Intrinsic difficulty of an optimisation problem

Maurice Clerc

2013-05-26 DRAFT

1 Definition

In [2], and in its English version [3], I defined the difficulty of a problem as the probability to find a solution just by chance. More formally, let S be a search space of dimension D, usually a D-rectangle, and let f be a non negative numerical function defined on each point $x = (x_1, \ldots, x_D)$ of S. We try to find a point on which f is minimal, or, at least, acceptable, i.e. smaller than a predefined value ε. Such a point is called a *solution,* and the set of all solutions is called the *acceptable space A*.

Now, if we draw a point at random according to the uniform distribution, what is the probability that this point is a solution? Clearly, if V_S is the volume of the search space, and V_A the volume of the acceptable space, this probability is simply V_A/V_S. In practice we do not consider the probability itself, but the opposite of its logarithm. Therefore, the formula that we use for the difficulty is

$$\delta = -\ln\left(\frac{V_A}{V_S}\right) = \ln(V_S) - \ln(V_A) \tag{1.1}$$

So, we manipulate only small numbers, and if the acceptable space is identical to the search space, then the difficulty is zero, which is quite conform to the intuition. Sometimes, it is possible, and even easy, if we have an analytical formula for f, to directly compute it. More often,,we can compute a mathematical approximation. But most of the time, the only way is to apply an algorithm that gives an estimation.

2 Exact and approximate computations

2.1 Discrete functions

An obvious case in which exact computation is possible is for a finite discrete search space of size N. If there are n solutions, the "volume" of the acceptable space is n, and the "volume" of the search space is N. Therefore the difficulty is simply

$$\delta = \ln(N) - \ln(n) \tag{2.1}$$

For example, for a combinatorial problem like the Travelling Salesman one with C cities, if there is just one solution, the difficulty is

$$\delta = \ln(C!) \tag{2.2}$$

Note that this is the *intrinsic* difficulty, which is not depending on any method or algorithm that could be used to find the solution. It means that the difficulty of a problem that seems easy (say, for TSP, all cities on a circle) is exactly the same that the one of a problem that seems quite hard (cities on a complicated network of roads). From this point of view, there is no difference between discrete problems, combinatorial or not, they are all equivalent: if the size of the search space is the same, then the difficulty is also the same.

2.2 Continuous functions

Let us consider now a few less obvious cases, taken from the numerical continuous optimisation. In all cases below, and for simplicity, we consider that the search space is $S = [-a, a]^D$. So, its volume is

$$V_S = (2a)^D \tag{2.3}$$

As said, for some functions found in the optimisation literature , the volume V_A of the acceptable space can be exactly computed. Also, there are a bit more functions for which the function, around the solution point, is equivalent to a polynomial one, and therefore V_A can be approximated. To derive the approximations, I applied the following classical formulae, when u is small

$$\begin{cases} (1+u)^n & \simeq & 1+nu \\ \cos(u) & \simeq & 1-\frac{u^2}{2} \\ e^u & \simeq & 1+u \end{cases}$$

2.2.1 Parabola/Sphere function (exact computation)

$$f(x) = \sum_{d=1}^{D} x_d^2 \tag{2.4}$$

When $D = 1$, the mathematical name of this function is Parabola, and Paraboloid for $D = 2$, but in optimisation literature and for any values of D, it is more often called Sphere, probably for the isolines indeed are D-spheres. Let us precisely consider the D-sphere of radius ρ so that $\rho^2 = \sum_{d=1}^{D} x_d^2 < \varepsilon$. Clearly, the acceptable space is the portion of this D-sphere that is inside the search space. We do suppose that ε is small enough so that this D-sphere is entirely included in the search space. Therefore the volume of the acceptable space A is

$$V_A = \varepsilon^{\frac{D}{2}} \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \tag{2.5}$$

2.2.2 Schwefel 2.21 (exact computation)

$$f(x) = \max(|x_d|) \tag{2.6}$$

Here the acceptable space is defined by the D-sphere of radius 2ε, and therefore its volume is

$$V_A = (2\varepsilon)^D \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \tag{2.7}$$

