On the Benchmarking of Multiobjective Optimization Algorithm

Mario Köppen

Fraunhofer IPK
Dept. Security and Inspection Technologies
Pascalstr. 8-9, 10587 Berlin, Germany

Abstract. The "No Free Lunch" (NFL) theorems state that in average each algorithm has the same performance, when no *a priori* knowledge of single-objective cost function f is assumed. This paper extends the NFL theorems to the case of multi-objective optimization. Further it is shown that even in cases of *a priori* knowledge, when the performance measure is related to the set of extrema points sampled so far, the NFL theorems still hold. However, a procedure for obtaining function-dependent algorithm performance can be constructed, the so-called tournament performance, which is able to gain different performance measures for different multi-objective algorithms.

1 Introduction

The "No Free Lunch" (NFL) theorems state the equal average performance of any optimization algorithm, when measured against the set of all possible cost functions and if no domain knowledge of the cost function is assumed [3]. Usually, the NFL theorem is considered in a context of design of algorithms, especially it became well-known in the scope of evolutionary computation. However, the NFL theorem has also some other facettes, one of which is the major concern of this paper. So, the NFL theorem can also be seen as stating the impossibility to obtain a concise mathematical definition of algorithm performance.

In this context, this paper considers multi-objective optimization and how the NFL theorems apply in this field. After recalling some basic definitions of multi-objective optimization in section 2, esp. the concept of Pareto front, the standard NFL theorem is proven for the multi-objective case in section 3. Then, the proof is extended to the case where sampling of extrema is also involved in the performance measure in section 4, proving that there is no gain in using such a measure. Only the case that two algorithms are compared directly give rise to so-called tournament performance and a heuristic procedure to measure algorithm performance. This will be presented in section 5.

2 Basic Definitions

In multi-objective optimization, optimization goal is given by more than one objective to be extreme [1]. Formally, given a domain as subset of R^n, there are assigned m functions $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$. Usually, there is not a single optimum but rather the so-called Pareto set of *non-dominated* solutions:

For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ it is said that $\boldsymbol{a}$ *(Pareto-)dominates* $\boldsymbol{b}$, when each component of $\boldsymbol{a}$ is less or equal to the corresponding component of $\boldsymbol{b}$, and at least one component is smaller:

$$\boldsymbol{a} >_D \boldsymbol{b} \longleftrightarrow \forall i(a_i \leq b_i) \wedge \exists k(a_k < b_k). \tag{1}$$

Note that in a similar manner Pareto dominance can be related to ">"-relation.

The subset of all vectors of a set M of vectors, which are not dominated by any other vector of M is the Pareto set (also Pareto front) PF. The Pareto set for univariate data (single objective) contains just the maximum of the data.

The task of multi-objective optimization algorithm is to sample points of the Pareto front. A second instantiation (often called *decision maker*) is needed to further select from the Pareto front.

3 NFL-Theorem for multi-objective optimization algorithms

A slight modification extends the proof of the single-objective NFL theorems given in [2] to the multi-objective case. Be X a finite set and Y a set of k finite domains Y_i with $i = 1, \ldots, k$. Then we consider the set of all sets of k cost functions $f_i : X \to Y_i$ with $i = 1, \ldots, k$, or $f : X \to Y$ for simplicity. Let m be a non-negative integer $< |X|$. Define d_m as a set $\{(d_m^x(i), d_m^y(i) = (f(d_m^x(i)))\}, \quad i = 1, \ldots, m$ where $d_m^x(i) \in X \; \forall\, i$ and $\forall\, i, j,\; d_m^x(i) \neq d_m^x(j)$.

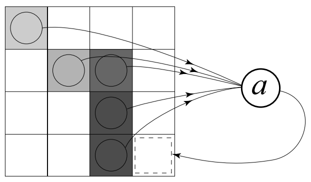

Fig. 1. A deterministic algorithm derives next sampling point $d_{m+1}^x(m+1)$ from the outcome of foregoing sampling d_m.

