When Nearer is Better

Maurice Clerc

WHEN NEARER IS BETTER

Maurice Clerc
Maurice.Clerc@WriteMe.com
Draft 2007-05-18 (replace Draft 2007-03-19). Modified definition, and fixed some mistakes. A few new results.

Abstract. We define a numerical "nearer is better " truth value that can be computed or estimated for all functions on a given definition space. The set of all these functions can be then partitioned into three subsets: the ones for which this truth value is positive, the ones for which it is negative, and the ones for which is is null. Most of classical functions belong to the first subset, as the second one is useful to design problems that are deceptive for algorithms like Particle Swarm Optimisation. Also on these subset the No Free Lunch Theorem does not hold. Therefore it may exist a best algorithm, and we suggest a way to design it for the first one.

1. Introduction

It has been already shown that in a lot of optimisation scenarios there can be no such thing as a No Free Lunch [5]. Actually it has been proved that the NFL Theorem (NFLT) is valid if and only if the set of problems (functions) is closed under permutations (c.u.p.) and each target function is equally likely [10]. It has then been proved that the number of such c.u.p. subsets can be neglected compared to the overall number of possible subsets [6]. As a consequence, in the same work, the authors define some large classes of functions on which NFLT does not hold. For example, if the number of local minima of every function f in a class is constrained to be smaller than the maximal possible one, then this class is not c.u.p., and therefore NFLT does not hold.
However being c.u.p. or not is not a feature that can be easily used by optimisation algorithms. On the contrary, a "nearer is better" (NisB) property is almost always assumed: most of iterative stochastic optimisation algorithms, if not all, at least from time to time look around a good point in order to find an even better one. This can be mathematical defined, and summarised by a single real value for any function. What is interesting is that it can be estimated (or even exactly calculated for not too big problems). The functions for which this value is positive define a huge class, which in particular contains most of classical and less classical tests functions that we have checked by now. And what is even more interesting is that NFLT does not hold on this class: we can explicitly define an algorithm that is better than random search, and therefore it is not unrealistic to look for the best possible algorithm.
When negative this NisB value indicates that the corresponding function is very probably deceptive for most of algorithms that precisely assume that a vague nearer is better property is true. Particle Swarm Optimisation is typically such an algorithm, and we will see thanks to this analysis how easy it is to design functions on which its classical variations are worse than random search.

2. Notations and Definitions

2.1. Search space and problems.

As in [11], we have a search space X,of size $|X|$, a set of fitness values Y, of size$|Y|$. An optimisation problem f is identified with a mapping $f : X \to Y$ and $\mathcal{F}$ is the space of all problems. It may be useful to quickly recall under which conditions NFLT holds.

Condition 2.1. *For any position in X all values of Y are possible.*

In such a case the size of $\mathcal{F}$ is obviously $|Y|^{|X|}$. A optimisation algorithm A is generating a time ordered sequence of points in the search space, associated with their fitness's, called a *sample*.

Condition 2.2. *The algorithm A does not revisit previously visited points*

So an optimisation algorithm A can be seen as a permutation of the elements of X.

Condition 2.3. *The algorithm may be stochastic (random), but under Condition 2.2.*

That is why Random Search (let us call it R) in NFLT context is not exactly the usual one in which each draw is independent of the previous ones. Here R is defined not only by "drawing at random according to an uniform distribution" but also by "... amongst points not already drawn". It means that under Condition 2.2 any algorithm, including R, is in fact an exhaustive search.

2.2. **Performance.**
In NFLT context, the performance of A after m iterations for a given problem f is defined for any kind of sample, and assuming the following condition.

Condition 2.4. *The performance is only depending on the fitness values, and not on the positions in the search space.*

It means, for example, that NFLT does not hold if the performance measure takes into account the distance to the solution point. We are interested here only on samples that contain at least one solution point, i.e. here where f reaches its minimum value. Let f be a problem, and A an algorithm that samples x_{α_t}at "time step" t, assuming that a each time step another point is drawn. There is a probability $p(f,t,a)$ that the sampled point is a solution point. We compute the following expectation.

$$r(f,A) = \sum_{t=1}^{|X|} p(f,t,A)\, t \tag{2.1}$$

Roughly speaking it means that the algorithm finds a solution after "in average" $r(f,A)$draws. Then we say that algorithm A is better than algorithm B for the problem f if $r(f,A) < r(f,B)$, i.e. if in average A finds a solution quicker than B. On a whole set of functions $\mathcal{F}'$ we can then define the global performance by the following mean.

$$\boldsymbol{r}(\mathcal{F}', A) = \frac{1}{|\mathcal{F}'|} \sum_{P \in \mathcal{F}'} r(f,A)$$

NFLT claims that when averaged over *all* $|Y|^{|X|}$ functions, i.e. over $\mathcal{F}$, "Any algorithm is equivalent to Random Search". It means that $\boldsymbol{r}(\mathcal{F}, A)$ is the same for all algorithms, and its value is $(|\mathcal{F}|+1)/2$.
As we have seen, this is true only under very precise conditions, the most important being precisely that "all". It is easy to define subsets of functions on which it does not hold. We are now going to study two such subsets which have interesting practical applications.

3. Nearer is Better Truth Value

3.1. **Definition.**

We assume here that we are looking for a minimum. Let f be a function, x_b, x_w two positions in its definition domain, so that $f(x_b) \leq f(x_w)$, and δ a distance defined on this domain. Let us define two subdomains.

$$\begin{array}{rcl} N_{b,w} & = & \{x, 0 < \delta(x_b, x) < \delta(x_b, x_w)\} \\ B_{w,b} & = & \{x, x \neq x_b, x \neq x_w, f(x) \leq f(x_w)\} \end{array} \tag{3.1}$$

That is, $N_{b,w}$ is the set of points to which x_b is closer than to x_w, with a nonzero distance, and $B_{w,b}$the set of positions that are better than x_w (or equivalent to), except the two points already drawn. This last definition, which could be called "better than the worse" may seem strange at first glance. Why not "better than the best?". Because the corresponding class of functions would not contain some "obviously good" functions (see Annexe 8.6).

If we now choose x completely at random in $X - \{x_b, x_w\}$, the probability to find a position better than x_w is simply $|B_{w,b}| / (|X| - 2)$. Note that for a perfect random landscape the expectation of this variable is not $1/2$ as one could think, but $2/3$, because the condition $f(x_b) \leq f(x_w)$(see 8.1).