2.2.3 Griewank (approximation)

$$f(x) = \frac{1}{4000}\sum_{d=1}^{D} x_d^2 - \prod_{d=1}^{D}\cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \tag{2.8}$$

The second order approximation is

$$\sum_{d=1}^{D}\left(\frac{1}{4000}+\frac{1}{2d}\right)x_d^2 \tag{2.9}$$

which is the formula of a D-ellipsoid. The volume of the acceptable space is then given by

$$V_A = \varepsilon^{\frac{D}{2}} \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \sqrt{\prod_{d=1}^{D}\frac{1}{\frac{1}{4000}+\frac{1}{2d}}} \tag{2.10}$$

2.2.4 Alpha-D-max function (exact computation)

$$f(x) = \alpha^D \max(|x_d|) \tag{2.11}$$

Here the acceptable space is defined by the D-cube of side $\frac{2\varepsilon}{\alpha^D}$, and its volume is therefore

$$V_A = \left(\frac{2\varepsilon}{\alpha^D}\right)^D \tag{2.12}$$

Note that here the volume decrease as soon as the dimension is greater than

$$\frac{1}{2}\left(\frac{\ln(2\varepsilon)}{\ln(\alpha)} - 1\right) \tag{2.13}$$

It means that when D increases, the difficulty first also increases, but then decreases to zero. For the example below, with $\varepsilon = 10^{-10}$ and $\alpha = 0.1$, it happens for $D \geq 5$.

2.2.5 Rastrigin (approximation)

$$f(x) = \sum_{d=1}^{D} \left(x_d^2 + 10\cos(2\pi x_d) + 10 \right) \tag{2.14}$$

The second order approximation is

$$\left(1 + 20\pi^2\right) \sum_{d=1}^{D} x_d^2$$

and therefore an approximation of the volume of the acceptable space is the volume of the D-sphere of radius $\rho = \sqrt{\frac{\varepsilon}{1+20\pi^2}}$

$$V_A = \left(\frac{\varepsilon}{1+20\pi^2} \right)^{\frac{D}{2}} \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \tag{2.15}$$

2.2.6 Ackley (approximation)

$$f(x) = -20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} x_d^2}\right)} - e^{\frac{1}{D}\sqrt{\sum_{d=1}^{D}\cos(2\pi x_d)}} + 20 + e \tag{2.16}$$

The second order approximation is

$$\frac{4r}{\sqrt{D}} + 2\pi^2 \frac{r^2}{D}$$

where $r = \sqrt{\sum_{d=1}^{D} x_d^2}$, and therefore the first order one is

$$\frac{4}{\sqrt{D}} \sqrt{\sum_{d=1}^{D} x_d^2}$$

So, an approximation of the volume of its acceptable space is the volume of the D-sphere of radius $\rho = \frac{\varepsilon\sqrt{D}}{4}$

$$V_A = \left(\frac{\varepsilon\sqrt{D}}{4} \right)^{D} \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)} \tag{2.17}$$

3 Estimation algorithm

When there is no way to exactly compute, or even to mathematically approximate the volume of the acceptable space, there is still the algorithmic method. In short, we draw at random a big number of points, and we count how many are solutions. However this kind of brute force method is not realistic because of the computation time, so we have to use a more clever one.

3.1 Brute force is not a good approach

Let us suppose we know the volume V_A of the acceptable space. We draw at random k points. How big should be k to be almost sure (confidence level φ) to find at least one solution? This is a classical probability exercise. At each attempt, the probability to find a solution is $p = \frac{V_A}{V_S}$. So, the probability to *not* find a solution after k attempts is $(1-p)^k$. And therefore the probability to indeed find a solution after k attempts is $1-(1-p)^k$. As we want it at least equal to φ, it means that the answer to our above question is

$$k \geq \frac{\ln(\varphi)}{\ln\left(1-\frac{V_A}{V_S}\right)} \tag{3.1}$$

Let us consider a simple problem: the Sphere function on $[-100, 100]^2$, with an acceptable error equal to 10^{-10}. According to the formula 2.5, we have $\frac{V_A}{V_S} = 7.85 \times 10^{-15}$. With a confidence not specially high, say 0.9, the formula 3.1 says that we will have to draw at least 2.92×10^{14} points. And this is just for a 2D problem. So, unless you have a hypercomputer at hand, a better method has to be used.