Now consider a *deterministic* search algorithm a which assign to every possible d_m an element of $X \setminus d_m^x$ (see fig. 1):

$$d_{m+1}^x(m+1) = a[d_m] \notin \{d_m^x\}. \tag{2}$$

Define $Y(f, m, a)$ to be the sequence of m Y values produced by m successive applications of the algorithm a to f. Let $\delta(.,.)$ be the Kronecker delta function that equals 1 if its arguments are identical, 0 otherwise. Then the following holds:

Lemma 1. *For any algorithm a and any d_m^y,*

$$\sum_f \delta(d_m^y, Y(f, m, a)) = \prod_{i=1}^{k} |Y_i|^{|X|-m}.$$

Proof. Consider all cost functions f_+ for which $\delta(d_m^y, Y(f_+, m, a))$ takes the value 1, 2 asf. of the sequence d_m^y:

i) $f_+(a(\emptyset)) = d_m^y(1)$
ii) $f_+(a[d_m(1)]) = d_m^y(2)$
iii) $f_+(a[d_m(1), d_m(2)]) = d_m^y(3)$
$\ldots$

where $d_m(j) \equiv (d_m^x(j), d_m^y(j))$. So the value of f_+ is fixed for exactly m distinct elements from X. For the remaining $|X| - m$ elements from X, the corresponding value of f_+ can be assigned freely. Hence, out of the $\prod_i |Y_i|^{|X|}$ separate f, exactly $\prod_i |Y_i|^{|X|-m}$ will result in a summand of 1 and all others will be 0.

Then, we can continue with the proof of NFL theorem in multi-objective case. Take any performance measure $c(.)$, mapping sets d_m^y to real numbers.

Theorem 1. *For any two deterministic algorithms a and b, any performance value $K \in \mathcal{R}$, and any $c(.)$,*

$$\sum_f \delta(K, c(Y(f, m, a))) = \sum_f \delta(K, c(Y(f, m, b))).$$

Proof. Since more than one d_m^y may give the same value of the performance measure K, for each K the l.h.s. is expanded over all those possibilities:

$$\sum_f \delta(K, c(Y(f, m, a))) =$$

$$= \sum_{f, d_m^y \in Y^m} \delta(K, c(d_m^y)) \delta(d_m^y, Y(f, m, a)) \tag{3}$$

$$= \sum_{d_m^y \in Y^m} \delta(K, c(d_m^y)) \sum_f \delta(d_m^y, Y(f, m, a))$$

$$= \sum_{d_m^y \in Y^m} \delta(K, c(d_m^y)) \prod_{i=1}^{k} |Y_i|^{|X|-m} \text{ (by Lemma 1)}$$

$$= \prod_{i=1}^{k} |Y_i|^{|X|-m} \sum_{d_m^y \in Y^m} \delta(K, c(d_m^y)) \tag{4}$$

The last expression does not depend on a but only on the definition of $c(.)$.

4 Benchmarking measures

The formal proof of the NFL theorems assumes no *a priori* knowledge of the function f. This can be easily seen in the proof of Theorem 1, when the expansion over f is made (line 3 of the proof): it is implicitely assumed that the performance measure $c(.)$ does not depend on f. There are performance measures depending on f, for which Theorem 1 does not hold and that can be easily constructed (as e.g. derived from the requirement to scan (x, y) pairs in a given order).

This is a reasonable assumption for evaluating an algorithm a. Domain knowledge of f could result in algorithm a somehow designed in a manner to show increased performance on some benchmark problems. However, common procedure to evaluate algorithms is to apply them onto a set of so-called "benchmark problems." This also holds in the multi-objective case. From a benchmark function f, usually analytic properties (esp. the extrema points) are given in advance. In [1], an extensive suite of such benchmark problems is proposed, in order to gain understanding of abilities of multi-objective optimization algorithms. So, for each benchmark problem, a description of the Pareto front of the problem is provided. The task given to a multi-objective optimization algorithm is to sample as many points from the Pareto front as possible. To name it here again: clearly, such a performance measure is related to *a priori* of f itself. NFL theorems given with Theorem 1 do not cover this case.

