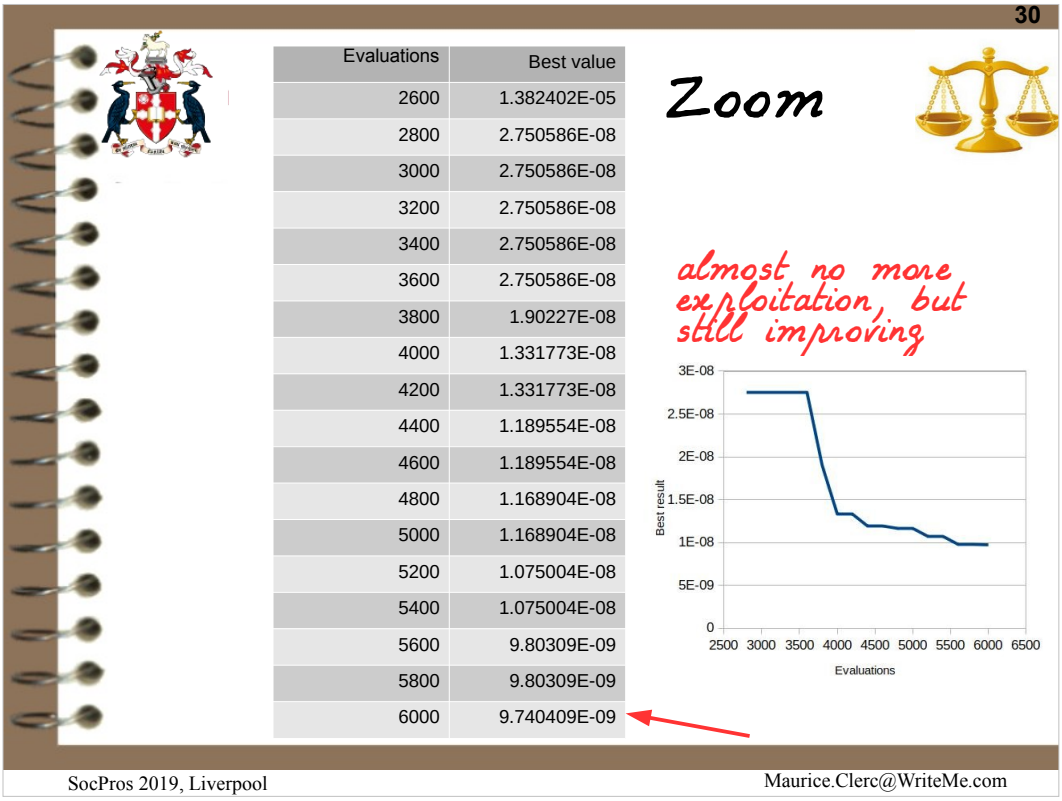
Why 6000? Because with this budget, APS finds a pretty good solution about 10 times more often than SPSO, so comparison is easy.

(Do we have time to say a few words about difficulty?)

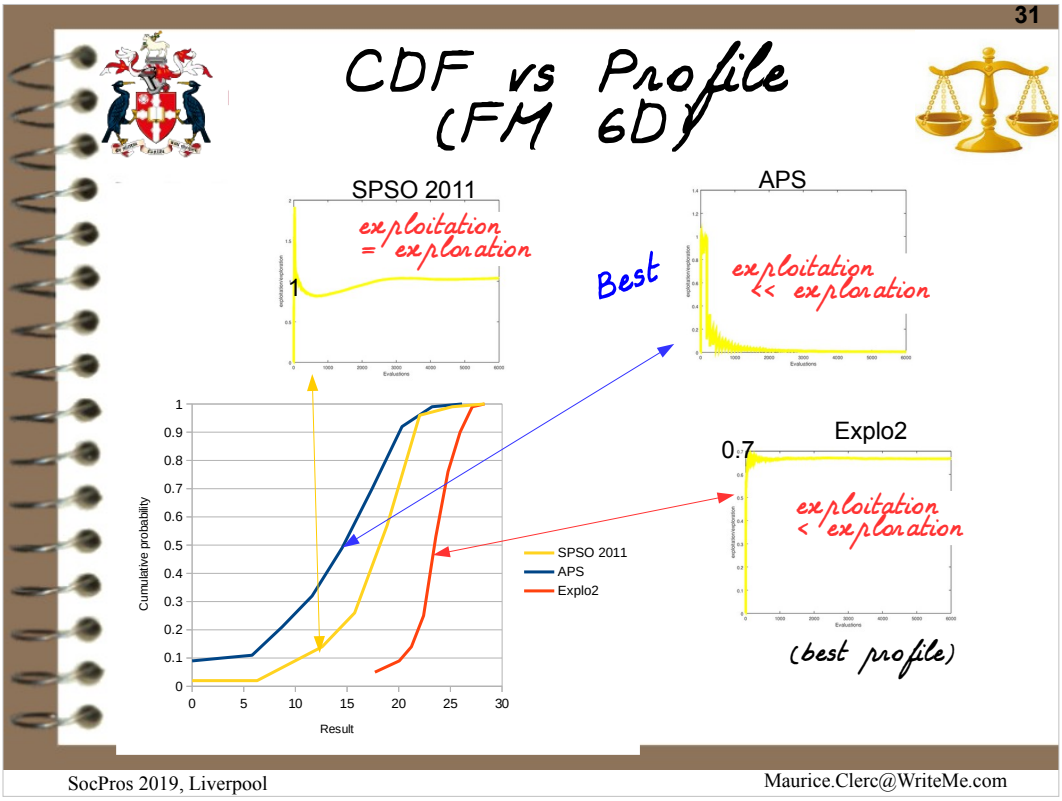


Again APS generates a decreasing profile, meaning it performs more and more exploration than exploitation.

Moreover, with this strategy, and as already said, APS continuously improves the best solution found, and the final error is very small.



Actually we can see it more clearly on this slide, which shows the decreasing error values found after 2000 evaluations and more.

