A review of No Free Lunch Theorems for search

Edgar A. Duéñez-Guzmán and Marte A. Ramírez-Ortegón

Abstract. Black-box search techniques are used as general purpose optimizers and are purported to robustly solve computationally complex problems. The No-free-lunch theory (NFL) aims to understand the limitations of black-box search. As its name implies, there are typically tradeoffs when designing an algorithm: in order to perform better in a particular scenario, it needs to perform worse in another one. Disagreements over the ultimate consequences of the theory have persisted over almost two decades, with some researchers insisting that NFL results have limited applicability, and others refining the extant results to address such limitations. Regardless of their immediate applicability, NFL theorems help us understand the underlying symmetries and structure of search algorithms. Here we review the main NFL theorems and discuss their application to real world problem solving.

Keywords. Benchmarks, Black-Box Search, Minimax, No Free Lunch, Optimization

Solutions to problems
are easy to find:
the problem's a great
contribution.
What's truly an art
is to wring from your mind
a problem to fit
a solution.

–Piet Hein

1. Introduction

Since the 1960s, search algorithms inspired by physical and biological processes have been a popular approach to general purpose optimization [**14, 17, 12, 23, 3, 8, 28, 11, 4**]. These algorithms are especially attractive to researchers working on computationally complex problems. They are frequently implemented as black-box search; that is, they are applied to optimization and search problems without exploiting *a priori* information about the objective function. Such algorithms acquire information about the objective function only by evaluating it. Many of these black-box search algorithms have been regarded as robust optimization techniques [**14, 12, 21, 13**].

2010 *Mathematics Subject Classification.* Primary 68Q99.

The *No free lunch* (NFL) theory aims to understand the fundamental limitations of algorithms. As the name implies, there are typically tradeoffs when designing an algorithm: an algorithm that performs better in one scenario performs worse in another. NFL theorems [**34, 35**] were conceived to investigate whether different search algorithms could differ in robustness. The original NFL theorem states that the graph of an algorithm's performance against all objective functions has the same expected, maximum, and minimum as any other algorithm's performance graph. In fact, all performance graphs are equal, up to a permutation. NFL-like results have been proposed in diverse areas including machine learning [**7, 36, 38**], multi-objective optimization [**5**], discrete Laplace operators [**29**], induction and combinatorial problems [**37**], Walsh analysis [**31**], noise prediction [**20**], and others. In this review, we concentrate on NFL theorems for search.

NFL theorems have attracted considerable controversy due to their potential implications for popular black-box search techniques such as evolutionary algorithms, tabu search, simulated annealing and other nature-inspired algorithms [**7, 9, 15, 1, 18**]. Auger & Teytaud argued that NFL does not apply to functions over continuous domains [**1**]. The apparent limitations discussed in [**1, 2**] arise from the probabilistic framework within which original NFL theorems were expressed. However, the set-theoretic NFL theorems of Rowe, *et al.* [**24**] obviate these limitations by dispensing with probability altogether.

Other criticisms of NFL, however, are still debated. Droste, *et al.* [**9**] observed that the likelihood of facing a particular optimization problem is often not uniform. They argued that weighted average performance is, thus, the most meaningful metric to summarize aggregate optimization behavior. The original NFL assumes a uniform distribution, and was later extended to nonuniform distributions [**35**]. However, the nonuniformities considered by Wolpert & Macready [**35**] do not address all the limitations pointed out in [**9**]. As an attempt to alleviate the requirement for uniform average performance, and the equal consideration of all target functions, Igel & Toussaint [**16**] proposed a Non-Uniform NFL, which still failed to address all limitations mentioned in [**9**]. The Non-Uniform NFL was later proven by Rowe, *et al.* [**24**] to be a corollary of the Sharpened NFL by Schumacher, *et al.* [**26, 27**].

Similarly, Domingos [**7**] pointed out that instead of measuring performance in all scenarios, real world problem-solvers are employed in special, restricted contexts. Progressively more specific NFL theorems have been proposed that hold if the set of functions considered, the objective function distribution, or the relationship between functions and algorithms has special properties; permutation closure in the case of the Sharpened NFL [**26, 27**] and focused sets in the case of Focused NFL [**32**]. In practice, the assumption of permutation closure is rarely satisfied [**15**]; however, the generality of focused sets remains unclear.

The pitfalls for NFL applicability noted above stem from the traditional focus in characterizing the classes of functions over which some or all algorithms have the same performance. Researchers of black-box search, instead, typically establish a benchmark, and evaluate multiple algorithms over it. Performance over the benchmark is hoped to be indicative of performance over a related group of target problems. In response to this, Whitley a & Rowe [**32**] changed the focus of NFL to arbitrary benchmarks, and Duéñez-Guzmán & Vose [**10**] have subsequently begun to develop this perspective. These results apply to collections of functions that need not be permutation closed, nor focused.