Now what happens if we choose x in $N_{b,w}$(assuming it is not empty)? The probability to find a better position than $x_w$is $|N_{b,w} \bigcap B_w, b| / |N_{b,w}|$. If this probability is greater than the previous one, this strategy may be interesting. We do not want it always happens, though, but just that it happens "on average". Let us call this method "NisB strategy". We are interested to know how much this NisB strategy is better than Random Search. To formalise this we define the set of acceptable pairs by.

$$\Omega = \{(x_b, x_w), x_b \neq x_w, f(x_b) \leq f(x_w), |N_{b,w}| > 0\} \tag{3.2}$$

and the two mean probabilities.

$$\left\{ \begin{array}{rcl} \xi(f) & = & \dfrac{1}{|\Omega|} \sum_{\Omega} \dfrac{|N_{b,w} \bigcap B_{w,b}|}{|N_{b,w}|} \\ \rho(f) & = & \dfrac{1}{|\Omega|} \sum_{\Omega} \dfrac{|B_{w,b}|}{|X| - 2} \end{array} \right. \tag{3.3}$$

Then we say "nearer is better" is true for the function f iif.

$$\nu(f) = \xi(f) - \rho(f) > 0 \tag{3.4}$$

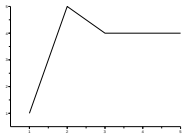
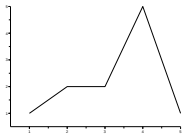
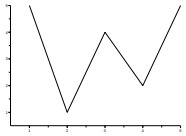
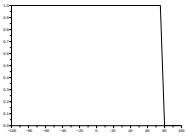
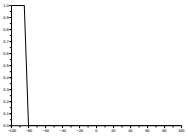
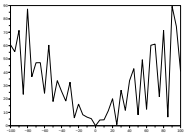
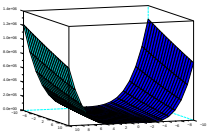
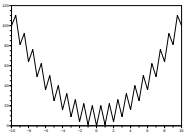
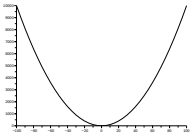
It can be seen as an estimation of how much the NisB strategy is better than Random Search on f. Or, intuitively, it can seen as an estimation of how much "nearer is better not just by chance". Note that $\nu(f)$ may be null even for not constant functions, when $|X|$ is small. For example, on dimension 1, it is the case with $X = (1, 2, 3, 4)$and $f(x) = (4, 2, 3, 4)$. This is because if we consider the "contribution" of a given pair.

$$\eta_{b,w}(f) = \frac{|N_{b,w} \bigcap B_{w,b}|}{|N_{b,w}|} - \frac{|B_{w,b}|}{|X| - 2}$$

it is obviously equal to zero when $f(x_w)$ is maximum. And in this example all the four possible pairs contain a x_w so that $f(x_w) = 4$.

From now on we call $\mathcal{F}^+$ the set of functions on X for which $\nu(f) > 0$, $\mathcal{F}^-$the set of functions for which $\nu(f) < 0$, and $\mathcal{F}^=$ the set of functions for which $\nu(f) = 0$. When $|X|$ is not too big this truth value can be computed by exhaustively considering all pairs of points, as in Table 2, but most the time we just estimate it by sampling. For simplicity we assume here discrete domains are defined by an interval and a "granularity". To build the table we used the following distance between two points $x = (x_1, ..., x_D)$ and $x' = (x'_1, ..., x'_D)$, sometimes noted L_∞:

$$\delta(x, x') = max(|x_i - x'_i|) \tag{3.5}$$

Function f	Domain	Granularity,$\lvert X\rvert$	Landscape	$\xi(f)$	$\nu(f)$
F1188	$[1,5]$	1, 5		0.67	-0.10
F171	$[1,5]$	1, 5		0.50	-0.07
F2585	$[1,5]$	1, 5		0.86	0.00
Step, plateau 90%	Step, plateau 90%	5,41		0.92	0.01
Step, plateau 10%	$[-100,100]$	5,41		0.995	0.05
Alpine	$[-100,100]$	5, 41		0.78	0.09
Rosenbrock	$[-10,10]^2$	1, 441		0.80	0,13
Rastrigin	$[-10,10]$	0.5, 41		0.83	0.15
Parabola	$[-100,100]$	5, 41		0.91	0.23

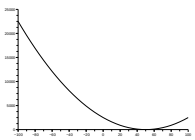
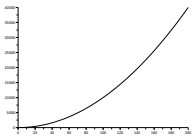
Function f	Domain	Granularity,$\lvert X\rvert$	Landscape	$\xi(f)$	$\nu(f)$
Parabola, offset 50	$[-100, 100]$	5, 41		0.96	0.29
Parabola	$[0, 200]$	5, 41		1.00	0.33

TABLE 2. "Nearer is Better" truth value $\nu(f)$ for some functions, and how much a pure NisB algorithm is better than Random Search

Of course on dimension one it is equal to the Euclidean distance. The table shows that $\mathcal{F}^-$is not empty: we can have a "nearer is worse" effect, i.e. a deceptive function (actually as $\mathcal{F}^+$ is not empty this is anyway a consequence of the NFLT). See F1188, F171 (c.f. 8.2.3 for more information about these functions). In such a case Random Search may be better than a more sophisticated algorithm.

3.2. **Probability to find the optimum.**

Assuming there are k solution points x^*(i.e. where the function reaches its optimum), we draw at random two points. One is x_b, and the other one is x_w, with $f(x_b) \leq f(x_w)$. We consider just the case where none of them is a x^*, (we will see why in 3.3). It implies in particular that all x^* are in $B_{w,b}$. Now we draw a third point x. If we draw it at random the probability to find a $x^*$is then.

$$p_1 = \frac{k}{|X| - 2}$$

However, if we choose it (at random) in $N_{b,w}$, the probability to find a x^* is the product of.

- $k\frac{|N_{b,w} \bigcap B_{w,b}|}{|B_{w,b}|}$, probability that at least one $x^*$is in $N_b \bigcap B_w$
- $\frac{1}{|N_{b,w}|}$, probability to draw a precise x in $N_{b,w}$

i.e. $p_1 = k\frac{|N_{b,w} \bigcap B_{w,b}|}{|N_{b,w}||B_{w,b}|}$.

Therefore, $p_2 > p_1$is equivalent to $\frac{|N_{b,w} \bigcap B_{w,b}|}{|N_{b,w}|} > \frac{|B_{w,b}|}{|X|-2}$. For a function in $\mathcal{F}^+$, we have then.

$$probability\,(p_2 > p_1) > 0.5 \tag{3.6}$$

In short, when choosing a point near to x_b we have a better chance to find a solution than when choosing it at random.

3.3. **Better than Random Search.**

Theorem 3.1. On the "Nearer is Better" class $\mathcal{F}^+$ there is at least one algorithm better than Random Search

Let f be a problem (function) belonging to $\mathcal{F}^+$. Let us consider the (non repeating) random algorithm R. On a given run it samples $x_{\alpha_1}, x_{\alpha_2}, x_{\alpha_3}, ...$ Let R_+be another algorithm that on a given run samples $x_{\beta_1}, x_{\beta_2}, x_{\beta_3}, ...$ We define it as similar to R, except sometimes for the choice of one point:

- $x_{\beta_1}$is chosen at random (uniform distribution) in X
- $x_{\beta_2}$is chosen at random (uniform distribution) in $X - \{x_{\beta_1}\}$
- if x_{β_1}or $x_{\beta_2}$is a solution point (or both), for this precise run $R_+$is simply R
- else let $x_b$be the best of the two positions $(x_{\beta_1}, x_{\beta_2})$, and x_w the worst of these two positions. We choose x_{β_3}in $N_{b,w}$.
- After that, for $k = 4$ to X, $x_{\beta_k}$is again chosen at random (uniform distribution) in the remaining positions

The idea is to prove that $R_+$is better than R, not on a given run, but averaged over all possible ones. For simplicity we suppose here there is just one solution point. Note that at any time t we have.

$$\left\{ \begin{array}{lcl} p(f,t,R) & = & \left(1-\frac{1}{|X|}\right)\left(1-\frac{1}{|X|-1}\right)...\left(1-\frac{1}{|X|-t+1}\right)\frac{1}{|X|-t} \\ & = & \frac{1}{|X|} \end{array} \right.$$