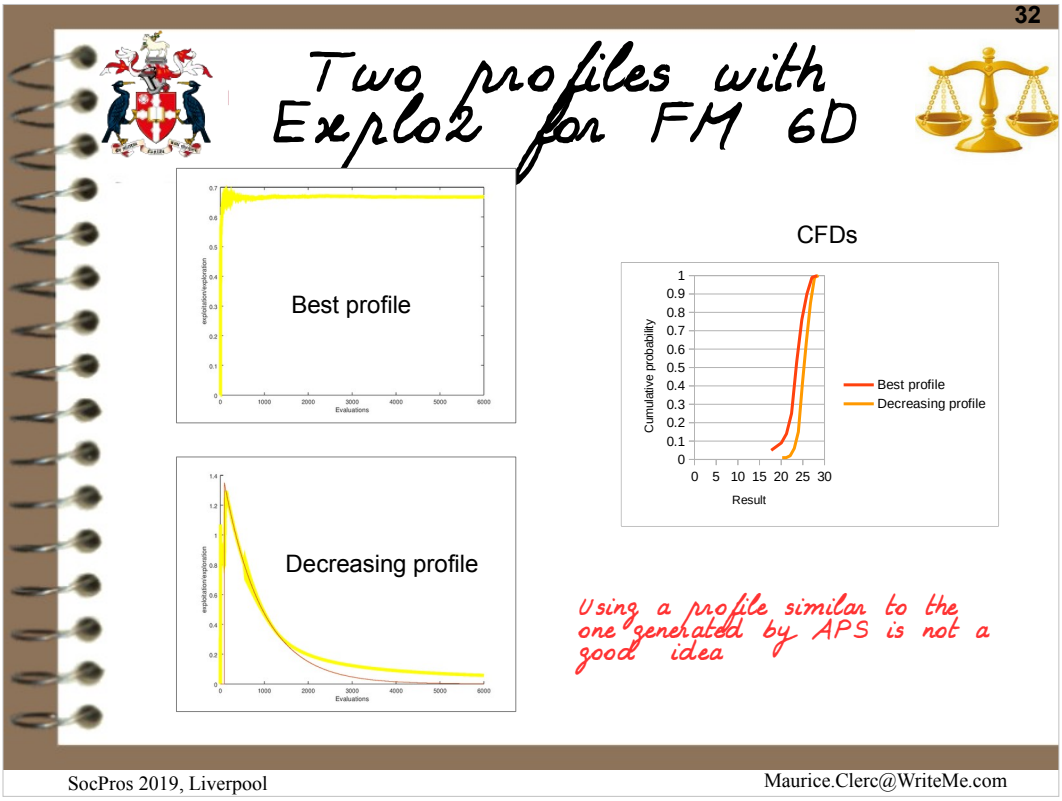


So, if we compare the CDFs for this difficult 6 dimensions problem, now APS is the best optimiser.

One may think that the generated profile looks a bit strange, but this is precisely for the problem is difficult. As the algorithm is adaptive, it largely increases the number of explorations in order to cope with this difficulty.

So, in such a case, trying to keep a “good balance” between exploitation and exploration would be inefficient.

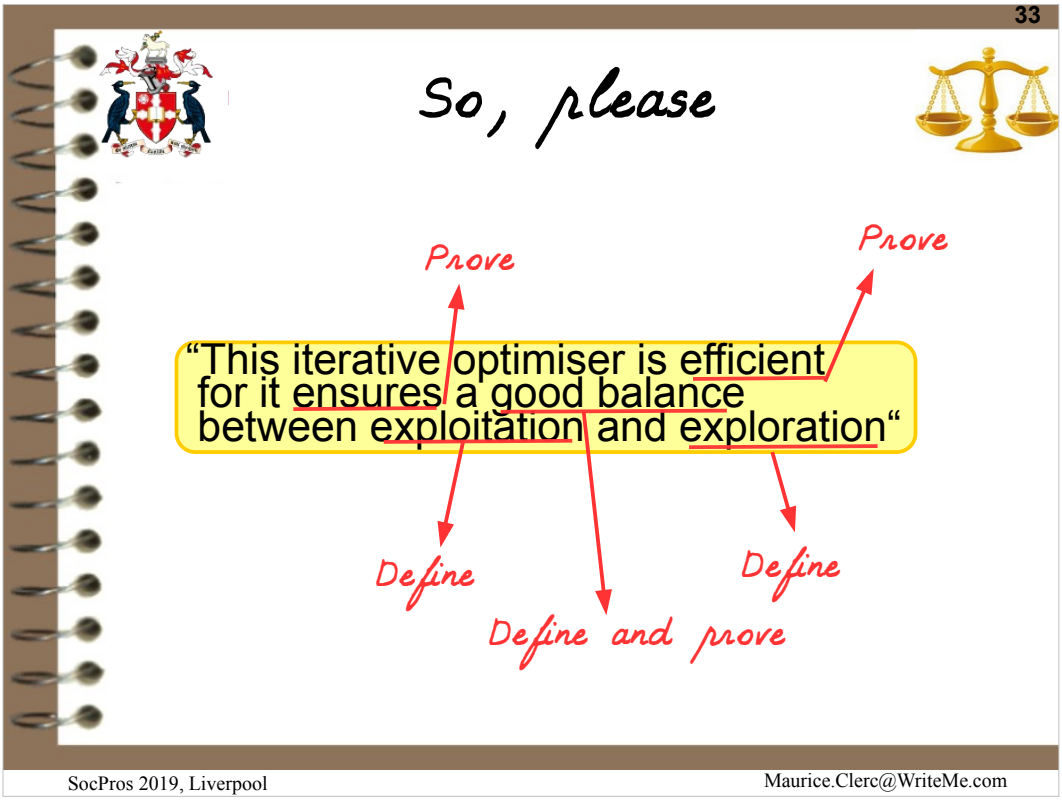
And of course, again, Explo 2 is largely the worst, for it never finds the solution, even by trying different predefined profiles.



For example one could give to Explo 2 a predefined profile very similar to the one generated by APS, that is quickly decreasing.

But in fact it is not the best choice, as you can see on THIS figure.

Actually, it confirms that the balance between exploration and exploitation is not a reliable indicator of the efficiency of an optimiser.



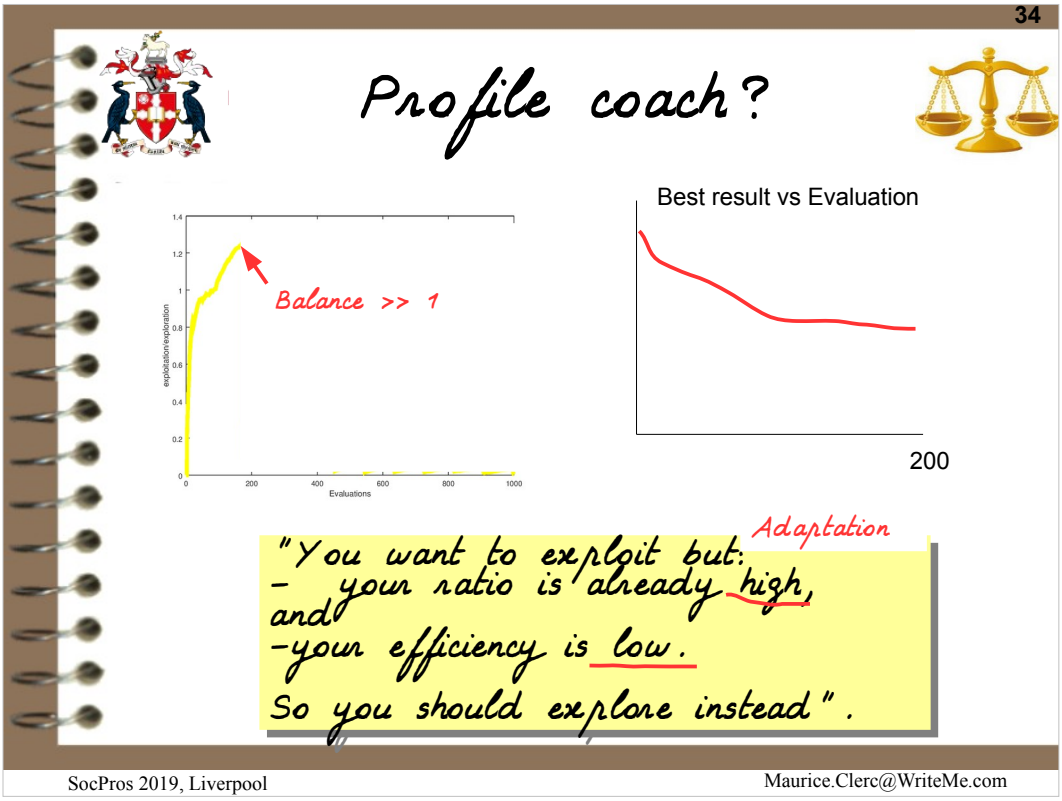
Finally, the main conclusion of all these experiments is that you have to be careful with the claim we have seen at the beginning of this talk.