3.2 A more clever method

The proposed algorithm is based on two assumptions:

- there is just one point x^* on which f reaches its minimum;
- the acceptable space is a connected area (which is therefore "around" x^*).

Moreover, in the current form, it really works only it the acceptable space A can be included in a D-rectangle whose volume is not "too big" compared to V_A. See 5.1 below for hint about how to generalise the algorithm.

As it does not matter where x^* is, we can suppose, for simplicity, that it is $(0, \ldots, 0)$. As said, and also for simplicity, the search space S is a D-cube. The idea is to define a test space T that is a D-rectangle $\otimes_{d=1}^{D}[t_{min,d}, t_{max,d}]$ centred on x^*. The algorithm is a cyclic one, and each cycle needs P_{max} points to draw. This number is an user defined parameter. Here is the description of the method:

At the beginning, the algorithm defines each side of T by dichotomy, so that $f((0, \ldots, t_{min,d}, \ldots, 0)) < 2\varepsilon < f((0, \ldots, 2t_{min,d}, \ldots, 0))$, and the same for $t_{max,d}$.

Loop:

Draw P_{max} points at random, compute their f values, and count how many are smaller than ε, say $P_{acceptable}$. We have three cases:

- $P_{acceptable} > P_{max}/2$. It means that T is probably too small. If it is smaller than the search space, we enlarge it (in practice we double it along each dimension), and launch another cycle of P_{max} points at random (go to Loop).

If it can not be enlarged (already equal to the search space), then the algorithm stops, saying that the difficulty is null.

- $P_{acceptable}$ is too small (typically smaller than $P_{max}/100$). It means that T is probably too big, and we reduce it (each side of the D-cube is divided by two), and then go to Loop. However, in order to avoid any infinite enlarge/reduce loop, this can be done only once.
- $P_{acceptable}$ is neither too big nor too small. Then the estimation of the volume of the acceptable space is given by

$$V_A \simeq \frac{P_{acceptable}}{P_{max}} \frac{V_T}{V_S} \tag{3.2}$$

where V_T is the volume of the test space, i.e. $T_E = \prod_{d=1}^{D} (t_{max,d} - t_{min,d})$. And the estimation of the difficulty is given by $\delta = \ln(V_S) - \ln(V_A)$. We will now see on some examples that this method gives pretty good results by using only a reasonable number of test points.

4 Examples and comparisons

First, we will compare the results of the algorithm to the ones we have exactly computed, or approximated. Then we will apply it to some other classical test functions, particular in order to see how the difficulty increases (or not) with the dimension.

4.1 Checking the precision

Let us apply our algorithm to the five functions for which we have a mathematical formula of the difficulty (even if approximated). The search space is $S = [-100, 100]^D$, and the acceptable error $\varepsilon = 10^{-10}$. The difficulty is estimated for dimensions D from 2 to 15. As we can see on the table 1 figure 4.1, the algorithmic estimation is pretty good, for the maximum error is quite small. The number of test points generated by the algorithm is typically $2 \times 10^6 D$, which is perfectly acceptable on any decent laptop.

Table 1: Maximum relative error between algorithmic estimation and exact/approximated difficulty, for dimension D from 2 to 15. $S = [-100, 100]^D$, $\varepsilon = 10^{-10}$.

Sphere	1.2×10^{-3}
Schwefel 2.21	3.6×10^{-5}
Griewank	5.7×10^{-4}
Rastrigin	2.2×10^{-4}
Ackley	2.7×10^{-3}
Alpha-D-max	2.5×10^{-5}

Table 2: Intrinsic difficulty of six quasi-real world problems.

