No Free Lunch Theorems: Limitations and Perspectives of Metaheuristics

Christian Igel

> [...] Thus not only our reason fails us in the discovery of the ultimate connexion of causes and effects, but even after experience has informed us of their constant conjunction, it is impossible for us to satisfy ourselves by our reason, why we should extend that experience beyond those particular instances, which have fallen under our observation. We suppose, but are never able to prove, that there must be a resemblance betwixt those objects, of which we have had experience, and those which lie beyond the reach of our discovery.
>
> (David Hume, 1739 [14, 10])

Abstract The No Free Lunch (NFL) theorems for search and optimization are reviewed and their implications for the design of metaheuristics are discussed. The theorems state that any two search or optimization algorithms are equivalent when their performance is averaged across all possible problems and even over subsets of problems fulfilling certain constraints. The NFL results show that if there is no assumption regarding the relation between visited and unseen search points, efficient search and optimization is impossible. There is no well performing universal metaheuristic, but the heuristics must be tailored to the problem class at hand using prior knowledge. In practice, it is not likely that the preconditions of the NFL theorems are fulfilled for a problem class and thus differences between algorithms exist. Therefore, tailored algorithms can exploit structure underlying the optimization problem. Given full knowledge about the problem class, it is in theory possible to construct an optimal algorithm.

1 Introduction

Metaheuristics such as evolutionary algorithms, simulated annealing, swarm algorithms, and tabu search are general in the sense that they can be applied to any objective (target of fitness) function $f : \mathscr{X} \to \mathscr{Y}$, where $\mathscr{X}$ denotes the search space

Department of Computer Science, University of Copenhagen, Universitetsparken 1, 2100 Copenhagen Ø, Denmark, e-mail: igel@diku.dk

and $\mathscr{Y}$ a set of totally ordered cost-values. The goal of designing metaheuristics is to come up with search or optimization methods that are superior to others when applied to instances of a certain class of problems. In this chapter, we ask under which conditions one heuristic can be better than another at all and derive answers based on the No Free Lunch (NFL) theorems.

Table 1 All possible functions $\{0,1\}^2 \to \{0,1\}$, which will be used as examples throughout this chapter.

(x_1,x_2)	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
$(0,0)$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$(0,1)$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$(1,0)$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$(1,1)$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

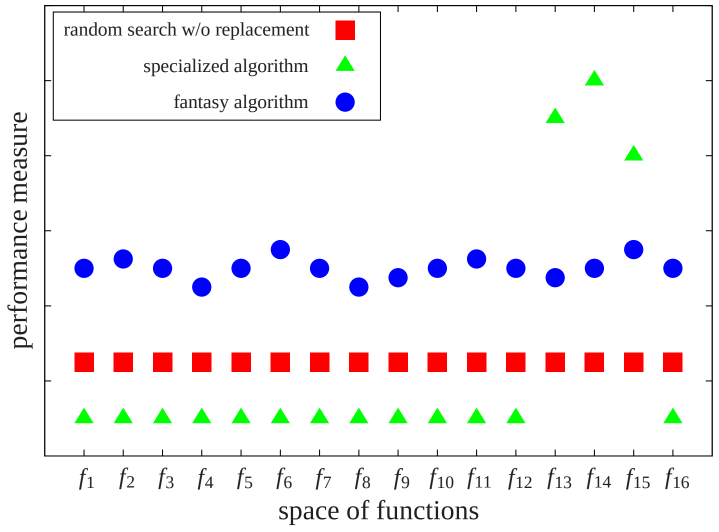

Fig. 1 Distribution of algorithm performance over a discrete space of problems. For random search without replacement the average performance on a function is shown. A performance profile such as depicted for the "fantasy algorithm" is not possible if the space has the property of being closed under permutation (e.g., as in the case of the problems shown in Table 1).

Example 1. Let us consider all objective functions mapping from some finite domain $\mathscr{X}$ to some finite set of values $\mathscr{Y}$, for example those given by all functions

$\{0,1\}^2 \to \{0,1\}$ listed in Table 1. Figure 1 depicts the performance of three algorithms over this set of functions. Performance could for example be measured in terms of the number of objective function evaluations needed to find the optimum or by the quality of the best solution found in m steps (or by any other measure consistent with the definition given in the next section). The baseline is given by the average performance of a random search algorithm, which picks search points uniformly at random without replacement (i.e., every point is visited only once). A highly specialized algorithm – which has been developed for many years based on extensive domain knowledge – may have a performance profile as indicated by the triangles. It performs extremely well on a very few problems, but very badly on all the others. Now, would it not be great to come up with an algorithm that performs well, in particular better than random search, across *all* problems as indicated by the circles in Fig. 1? Could this be achieved by designing a metaheuristic using principled methods? Unfortunately, the answer to these questions is *no*.

The NFL theorem for optimization – and we do not want to distinguish between search and optimization in this chapter – roughly speaking states that all non-repeating search algorithms have the same mean performance when averaged uniformly over *all* possible objective functions $f : \mathscr{X} \to \mathscr{Y}$ [35, 24, 8, 36, 21, 6]. In practice, of course, algorithms need not perform well on all possible functions, but only on a subset that arises from the application at hand.

In this chapter, we discuss extensions and implications of this fundamental result. In the next section, we introduce the basic notation and formally state the original NFL theorem as well as some of its refinements, in particular NFL results for restricted problem classes [28, 29, 20]. Section 3 discusses the optimization scenario underlying these theorems. Section 4 studies how many problem classes fulfill the prerequisites for a NFL result and if these problem classes are likely to be observed in real-world applications. It ends with a discussion of the Almost NFL theorem [6]. In the more research oriented section 5 the link between optimization problems and Markov decision processes is established and it is shown how to construct optimal algorithms in this framework. Finally, the main results are summarized and further general conclusions are drawn.

2 The NFL Theorem for Search

In the following, we first fix the notation before we state the basic NFL theorem and its extensions.

2.1 Basic Definitions

Let us assume a finite search space $\mathscr{X}$ and a finite set of cost-values $\mathscr{Y}$. Let $\mathscr{F}$ be the set of all objective functions $f : \mathscr{X} \to \mathscr{Y}$ to be optimized (also called target,

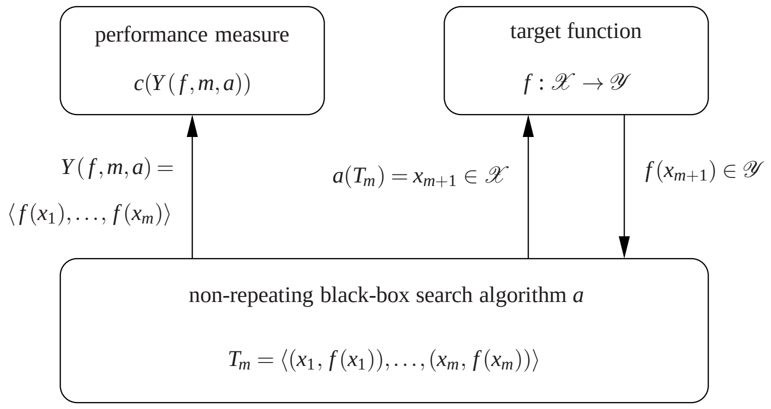
performance measure
$c(Y(f,m,a))$
target function
$f:\mathscr{X}\to\mathscr{Y}$
$Y(f,m,a) =$
$\langle f(x_1),\dots,f(x_m)\rangle$
$a(T_m) = x_{m+1}\in\mathscr{X}$
$f(x_{m+1})\in\mathscr{Y}$
non-repeating black-box search algorithm a
$T_m = \langle (x_1, f(x_1)),\dots,(x_m, f(x_m))\rangle$

Fig. 2 Scheme of the optimization scenario considered in NFL theorems. A non-repeating black-box search algorithm a chooses a new exploration point in the search space depending on the sequence T_m of the already visited points with their corresponding cost-values. The target function f returns the cost-value of a candidate solution as the only information. The performance of a is determined using the performance measure c, which is a function of the sequence $Y(f,m,a)$ containing the cost-values of the visited points.

fitness, energy, or cost functions). No Free Lunch theorems make statements about non-repeating search algorithms (referred to as algorithms) that explore a new point in the search space depending on the history of previously visited points and their cost-values. Non-repeating means that no search point is evaluated more than once. Let the sequence $T_m = \langle (x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_m, f(x_m))\rangle$ represent m pairs of different search points $x_i \in \mathscr{X}$, $\forall i,j : x_i \neq x_j$ and their cost-values $f(x_i)\in\mathscr{Y}$. An algorithm a appends a pair $(x_{m+1}, f(x_{m+1}))$ to this sequence by mapping T_m to a new point x_{m+1} with $x_{m+1}\neq x_i$ for $i=1,\dots,m$.