If you read such a claim or, worse, if you are tempted to write one, be sure exploitation and exploration are rigorously defined.

Be sure to rigorously define what "good balance" means, and prove that the algorithm indeed ensures it.

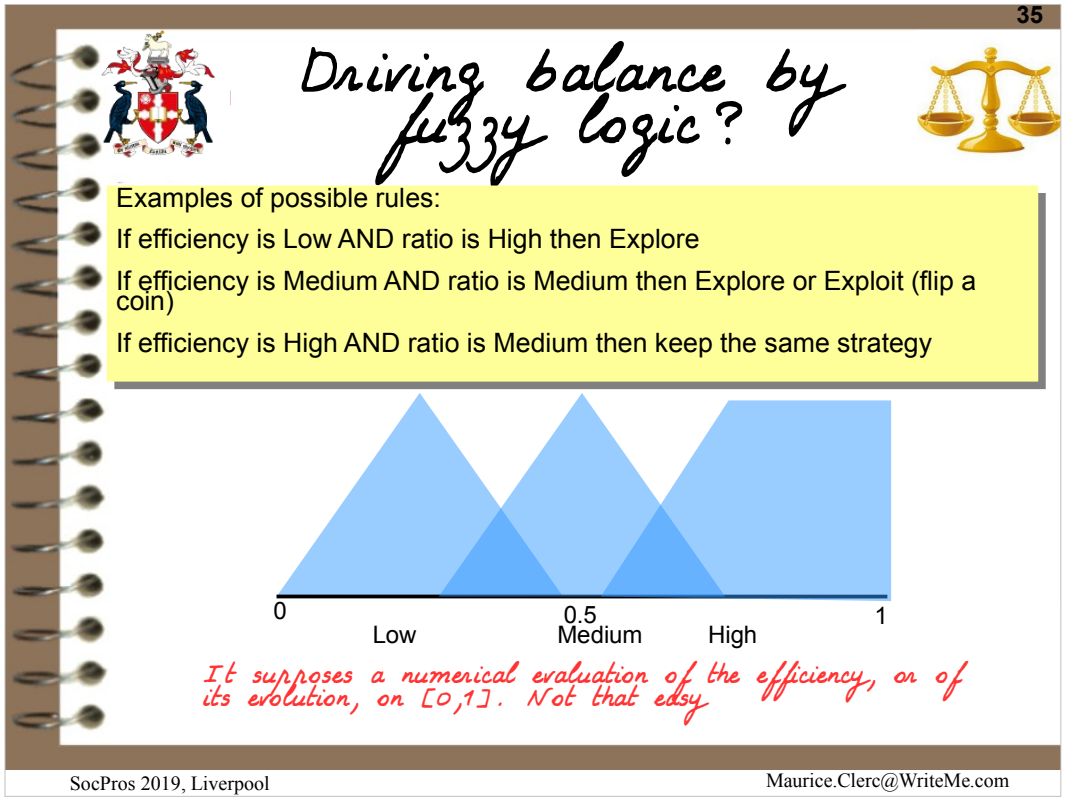
And last but not least, be sure that the efficiency is well proved, at least statistically.

Note a side effect of this study: it suggests possible ways to improve existing optimisers or even to design new ones.



Adaptive optimisers are not new.
For example, in APS, there is a rule saying “If there is probably a stagnation, then perform a partial restart of the worst agents”.

A similar rule, based on the generated profile, could be used.
Note the terms, High, and Low.
This is typically something that could be formalised by fuzzy sets.



As usual you can define fuzzy sets for the three terms Low, Medium and High.

The main difficulty is to define the efficiency. Actually an interesting approach could be a definition based on the evolution of the best solution.
Then High would mean “rapidly decreasing”, and so on.

To implement such a fuzzy approach, we need to build a complete table of the nine possible cases.

A table of possible fuzzy rules

Efficiency

exploitation / *exploration*

	Low	Medium	High
High	explore	explore or exploit (flip a coin)	exploit
Medium	switch to the other strategy	flip a coin	keep the same strategy
Low	exploit	flip a coin	explore

Logic: I already have gobd solutions, so I can look elsewhere

Here is such a table.
Of course other rules are perfectly possible, but discussing them is out of the scope of this talk.

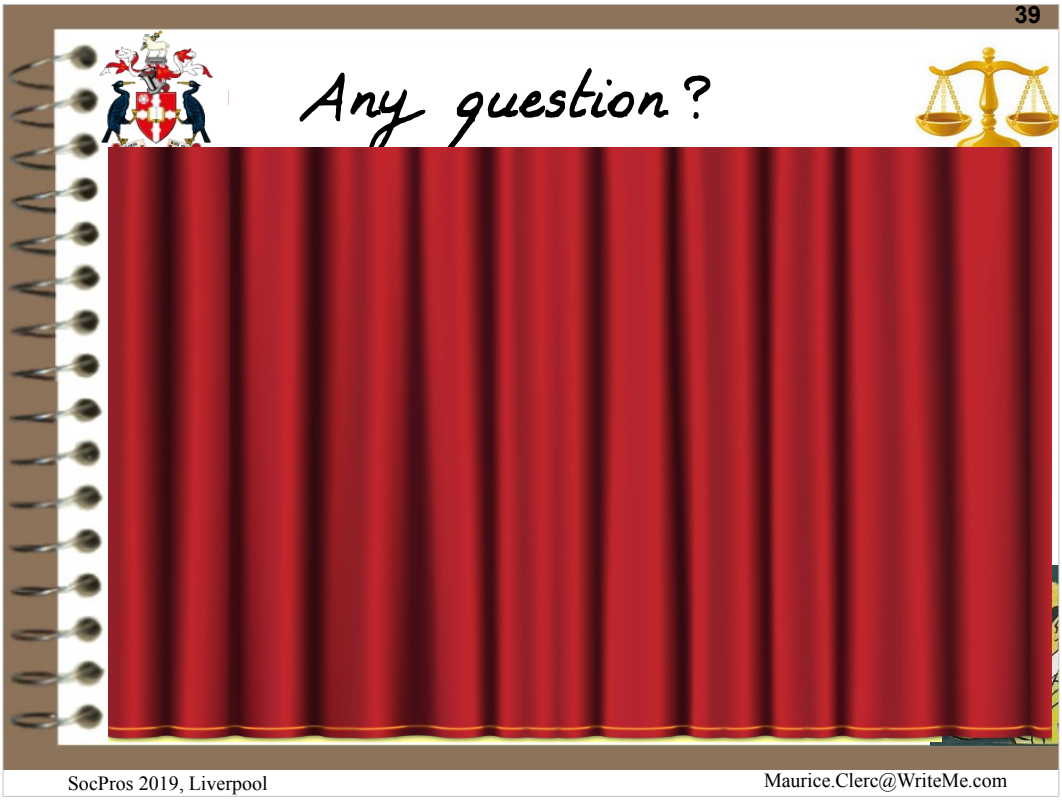
So, before closing, just a last slide with a little advertising.