Importantly, the cautionary observations mentioned above do not invalidate NFL results, but instead warn against their misapplication. An example for which a particular NFL theorem does not apply indicates a potential opening for future research.

In this article we introduce the main theorems of NFL, focusing on their interpretation and implications for optimization problems. To facilitate reading, we exclude all proofs; the interested reader is invited to look at the original sources for these. We will first discuss deterministic algorithms and later examine stochastic algorithms. For each, we review the theoretical background and major NFL theorems. We conclude by discussing general implications of NFL for black-box search, and potential future directions of NFL research.

Part 1. Deterministic Algorithms

2. Theoretical background

The original No Free Lunch (NFL) theorems were expressed for functions of finite domain and codomain and in the language of probability [**34, 35**]. However, Auger & Teytaud [**1, 2**] showed that probability is inadequate to affirm unconstrained NFL results in continuous cases. An alternative set-theoretic expression which eliminates the limitations of probability on NFL theorems has been employed by several researchers [**26, 27, 32, 24, 25, 10**]. In this article, we focus on this set-theoretic expression of NFL.

Before we can enunciate the NFL theorems, we need to formalize some concepts that are commonly used in NFL theorems. These concepts include traces, algorithms, black-box search and permutations of functions.

Start by fixing two (arbitrary) finite sets $\mathcal{X}$ and $\mathcal{Y}$. A target function (or simply, a function) $f : \mathcal{X} \to \mathcal{Y}$ is an element of the space of all functions $\mathcal{Y}^{\mathcal{X}}$ with domain $\mathcal{X}$ and codomain $\mathcal{Y}$. A set of target functions will be called a *benchmark*. Let y_i denote $f(x_i)$. A function itself can be considered as a set of pairs (x_i, y_i) with the condition that $x_i = x_j$ implies $y_i = y_j$. A *sequence* $S = \langle s_0 \ldots s_i \ldots \rangle$ can be considered as a function mapping i to s_i. As an example, the space of all sequences of elements of $\mathcal{Y}$ of length 3 is given by $\mathcal{Y}^3$, where the exponent 3 in this case is interpreted as a set containing three elements ($\{0, 1, 2\}$). A sequence of values from $\mathcal{Y}$ will be called a *performance vector*.

Of central importance to NFL is the concept of a *trace*. A *trace* T corresponding to f is a finite sequence of elements from f,

$$T = \langle (x_0, y_0), \ldots \rangle \ ,$$

where the x-components are unique. A trace represents a particular search history of the space $\mathcal{X}$ and its corresponding target values in $\mathcal{Y}$. For instance, one can interpret trace $\langle (x_0, y_0), (x_1, y_1), (x_2, y_2) \rangle$ as a process visiting first x_0, evaluating f on x_0 to obtain y_0, the process later visits x_1 obtaining y_1, and finally visits x_2 obtaining y_2.

To refer to the sub-components of a trace the following notation is used

$$\begin{array}{lcll} T_x & = & \langle x_0, \ldots \rangle & \text{sequence of } x\text{-components} \\ T_y & = & \langle y_0, \ldots \rangle & \text{sequence of } y\text{-components} \end{array}$$

The performance vector associated with T is T_y.

For a sequence S, let S^* be the range of S (without order). In particular, $T^* \subset f$. Trace T is *total* if $T_x^* = \mathcal{X}$ (equivalently, if $T^* = f$). A trace that is not total is *partial*.

Let $\mathcal{T}(f)$ be the set of all partial traces corresponding to f, and let the set of all partial traces be

$$\mathcal{T} = \bigcup_{f \in \mathcal{Y}^{\mathcal{X}}} \mathcal{T}(f)$$

A *search operator* is a function $g : \mathcal{T} \to \mathcal{X}$ such that $g(T) \notin T_x^*$. A *non-repeating/non-revisiting deterministic black box search algorithm* $\mathcal{A}$ corresponds to a search operator g, and will be referred to simply as an *algorithm*. Algorithm $\mathcal{A}$ applied to function f is denoted by $\mathcal{A}_f$, and maps traces to traces

$$\mathcal{A}_f(T) \;=\; \begin{cases} T \,|\, \langle (g(T), f \circ g(T)) \rangle & \text{if } T \in \mathcal{T}(f) \\ T & \text{otherwise} \end{cases}$$

where $|$ denotes concatenation.