Now, when one of the two first points is a solution point, R and R_+ are equivalent, by definition. Else, at time 3, $p(f,R_+,3)$ is not anymore equal to $1/|X|$. It can be written.

$$\left\{ \begin{array}{lcl} p(f,3,R_+) & = & \frac{|X|-1}{|X|}\frac{|X|-2}{|X|-1}\left(\frac{1}{|X|-2}+\varepsilon\right) \\ & = & \frac{1+\varepsilon(|X|-2)}{|X|} \end{array} \right. \tag{3.7}$$

Let us consider the case $\varepsilon > 0$. At time $t > 3$, the probability to draw a solution is again constant, but equal to.

$$\frac{1}{|X|}\left(1-\varepsilon\frac{|X|-2}{|X|-3}\right) \tag{3.8}$$

So to compare the performances, we just have to evaluate.
(3.9)

$$\begin{array}{lcl} r(f,R)-r(f,R_+) & = & 3\left(\frac{1}{|X|}-p(f,3,R_+)\right)+\sum_{t=4}^{|X|}\left(\frac{1}{|X|}-\frac{1}{|X|}\left(1-\varepsilon\frac{|X|-2}{|X|-3}\right)\right)t \\ & = & \frac{-3\varepsilon(|X|-2)}{|X|}+\frac{\varepsilon(|X|-2)}{|X|(|X|-3)}\sum_{t=4}^{|X|}t \\ & = & \varepsilon\frac{|X|-2}{|X|}\left(\frac{|X|+4}{2}-3\right) \\ & > & 0 \end{array}$$

It means that $R_+$is better than R for this precise run. Now for another run it may happen that $\varepsilon \leq 0$. However, because of inequality 3.6, the probability of this event is smaller than 0.5. So, over all possible different runs, $R_+$is better than R with a probability greater than 0.5. Of course, it is still a quite bad algorithm, but now we know that not all algorithms are equivalent, it is worth looking for the best one. In order to do that, we are now considering the distribution of solutions points for functions of $\mathcal{F}^+$.

3.4. **Characteristic distribution.**

3.4.1. *Definition and examples.*

For each position x_i in X we can count how many times it is a solution point when considering all functions of $\mathcal{F}^+$. Let $n(x_i)$ be this number and let us define.

$$s(x_i) = \frac{1}{\left|\mathcal{F}^+\right|}n(x_i) \tag{3.10}$$

i	1	2	3	4
$n(x_i)$	15	21	21	15
S^+	0.38	0.53	0.53	0.38

TABLE 3. Distribution of solution positions for Example 3.3

This can be seen as the probability to have a solution point in x_i when choosing at random (uniform distribution) a function f in $\mathcal{F}^+$. So we define a *characteristic distribution* $S^+ = \left(s(x_1), ..., s\left(x_{|X|}\right)\right)$. Note that it is not a normalised probability distribution, for some functions have several solutions points, and therefore $\sum_i s(x_i) > 1$. We suggest the following property is true.

Conjecture 3.2. The characteristic distribution of the $\mathcal{F}^+$ class is unimodal (possibly with a plateau) with the minimum on the bound

Note that if the search space is symmetrical, and has "centre", like in the examples below, then the characteristic distribution is obviously also symmetrical around this centre.
This conjecture is well supported by experiments. Let us give some examples on dimension 1 and 2.

Example 3.3.

$X = (1, 2, 3, 4)$, $Y = (1, 2, 3, 4)$. $|\mathcal{F}| = 256$, $|\mathcal{F}^+| = 40$

Example 3.4.

$X = (1, 2, 3, 4, 5)$, $Y = (1, 2, 3, 4, 5)$. $|\mathcal{F}| = 3125$, $|\mathcal{F}^+| = 1090$

i	1	2	3	4	5
$n(x_i)$	205	411	478	411	205
S^+	0.19	0.38	0.44	0.38	0.19

Example 3.5.

$X = ((1,1), (2,1), (3,1), (1,2), (2,2), (3,2))$, $Y = (1, 2, 3, 4, 5, 6)$. $|\mathcal{F}| = 46656$, $|\mathcal{F}^+| = 18620$

i	1	2	3
	4	5	6
$n(x_i)$	3963	6580	3963
	3963	6580	3963
S^+	0.212	0.352	0.212
	0.212	0.352	0.212

When using Euclidean distance, instead of the one defined by 3.5, we have. $|\mathcal{F}^+| = 21772$

i	1	2	3
	4	5	6
$n(x_i)$	4739	7636	4739
	4739	7636	4739
S^+	0.213	0.344	0.213
	0.213	0.344	0.213

3.4.2. *Practical computation.*
Computing a characteristic distribution is quite time consuming, as soon as the search space is not very small. However, $\mathcal{F}$ can be divided into equivalence classes, on which most of algorithms have the same behaviour (see Annexe 8.2 for details). All the functions in a class have the same NisB truth value , and the same "profile". Actually it would be even possible to define fuzzy classes, but it is outside the scope of this paper.

The point is that as soon as we have defined these equivalence classes, we can work on them to compute the characteristic distribution. On the whole the process is far cheaper than a direct computation. That is why even the small tables for Example 3.4 and Example 3.5 have been built by using this method.

4. The best algorithm?

4.1. **Permutational algorithms.**

4.1.1. *Definition.*
As we have seen any run of a non repeating algorithm can be identified to a permutation of the elements of X , say $\left(x_{\alpha_1}, x_{\alpha_2}, ..., x_{\alpha_{|X|}}\right)$. The algorithm is called *permutational* if this permutation is always the same, for any function and for any run (no randomness, in particular). For example, on $X = (1, 2, 3, 4)$ the algorithm $A \equiv (4, 3, 2, 1)$ is "draw $x_4$then $x_3$then $x_2$then x_1".

4.1.2. *Performance conjectures.*
On $\mathcal{F}^+$ we evaluate the global performance for each permutational algorithm, and call $\boldsymbol{r_b}(\mathcal{F}^+)$ the best one (the minimum one). We suggest the following property is true.

Conjecture 4.1. On $\mathcal{F}^+$ there exists at least one permutational algorithm A so that $\boldsymbol{r}(\mathcal{F}^+, A) > \boldsymbol{r_b}(\mathcal{F}^+)$

It means that not all permutational algorithms are equivalent: at least one is strictly worse than the best one. For Example 3.3, we have $|\mathcal{F}| = 256$, $|\mathcal{F}^+| = 40$, and there are $4! = 24$ permutational algorithms. Their best global performance on $\mathcal{F}^+$ is $\boldsymbol{r_b}(\mathcal{F}^+) = 1.85$ (by running for example $A \equiv (2, 3, 1, 4)$). Not surprisingly, this is smaller (better) than that on the whole $\mathcal{F}$ (global performance 2.5). However there are several permutational algorithms with a strictly worse performance than the best one. For example with $A \equiv (1, 2, 3, 4)$ we have $\boldsymbol{r}(\mathcal{F}^+, A) = 2.1$. The conjecture says that this is true for any $\mathcal{F}$. Note that it is weaker than (and implied by) Conjecture 3.2 so we may quite safely suggest a more interesting one.

Conjecture 4.2. On $\mathcal{F}^+$ there exists no better algorithm than the best permutational one

This conjecture says nothing about how to find a best algorithm without having to try all of the permutational ones. However under Conjecture 3.2 it is indeed possible.