This talk is mainly based on the two easiest chapters of THIS book.
The other chapters are quite technical, particularly the ones in which I define classes of problems and estimate their relative sizes.
Note that the original book is in French. As my wife said “It is not my field, so I can’t say anything about the content, but the coloured figures are very nice”.
Unfortunately, in the English version, they are black and white.

THIS other book is completely different. Far less technical, but, as a reviewer, I would say that you should read it before to write your next paper.
There is a French version, too.

Now I can close the curtains.

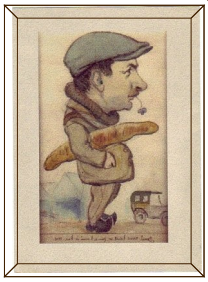
Unless, of course, you have any questions.



Actually I tried to anticipate some questions, that is why there are a few more slides in the Appendix.

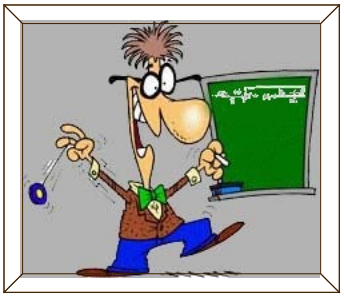
Appendix

For normal guys



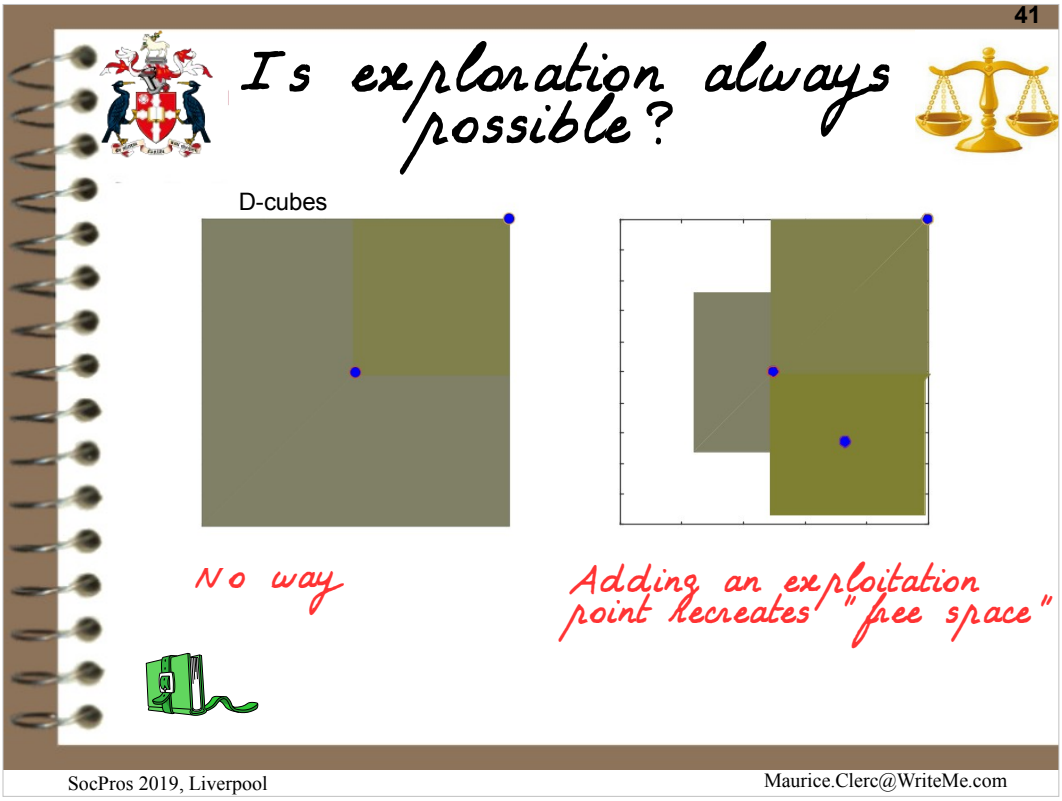
* Counting exploitation points
* Is Exploration always possible?
* Nearer is better

For maths addict



* CDF, Qu'es acò?
* Success proba vs numberb of runs
* Stochastic geometry, a simple example

* Difficulty measure
* Difficulty vs Dimension

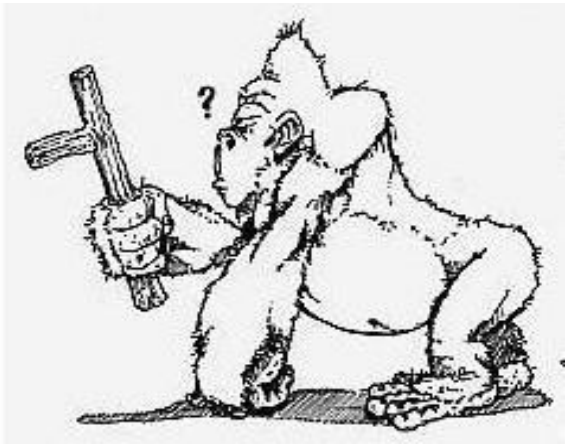


On this example the exploitation domains are squares. On the left no exploration is possible any more. The two exploitation domains entirely cover the search space.

So we have to cheat a bit and to force an exploration point inside an exploitation domain, say the largest one.

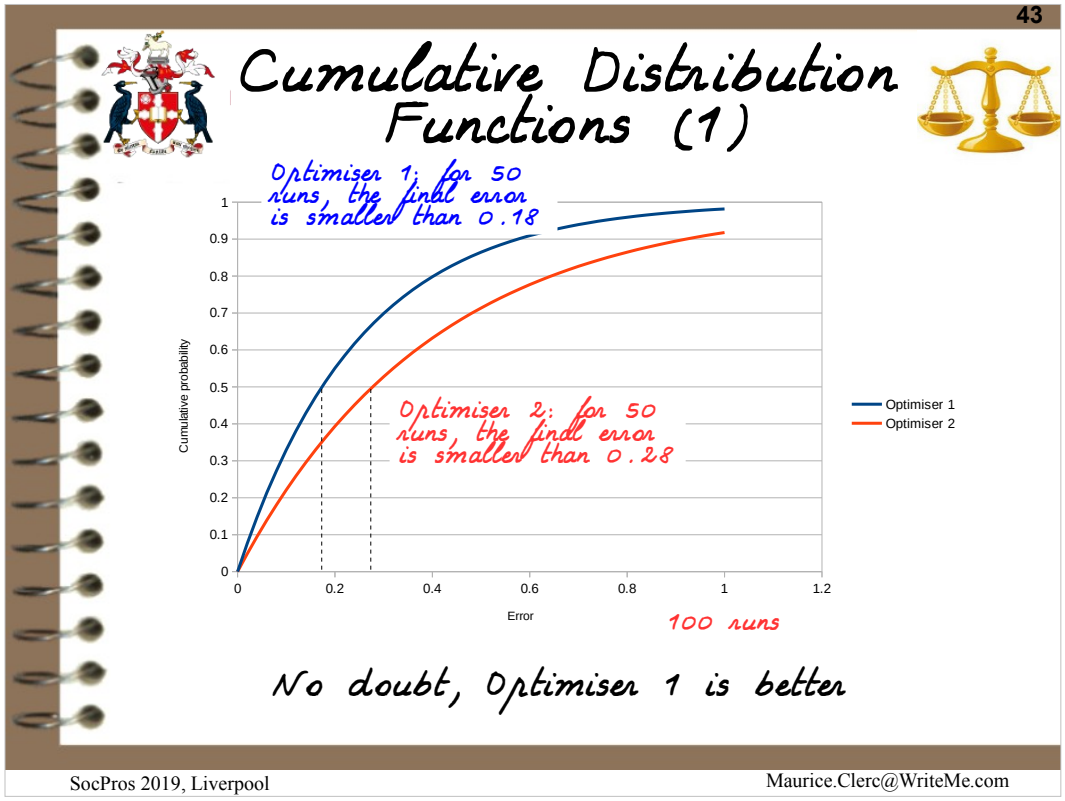
And as soon as we do that, free space is generated because some exploitation domains are now smaller.