We assume that the performance of an algorithm a after $m \leq |\mathscr{X}|$ iterations with respect to a function f depends only on the sequence

$$Y(f,m,a) = \langle f(x_1), f(x_2), \dots, f(x_m)\rangle$$

of cost-values the algorithm has produced. Let the function c denote a performance measure mapping sequences of cost-values to the real numbers. Figure 2 depicts the optimization scenario assumed in NFL theorems.

Example 2. In the case of function minimization a performance measure that returns the minimum cost-value in the sequence could be a reasonable choice. Alternatively, the performance measure could return the number of objective function evaluations that were needed to find a search point with a cost-value below a certain threshold.

In general, one must distinguish between an algorithm and its search behavior. For finite $\mathscr{X}$ and $\mathscr{Y}$ and thus finite $\mathscr{F}$, there are only finitely many different non-repeating, deterministic search behaviors. For a given function f, there exist only $|\mathscr{X}|!/(|\mathscr{X}|-m)!$ different search behaviors corresponding to the possible orders in which the search points can be visited. In practice, two algorithms that always exhibit the same search behavior (i.e., produce the same sequence T_m given the same function f and number of steps m) may differ, for example, in their space and run time requirements (see section 3).

Example 3. Given some function $f : \{0,1\}^2 \to \{0,1\}$ there are 12 possible search behaviors for 2-step search resulting in the following sequences:
$\langle((0,0), f((0,0))),((0,1), f((0,1)))\rangle$, $\langle((0,0), f((0,0))),((1,0), f((1,0)))\rangle$,
$\langle((0,0), f((0,0))),((1,1), f((1,1)))\rangle$, $\langle((0,1), f((0,1))),((0,0), f((0,0)))\rangle$,
$\langle((0,1), f((0,1))),((1,0), f((1,0)))\rangle$, $\langle((0,1), f((0,1))),((1,1), f((1,1)))\rangle$,
$\langle((1,0), f((1,0))),((0,0), f((0,0)))\rangle$, $\langle((1,0), f((1,0))),((0,1), f((0,1)))\rangle$,
$\langle((1,0), f((1,0))),((1,1), f((1,1)))\rangle$, $\langle((1,1), f((1,1))),((0,0), f((0,0)))\rangle$,
$\langle((1,1), f((1,1))),((0,1), f((0,1)))\rangle$, $\langle((1,1), f((1,1))),((1,0), f((1,0)))\rangle$.

Often we are not interested in statements that assume that each function in $\mathscr{F}$ is equally likely to be the objective function. Instead, we want to make statements that refer to *problem classes*. A reasonable general working definition of a problem class is:

Definition 1 (problem class). Given a finite search space $\mathscr{X}$ and a finite set of cost-values $\mathscr{Y}$, a *problem class* can be defined by a probability distribution $p_{\mathscr{F}}$ over the space $\mathscr{F}$ of all possible objective functions $f : \mathscr{X} \to \mathscr{Y}$, where $p_{\mathscr{F}}(f)$ is the probability that $f \in \mathscr{F}$ is the objective function faced by an algorithm.

In the special case that all functions that have a non-zero probability of being the target function are equally likely, we can identify a problem class by the subset $F \subseteq \mathscr{F}$ with $f \in F \Rightarrow p_{\mathscr{F}}(f) = 1/|F| > 0$.

2.2 The NFL Theorem

The original and most popular version of the NFL theorem for optimization was formally stated and proven by Wolpert and Macready in 1995 [35, 36]. It can be expressed as:

Theorem 1 (NFL theorem [36]). *For any two algorithms a and b, any $k \in \mathrm{I\!R}$, any $m \in \{1,\dots,|\mathscr{X}|\}$, and any performance measure c*

$$\sum_{f\in\mathscr{F}} \delta(k, c(Y(f,m,a))) = \sum_{f\in\mathscr{F}} \delta(k, c(Y(f,m,b))) \quad . \tag{1}$$

Herein, δ denotes the Kronecker function ($\delta(i, j) = 1$ if $i = j$, $\delta(i, j) = 0$ otherwise). Proofs can be found in [35, 36, 27, 21]. Equation (1) implies

$$\sum_{f\in\mathscr{F}} c(Y(f,m,a)) = \sum_{f\in\mathscr{F}} c(Y(f,m,b)) \tag{2}$$

for any two algorithms a and b, any $m \in \{1,\dots,|\mathscr{X}|\}$, and any performance measure c.

This proves that statements such as "averaged over all functions, my search algorithm is the best" are misconceptions. Radcliffe and Surry were among the first who appreciated and discussed this theorem [24]. Without rigorous proof, the basic NFL result was stated already earlier by Rawlins in 1991 [25].

When looking at the proof of the NFL theorem, it implies the following statements (see [33]):

Corollary 1. *For any $m \in \{1,\dots,|\mathscr{X}|\}$ and any performance measure c we have:*

1. *For any two algorithms a and b, for each $f_a \in \mathscr{F}$ it holds*

$$c(Y(f_a,m,a)) = k \Rightarrow \exists f_b \in \mathscr{F} : c(Y(f_b,m,b)) = k \ . \tag{3}$$

2. *For any two algorithms a and b and any subset of functions $F \subset \mathscr{F}$ with complement $F^c = \mathscr{F} \setminus F$ it holds*

$$\sum_{f\in F} c(Y(f,m,a)) > \sum_{f\in F} c(Y(f,m,b)) \Rightarrow \sum_{f\in F^c} c(Y(f,m,a)) < \sum_{f\in F^c} c(Y(f,m,b)) \ . \tag{4}$$

The first statement says that for any function $f_a \in \mathscr{F}$, there is a function $f_b \in \mathscr{F}$ on which algorithm b has the same performance as algorithm a on f_a. That is, if my algorithm outperforms your algorithm on some function then there is also an objective function on which your algorithm outperforms mine. And if my algorithm outperforms yours on some set of benchmark problems then your algorithm is better than mine averaged over the remaining problems.

2.3 The Sharpened NFL Theorems

Theorem 1 assumes that all possible objective functions in $\mathscr{F}$ are equally likely. Given that it is fruitless to design a metaheuristic for all possible objective functions, the question arises under which constraints the average performance of one algorithm can be better than another when the average is taken only over a subset $F \subset \mathscr{F}$. This is a relevant scenario if the goal is to develop metaheuristics for certain (e.g., "real-world") problem classes.