This formal definition of an algorithm corresponds to the intuitive notion of a program running on a particular target problem as follows. Algorithm $\mathcal{A}$ runs on function f by beginning with the empty trace $\emptyset$, and repeatedly applying $\mathcal{A}_f$. At each step, the search operator of $\mathcal{A}$ produces the new point in the search space $\mathcal{X}$ that the algorithm will visit, and it makes this decision based on the complete history of which points the algorithm has already visited. Of course a particular search operator (and thus, an algorithm) might only use a subset of this information, or none of it at all.

Notice that this formal definition of an algorithm abstracts many details of its implementation. For instance, algorithmic complexity and running time are ignored. The restriction to deterministic algorithms is weaker that it might seem if we consider that the implementation of many stochastic algorithms, in fact, uses pseudo-random number generators that are deterministic. A pseudo-stochastic algorithm will behave differently if and only if the random seed used is different. Therefore, we simply consider the combination of the algorithm and a particular seed as a new deterministic algorithm. In Part 2, we will relax the deterministic constrain and discuss NFL results for truly stochastic algorithms.

In addition, many real-world algorithms do revisit points in the search space $\mathcal{X}$. However, such repeating algorithms produce the same trace as a non-repeating algorithm by ignoring repeats in the trace (equivalently, taking as output of

the search operator the first point in T_x not previously visited). NFL theory typically ignores the case of algorithms that never visit the whole of the search space, regardless of running time. Note that such algorithms can have no better performance than alternative implementations of the same algorithm where, when caught in such a loop, simply restarts at a previously unvisited point.

Following [**27**], denote by $\mathcal{A}(f)$ the total trace produced (by running $\mathcal{A}$ on f to convergence). Note that $\mathcal{A}(f)^* = f$. Algorithms $\mathcal{A}$ and $\mathcal{A}'$ are regarded as equal if and only if $\mathcal{A}(f) = \mathcal{A}'(f)$ for all f. Let $\tilde{\mathcal{A}}(f)$ denote the performance vector $\mathcal{A}(f)_y$ associated with running to completion algorithm $\mathcal{A}$ on function f.

Permutations of the search space are central in NFL theory, and are used to define group actions on functions and algorithms [**22, 26**]. Symmetries of the space of functions or algorithms are expressed as permutations of the search space. Let us denote by $\mathcal{X}!$ the set of permutations (bijections) of $\mathcal{X}$. Given a permutation (i.e. a bijection) $\sigma : \mathcal{X} \to \mathcal{X}$, the *permutation of f by σ* (that is, the action of σ on the function f) is a new function denoted σf and defined by

$$(\sigma f)(x) = f(\sigma^{-1}(x)).$$

Thus, a permutation σ of $\mathcal{X}$ may also be considered as a permutation of the space of functions $\mathcal{Y}^{\mathcal{X}}$ (i.e. an element of $\mathcal{Y}^{\mathcal{X}}!$) via $f \mapsto \sigma f$. Intuitively, the function σf behaves exactly as f but over a permutation of $\mathcal{X}$. For example, if $\mathcal{X} = \{0, 1, 2\}$, $\mathcal{Y} = \{0, 1\}$, $\sigma(x) = (x + 1) \mod (3)$ and

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x = 1 \\ 0 & \text{if } x = 2 \end{cases}, \quad \text{then} \quad (\sigma f)(x) = \begin{cases} 0 & \text{if } x = 0 \\ 0 & \text{if } x = 1 \\ 1 & \text{if } x = 2 \end{cases}.$$

One can also define the action of a permutation on algorithms. Given $\sigma \in \mathcal{X}!$, define the corresponding function σ_x which maps traces to traces by $\sigma_x(\langle (x_0, y_0), \ldots \rangle) = \langle (\sigma(x_0), y_0), \ldots \rangle$. Intuitively, σ_x operates on the x-components of a trace by applying σ to each of them while leaving the y-components unchanged. The *permutation $\sigma\mathcal{A}$ of $\mathcal{A}$ by σ* is the algorithm corresponding to search operator σg defined by $(\sigma g)(T) = \sigma^{-1}(g(\sigma_x(T)))$, where g is the search operator of $\mathcal{A}$.

Actions of permutations on algorithms and functions are intimately intertwined. The following theorem shows precisely how they correspond to one another, and was first proved in [**26**].