CDF, Qu'es acò?



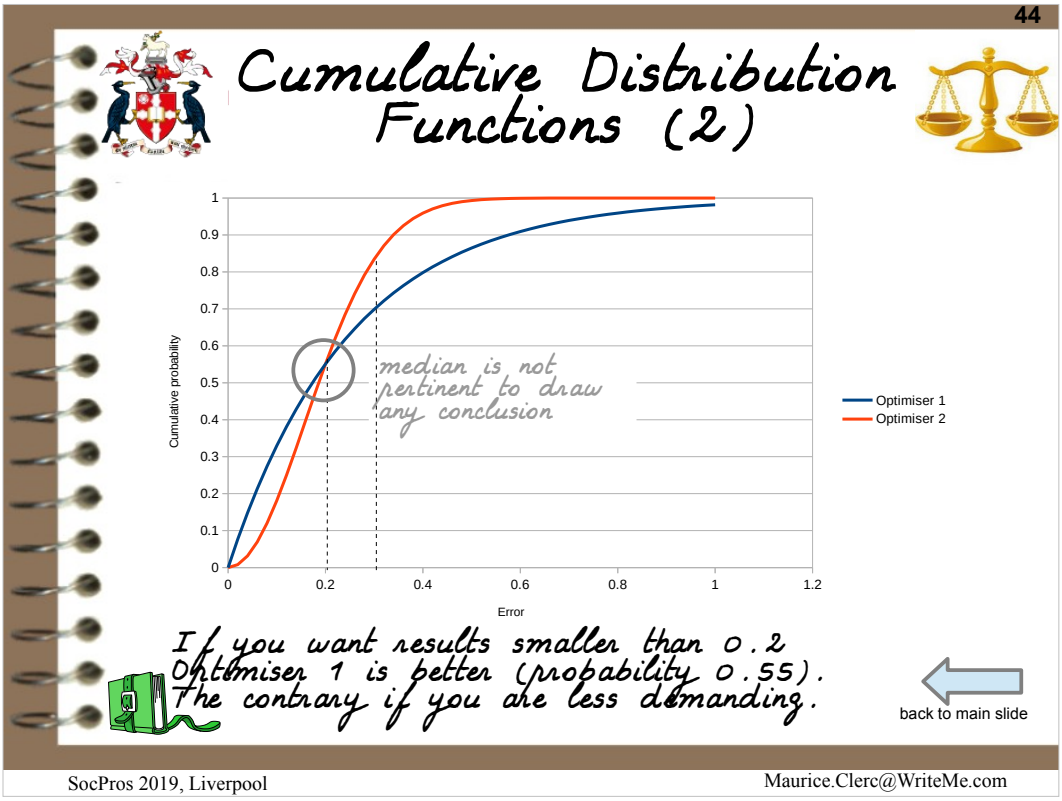
Maurice.Clerc@WriteMe.com

CDF is for Cumulative Distribution Function.
It may be not enough to say that, so let's explain.



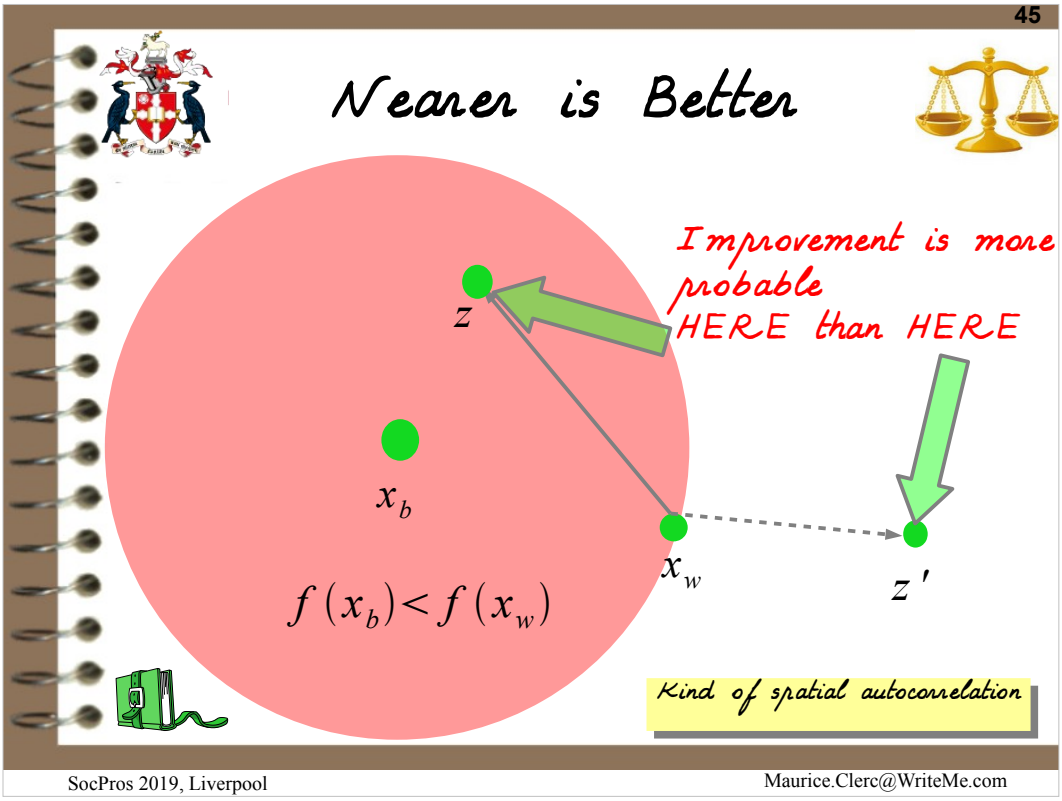
This concept is valid only for stochastic optimisers, not for deterministic ones.
On a given problem, you run the optimiser a lot of times, and for each final error value you count how many runs can be said to be successful, that is giving a result at most equal to this error value.
You do that for the two optimisers to compare, and you plot the normalised curves.
If one is entirely “above” the other one, then the superiority is clear.

However, it is not always the case.



As an user, I have a given budget, say a given maximum number of evaluations.
So I do not really care of the mean or of the median.
What I want to know is the probability to find a good solution.
But “good solution” is depending on my requirement. Here, if I am satisfied with an error value smaller than 0.3 then the optimiser 2 is largely better.