However, in the following, it will be shown that NFL theorems even apply in such a case. It is based on the following lemma:

Lemma 2. *For any algorithm a it holds*

$$\bigcup_f \{a \circ f\}\,|_Y \; = Y^{|X|}$$

A given algorithm a applied to any f gives a sequence of values from Y. The union of all those sequences will be the set of all possible sequences of $|X|$ elements chosen from Y. Or, in other words: each algorithm, applied to all possible f will give a permutation of the set of all possible sequences, with each sequence appearing exactly once.

Proof. Assume that for two functions f_1 and f_2 algorithm a will give the same sequence of y-values $(y_1, y_2, \ldots, y_{|X|})$. This also means that the two corresponding x-margins are permutations of X. Via induction we show that then follows $f_1 = f_2$.
Verification. Since we are considering deterministic algorithms, the choice of the first element x_1 is fixed for an algorithm a (all further choices for x values are functions of the foregoing samplings). So, both f_1 and f_2 map x_1 to y_1.
Step. Assume $f_1(x_i) = f_2(x_i)$ for $i = 1, \ldots, k$ (and $k < |X|$). Then according to eq. 2, algorithm a will compute the same $d^x_{k+1}(k+1)$ since this computation only depends on the sequence d_m that is equal for f_1 and f_2 by proposition. Since the y-margins are also equal in the position $(k+1)$, for both f_1 and f_2 $x_{k+1} = d^x_{k+1}(k+1)$ is mapped onto y_{k+1}.

This completes the proof. It has to be noted that not each permutation of $Y^{|X|}$ can be accessed by an algorithm (what can be easily seen from the fact that there are much more permutations than possible algorithm specifications).

Following this lemma, all performance calculations that are independent of the sorting of the elements of $Y^{|X|}$ will give the same average performance, independent on a. Sampling of Pareto front elements after m algorithm steps is an example for such a measure. For illustration, table 1 gives these compuations for the simple case $Y = \{0,1\} \times \{0,1\}$ and $X = \{a,b,c\}$.

Table 1. Performance measure Pareto sampling after two steps in the example case $Y = \{0,1\}^2$ and $|X| = 3$.

y_1	y_2	y_3	PF	$c(2)$	y_1	y_2	y_3	PF	$c(2)$	y_1	y_2	y_3	PF	$c(2)$	y_1	y_2	y_3	PF	$c(2)$
00	00	00	00	1	01	00	00	00	1	10	00	00	00	1	11	00	00	00	1
00	00	01	00	1	01	00	01	00	1	10	00	01	00	1	11	00	01	00	1
00	00	10	00	1	01	00	10	00	1	10	00	10	00	1	11	00	10	00	1
00	00	11	00	1	01	00	11	00	1	10	00	11	00	1	11	00	11	00	1
00	01	00	00	1	01	01	00	00	0	10	01	00	00	0	11	01	00	00	0
00	01	01	00	1	01	01	01	01	1	10	01	01	01, 10	2	11	01	01	01	1
00	01	10	00	1	01	01	10	01, 10	1	10	01	10	01, 10	2	11	01	10	01, 10	1
00	01	11	00	1	01	01	11	01	1	10	01	11	01, 10	2	11	01	11	01	1
00	10	00	00	1	01	10	00	00	0	10	10	00	00	0	11	10	00	00	0
00	10	01	00	1	01	10	01	01, 10	2	10	10	01	01, 10	1	11	10	01	01, 10	1
00	10	10	00	1	01	10	10	01, 10	2	10	10	10	10	1	11	10	10	10	1
00	10	11	00	1	01	10	11	01, 10	2	10	10	11	10	1	11	10	11	10	1
00	11	00	00	1	01	11	00	00	0	10	11	00	00	0	11	11	00	00	0
00	11	01	00	1	01	11	01	01	1	10	11	01	01, 10	1	11	11	01	01	0
00	11	10	00	1	01	11	10	01, 10	1	10	11	10	10	1	11	11	10	10	0
00	11	11	00	1	01	11	11	01	1	10	11	11	10	1	11	11	11	11	1
Sum				16					16					16					11
															Average Performance $59/64 \sim 0.92$				

Table 1 displays all possible functions $f : X \rightarrow Y$ and the corresponding Pareto set PF. The column $c(2)$ shows the number of Pareto set elements that have already been sampled after two steps. The computation of the average performance does not depend on the order in which the functions are listed, thus each algorithm a will have the same average performance $c_{av}(2) = 59/64$.