The NFL theorem has been extended to subsets of functions with the property of being closed under permutation (c.u.p.). Let $\pi : \mathscr{X} \to \mathscr{X}$ be a permutation of $\mathscr{X}$. The set of all permutations of $\mathscr{X}$ is denoted by $\Pi(\mathscr{X})$. A set $F \subseteq \mathscr{F}$ is said to be c.u.p. if for any $\pi \in \Pi(\mathscr{X})$ and any function $f \in F$ the function $f \circ \pi$ is also in F. In [27, 28] the following result is proven:

Theorem 2 (Sharpened NFL theorem [27, 28]). *If and only if F is c.u.p., then for any two algorithms a and b, any $k \in \mathbb{R}$, any $m \in \{1, \ldots, |\mathscr{X}|\}$, and any performance measure c*

$$\sum_{f \in F} \delta(k, c(Y(f,m,a))) = \sum_{f \in F} \delta(k, c(Y(f,m,b))) \quad . \tag{5}$$

This is an important extension of theorem 1, because it gives necessary and sufficient conditions for NFL results for subsets of functions.

Example 4. Consider again the mappings $\{0,1\}^2 \to \{0,1\}$, denoted by f_0, f_1, $\ldots$, f_{15} as shown in Table 1. Then $\{f_1, f_2, f_4, f_8\}$ and $\{f_0, f_1, f_2, f_4, f_8\}$ are examples of sets that are c.u.p.. The set $\{f_1, f_2, f_3, f_4, f_8\}$ is not c.u.p., because some functions are "missing". These missing functions include f_5, which results from f_3 by switching the elements $(0,1)$ and $(1,0)$.

Example 5. Theorem 1 tells us that on average all algorithms need the same time to find a desirable, say optimal, solution – but how long does it take? The average number of evaluations needed to find an optimum (the mean hitting time) depends on the cardinality of the search space $|\mathscr{X}|$ and the number n of search points that are mapped to a desirable solution. As the set of *all* functions where n search points represent desirable solutions is c.u.p., it is sufficient to compute the average time to find one of these points for an arbitrary algorithm, which is given by $(|\mathscr{X}|+1)/(n+1)$ [17] (a proof for $n=1$ is given in chapter **??[reference to Thomas' Black-Box chapter]**).

In theorems 1 and 2 it is implicitly assumed that each function in $\mathscr{F}$ and F, respectively, has the same probability to be the target function, because the summations in (1) and (5) average uniformly over the functions. However, it is more realistic to assume that different functions can have different probabilities to be the target function. In this more general case, a problem class is described by a probability distribution assigning each function f its probability $p_{\mathscr{F}}(f)$ to be the objective function.

To derive results for this general scenario it is helpful to introduce the concept of $\mathscr{Y}$-histograms. A *$\mathscr{Y}$-histogram* (*histogram* for short) is a mapping $h : \mathscr{Y} \to \mathbb{N}_0$ such that $\sum_{y \in \mathscr{Y}} h(y) = |\mathscr{X}|$. The set of all histograms is denoted by $\mathscr{H}$. Any function $f : \mathscr{X} \to \mathscr{Y}$ implies a histogram $h_f(y) = |f^{-1}(y)|$ that counts the number of elements in $\mathscr{X}$ that are mapped to the same value $y \in \mathscr{Y}$ by f. Herein, $f^{-1}(y)$ returns the preimage $\{x | f(x) = y\}$ of y under f. Further, two functions f and g are called *h-equivalent* if and only if they have the same histogram. The corresponding h-equivalence class $B_h \subseteq \mathscr{F}$ containing all functions with histogram h is termed a *basis class*. It holds:

Lemma 1 ([18]). *Any subset $F \subseteq \mathscr{F}$ that is c.u.p. is uniquely defined by a union of pairwise disjoint basis classes. B_h is equal to the permutation orbit of any function f with histogram h, i.e., $\forall f \in F : B_{h_f} = \bigcup_{\pi \in \Pi(\mathscr{X})} \{f \circ \pi\}$.*

Example 6. Consider the functions in Table 1. The $\mathscr{Y}$-histogram of f_1 contains the value zero three times and the value one one time, i.e., we have $h_{f_1}(0) = 3$ and

$h_{f_1}(1) = 1$. The mappings f_1, f_2, f_4, f_8 have the same $\mathscr{Y}$-histogram and are therefore in the same basis class $B_{h_{f_1}} = \{f_1, f_2, f_4, f_8\}$. The set $\{f_1, f_2, f_4, f_8, f_{15}\}$ is c.u.p. and corresponds to $B_{h_{f_1}} \cup B_{h_{f_{15}}}$.

The following ramification of the Sharpened NFL theorem (derived independently in [19], [29], and [7] generalizing the results in [9]) gives a necessary and sufficient condition for a NFL result in the general case of arbitrary distributions $p_{\mathscr{F}}$ over $\mathscr{F}$:

Theorem 3 (non-uniform Sharpened NFL theorem [19, 29, 20, 7]). *If and only if for all histograms h*

$$f, g \in B_h \Rightarrow p_{\mathscr{F}}(f) = p_{\mathscr{F}}(g) \quad , \tag{6}$$

then for any two algorithms a and b, any value $k \in \mathbb{R}$, any $m \in \{1, \ldots, |\mathscr{X}|\}$, and any performance measure c

$$\sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f)\, \delta(k, c(Y(f,m,a))) = \sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f)\, \delta(k, c(Y(f,m,b))) \quad . \tag{7}$$

This observation is the "sharpest" NFL so far. It gives necessary and sufficient conditions for NFL results for general problem classes.

3 The Preconditions in the NFL Theorem and the Sharpened NFL Theorems

The NFL theorems 1, 2, and 3 consider a certain optimization scenario. In the following, we discuss its basic preconditions.

Independence of algorithmic complexity

It is important to note that the NFL theorems make no assumptions about the space and time complexity of computing the next search point. For example, it makes no difference if the algorithm simply enumerates all elements of $\mathscr{X}$ or comes up with a decision after running some complex internal simulation.

For many practical applications, it is indeed reasonable to assume that the time needed to evaluate the fitness dominates the computational costs of computing the next step and that memory requirements are not an issue. Still, this needs not always be true.

Example 7. The update of the strategy parameters in CMA-ES scales quadratically with the search space dimension ([13, 30], see chapter **??[reference to Niko's**

CMA-ES chapter]). For extremely high-dimensional fitness functions that can be evaluated quickly this may have a significant impact on the run time (however, in most practical applications I dealt with, the evaluation of the objective function was by far the dominating term).

This topic is discussed in more detail in section **??** of chapter **??[reference to Thomas' Black-Box chapter]**, which demonstrates the differences between *algorithmic complexity* and *black-box complexity* of search algorithms.

Non-repeating algorithms

Metaheuristics often do not have the non-repeating property. If a randomized algorithm searches locally in discrete (or discretized) domains, the chance to resample a search point can be high even if the search space is huge. In evolutionary algorithms, the variation operators that are used to generate a new solution based on an existing one are often symmetric in the sense that the probability to generate some solution x' from x is equal to generating x from x'. Thus, there is always a chance to jump back to an already visited point.

We can always turn a repeating search algorithm into a non-repeating one by adding a look-up table for already visited points. The algorithm then internally uses this look-up table and only evaluates the objective function (i.e., "makes a step") for previously unseen points. Of course, this increases the memory requirements of the algorithm.

Example 8. Studies in which evolutionary algorithms searching a space of graphs are turned into non-repeating algorithms by coupling them with a search-point database are described in [16, 23].