4.2. **A candidate.**

We explain here how to design the best permutational algorithm (or more precisely one of the possible ones for there usually are several equivalent ones). The idea is very simple: we build a permutation by choosing at each step the most probable solution position not already chosen. Although it obviously finally indeed gives the best algorithm, this method is impracticable as usually we do not know the characteristic distribution. However, under Conjecture 3.2 this method can be replaced by the following one for a symmetric search space of "gravity centre" G:

Algorithm 4.3. R_{++} - *Finding the best permutation*

At each step, choose the position (not already chosen) which is the nearest one to G (the centre of the search space). In case of equivalence, choose at random

4.2.1. *Examples.*
For Example 3.3 this method builds the following possible permutations: $(2, 3, 1, 4)$, $(2, 3, 4, 1)$, $(3, 2, 1, 4)$, $(3, 2, 4, 1)$
All the corresponding permutational algorithms have a global performance on $\mathcal{F}^+$ equal to 1.85, the best possible one.

For Example 3.5, one of the possibilities is $(2, 5, 1, 6, 3, 4)$, and the corresponding permutational algorithm has a global performance on $\mathcal{F}^+$ equal to 2.55, which is also the best one.

5. Some practical consequences

Although our analysis is mainly a theoretical one it gives some interesting enlightments and suggests some applications. Here are some of them.

5.1. **About centre bias.**

The bigger X the bigger $\mathcal{F}^+$. It means that R_{++} tends to build an algorithm that is simply "*At each step choose a point near to the centre of the search space*". Although still the best algorithm averaged on the whole $\mathcal{F}^+$, it is of course quite bad on any classical benchmark set, which is always far smaller than $\mathcal{F}^+$, and when compared to more specific algorithms. However it nevertheless shows that for such algorithms having a bias in favour of the centre of the search space is not necessarily a weakness, on the contrary. It may explain for example why Particle Swarm Optimisation (PSO), which is indeed biased [8], is so robust. How important should be this bias has nevertheless still to be found.

5.2. **Divide and conquer.**

According to 4.2 there exists a best algorithm A on $\mathcal{F}^+$, and at least another one B that is strictly worse. Now let us consider the "Nearer is worse " subset $\mathcal{F}^-$. According to the NFLT A and B are equivalent on $\mathcal{F}$. So B is necessarily better than A on $\mathcal{F}^-$. It implies that there exists a best algorithm on $\mathcal{F}^-$. In other words $\mathcal{F}$ can be partitioned into two classes, each one corresponding to a interval of the NisB truth value ν, and for each one there exists a best algorithm. Whether it is true or not for any interval, and, more important, if it is still possible to explicitly define the corresponding best algorithm, is an interesting open question. Note that the answer is positive for some particular cases, for example for $\nu = 1$.

5.3. **Benchmarks.**

For an algorithm that makes use of the NisB property, it may be a good idea to design a benchmark as diversified as possible from this point of view, i.e. whose NisB truth values cover a large interval. Indeed, when computing the NisB truth values for existing benchmarks it appears they not "cover" all the possible values. In particular they are almost never negative.
Moreover for most of usual functions we can just estimate the NisB truth value and not exactly compute it, as knowing the exact values would useful for a better classification of algorithms. Fortunately there is a simple way is to design "on demande" functions by starting from small discrete ones and by transforming them into piece-wise ones (see Annexe8.5).

5.4. **Adaptive algorithms.**

We are particularly interested here on iterative adaptive algorithms that can modify their search strategy according to what they learn about the fitness function during the sampling process. Note that algorithms that perform only *partial* adaptations may fail badly, as proved in [4], so it would be better to choose a completely parameter-free method, like Tribes, a PSO variation [2, 3, 9].
The idea is the following: from time to time, or even at each time step, the algorithm computes the NisB truth value $\nu(f)$ of the function f, as it "sees" it thanks to the sampled points, and and it modifies its search strategy according to this value. In Tribes, for example, each particle has to choose between three or four strategies, and the choice is depending one several criteria (status of the particle, status of the tribes it belongs to, etc.). So $\nu(f)$ could be another criterion.

Some strategies are better for "exploration", some others better for "exploitation". Typically, a high $\nu(f)$ value should favours exploitation, and vice-versa. In other words an adaptive algorithm may perfectly switch from a "nearer is better" assumption to a "nearer is worse" one, and vice-versa. Note that this approach is already empirically used by some algorithms that try to take into account unsuccessful sampled positions either by direct computation or thanks to a repulsive "force" [7, 1]. Because of NFLT such an algorithm can not be good for all functions of $\mathcal{F}$ but it could be effective for a class of functions that contains some functions of $\mathcal{F}^+$ and some functions of $\mathcal{F}^-$.

6. Generalisations

For simplicity we have assumed that both X and Y are finite. Extending our definitions to infinite or unbounded ones is straightforward. Actually some formulae may even be simplified for, for example, the measure of an infinite space is the same when you remove just two points.

6.1. **Infinite bounded search space.**

Here "infinite" means "contains an infinite number of points". A typical case is $X = [x_{min}, x_{max}]^D$. Formulae 3.1 and 3.2 that define $N_{b,w}$, $B_{w,b}(f)$ and Ω are still valid. We do assume that the measures like $|X|$ have a meaning. For example, with the above X we have $|X| = |x_{max} - x_{min}|^D$. Then formulae 3.3 that define ξ and ρ could easily be replaced by two integrals. However in order to avoid some infinite quantities, it is better to directly compute the NisB truth value ν by the following formula:

$$\nu(f) = \int_{\Omega} \left(\frac{|N_{b,w} \bigcap B_{w,b}|}{|N_{b,w}|} - \frac{|B_{w,b}|}{|X|} \right)$$

Are the properties we have found still true? Probably yes, although of course it has to be proved. The reason is that what is important is the unimodality of the distribution of the solutions points, and it is only depending on the geometry of the search space, and of the definition of $\mathcal{F}^+$, which is still the same (set of functions f so that $\nu(f) > 0$), no matter X and Y are finite or not, bounded or not.
Let $\theta(x, r) = \{z, z \in X, \delta(x, z) < r\}$ be the set of points in X that are also in the "sphere" of centre x and radius r, and G the "gravity centre" of X. Then, more precisely, this unimodality is probably true as soon as the following condition holds:

Condition 6.1. For any pair of points (x, x'), for any "radius" r, we have

$$\begin{cases} \delta(x, G) > \delta(x', G) \Rightarrow |\theta(x, r)| \leq |\theta(x', r)| \\ \exists \delta_{max}, \delta_{max} > \delta(x, G) > \delta(x', G) \Rightarrow |\theta(x, r)| < |\theta(x', r)| \end{cases}$$

Roughly speaking, it means that a point "near" the bound has less "neighbours" than a point near the centre. In particular this condition is true for a bounded convex space. This generalisation is particularly interesting to design deceptive functions on a continuous search space (see Annexe 8.5): any algorithm that assumes that "nearer is better" is more or less true will be worse than Random Search on any function of $\mathcal{F}^-$.

6.2. **Unbounded search space.**

Some iterative optimisation algorithms are allowed to sample the definition space of the function outside an initially given search space X. It is typically the case for some PSO variations. However in such a case the sampled position is usually simply not evaluated. It means that the user is sure there is no solution outside X, or, at least, is not interested on any solution that could be outside X.
Then the above remark about Condition 6.1 is still valid, for anyway all points that are really taken into account are still inside X.

6.3. **A more general formulation.**

Finally a reasonable aim for a future work may be to prove the following.