	Dimension	Acceptable error	Difficulty
Tripod	2	10^{-10}	56
Compression Spring	3	10^{-10}	71.8
Gear Train	4	10^{-13}	23.3
Pressure Vessel	4	10^{-5}	87.3
Lennard-Jones	15	10^{-6}	145.3
Frequency Modulation	16	10^{-6}	70

4.2 Applying on some problems

As we have seen that the estimation algorithm gives good results, we can now safely apply it to more functions.

4.2.1 Scalable problems

The six first ones are coming from the CEC 2008 competition [1]. They are shifted, but, as said, it changes nothing for the intrinsic difficulty. Again, the search space is $S = [-100, 100]^D$, and the acceptable error $\varepsilon = 10^{-10}$. When the dimension increases, and for all of them, the difficulty increases linearly. It means that around the solution point all these functions have the behaviour of a D-polynom. In other words, these functions are locally more similar that one could think.

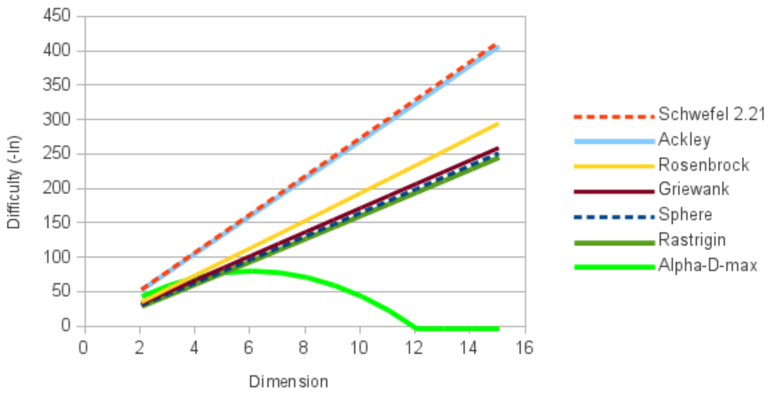
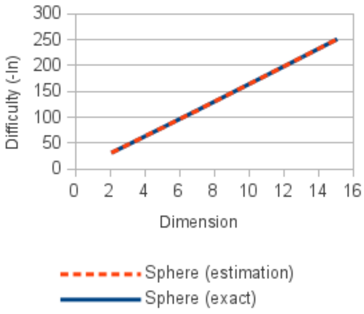
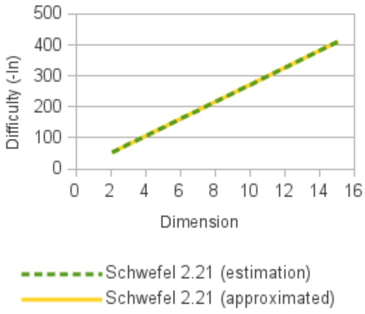


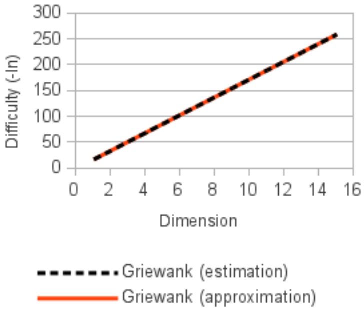
Figure 4.2: Evolution of the intrinsic difficulty with the dimension. The CEC 2008 problems have all the same local behaviour around the solution point.



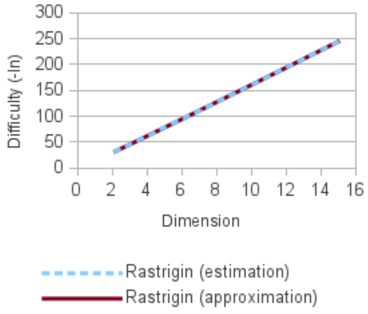
(a)



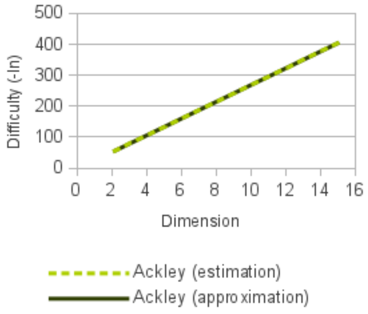
(b)