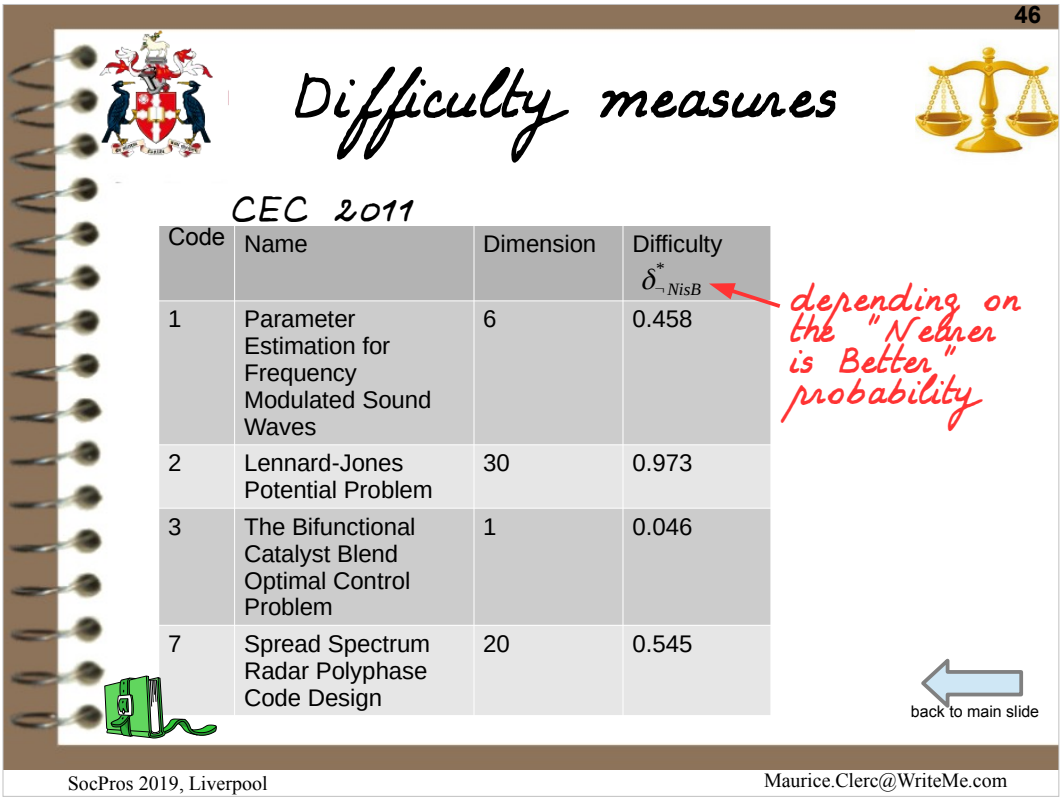
But if I absolutely want a value smaller than 0.2, then the optimiser 1 is preferable, even if the corresponding probability is only about 50%.



A difficulty measure can be based on the Nearer is Better concept. What is it?
The point (xb) is better than this one (xw).
Now, starting from here, we can either go farther from the best point (z'), or, on the contrary, nearer to it (z).
The Nearer is Better assumption is that z will more probably improve xw than z'.

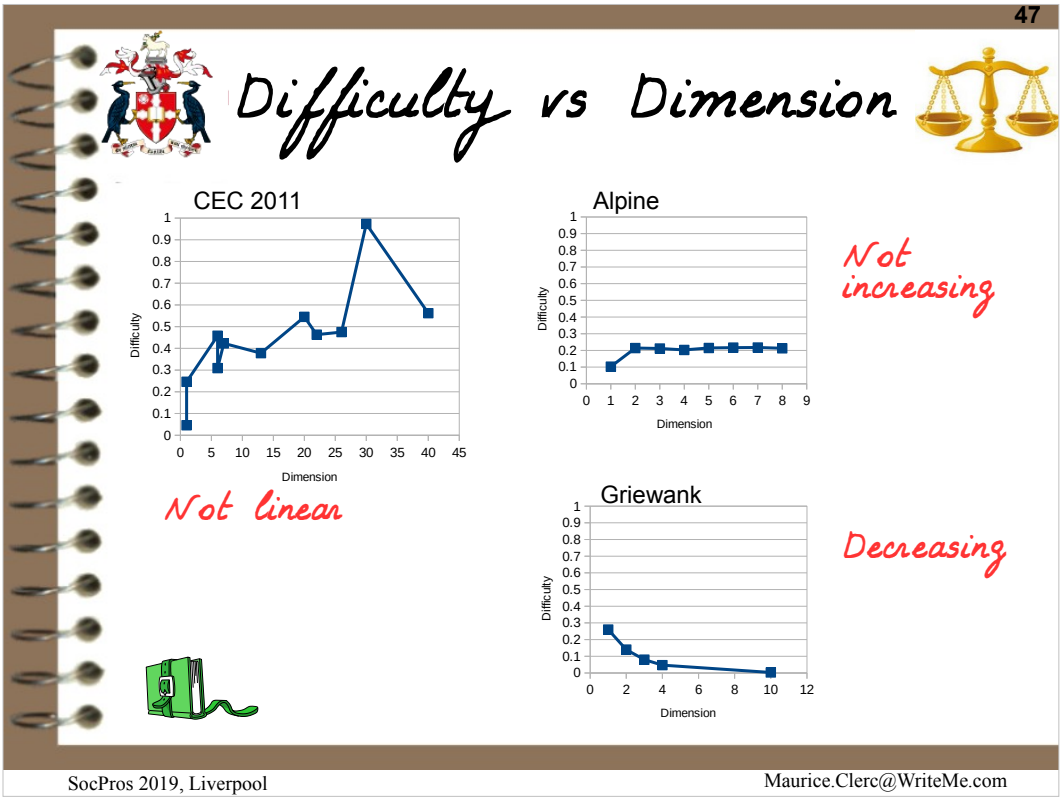
I don't go into detail, but this can be formalised. For a given function, one can define a correlation coefficient that measures how much the assumption is true.

For many many problems, it is true, even for combinatorial ones. However, in that case, the correlation is just slightly positive.
On the other hand, except for strictly monotonic functions. the correlation is smaller than 1.



For example, by using such a measure, you can evaluate how difficult are the problems of THIS well known benchmark.