Deterministic and randomized algorithms

In general, NFL results hold for deterministic as well as randomized algorithms. A randomized search algorithm a can be described by a probability distribution p_a over deterministic search behaviors [6]: Every search behavior generated by a single application of a randomized algorithm has a certain probability. The same behavior could have been generated by some deterministic algorithm. One can view the application of a randomized algorithm a as picking – according to a fixed, algorithm dependent distribution p_a – a deterministic search behavior at random and applying it to the problem (we view the deterministic algorithms as a subset of the randomized algorithms having degenerated probability distributions). An alternative way to see this is to think of drawing all realizations of random variables required by a randomized search method at once prior to the search process and to use these events as inputs to a deterministic algorithm (see chapter **??[reference to Thomas' Black-Box chapter]** for a detailed discussion of this issue).

Let the set A contain all deterministic search behaviors operating on $\mathscr{F}$. The performance of a randomized search algorithm a corresponds to the expectation over the possible search behaviors A w.r.t. p_a [22]:

$$\mathbb{E}\{c(Y(f,m,a))\} = \sum_{a' \in A} p_a(a')c(Y(f,m,a')) \tag{8}$$

For deterministic algorithms, the previous theorems do not only state that the average performance of two algorithms is the same across all functions, but also *any* statistic – in particular the variance – of the performance values is the same. We can derive a similar result for randomized algorithms. Let us assume that the conditions of theorem 3 are met. We consider two randomized algorithms described by p_a and p_b and extend the NFL theorems using simple transformations:

$$\begin{aligned}
&\sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f) \sum_{a' \in A} p_a(a')\delta(k, c(Y(f,m,a'))) \\
&\quad = \sum_{a' \in A} p_a(a') \underbrace{\sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f)\, \delta(k, c(Y(f,m,a')))}_{\substack{\text{independent of } a' \\ \text{because of theorem 3}}} \\
&\quad \overset{\substack{\text{for any} \\ \text{algorithm } z}}{=} \sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f)\, \delta(k, c(Y(f,m,z))) \sum_{a' \in A} p_a(a') \\
&\quad = \sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f)\, \delta(k, c(Y(f,m,z))) \sum_{b' \in A} p_b(b') \\
&\quad \overset{\substack{\text{reversing the} \\ \text{prev. arguments}}}{=} \sum_{f \in \mathscr{F}} p_{\mathscr{F}}(f) \sum_{b' \in A} p_b(b')\delta(k, c(Y(f,m,b')))
\end{aligned} \tag{9}$$

Equation (2) holds for randomized algorithms if we simply replace $c(\cdot)$ by $\mathbb{E}\{c(\cdot)\}$.[1]

Finiteness of domain and co-domain

If optimization problems solved on today's digital computers are considered, the restriction to finite domains and co-domains is no restriction at all. Still, NFL results for continuous search spaces are very interesting from the theoretical point of view. The reader is referred to [26] and [1] for different views on this topic as well as to chapter **??[reference to Marc's Bayesian Search Game chapter]**, which explains why assumptions on the Lebesgue measurability of $p_{\mathscr{F}}$ play a role when studying NFL in continuous search spaces.

Restriction to a single objective

The basic NFL theorems considers single-objective optimization. However, it also holds for multi-objective (vector) optimization as proven in [5].

[1] In general, this cannot be done in the theorems because $\sum_{f \in F} \delta(k, \sum_{a' \in A} p_a(a')c(Y(f,m,a'))) \neq \sum_{f \in F} \sum_{a' \in A} p_a(a')\delta(k, c(Y(f,m,a')))$.

Fixed objective functions

The framework considered in this chapter does not include repeated game scenarios in which the "objective function" for some agent can vary based on the behavior of other agents as it is the case in coevolutionary processes. For results and discussions of NFL in such game-like scenarios the reader is referred to the work by Wolpert and Macready presented in [37].

Averaging over all search behaviors and all performance criteria

The NFL theorems rely on averages over all possible search behaviors and all possible performance criteria. However, in practice we are most often concerned with the comparison of a subset of algorithms A using a fixed performance measure c and some fixed number of steps m. This scenario is considered in *Focused NFL theorems* [32]. Given a set of algorithms A and a performance measure c and some fixed number of steps m, Whitley and Rowe define a *focus set* as a set of functions on which the algorithms in A have the same mean performance measured by c (after m steps). The *orbit* of an objective function f is the smallest of these focus sets containing f. It is important to note that focus sets need not be c.u.p..

Example 9. Assume we want to compare two deterministic algorithms a and b run for m steps using performance measure c. Further, assume two objective functions f_1 and f_2 with $c(Y(f_1,m,a)) = c(Y(f_2,m,b))$ and $c(Y(f_2,m,a)) = c(Y(f_1,m,b))$. Then $\{f_1, f_2\}$ is clearly a focus set for $A = \{a,b\}$, c, and m – regardless whether $\{f_1, f_2\}$ is c.u.p. or not.

For details about Focused NFL theorems the reader is referred to [32] and **??[reference to Darrell's chapter]**.

4 Restricted Function Classes and NFL

In this section we take a look at restricted function classes. First, we compute the probability that a randomly chosen function class meets the conditions of the NFL theorems. Then it is argued that there are certain restrictions that are likely to constrain problem classes corresponding to "real-world " problems. It is shown that these constraints are not compatible with the conditions of the NFL theorems. This is an encouraging result for the design of metaheuristics. However, it is difficult to evaluate heuristics empirically, because good performance on some functions does not necessarily imply good performance on others – even if these functions seem to be closely related. This is underlined by the Almost NFL theorem discussed at the end of this section.

4.1 How Likely Are the Conditions for NFL?

The question arises whether the preconditions of the NFL theorems are ever fulfilled in practice. How likely is it that a randomly chosen subset is c.u.p.? There exist $2^{\left(|\mathscr{Y}|^{|\mathscr{X}|}\right)} - 1$ non-empty subsets of $\mathscr{F}$ and it holds:

Theorem 4 ([18]). *The number of non-empty subsets of $\mathscr{Y}^{\mathscr{X}}$ that are c.u.p. is given by*

$$2^{\binom{|\mathscr{X}|+|\mathscr{Y}|-1}{|\mathscr{X}|}} - 1 \tag{10}$$

and therefore the fraction of non-empty subsets c.u.p. is given by

$$\left(2^{\binom{|\mathscr{X}|+|\mathscr{Y}|-1}{|\mathscr{X}|}} - 1\right) \Big/ \left(2^{\left(|\mathscr{Y}|^{|\mathscr{X}|}\right)} - 1\right) \ . \tag{11}$$

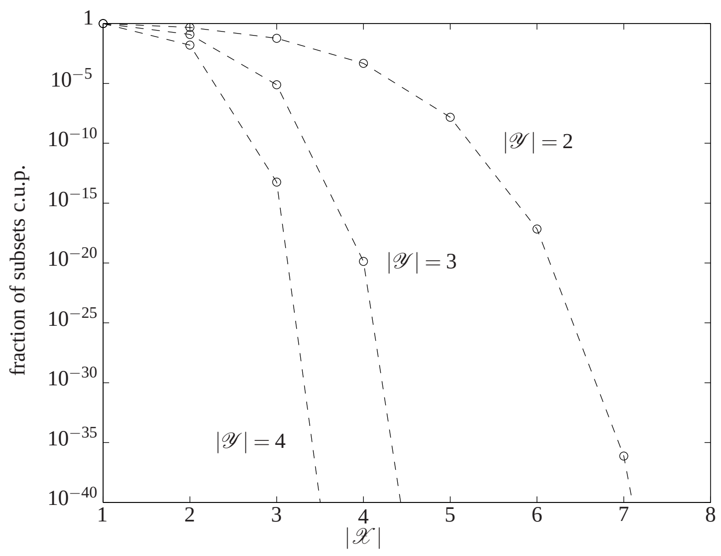

Fig. 3 Fraction of subsets of objective functions that are closed under permutation (c.u.p.) of all possible subsets depending on the number of search points and the number of possible objective function values. The ordinate gives the fraction of subsets c.u.p. on logarithmic scale given the cardinality $|\mathscr{X}|$ of the search space. The different curves correspond to different cardinalities of the space of objective function values $\mathscr{Y}$.