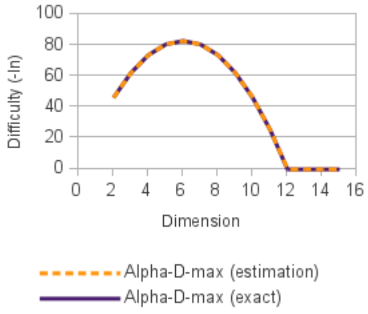
(c)



(d)



(e)



(f)

Figure 4.1: Algorithmic estimation of the difficulty vs exact or approximated value.

4.2.2 Quasi-real world problems

4.3 Comparing with an optimisation method

If we apply a reasonably good optimisation algorithm on some problems, is there a relationship between the success rate and the intrinsic difficulty? As we can see on the Table 3 and the Figure 4.3, the answer is negative. I have used APS (Adaptive Population Based Simplex [4]) on the six CEC 2008 problems we have seen above, in 10 dimensions. Clearly, APS (like in fact any "good" algorithms) takes advantage of the structure of the landscape of the function. No matter the intrinsic difficulty is, it easily finds the solution when the problem is unimodal, or separable (or both!).

For Griewank, which is neither unimodal nor separable, it takes advantage of the fact that the landscape has a global unimodal shape, just "perturbed" by a lot of small local minimums, from which it can easily escape (and the higher D is, the more it is easy). On the contrary, for Rosenbrock, whose intrinsic difficulty is smaller than the one of, say, Ackley, the global shape is quite deceptive.

Table 3: Intrinsic-difficulty, and success rate with APS.

CEC 2008 problem			Difficulty	Success rate (%)
Rastrigin		separable	80.7	98
Sphere	unimodal	separable	116.2	100
Griewank			156.7	82
Rosenbrock			215.9	2
Ackley		separable	354.9	100
Schwefel 2.21	unimodal		414.5	100

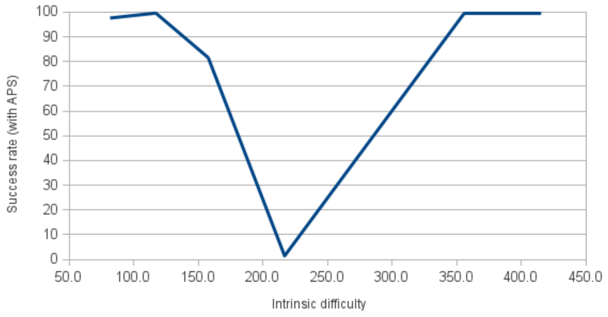


Figure 4.3: Success rate (with APS) vs Intrinsic difficulty. There is no relationship, because the optimisation algorithm takes advantage of the structure of the landscape, as the intrinsic difficulty does not.

5 When it does not work ...

On some problems, the above estimation method can not work, and the mathematical computation is far more difficult, because the acceptable space is not more or less "around" the solution point. Let us consider for example the following Product function (Schwefel 2.21):

Search space $S = [0, a]^D$.

$$f(x) = \prod_{d=1}^{D} x_d \tag{5.1}$$

Each hyperplan for which $x_d = 0$, is an infinite set of solutions. Therefore the acceptable space looks like a "star" with long thin rays that go through the whole search space. It is still connected, but can not be any more included in a single small D-rectangle.

5.1 ... and how to improve it

Let us consider the case $D = 2$. In this case, the volume of the acceptable space is easy to compute

$$\begin{aligned} V_A &= \varepsilon + \int_{\frac{\varepsilon}{a}}^{a} \tfrac{\varepsilon}{u} du \\ & \varepsilon + \varepsilon \left(2 \ln(a) - \ln(\varepsilon)\right) \end{aligned} \tag{5.2}$$

The difficulty is then $\delta = \ln\left(a^2\right) - \ln\left(V_A\right)$.