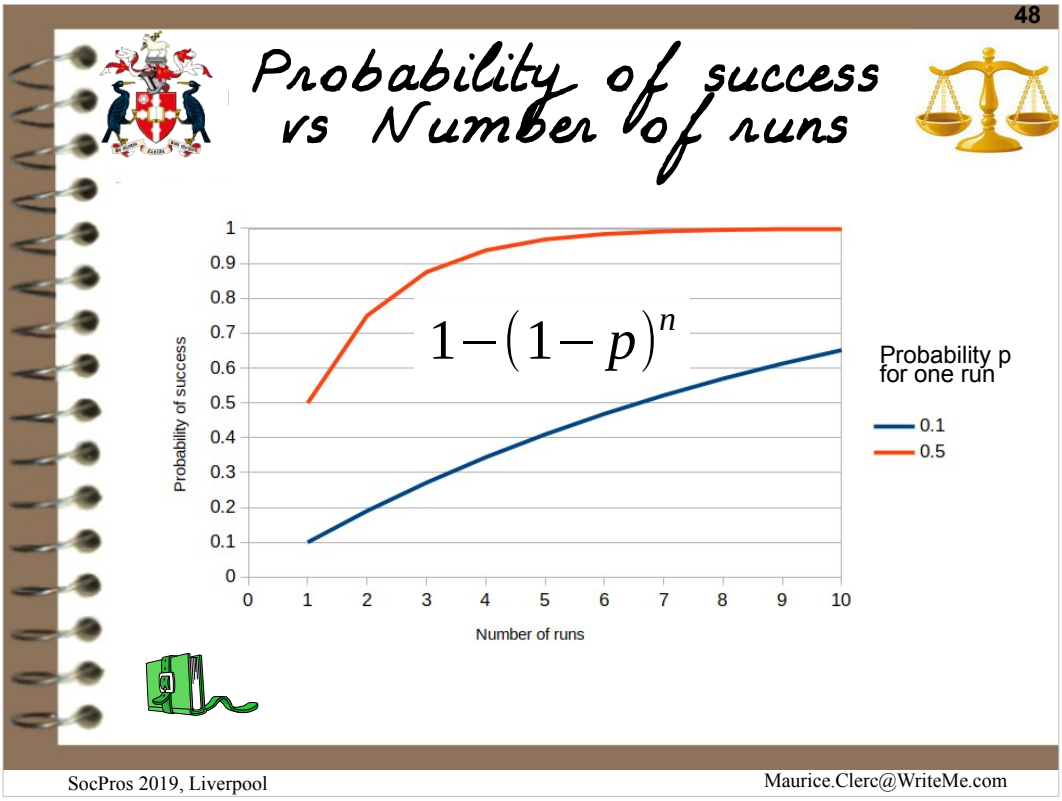
It is sometimes said that the difficulty is linearly increasing with the dimension.
Even with a clear definition of the difficulty, which is not always given, this claim is rarely true.



It is of course wrong when you consider the different problems of a benchmark, like HERE.

But it is also wrong when you consider a so called “scalable” problem, like Alpine, or Griewank.

In this last case, you may even note that the difficulty is decreasing with the dimension. This is because the number of local minima indeed increases with the dimension, but their sizes are also quickly decreasing, so it is easy to escape.



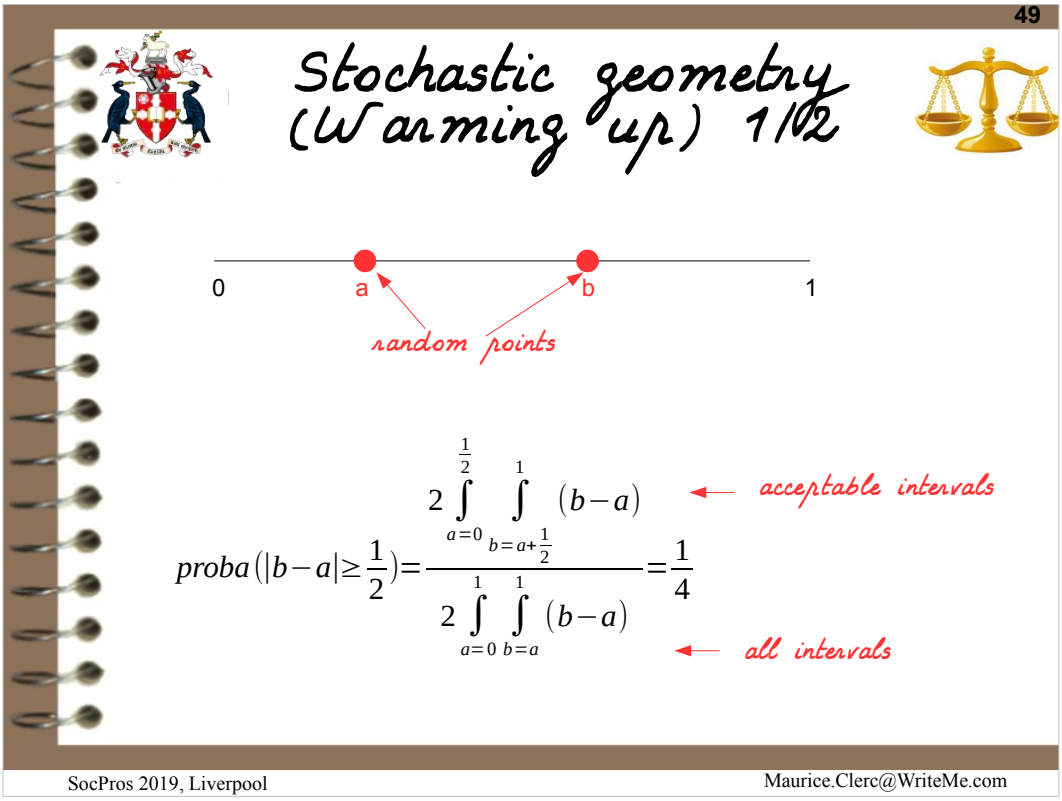
Even if the probability of success is low, say 0.1, you can increase it if your budget is sufficient.

Here, after only 7 runs, the probability increases to more than 0.7.

Now, how to spend the total allowed budget?
Should you launch just a few long runs, or many short ones?

Or something more sophisticated, like a few short runs first and then longer ones?

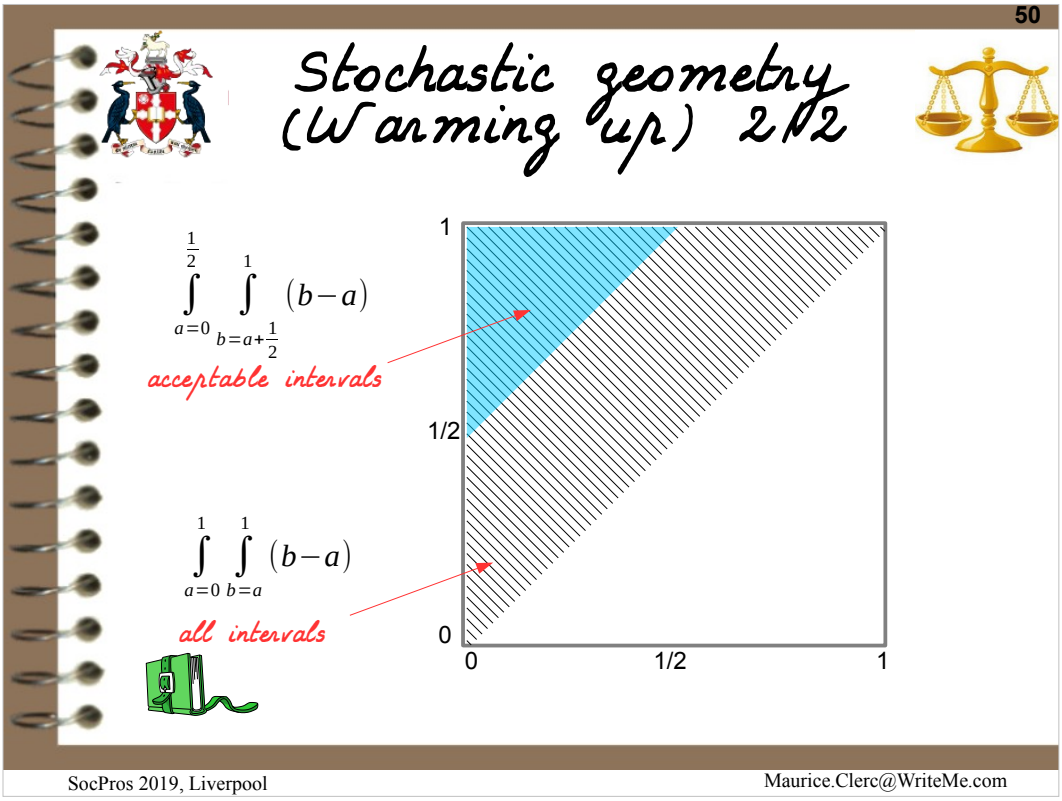
I don't elaborate here, but it is possible to define strategies that optimise the way you use your budget.