Figure 3 shows a plot of the fraction of non-empty subsets c.u.p. versus the cardinality of $\mathscr{X}$ for different values of $|\mathscr{Y}|$. The fraction decreases for increasing $|\mathscr{X}|$ as well as for increasing $|\mathscr{Y}|$. More precisely, using bounds for binomial coefficients one can show that (11) converges to zero double exponentially fast with increasing

$|\mathscr{X}|$ (for $|\mathscr{Y}| > e|\mathscr{X}|/(|\mathscr{X}| - e)$, where e is Euler's number). Already for small $|\mathscr{X}|$ and $|\mathscr{Y}|$ the fraction almost vanishes.

Thus, the statement "I'm only interested in a subset F of all possible functions and the precondition of the Sharpened NFL theorem is not fulfilled" is true with a probability close to one if F is chosen uniformly at random and $\mathscr{Y}$ and $\mathscr{X}$ have reasonable cardinalities.

The probability that a randomly chosen distribution over the set of objective functions fulfills the preconditions of theorem 3 has measure zero. This means that in this general and realistic scenario the conditions for a NFL result do not hold almost surely.

4.2 Structured Search Spaces and NFL

Although the fraction of subsets c.u.p. is close to zero already for small search and cost-value spaces, the absolute number of subsets c.u.p. grows rapidly with increasing $|\mathscr{X}|$ and $|\mathscr{Y}|$. What if these classes of functions are the relevant ones in the sense that they correspond to the problems we are dealing with in practice?

It can be argued that presumptions can be made for most of the functions relevant in real-world optimization: First, the search space has some structure. Second, the set of objective functions fulfills some constraints defined based on this structure. More formally, there exists a non-trivial neighborhood relation on $\mathscr{X}$ based on which constraints on the set of functions under consideration are formulated. Such constraints include upper bounds on the ruggedness or on the maximum number of local minima. Concepts such as ruggedness or local optimality require a notion of neighborhood.

A neighborhood relation on $\mathscr{X}$ is a symmetric function $n : \mathscr{X} \times \mathscr{X} \to \{0,1\}$. Two elements $x_i, x_j \in \mathscr{X}$ are called neighbors if $n(x_i, x_j) = 1$. A neighborhood relation is called non-trivial if $\exists x_i, x_j \in \mathscr{X} : x_i \neq x_j \wedge n(x_i, x_j) = 1$ and $\exists x_k, x_l \in \mathscr{X} : x_k \neq x_l \wedge n(x_k, x_l) = 0$. There are only two trivial neighborhood relations, either every two points are neighbored or no points are neighbored. Non-trivial neighborhood relations are not preserved under permutations, it holds:

Theorem 5 ([18]). *A non-trivial neighborhood relation on $\mathscr{X}$ is not invariant under permutations of $\mathscr{X}$, i.e.,*

$$\exists x_i, x_j \in \mathscr{X}, \pi \in \Pi(\mathscr{X}) : n(x_i, x_j) \neq n(\pi(x_i), \pi(x_j)) \quad . \tag{12}$$

This result is quite general. Assume that the search space $\mathscr{X}$ can be decomposed as $\mathscr{X} = \mathscr{X}_1 \times \cdots \times \mathscr{X}_l$, $l > 1$, and let a non-trivial neighborhood $n_i : \mathscr{X}_i \times \mathscr{X}_i \to \{0,1\}$ exists on one component $\mathscr{X}_i$. This neighborhood induces a non-trivial neighborhood on $\mathscr{X}$, where two points are neighbored if their i-th components are neighbored with respect to n_i regardless of the other components. Thus, the constraints discussed in the examples below need only refer to a single component. Note that the neighborhood relation need not be the canonical one (e.g., the Hamming-distance for Boolean search spaces). For example, if integers are encoded by bit-strings, then the bit-strings can be defined as neighbored iff the corresponding integers are. The following examples illustrate further implications of theorem 5 [18].

Example 10. Consider a non-empty subset $F \subset \mathscr{F}$ where the co-domains of the functions have more than one element and a non-trivial neighborhood relation exists. If for each $f \in F$ it is not allowed that a global maximum is neighbored to a global minimum (i.e., we have a constraint "steepness"), then F is not c.u.p. (because for every $f \in F$ there exists a permutation that maps a global minimum and a global maximum of f to neighboring points). An example of such a function class is OneMax* as described in section **??** of chapter **??[reference to Thomas' Black-Box chapter]**.

Example 11. Imposing an upper bound on the complexity of possible objective functions can lead to function classes that are not c.u.p. Let us assume without loss of generality minimization tasks. Then the number of suboptimal local minima is often regarded as a measure of complexity of an objective function [31] (see section 4.3 for other notions of complexity). Given a neighborhood relation on $\mathscr{X}$, a local minimum can be defined as a point whose neighbors all have worse fitness. As a concrete example, consider all mappings $\{0,1\}^\ell \to \{0,1\}$ not having maximum complexity in the sense that they have less than the maximum number of 2^{n-1} local minima w.r.t. the ordinary hypercube topology on $\{0,1\}^\ell$. For example, this set does not contain mappings such as the parity function, which is one iff the number of ones in the input bitstring is even. This set is not c.u.p. The general statement that imposing an upper bound on the number of local minima leads to function classes not c.u.p. can be formally proven using the following line of arguments. Let $l(f)$ be the number of local minima of a function $f \in \mathscr{F}$ and let $B_{h_f} \subseteq \mathscr{F}$ be the set of functions in $\mathscr{F}$ with the same $\mathscr{Y}$-histogram as f (i.e., functions where the number of points in $\mathscr{X}$ that are mapped to each $\mathscr{Y}$-value is the same as for f). Given a function f we define $l^{\max}(f) = \max_{f' \in B_{h_f}} l(f')$ as the maximal number of local minima that functions in $\mathscr{F}$ with the same $\mathscr{Y}$-histogram as f can possibly have. For a non-empty subset $F \subset \mathscr{F}$ we define $l^{\max}(F) = \max_{f \in F} l^{\max}(f)$. Let $g(F) \in \mathscr{F}$ be a function with $l(g(F)) = l^{\max}(F)$ local minima and the same $\mathscr{Y}$-histogram as a function in F. Now, if the number of local minima of every function $f \in F$ is constrained to be smaller than $l^{\max}(F)$ (i.e., $\max_{f \in F} l(f) < l^{\max}(F)$), then F is not c.u.p.—because $\exists f \in F$ with the same $\mathscr{Y}$-histogram as g and thus $\exists \pi \in \Pi(\mathscr{X}) : f \circ \pi = g$.

4.3 The Almost NFL Theorem

The results presented above are encouraging, because they suggest that for a restricted problem class the NFL results will most likely not apply. However, this does not promise "free lunch". For a problem class violating the conditions for a NFL result, the average performance of two algorithms may differ. But this difference may be negligible.

The performance on two functions, although they are similar according to some reasonable similarity measure, may differ significantly. In particular, for any function f on which algorithm *a* performs well, there may be many functions on which it performs badly and that are similar to f in a reasonable sense. This becomes clear from the Almost NFL (ANFL) theorem derived by Droste, Jansen, and Wegener [6], which proves such statements for a particular scenario:

Theorem 6 (Almost NFL theorem [6]). *Let $\mathscr{F}$ be the set of objective functions $f : \{0,1\}^n \to \{0,1,\dots,N-1\}$ for fixed positive integers n and N. For any algorithm a operating on $\mathscr{F}$ and any function $f \in \mathscr{F}$ there exist at least $N^{2^{n/3}-1}$ functions in $\mathscr{F}$ that agree with f on all but at most $2^{n/3}$ inputs for which a does not find their optimum within $2^{n/3}$ steps with a probability bounded above by $2^{-n/3}$.*

Exponentially many of these functions have the additional property that their evaluation time, circuit size representation, and Kolmogorov complexity is only by an additive term of $O(n)$ larger than the corresponding complexity of f.