As we can see on the figure 5.1, the acceptable space can be "covered" by a finite number of D-rectangles, and, therefore, the same (generalised) estimation approach could be applied. We can try to code an algorithm that does that, i.e. that can cover any connected acceptable space thanks to D-rectangles, and test it on some problems like the above one, for which the result can be directly computed. A future work?

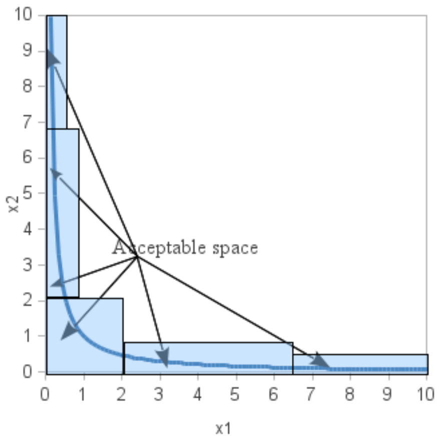


Figure 5.1: Product function ($a = 10$, $\varepsilon = 1$). Covering of the acceptable space by D-rectangles. The open question is “How to do it thanks to an algorithm?”

6 Appendix

6.1 More detailed results

	Schwefel 2.21		Ackley		Rosenbrock	Griew
D		approxim.		approxim.		
2	55.260049	55.2620422319	55.302678	55.3041796824	37.012832	35.02
3	82.893196	82.8930633478	82.631236	82.632754732	57.151654	52.47
4	110.522579	110.5240844637	109.913614	109.9152121843	76.880092	69.85
5	138.155248	138.1551055796	137.169684	137.1706959787	96.546279	87.23
6	165.786622	165.7861266956	164.408169	164.4084616505	116.22078	104.6
7	193.417895	193.4171478115	191.628667	191.6336950696	136.016977	121.9
8	221.045209	221.0481689274	218.842511	218.8495951156	155.88727	139.3
9	248.677717	248.6791900434	246.091253	246.0582742531	175.848609	156.7
10	276.311145	276.3102111593	273.22007	273.2612005928	195.934864	174.1
11	303.947487	303.9412322752	300.454166	300.4594358059	215.879029	191.4
12	331.575962	331.5722533911	327.539561	327.6537724912	235.939732	208.8
13	359.197231	359.2032745071	354.873567	354.8448180227	255.976937	226.1
14	386.834834	386.834295623	383.049241	382.0330480947	276.4605	243.5
15	414.46932	414.4653167389	408.994412	409.2188422215	296.582691	260.9

	Sphere		Rastrigin		Alpha-D-max	
D		exact		approxim.		exact
2	32.478106	32.4777557772	31.77626	31.7765365452	46.051832	46.05
3	49.060892	49.0013165363	47.948867	47.9494876883	62.170141	62.16
4	65.717632	65.6486587349	64.387733	64.246220271	73.682959	73.68
5	82.45845	82.3953630453	80.739574	80.6423149654	80.590167	80.59
6	99.27929	99.2250268008	97.179972	97.1213691049	82.890956	82.89
7	116.162388	116.125881791	113.719736	113.6716144791	80.590816	80.59
8	133.105588	133.0890769391	130.313968	130.2842000112	73.683374	73.68
9	150.123693	150.107714965	146.975234	146.9522284211	62.169415	62.16
10	167.192559	167.1762706287	163.686877	163.6701744688	46.051981	46.05
11	184.307793	184.2902188344	180.438309	180.4335130585	25.328462	25.32
12	201.456385	201.4457858751	197.221674	197.2384704833	0	0
13	218.598244	218.6397767885	214.059771	214.0818517806	0	0
14	235.884914	235.8694518014	230.939306	230.9609171776	0	0
15	253.207271	253.1324355862	247.851842	247.8732913464	0	0

References

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

[2] Maurice Clerc. *L'optimisation par essaims particulaires. Versions paramétriques et adaptatives.* Hermès Science, 2005.

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

[4] Mahamed G. H. Omran and Maurice Clerc. An adaptive population-based simplex method for continuous optimization. 2013 (submitted).