Conjecture 6.2. *When taking into account only solutions points that are inside a convex space X, their distribution for functions in $\mathcal{F}^+$ is unimodal, with its minimum on the bound.*

7. Conclusion

For optimisation algorithms that use the "Nearer is Better" property that is defined here, i.e. for most of them, there is an infinite class of problems on which there are algorithms better than Random Search, and therefore it is worth looking for the best one. We suggest a candidate. Of course, being the best relatively to the whole "Nearer is Better" class usually means being quite bad on a specific sub-class when compared to classical optimisation algorithms. In particular all commonly used benchmarks are such sub-classes. However preliminary results seem to show that each of these benchmark can be included in a sub-class that is characterised by a positive threshold for the "Nearer is Better" truth value. It may be a hint to explicitly design the best algorithm for each benchmark set. Or, conversely, the NisB truth value may be a tool to design benchmarks that contains functions that are deceptive for most of optimisation algorithms.

8. Annexe

8.1. **Perfect Random Landscape.**

8.1.1. *Definition.*

Let us define a perfect random distribution f from X to Y as follows:

- f is surjective
- the density probability of f is 1, i.e. for any u in Y, $probability\left(f\left(x\right) < u\right) = u$

8.1.2. *Properties.*

Let X' be a subset of X, $f_{min,X'}$and $f_{max,X'}$the minimum and the maximum values of $f$on X'. Then we have.

Fact 8.1. The expectation is given by

$$\int_{X'} xf\left(x\right)dx = \left|X'\right| \frac{f_{min,X'} + f_{max,X'}}{2} \tag{8.1}$$

where $|X'|$ is the measure of X' consistent with the integration, i.e. $|X'| = \int_{X'} dx$. In particular the expectation of f is.

$$E\left(f\right) = \frac{1}{2}$$

Fact 8.2. The probability density of $f^{-1}$is equal to 1.

In particular, it means that for any given a in $[0, 1]$ the measure of the set of points x for which $f\left(a\right) \leq f\left(x\right)$is given by:

$$\left|\left\{x, f\left(a\right) < f\left(x\right)\right\}\right| = 1 - f\left(a\right)$$

and similarly we have.

$$\left|\left\{x, f\left(x\right) \leq f\left(a\right)\right\}\right| = f\left(a\right)$$

8.1.3. *Example.*

Δx	$E(g)$
0.5	1
0.4	0.8
0.3	0.675
0.2	0.673
0.1	0.634
0.05	0.680
0.01	0.669

TABLE 4. Mean probability of improvement relatively to $f(x_w)$, knowing that $f(x_w) \geq f(x_b)$, with (x_b, x_w)drawn at random, for non repeating random search on $[0,1]$, for a perfect random landscape and for different granularities Δx

For simplicity, we choose $X = [0,1]$, $Y = [0,1]$, but the final result of this example is valid for any other intervals. For a given pair (x_1, x_2) of elements of X, we define.

$$\left\{ \begin{array}{lcl} X'(x_1) & = & \{x, x \in, f(x) \leq f(x_1)\} \\ X'(x_2) & = & \{x, x \in, f(x) \leq f(x_2)\} \end{array} \right.$$

i.e. $X'(x_i)$is the set of points that are "better than" x_i from a minimisation point of view. If we draw a point at random in X, according to an uniform distribution, the probability of improvement, relatively to $f(x_i)$, is then $|X'(x_i)| = f(x_i)$. Now let us consider the following function $g(x_1, x_2)$, defined on $X \times X$:

- if $f(x_1) \leq f(x_2)$, $g(x_1, x_2) = |X'(x_2)|$
- if $f(x_2) \leq f(x_1)$, $g(x_1, x_2) = |X'(x_1)|$

In other words, for each pair of points, we consider the probability of improvement relatively to the worst one, if we draw at random a third point. We can compute the mean value (the expectation) of this probability, over all pairs. For symmetry reason we have.

$$\left\{ \begin{array}{lcl} E(g) & = & 2\int_0^1 x_1 \left(\int_{\{x_2, f(x_1) \leq f(x_2)\}} f(x_2)\, x_2 dx_2 \right) dx_1 \\ & = & 2\int_0^1 x_1 (1 - f(x_1)) \frac{1+f(x_1)}{2} dx_1 \\ & = & \int_0^1 x_1 \left(1 - f(x_1)^2\right) dx_1 \\ & = & E\left(1 - f^2\right) \end{array} \right.$$

We can directly compute the expectation of $1-f^2$ by noting that *probability* $\left(1 - f^2(x) < u\right) =$ $1 -$ *probability* $\left(f(x) < \sqrt{1-u}\right)$. So the density is $1/\left(2\sqrt{1-u}\right)$. We have then.

$$\left\{ \begin{array}{lcl} E\left(1 - f^2\right) & = & \frac{1}{2}\int_0^1 \frac{u}{\sqrt{1-u}} du \\ & = & -\frac{1}{3}\left[(u+2)\sqrt{1-u}\right]_0^1 \\ & = & \frac{2}{3} \end{array} \right.$$

So, finally.

$$E(g) = \frac{2}{3} \simeq 0.667 \tag{8.2}$$

8.1.4. *Discrete finite case.*
On a discrete finite space, formula 8.1 is not exactly valid, and gives just an estimation. Consequently, a formula like **??** is also just an estimation. In particular if the definition space is given by an interval (say still $[0,1]$) and a "granularity" Δx, we obtain different values, depending on Δx, as we can see on table 4.

8.2. **Equivalences and simplifications.**

8.2.1. *Equivalent functions.*

Let us consider two functions defined on the same search space $X = (1, 2, 3, 4)$by.

$$\begin{cases} f &= (3, 4, 2, 1) \\ g &= (2, 10, 1, 0) \end{cases}$$

They have the same "profile", and this can be formalised. Two functions f and g on X are *equivalent* if for *any* pair of points (x, x') of $X \times X$ we have $f(x) < f(x') \Leftrightarrow g(x) < g(x')$. In particular, if $f(x) = f(x')$ we also have $g(x) = g(x')$.
We are mainly interested here on optimisation algorithms that make use only of relationships between function values in order to choose the next point to draw. Typically something like "if $f(x) < f(x')$ then draw a new point according to Rule 1", where Rule 1 is depending only on already drawn positions, and not on some values of the function. A classical example is Particle Swarm Optimisation. Running such an algorithm on two equivalent functions produces two identical sequences of positions (assuming the same pseudo-random number generator is used).
Now, instead of considering all possible functions on X, we can consider just one representative for each equivalence class. From now on, we will work on such a set of functions, and we call it $\tilde{\mathcal{F}}$. When speaking of a function, we in fact speak of any representative of an equivalence class.

8.2.2. *Equivalence class coding.*
Let us assign a rank to each position:

$$X = (x_1, ..., x_i, ...x_m)$$

and let us consider a function f on X. To each pair (x_i, x_j) where $i > j$ we assign the number $j + (i-1)m$. There are $K = m(m-1)/2$ such pairs. The profile of the function is defined by a K-vector r as follows.