The last sentence formalizes that extremely bad behavior can be expected on functions which are similar to our reference (e.g., benchmark) function f. These functions do not only coincide on $2^{n/3}$ inputs with f, they also have a similar complexity. Complexity can either be measured in the number of steps needed to evaluate the objective function at a certain point (evaluation time), the length of the shortest program implementing f (Kolmogorov complexity), or by the size of a circuit representation of f (circuit size representation).

5 Search and Markov Decision Processes

If a problem class does not fulfill the assumptions for NFL, then there may be differences in performance between algorithms on this class. The question arises: what is then the best method given a problem class $p_{\mathscr{F}}$, a performance measure, and a number of steps *m*? In the following, it is shown how to determine an optimal algorithm using dynamic programming (DP) [15]. For a similar approach see [1] and chapter **??[reference to Olivier's Optimal Search chapter]**. However, it is not claimed that this way is efficient.

The problem of finding an optimal search algorithm can be transformed to a *finite horizon optimal control problem* that can be solved by DP. In the context of mathematical optimization for optimal control, DP is concerned with some discrete-time dynamic system, in which at time *t* the state s_t of the system changes according to given transition probabilities that depend on some action a_{t+1} (I adopt a standard formalism and use *a* for actions instead of algorithms in this section). Each transition results in some immediate reward r_{t+1} (or, alternatively, immediate cost) [3]. The dynamic system can be described by a finite Markov decision process:

Definition 2 (finite MDP). A finite Markov decision process $[\mathscr{S},\mathscr{A},\mathscr{P},\mathscr{R}]$ (finite MDP) is given by:

1. a finite set $\mathscr{S}$ of states;

2. a finite set of actions $\mathscr{A}$, where $\mathscr{A}(s)$ denotes the set of possible actions in state $s \in \mathscr{S}$;
3. transition probabilities $\mathscr{P}^a_{ss'} = \Pr\{s_{t+1} = s' \,|\, s_t = s, a_t = a\}$ describing how likely it is to go from $s \in \mathscr{S}$ to $s' \in \mathscr{S}$ when taking action $a \in \mathscr{A}$, and
4. expected reward values $\mathscr{R}^a_{ss'} = \mathrm{E}\{r_{t+1} \,|\, s_{t+1} = s', s_t = s, a_t = a\}$ expressing the expected reward when going from $s \in \mathscr{S}$ to $s' \in \mathscr{S}$ after taking action $a \in \mathscr{A}$.

The goal of a *finite horizon problem*[2] is to find a policy $\varpi : \mathscr{S} \to \mathscr{A}$ that maximizes the accumulated m-step reward

$$R_m = \sum_{t'=1}^{m} r_{t'} \quad . \tag{13}$$

Given $[\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}]$ the optimal policy can be computed using established standard techniques [3, 2].

Now we turn an optimization problem (as illustrated Fig. 2) into a finite MDP. First, we need to fix some additional notation. The length of a sequence $T_m = \langle (x_1, f(x_1)), \dots, (x_m, f(x_m)) \rangle$ is given by $\text{length}(T_m) = m$. Adding an element to a sequence is indicated by $\langle x_1, x_2, \dots, x_n \rangle \oplus x' = \langle x_1, x_2, \dots, x_n, x' \rangle$. Given T_M, we denote the corresponding sequence and set of search points by $X(T_m) = \langle x_1, x_2, \dots, x_m \rangle$ and $\mathscr{X}(T_m) = \{x_1, x_2, \dots, x_m\}$, respectively, and the sequence of cost values by $Y(T_m) = \langle f(x_1), f(x_2), \dots, f(x_m) \rangle$.

We define the set $F(T_m)$ of all functions that are *compatible with T_m in $F \in \mathscr{F}$* by

$$F(T_m) = \{g \,|\, g \in F \wedge \forall x \in \mathscr{X}(T_m) : g(x) = f(x)\} \tag{14}$$

The *search MDP* (S-MDP) given an optimization scenario can now be formalized as follows:

Definition 3 (S-MDP). Given a problems class $p_{\mathscr{F}}$, a performance measure c, and a number of steps $m \in \{1, \dots, |\mathscr{X}|\}$, the search MDP (S-MDP) is defined by

1. $\mathscr{S} =$ set of all possible traces T_n with $0 \le n \le m$;
2. $\mathscr{A}(s) = \mathscr{X} \setminus \mathscr{X}(s)$;
3. $\mathscr{P}^a_{ss'} = \begin{cases} 0 & X(s') \neq X(s) \oplus a \\ \sum\limits_{g \in F(s')} p_{\mathscr{F}}(g) \Big/ \sum\limits_{h \in F(s)} p_{\mathscr{F}}(h) & \text{otherwise} \end{cases}$;
4. $\mathscr{R}^a_{ss'} = \begin{cases} 0 & \text{if length}(s) \neq m-1 \\ c(Y(s')) & \text{otherwise} \end{cases}$.

This MDP has been constructed such that the following theorem holds, which establishes the link between optimal search algorithm and optimal policy:

Theorem 7. *For any performance measure c, any problem class $p_{\mathscr{F}}$ over $\mathscr{F}$, and any $m \in \{1, \dots, |\mathscr{X}|\}$, an algorithm b that maximizes*

[2] It is also possible to map the optimization to an *infinite horizon problem* with appropriate *absorbing states*.

$$\sum_{f\in F} p_{\mathscr{F}}(f)\, c(Y(f,m,b))$$

is given by the optimal policy for the corresponding S-MDP (with $t=0$ and $s_0=\langle\rangle$).

Proof. By definition, every policy operating on a S-MDP is an algorithm. The states of the dynamic system correspond to the sequences of previously evaluated search points and an action correspond to exploring a new search point.

First, we verify that the definition of $\mathscr{P}^a_{ss'}$ leads to proper probability distributions, that is, that $\forall s\in\mathscr{S}:\forall a\in\mathscr{A}(s):\sum_{s'\in\mathscr{S}}\mathscr{P}^a_{ss'}=1$. We rewrite the definition of $\mathscr{P}^a_{ss'}$ as

$$\mathscr{P}^a_{ss'}=\delta(X(s'),X(s)\oplus a)\sum_{g\in F(s')} p_{\mathscr{F}}(g)\Big/\sum_{h\in F(s)} p_{\mathscr{F}}(h)\quad . \tag{15}$$

For every $f\in F(s)$ and $a\in\mathscr{A}(s)$ there is exactly one $s'\in\mathscr{S}$ with $X(s')=X(s)\oplus a$ and $f\in F(s')$. Thus we have

$$\sum_{h\in F(s)} p_{\mathscr{F}}(h)=\sum_{\substack{s'\in\mathscr{S}\\ \wedge X(s')=X(s)\oplus a}}\;\sum_{g\in F(s')} p_{\mathscr{F}}(g)=\sum_{s'\in\mathscr{S}}\delta(X(s'),X(s)\oplus a)\sum_{g\in F(s')} p_{\mathscr{F}}(g) \tag{16}$$

and the normalization is correct.