A very classical problem.
a and b are two random numbers (uniform distribution).

What is the probability that the distance between them is greater than 0.5?

The nice formula here gives the value, but it is possible to guess it, just by looking at a figure

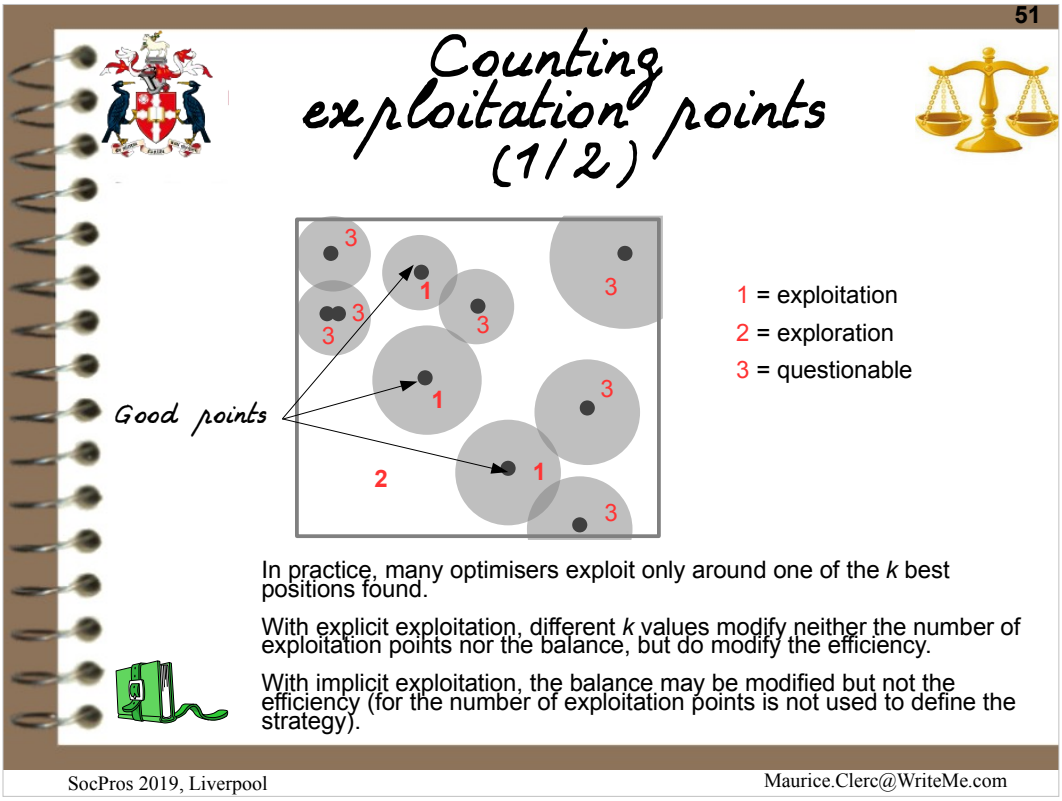


First, one can consider only the case b greater than a.

Then, the “volume" of all possible intervals, if I dare say, is the grey triangle.

And the volume of all acceptable intervals is given by the blue triangle.

Hence the ratio ¼.



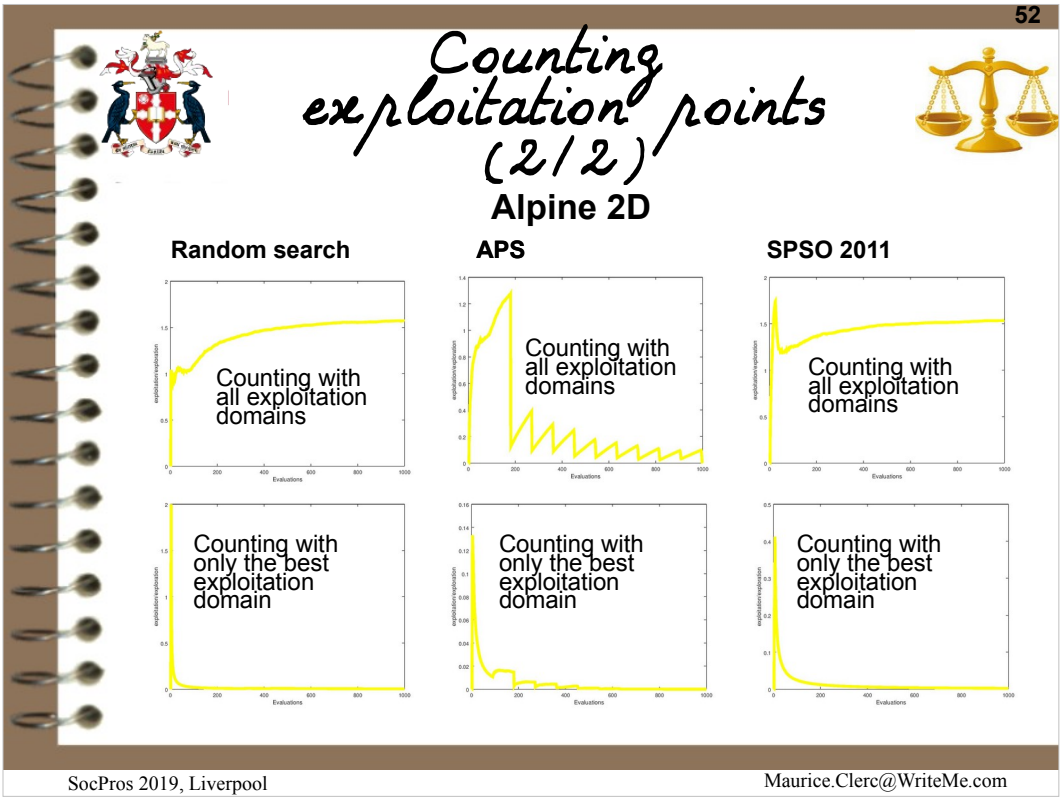
Quite often, not all exploitation domains are taken into account, but only the ones of the best points.

In that case, exploration is not any more the logical negation of exploitation.

More precisely, with some algorithms, a point can be sampled inside a “bad” exploitation domain, and seen as exploration, as in fact it is not true.

This can lead the algorithm to a wrong strategy

Of course, this bias also appears when not all positions are saved. A point can perfectly be sampled very near to a bad one that has been forgotten, and this is often a waste of time.



With random search, the difference is easy to explain. The probability to sample a point inside the only exploitation domain of the best point tends to zero, because the size of this domain is decreasing.
Therefore, the balance also tends to zero.

With an adaptive algorithm like APS, the difference is not that important. The balance is decreasing, no matter how you count the number of exploitation points.

With SPSO, the explanation of the difference is similar to the one for random search, because SPSO is not adaptive.

But for both, APS and SPSO, the behaviour of the algorithm is exactly the same, for the number of exploitation points is not used to modify the strategy.