- pairs are sorted by lexicographic order
- for the pair(x_i, x_j) of rank k, we define

$$\begin{cases} r(k) &= 0 \text{ if } f(x_i) = f(x_j) \\ &= 1 \text{ if } f(x_i) < f(x_j) \\ &= 2 \text{ if } f(x_i) > f(x_j) \end{cases}$$

Then the profile r can be "summarised" by an unique number Π:

$$\Pi(f) = \sum_{k=1}^{K} r(k) 3^{k-1}$$

Note that a lot of codes are not used, for the "dominance" relations like $f(x_i) < f(x_j)$ are not independent. So the number of equivalence classes is not 3^K, but far smaller, depending on the dimensionnality of the search space, on its size, and on the size of the value space.

8.2.3. *An example of classes of a "Nearer is Better" subset.*
Let us consider the set of all possible unidimensional functions on X with values in Y, with.

$$\begin{cases} X &= (1, 2, 3, 4, 5) \\ Y &= \{1, 2, 3, 4, 5\} \end{cases}$$

We can call in short this example X5Y5. There are $|Y|^{|X|} = 3125$ such functions.They can be easily generated by a program and we have numbered them from F1 to F3125. Note that, by definition, the equivalence classes are not depending on $|Y|$ as soon as it is at least equal to $|X|$. Of course, however, the number of representatives in each class *is* depending on $|Y|$. Here the number of the classes is 541. In Table 5 we present some of them, along with the ν and ρ values. Assigning these values to a whole class is possible thanks to the consistency theorem 8.3 (see below). On figure 8.1 we can also see the 541 $(\xi(f), \rho(f))$ points. Such a figure

$\xi(f)$	$\rho(f)$	Example	%	$\xi(f)$	$\rho(f)$	Example	%
1	1	F33: $(1,1,2,2,3)$	0.16%	0.62	0.62	F1456: $(3,2,4,2,1)$	0.16%
0.95	0.81	F1377: $(3,2,1,1,2)$	0.32%	0.60	0.63	F128: $(1,2,1,1,3)$	0.32%
0.91	0.78	F626: $(2,1,1,1,1)$	0.32%	0.48	0.52	F1331: $(3,1,4,2,1)$	0.16%
0.90	0.79	F1382: $(3,2,1,2,2)$	0.32%	0.52	0.57	F761: $(2,2,1,3,1)$	0.32%
0.88	0.81	F778: $(2,2,2,1,3)$	0.32%	0.62	0.67	F1661: $(3,4,2,3,1)$	0.16%
0.81	0.71	F1429: $(3,2,3,1,4)$	0.16%	0.45	0.52	F1291: $(3,1,2,4,1)$	0.16%
0.76	0.67	F2833: $(5,3,4,2,3)$	0.03%	0.47	0.56	F1791: $(3,5,2,4,1)$	0.03%
0.93	0.90	F1403: $(3,2,2,1,3)$	0.32%	0.67	0.72	F966: $(2,3,4,4,1)$	0.16%
0.78	0.72	F2679:$(5,2,3,1,4)$	0.03%	0.36	0.48	F411: $(1,4,2,3,1)$	0.16%
0.86	0.83	F753: $(2,2,1,1,3)$	0.32%	0.69	0.75	F762: $(2,2,1,3,2)$	0.32%
0.67	0.62	F1452: $(3,2,4,1,2)$	0.16%	0.73	0.75	F903: $(2,3,2,1,3)$	0.32%
0.62	0.57	F679: $(2,1,3,1,4)$	0.16%	0.69	0.76	F901: $(2,3,2,1,1)$	0.32%
0.69	0.67	F2010: $(4,2,1,3,2)$	0.12%	0.67	0.75	F162:$(1,2,2,3,2)$	0.32%
0.69	0.67	F1386: $(3,2,1,3,1)$	0.32%	0.35	0.52	F256:$(1,3,1,2,1)$	0.32%
...	...	...	...	...	...	...	...
		Total $\xi(f) > \rho(f)$	35%			Total $\xi(f) \leq \rho(f)$	65%

TABLE 5. Some examples of equivalence classes for the X5Y5 example

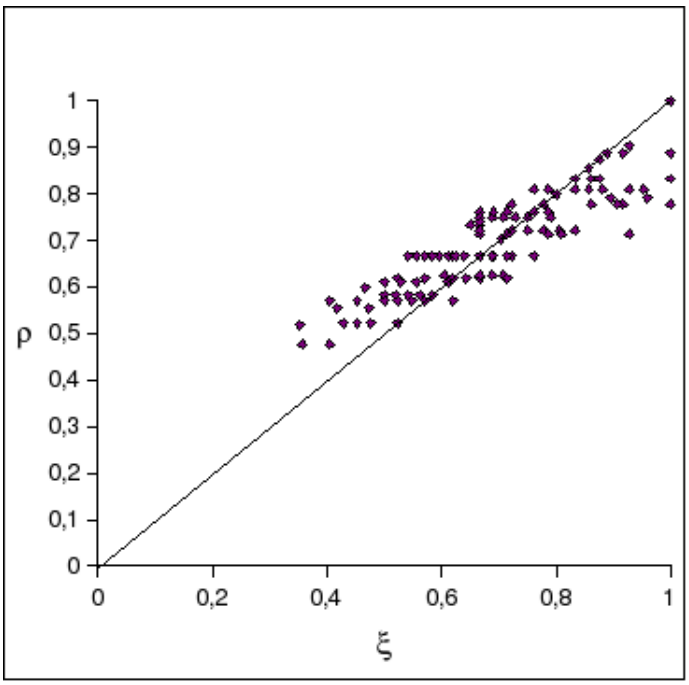


FIGURE 8.1. The 541 $(\xi(f), \rho(f))$ points of the X5Y5 example

suggests that there exists a positive minimum for these both functions, and also for their difference ν (see the Open question 8.7.1).

8.2.4. *Consistency theorems.*

Theorem 8.3. *If two functions f and g are equivalent, then $\xi(f) = \xi(g)$, and $\rho(f) = \rho(g)$*

Actually this is almost obvious. To evaluate the NisB truth value ξ, we need to build some $N_{b,w}$ and $B_{w,b}$ sets, as defined by 3.1. Let us consider two equivalent

Profile						Class size	Class codeΠ	Representative	$n(1)$	$n(2)$	$n(3)$	$n(4)$
2	2	2	0	0	0	6	26	(3,2,2,2)		6	6	6
2	2	2	2	2	0	4	242	(4,2,1,1)			4	4
2	2	2	2	0	1	4	323	(4,2,1,2)			4	
0	0	1	0	1	1	6	333	(1,1,1,4)	6	6	6	
1	1	1	0	1	1	4	337	(2,3,3,4)	4			
2	0	1	1	1	1	4	362	(3,1,3,4)		4		
0	1	1	1	1	1	4	363	(2,2,3,4)	4	4		
1	1	1	1	1	1	1	364	(1,2,3,4)	1			
2	1	1	1	1	1	1	365	(2,1,3,4)		1		
2	2	2	2	2	1	1	485	(4,3,1,2)			1	
2	2	2	0	2	2	4	674	(4,2,2,1)				4
2	2	2	2	2	2	1	728	(4,3,2,1)				1
								Total	**15**	**21**	**21**	**15**

TABLE 6. $\tilde{\mathcal{F}}^+$ list for Example 3.3. For each class the size is given, the Π code, and also a representative function. How to compute the characteristic distribution is given in the four last columns

functions f and g. A $N_{b,w}$ set is depending on positions, not on functions values. A $B_{w,b}$ set is depending on a relationship like $f(x) \leq f(x')$. As we have $f(x) \leq f(x') \Leftrightarrow g(x) \leq g(x')$, the set is the same when considering f or g. Same reasoning, even simpler, for ρ.