Next, we show that maximizing the accumulated reward $\mathbb{E}\{R_0\,|\,s_0=\langle\rangle\,,\varpi\}$ maximizes the performance measure $\sum_{f\in F} p_{\mathscr{F}}(f)\,c(Y(f,m,\varpi))$. The accumulated immediate reward reduces to

$$R_0=\sum_{t'=1}^{m} r_{t'}=r_m\quad . \tag{17}$$

Given a function $f\in\mathscr{F}$ and a policy ϖ we have $r_m=c(Y(f,m,\varpi))$ and therefore

$$\mathbb{E}\{R_0\,|\,s_0=\langle\rangle\,,f,\varpi\}=\mathbb{E}\{r_m\,|\,s_0=\langle\rangle\,,f,\varpi\}=c(Y(f,m,\varpi)) \tag{18}$$

as the process is deterministic for fixed f. Thus

$$\mathbb{E}\{R_0\,|\,s_0=\langle\rangle\,,\varpi\}=\sum_{f\in\mathscr{F}} p_{\mathscr{F}}(f)\,c(Y(f,m,\varpi))\quad , \tag{19}$$

which completes the proof. □

Solving the S-MDP leads to an optimal algorithm, which "plans" the next search points in order to maximize the reward (e.g., to find the best solution in m steps). Although solving a MDP can be done rather efficiently[3] in terms of scaling with $|\mathscr{S}|$, $|\mathscr{A}|$, and m, *solving the S-MDP is usually intractable* because $|\mathscr{S}|$ obviously scales

[3] In a S-MDP no state is ever revisited. Hence, for any policy, the transition probability graph is acyclic and thus *value iteration* finds the optimal policy after at most m steps, where each step needs $O(|\mathscr{A}||\mathscr{S}|)^2$ computations (see [3, sec. 2.2.2] or [2]).

badly with the dimensionality of the optimization problem. In chapter **??[reference to Olivier's Optimal Search chapter]**, Teytaud and Vazquez discuss this issue and optimal search algorithms in general in more detail.

6 What Can We Learn from the NFL Results for the Design of Metaheuristics?

There is no universal best search or optimization algorithm. A "careful consideration of the No Free Lunch theorems forces us to ask what set of problems we want to solve and how to solve them" [33]. The NFL theorem "highlights the need for exploiting problem-specific knowledge to achieve better than random performance" [37]. Its essence is nicely captured in the discussion of the NFL results for learning [34] by Bousquet et al. [4]. If "there is no a priori restriction on the possible phenomena that are expected [...] there is no better algorithm (any algorithm would be beaten by another one on some phenomena)" [4]. If their words are adapted for the search and optimization scenario, the main message of the NFL theorems may be summarized as follows:

If there is no restriction on how the past (already visited points) can be related to the future (not yet explored search points), efficient search and optimization is impossible.

Between the two extremes of knowing nothing about a problem domain and knowing a domain so well that an efficient, highly specialized algorithm can be derived in reasonable time, there is enough room for the application of well designed metaheuristics. Often we have only little, quite general knowledge about the properties of a problem class. In such a case, using quite general metaheuristics exploiting these properties may be a good choice. In practice, sometimes fine tuning an algorithm to a problem class is not efficient in terms of development time and costs. Then a more broadly tuned metaheuristic may be preferable.

Finally, let me mention some further conclusions that can be drawn from the results reviewed in this chapter.

The preconditions of the NFL theorem are not met in practice

I argue that if we consider a restricted class of problems, it is reasonable to assume that the necessary conditions in the NFL theorems are not fulfilled. First, if we would select a problem class at random, the probability that the conditions hold is extremely low. Second, if the objective functions in a problem class obey some constraints that are defined with respect to some neighborhood relation on the search space, then the necessary prerequisites are likely to be violated. Thus, we can hope to come up with metaheuristics showing above average performance for real world problem classes.

However, one has to keep in mind that the strict NFL results refer to averages over all possible search behaviors and performance criteria. If we compare a selection of algorithms using a fixed criterion, there can exist sets of functions on which the algorithms have the same mean performance even if the sets do not meet the conditions of the general NFL theorems. This is investigated in the context of Focused NFL theorems, see [32] and **??[reference to Darrell's chapter]**.

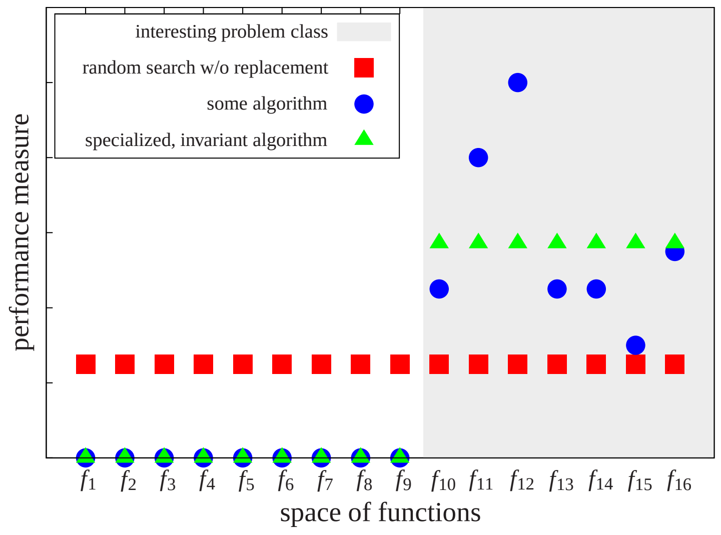

Fig. 4 Distribution of mean performance over a discrete space of problems for some randomized algorithms. The algorithms indicated by triangles and circles both have a similar mean performance on the interesting problems (i.e., some problem class we are developing a metaheuristic for), but a different variance. Both are biased towards the interesting problems. Uniform random search and the algorithm indicated by triangles are invariant under the choice of the problem instance from the class of interesting problems.

Generalization from benchmark problems is dangerous

The Almost NFL theorem and the second statement of corollary 1 show that drawing general conclusions about the ranking of algorithms – even if referring to a restricted problem class – based on evaluating them on single benchmark functions is dangerous (see the discussion in [33]). For every set of functions on which algorithm *a* outperforms algorithm *b* there is a set of function on which the opposite is true. Thus, one must ensure that the results on the considered benchmark functions can be generalized to the whole class of problems the algorithms are designed for. If algorithm *b* outperforms *a* on some test functions that are not likely to be observed in practice (in my opinion this includes functions frequently used in benchmark suites, e.g., some “deceptive” problems), this can even be regarded as an argument in favor of a.

There are ways to derive valid statements about algorithm performance on a problem class without testing the algorithm on every instance of the class. First, one can derive formal proofs about the performance of the algorithms. Of course, this can be extremely difficult. Second, it is possible to generalize from empirical results on single benchmark functions to a class of problems in a strict sense if the algorithms have certain invariance properties, see [11, 12] and chapter **??[reference to chapter discussing invariance by Niko]** for a discussion.

Example 12. Evolutionary algorithms in which the selection is based on ranking, such as the CMA-ES [13] presented in chapter **??[reference to Niko’s CMA-ES chapter]**, are invariant under order-preserving transformations of the fitness function. For example, if the fitness values given by a function $f(x)$ are all positive, the performance of a purely rank-based algorithm on f generalizes to $f(x)^2, \log(f(x)), \dots$. The CMA-ES has several additional invariance properties, in particular it is invariant under rotation of the search space.

As there is no well-performing universal search algorithm, the metaheuristics we are developing must be biased towards certain problem classes. There is some trade-off between the specialization to a problem class required for efficient optimization and invariance properties. This becomes obvious by the invariance properties of uniform random-search, which is fully unbiased. Figure 4 illustrates specialization and invariance properties, especially showing the performance of an algorithm that is biased towards an interesting problem class and at the same time invariant under the choice of the problem instance from this class. Understanding, formulating, and achieving the right bias and the right invariance properties is perhaps the most important aspect when designing metaheuristics.