A remark on the single-objective case: in the foregoing discussion, multi-objectivity of f was not referenced explicitely. Hence, the discussion holds also for the "single-objective" version, in which an algorithm is judged by its ability to find extrema points within a fixed number of steps. The NFL theorems also appy to this case.

5 Tournament performance

Among the selection of function-dependent performance measures, one should be pointed out in the rest of this paper. For obtaining "position-dependence" of the measure on a single function f, the value obtained by applying a *base algorithm* A is taken. Algorithm a now runs competively against A. In such a case, the NFL theorem does not hold. For seeing this, it is sufficient to provide a counterexample.

Before, we define the difference of two Pareto sets PF_a and PF_b as the set $PF_a \setminus PF_b$, in which in PF_a all elements are removed, which are dominated by any element of PF_b.

Be $Y = \{0,1\}^2$ and $X = \{a,b,c\}$, as in the foregoing example. Now algorithm A is as follows:

Algorithm A: Take the d_m^x in the following order: a, b, c.

And be

Algorithm a: Take a as first choice. If $f(a) = \{0,1\}$ then select c as next point, otherwise b.

Table 2 shows the essential part of all possible functions f, in which algorithms A and a behave different.

Table 2. Tournament performance of algorithm a after two steps.

$f(a)$	$f(b)$	$f(c)$	$PF(d_2^y(A,f))$	$PF(d_2^y(a,f))$	$PF(d_2^y(A,f)) \setminus PF(d_2^y(a,f))$	$\triangle\lvert.\rvert$
01	00	00	00	00	00	0
01	00	01	00	01	00	0
01	00	10	00	01, 10	00	0
01	00	11	00	01	00	0
01	01	00	01	00	-	-1
01	01	01	01	01	01	0
01	01	10	01	01, 10	01	0
01	01	11	01	01	01	0
01	10	00	01, 10	00	-	-2
01	10	01	01, 10	01	01, 10	0
01	10	10	01, 10	01, 10	01, 10	0
01	10	11	01, 10	01	01, 10	0
01	11	00	01	00	-	-1
01	11	01	01	01	01	0
01	11	10	01	01, 10	01	0
01	11	11	01	01	01	0
Total						-4

For "measuring" the performance of a at step m, we compute the size of the Pareto set difference

$$\left| PF\left(d_m^y(A, f)\right) \setminus PF\left(d_m^y(a, f)\right) \right| . \tag{5}$$

and take the average over all possible f as average performance. For functions that do not start with $f(a) = \{0, 1\}$, both algorithms are identical, so in these cases $\triangle = 0$. For functions mapping x-value a onto $\{0, 1\}$, we see $\triangle = -4$.

Now taking other algorithms:

Algorithm b: Take a as first choice. If $f(a) = \{1, 0\}$ then select c as next point, otherwise b.

Algorithm c: Take a as first choice. If $f(a) = \{1, 1\}$ then select c as next point, otherwise b.

For b and c, a similar computation gives $\triangle = -4$ and $\triangle = -5$ respectively. In this sense "strongest" algorithm (i.e. in comparison to A) is to sample in the order a, c, b with a performance $\triangle = -13$.

It should be noted that this performance measure is also applicable to the single-objective case. However, more studies on this measure have to be performed.

Based on this, a heuristic procedure to measure performance of multi-objective optimization algorithm a might look like:

1. Let algorithm a run for k evaluations of cost function f and take the set M_1 of non-dominated points from Y obtained by the algorithm.
2. Select k random domain values of X and compute the Pareto set M_2 of the corresponding Y values.
3. Compute the set M_3 of elements of M_2 that are not dominated by any element of M_1.

The relation of $|M_1|$ to $|M_3|$ gives a measure how algorithm a performs against random search.

References

1. Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, 2002.
2. Mario Köppen, David H. Wolpert and William G. Macready, "Remarks on a recent paper on the "no free lunch" theorems," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 295–296, 2001.
3. David H. Wolpert and William G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.