Theorem 8.4. *For a given profile (i.e. a given equivalence class) there is just one possible set of positions of the minima*

Or, in other words, no matter which representative is chosen in an equivalence class, the positions of the minima are always the same. Again, it is almost obvious. Let us consider a given function f. Let X_{min} be the set of positions of the minima of this function. It means that for all pairs of points (x_i, x_j) of X_{min} the relationship is $f(x_i) = f(x_j)$, and for all pairs of points $(x_i, x_j)$with x_i in $X - X_{min}$ and x_j in X_{min} we have $f(x_i) > f(x_j)$. For an equivalent function g the relationships are exactly the same, and it proves that any minimum of f is a minimum of g. The same reasoning shows that any minimum of g is a minimum of f.
Thanks to these consistency theorems, it is easier to compute the characteristic distribution. Let us consider again Example 3.3, with $X = (1, 2, 3, 4)$, and $Y = (1, 2, 3, 4)$. There are 75 equivalence classes, and 12 of them have the NisB property. The list is given in Table 6 . Now to compute the characteristic distribution, we can do it by considering just one representative by class, i.e. 12 functions, and by "weighting" each solution position by the corresponding class size. Of course we have first to compute the classes, but it is worth doing it, for in the whole process the most time consuming operation is to compute the NisB truth value for each function.
From Table 3, which shows the number of solution points on each position, we can derive the distribution $D^+ = (0.33, 0.42, 0.42, 0.33)$.

p	$\left\|\mathcal{F}^{+}\right\|$
1	20788
1.5	21772
2	21772
2.5	21772
10	21772
∞	18620

TABLE 7. Nearer is Better class size for different kinds of distance, and for Example 3.5

8.3. **About distances.**
We have seen that the Nearer is Better class $\mathcal{F}^{+}$ is depending of the kind of distance (at least as soon as dimension is greater than 1). Let us consider the general formula of the Minkowsky distance of order p (p-norm distance):

$$L_p\left(x,x'\right)=\left(\sum_i\left|x_i-x_i'\right|^p\right)^{1/p}$$

When p tends to infinity, we obtain the Chebyshev ("max") distance defined in 3.5. $L_1$is the taxicab or Manhattan distance, and L_2 the classical Euclidean one. In any case there is a p value (not necessary an integer, but greater than 1) for which $\left|\mathcal{F}^{+}\right|$ is maximum, say $p_M\left(D\right)$, and one for which it is minimum, say $p_m\left(D\right)$. In Table 7 we can see that for Example 3.5.
Now let us suppose we have an algorithm that precisely makes use of the NisB property, by using distance L_p. So its most robust variation, i.e. the one which is valid for the highest number of functions, is for $p=p_M\left(D\right)$, and its most specific one for $p=p_m\left(D\right)$.

8.4. $\mathcal{F}^{+}$**size.**
Unfortunately we do not have yet a formula that would directly give the size of the subset $\mathcal{F}^{+}$. However there is already possible to say a few things. For a given $\left|X\right|$ let us suppose that for $\left|Y\right|=2$, $\left|Y\right|=3$, ..., $\left|Y\right|=\left|X\right|$, we do know the number of functions in $\mathcal{F}^{+}$ that make use of exactly $\left|Y\right|$ values, and let us call it $a\left(\left|X\right|,\left|Y\right|\right)$. Note that by definition $a\left(\left|X\right|,j\right)=0$ as soon as $\left|Y\right|$ is greater than $\left|X\right|$.Then it is possible to compute the $\mathcal{F}^{+}$size for any Y by using the following formula:

$$\left|\mathcal{F}^{+}\right|=\sum_{j=2}^{min(\left|X\right|,\left|Y\right|)}a\left(\left|X\right|,j\right)C_{\left|Y\right|}^{j} \tag{8.3}$$

Let us give immediately an example for $\left|X\right|=5$ and dimension 1 (table 8). Remember that the $a\left(\left|X\right|,\left|Y\right|\right)$ values are for the moment known just experimentally. For $\left|Y\right|=2$ we have 5 functions that are in $\mathcal{F}^{+}$, like say $(1,1,1,2,2)$. For $\left|Y\right|=3$, we have $C_3^2=3$ possibilities to make use of exactly two values. Each of them "generates" then 5 functions for, because of equivalence, if a function like $(1,1,1,2,2)$ is in $\mathcal{F}^{+}$, then $(1,1,1,3,3)$and $(2,2,2,3,3)$also are in $\mathcal{F}^{+}$. By adding these 15 functions to the 53 ones (i.e. $a\left(5,3\right)C_3^3=a\left(5,3\right)$ in formula 8.3), we find that the size of $\mathcal{F}^{+}$ is 68. Same reasoning for the next lines of the table.
What is interesting is the rate $\left|\mathcal{F}^{+}\right|/\left|\mathcal{F}\right|$, for as soon as $\left|Y\right|$ is similar to $\left|X\right|$, or greater, it depends almost only on $a\left(\left|X\right|,\left|X\right|\right)$. It is easy to derive its limit value from formula 8.3.

$$lim\left(\frac{\left|\mathcal{F}^{+}\right|}{\left|\mathcal{F}\right|}\right)_{\left|Y\right|/\left|X\right|\to\infty}=\frac{a\left(\left|X\right|,\left|X\right|\right)}{\left|X\right|!}$$

$\|Y\|$	$a(\|X\|,\|Y\|)$	$j=2$	$j=3$	$j=4$	$j=5$	$\|\mathcal{F}^+\|$	$\|\mathcal{F}\|$	rate	
2	5	5				5	32	0.16	
3	53	15				68	243	0.28	
4	94	30	212			336	1024	0.33	
5	**40**	50	530	470		1090	3125	0.35	
6	0	75	1060	1410	240	2785	7776	0.36	
100	0	24750	8570100	368595150	3011500800	3388690800	10^{10}	0.34	

TABLE 8. How to compute $|\mathcal{F}^+|$ when knowing $a(|X|,|Y|)$. The limit rate is here $40/5! = 0.33$

This value can be seen as an estimation of the rate when $|Y| = |X|$. As $a(|X|,|Y|)$ is actually also depending on the dimension D of the search space X, and on the kind of distance L that is used, we can name it $a^*(|X|,D,L)$. It may be worth re-explaining its meaning. When the search space and the function value space have the same size m, this is the proportion of bijective functions that also have the NisB property. In other words, if we could say "for x% of the *bijective* functions the NisB truth value is positive", then we could estimate "on the whole, the size of $\mathcal{F}^+$ is $x\,|X|^{|X|}$".