Further Reading

The book chapter on “Complexity Theory and the No Free Lunch Theorem” by Whitley and Watson is recommended for an alternative review of NFL for search and optimization [33]. The proof of the basic NFL theorem can found in [36], the proof of the Sharpened NFL theorem in [27]. Theorem 3 is proven in [20] and the results in section 4.1 and 4.2 in [18]. The Almost NFL theorem is derived in [6], and Focused NFL is discussed in [32].

Acknowledgments

Parts of this text are based on former joint work with Marc Toussaint. The author acknowledges support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience – Bernstein Fokus: “Learning behavioral models: From human experiment to technical assistance”, grant FKZ 01GQ0951.

References

1. Auger, A., Teytaud, O.: Continuous lunches are free plus the design of optimal optimization algorithms. Algorithmica (2009)
2. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific **2** (2007)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming. Athena Scientific (1996)
4. Bousquet, O., Boucheron, S., Lugosi, G.: Introduction to statistical learning theory. In: O. Bousquet, U. von Luxburg, G. Rätsch (eds.) Advanced Lectures in Machine Learning, *LNAI*, vol. 3176, pp. 169–207. Springer-Verlag (2004)
5. Corne, D.W., Knowles, J.D.: No free lunch and free leftovers theorems for multiobjective optimisation problems. In: Evolutionary Multi-Criterion Optimization (EMO 2003), LNCS, pp. 327–341. Springer-Verlag (2003)
6. Droste, S., Jansen, T., Wegener, I.: Optimization with randomized search heuristics – The (A)NFL theorem, realistic scenarios, and difficult functions. Theoretical Computer Science **287**(1), 131–144 (2002)
7. English, T.: On the structure of sequential search: Beyond “No free lunch”. In: J. Gottlieb, G.R. Raidl (eds.) Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004), *LNCS*, vol. 3004, pp. 95–103. Springer-Verlag (2004)
8. English, T.M.: Evaluation of evolutionary and genetic optimizers: No free lunch. In: L.J. Fogel, P.J. Angeline, T. Bäck (eds.) Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP V), pp. 163–169. MIT Press (1996)
9. English, T.M.: Optimization is easy and learning is hard in the typical function. In: A. Zalzala, C. Fonseca, J.H. Kim, A. Smith (eds.) Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), pp. 924–931. IEEE Press (2000)
10. Giraud-Carrier, C., Provost, F.: Toward a justification of meta-learning: Is the No Free Lunch Theorem a show-stopper? In: Proceedings of the ICML-2005 Workshop on Meta-learning (2005)
11. Hansen, N.: Invariance, self-adaptation and correlated mutations and evolution strategies. In: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J.M. Guervós, H.P. Schwefel (eds.) Parallel Problem Solving from Nature (PPSN VI), *LNCS*, vol. 1917, pp. 355–364. Springer-Verlag (2000)
12. Hansen, N.: Adaptive encoding: How to render search coordinate system invariant. In: G. Rudolph, T. Jansen, S.M. Lucas, C. Poloni, N. Beume (eds.) Parallel Problem Solving from Nature (PPSN X), *LNCS*, vol. 5199, pp. 205–214. Springer-Verlag (2008)
13. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2), 159–195 (2001)
14. Hume, D.: A Treatise of Human Nature, chap. Sect. vi. Of the Inference From the Impression to the Idea. Web edition published by eBooks@Adelaide, 2006 (1739)
15. Igel, C.: Recent results on no-free-lunch for optimization. In: H. Beyer, T. Jansen, C. Reeves, M.D. Vose (eds.) Theory of Evolutionary Algorithms, no. 04081 in Dagstuhl Seminar Proceedings, Abstract Collection. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl (2004)

16. Igel, C., Stagge, P.: Graph isomorphisms effect structure optimization of neural networks. In: International Joint Conference on Neural Networks (IJCNN 2002), pp. 142–147. IEEE Press (2002)
17. Igel, C., Toussaint, M.: Neutrality and self-adaptation. Natural Computing **2**(2), 117–132 (2003)
18. Igel, C., Toussaint, M.: On classes of functions for which No Free Lunch results hold. Information Processing Letters **86**(6), 317–321 (2003)
19. Igel, C., Toussaint, M.: Recent results on no-free-lunch theorems for optimization. arXiv preprint cs.NE/0303032 (2003). Http://arxiv.org/abs/cs.NE/0303032
20. Igel, C., Toussaint, M.: A No-Free-Lunch theorem for non-uniform distributions of target functions. Journal of Mathematical Modelling and Algorithms **3**(4), 313–322 (2004)
21. Köppen, M., Wolpert, D.H., Macready, W.G.: Remarks on a recent paper on the "No Free Lunch" theorems. IEEE Transactions on Evolutionary Computation **5**(3), 295–296 (1995)
22. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
23. Niehaus, J., Igel, C., Banzhaf, W.: Reducing the number of fitness evaluations in graph genetic programming using a canonical graph indexed database. Evolutionary Computation **15**(2), 199–221 (2007)
24. Radcliffe, N.J., Surry, P.D.: Fundamental limitations on search algorithms: Evolutionary computing in perspective. In: J. van Leeuwen (ed.) Computer Science Today: Recent Trends and Development, *LNCS*, vol. 1000, pp. 275–291. Springer-Verlag (1995)
25. Rawlins, G.J.E.: Introduction. In: G.J.E. Rawlins (ed.) Foundations of Genetic Algorithms (FOGA), pp. 1–10. Morgan Kaufmann Publishers (1991)
26. Rowe, J.E., Vose, M.D., Wright, A.H.: Reinterpreting no free lunch. Evolutionary Computation **17**(1), 117–129 (2009)
27. Schumacher, C.: Fundamental limitations of search. Ph.D. thesis, University of Tennessee (2000)
28. Schumacher, C., Vose, M.D., Whitley, L.D.: The No Free Lunch and description length. In: L. Spector, E. Goodman, A. Wu, W. Langdon, H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, E. Burke (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 565–570. Morgan Kaufmann (2001)
29. Streeter, M.J.: Two broad classes of functions for which a No Free Lunch Result does not hold. In: E. Cantú-Paz, J.A. Foster, K. Deb, D. Davis, R. Roy, U.M. O'Reilly, H.G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M.A. Potter, A.C. Schultz, K. Dowsland, N. Jonoska, J. Miller (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003), no. 2724 in LNCS, pp. 1418–1430. Springer-Verlag (2003)
30. Suttorp, T., Hansen, N., Igel, C.: Efficient covariance matrix update for variable metric evolution strategies. Machine Learning **75**(2), 167–197 (2009). DOI 10.1007/s10994-009-5102-1
31. Whitley, D.: A Free Lunch proof for gray versus binary encodings. In: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), vol. 1, pp. 726–733. Morgan Kaufmann (1999)
32. Whitley, D., Rowe, J.: Focused no free lunch theorems. In: M. Keijzer, et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 811–818. ACM (2008)
33. Whitley, D., Watson, J.P.: Complexity theory and the No Free Lunch theorem. In: E.K. Burke, G. Kendall (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, chap. 11, pp. 317–339. Springer-Verlag (2005)
34. Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. Neural Computation **8**(7), 1341–1390 (1996)
35. Wolpert, D.H., Macready, W.G.: No Free Lunch theorems for search. Tech. Rep. SFI-TR-05-010, Santa Fe Institute, Santa Fe, NM, USA (1995)
36. Wolpert, D.H., Macready, W.G.: No Free Lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)

37. Wolpert, D.H., Macready, W.G.: Coevolutionary free lunches. IEEE Transactions on Evolutionary Computation **9**(6), 721–735 (2005)