8.5. **From discrete to continuous search space.**
As discrete functions are easier to study from a NisB point of view, some algorithms like PSO do prefer "moving" in a continuous search space. We are interesting here on how to design a benchmark of functions on such a space. Actually there is a quite easy way: piece-wise functions. Let us give an example.
We want to design a deceptive continuous function. We can start from any discrete one that belongs to $\mathcal{F}^-$, say $f = (0,3,2,4,2)$ on $X = (1,2,3,4,5)$. Here we have $\nu(f) = -0.17$. Then we can derive a piece-wise function g on say $[0,5[$ by:

$$x \in [i-1, i[\Rightarrow g(x) = f(i)$$

where i is in X. On this function the probability of success of Random Search after at most n attempts is given by $p(n) = 1-(1-1/5)^n$. We can compare with the result obtained by a classical PSO with say five particles (swarm size $S = 5$), as shown on Table 9. Of course when the number of attempts is precisely equal to the number of particles, PSO is equivalent to R because only the random initialisation phase is performed.
Note that the function obtained by this method may be not deceptive anymore when the number of attempts (i.e. sampled points) increases. Why? Because in any case, for a function f on a continuous search space X, a conjecture is that $\nu(f) \geq 0$ (see Open question 8.7.1 below). It would mean that it can not be intrinsically deceptive. However, the way PSO *sees* it can be. This is because the algorithm samples X, and moreover keeps only at most $2S$ points. Therefore, just after initialisation, it may easily tend to follow a wrong way, i.e. tend to sample some points that do not really contribute to increase the ν value, then it still tends to follow the wrong way, etc. On the contrary, when the sampling is completely "regular", and keeping all

Number of attempts	Random search	PSO 5 particles
5	0.67	0.67
20	0.99	0.73

TABLE 9. Comparison of Random Search and PSO on a piece-wise deceptive function. For PSO the success rate is estimated over 5000 runs

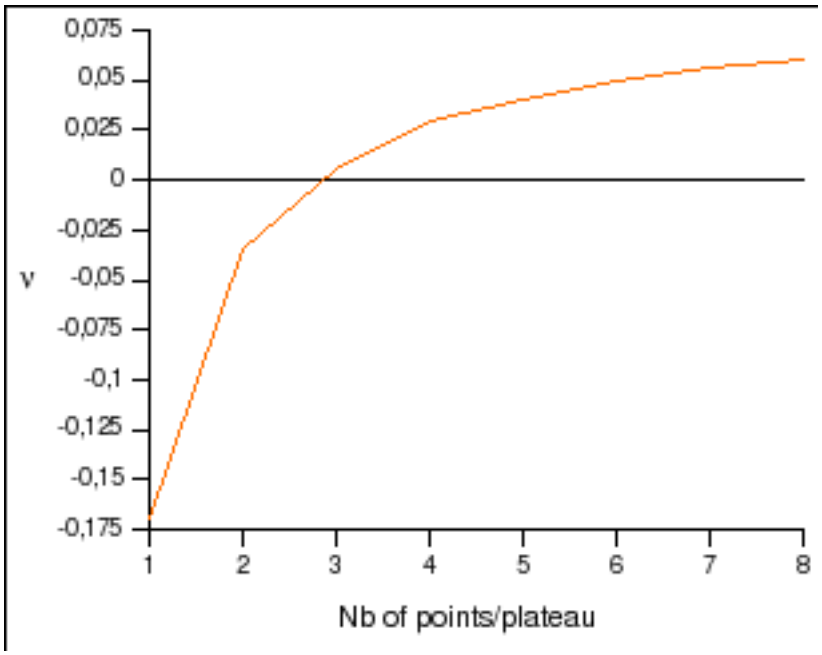


FIGURE 8.2. NisB truth value for the functions (0,3,2,4,2), (0,0,3,3,2,2,4,4,2,2) etc. First negative, then positive

points, when t increases so does the ν value. It can be negative at the beginning, then positive, as we can see on figure 8.2.

8.6. **Nearer is Better than the best.**

It is perfectly possible to define "Nearer is Better" in a apparently more restrictive way. In Equation 3.1 we can replace $B_{w,b}$ by.

$$B_{w,b} = \{x, x \neq x_b, x \neq x_w, f(x) \leq f(x_b)\}$$

i.e. we are considering not anymore the positions that are better than x_w but better than x_b, i.e. "better than the best" of the pair.
However, contrarily to the intuition, the corresponding class of functions, that we can denote $\mathcal{F}^{+b}$ is not included into $\mathcal{F}^{+}$. Their intersection is not empty, that is all. And also, even an "obviously good" function like $(1, 2, 3, 4)$, which is in $\mathcal{F}^{+}$, is not in $\mathcal{F}^{+b}$. That is why studying $\mathcal{F}^{+}$ seems more interesting.

8.7. **Open Questions.**

8.7.1. *Perfect deceptive function.*

What is the smallest possible value for $\nu(f)$? Intuitively, it seems it should be obtained with the perfect random landscape. If it is true, we have a quite simple lowest bound.

$$\nu_{min} \geq \left(1 - \frac{|X|}{|X| - 2}\right) \frac{1}{2} \tag{8.4}$$

For$|X| = 5$, it gives $-1/3$. Note that if formula 8.4 is true, it has an important consequence: although always negative, ν_{min} could be as near to zero as we want, just by increasing $|X|$. It would ever mean that if $|X|$ is intinite, then $\mathcal{F}^{-}$ is empty. An interesting question.

8.7.2. *Extending a non-NisB function.*

Let f be a non-NisB function (i.e. whose NisB truth value is negative) defined on X, and f' a non-NisB function defined on X'. We define a new function g on $X \bigcup X'$ by.

$$\begin{cases} g(x) & = & f(x) \text{ if } x \in X \\ & = & f'(x) \text{ if } x \in X' \end{cases}$$

Is g also a non-NisB function? Note that the contrary is obviously wrong, i.e. if f and f' are two NisB functions, then g is not always also a NisB one.

References

[1] Tim M. Blackwell and P. J. Bentley. Don't Push Me! Collision-Avoiding Swarms. In *Congress of Evolutionary Computation*, pages 1691–1696, Piscataway, New Jersey, 2002.

[2] Maurice Clerc. TRIBES - Un exemple d'optimisation par essaim particulaire sans paramÃĺtres de contrÃťle. In *OEP'03 (Optimisation par Essaim Particulaire)*, Paris, 2003. 14 pages.

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

[4] Paul Darwen. Black magic: Interdependence prevents principled parameter setting. pages 227–237, 2000. Institute for Food and Crop Research, Christchurch, New Zealand.

[5] T. Wegener I. Droste, S. Jansen. Perhaps not a free lunch but at least a free appetizer. pages 833–839. GECCO, Morgan Kaufmann Publishers, 1999.

[6] C. Igel and M. Toussaint. On classes of functions for which no free lunch results hold. *Information Processing Letters*, 86:317–321, 2003.

[7] G. A. Jastrebski and D. V. Arnold. Improving evolution strategies through active covariance matrix adaptation. pages pp. 9719–9726, Vancouver,, 2006. IEEE World Congress on Computational Intelligence, IEEE Press, Piscataway, New Jersey.

[8] Christopher K. Monson and Kevin D. Seppi. Exposing Origin-Seeking Bias in PSO. In *GECCO'05*, pages 241–248, Washington, DC, USA, 2005. Nice use of TRIBES. PSOGauss (constricted PSO with Gaussian noise).

[9] Godfrey C. Onwubolu. TRIBES application to the flow shop scheduling problem. In *New Optimization Techniques in Engineering*, pages 517–536. Springer, Heidelberg, Germany, 2004. particl swarm adaptive.

[10] M. D. Whitley D. Schumacher, C. Vose. The no free lunch and description length. In A. Wu W. Langdon H.-M. Voigt M. Gen S. Sen M. Dorigo S. Pezeshk M. Garzon L. Spector, E. Goodman and E. Burke, editors, *Genetic and Evolutionary Computation Conference (GECCO 2001), San Francisco, CA, USA, Morgan Kaufmann.*, pages 565–570, San Francisco, CA, USA, 2001. Genetic and Evolutionary Computation Conference, Morgan Kaufmann.

[11] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.