THEOREM 1. *(Duality) For any algorithm $\mathcal{A}$, permutation $\sigma \in \mathcal{X}!$, and function $f \in \mathcal{Y}^{\mathcal{X}}$,*

$$\sigma_x(\mathcal{A}(\sigma f)) = \sigma\mathcal{A}(f)$$

This theorem shows that the trace of a permuted algorithm ($\sigma\mathcal{A}$) is the same as the trace of the algorithm over the permuted function (σf) after the permutation has also been applied to the search space of the trace (σ_x). The strength of this result is perhaps best shown by projecting both sides of the equation to their y-components. This yields $\tilde{\mathcal{A}}(\sigma f) = \widetilde{\sigma\mathcal{A}}(f)$ where on the left hand side the permutation σ is acting on function f (it is regarded as an element of $\mathcal{Y}^{\mathcal{X}}!$), and on the right-hand side the permutation is acting on algorithm $\mathcal{A}$.

The map $\tilde{\mathcal{A}}$ is a bijection between functions and performance vectors. Theorem 1 implies that the performance vector of running $\sigma\mathcal{A}$ on function f is the same as the performance vector of algorithm $\mathcal{A}$ ran on the permuted function σf.

A set of functions $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$ is *closed with respect to a set $S \subset \mathcal{X}!$ of permutations* if and only if

$$\mathcal{F} = \{\sigma f : f \in \mathcal{F}, \sigma \in S\}$$

that is, every permutation σ in S, when applied to a function f in $\mathcal{F}$, yields another (possibly the same) function also in $\mathcal{F}$. For example, if $\mathcal{X} = \{0, 1, 2\}$, $\mathcal{Y} = \{0, 1\}$, $\sigma_i(x) = (x + i) \mod (3)$ and

$$f_j(x) = \begin{cases} 1 & \text{if } x = j \\ 0 & \text{otherwise} \end{cases},$$

then $\mathcal{F} = \{f_0, f_1, f_2\}$ is closed with respect to $S = \{\sigma_1, \sigma_2\}$, and any benchmark is closed with respect to σ_0, since σ_0 is the identity.

The set $\mathcal{F}$ is *permutation closed* if and only if it is closed with respect to $\mathcal{X}!$ (all permutations of $\mathcal{X}$).

3. No Free Lunch theorems

The performance of algorithm $\mathcal{A}$ over a given function f is, in principle, a function of the whole trace $\mathcal{A}(f)$. In practice, performance is independent on the order of points visited in the search space ($\mathcal{X}$). Rather, it depends only on the values (in $\mathcal{Y}$) the function takes at those points. This leads to the definition of a performance measure as given in [**24**]:

Definition 2. A *performance measure with respect to* $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$ is any function m_F defined over the collection of all search algorithms such that $m_{\mathcal{F}}(\mathcal{A})$ is a function of the multiset $\mathcal{A}(\mathcal{F}) = \{\!\{\tilde{\mathcal{A}}(f) : f \in \mathcal{F}\}\!\}$.
Note that for finite domains, one can think of a peformance measure as a function mapping performance vectors (of size $|\mathcal{X}|$) to real values. This specialized definition of performance measures leads naturally to the original NFL theorem.

Theorem 3. *(Original NFL) Every algorithm $\mathcal{A}$ has the same average performance over the set of all target functions $f \in \mathcal{Y}^{\mathcal{X}}$.*

This theorem, originally enunciated by Wolpert & Macready [**34**], is in fact a corollary of Theorem 1.

When the multi-set of performance vectors is the same for two algorithms $\mathcal{A}$ and $\mathcal{B}$ ($\mathcal{A}(\mathcal{F}) = \mathcal{B}(\mathcal{F})$), then, regardless of the performance measure m both algorithms necessarily have the same average performance. It should be noted that this is a sufficient, but not necessary, condition for equal average performance. However, the NFL literature preferentially uses this definition of equal performance for it allows ignoring the role of performance measures. In the remainder of this review we might simply say *equal performance* to refer to this notion of equal average performance.

While Theorem 3 shows that any two algorithms have equal performance over all target functions, it does not characterize all sets of functions for which this is true. This characterization is the purpose of the Sharpened NFL by Schumacher [**26, 27**].

Theorem 4. *(Sharpened NFL) Let $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$.*

$$\forall \mathcal{A}, \mathcal{B} \textit{ algorithms}. \mathcal{A}(\mathcal{F}) = \mathcal{B}(\mathcal{F})$$

if and only if $\mathcal{F}$ is permutation closed.

This characterization shows that equal performance of all possible algorithms over a benchmark is equivalent to the benchmark having all possible symmetries (i.e. being permutation closed). For example, all algorithms have equal performance over the benchmark $\mathcal{F} = \{f_0, f_1, f_2\}$ defined at the end of Section 2.

The Sharpened NFL has also been extended to arbitrary sets $\mathcal{X}$ and $\mathcal{Y}$ (e.g., countable, continuous, etc.) [**24**] with only the additional requirement that algorithms be *efficient*.

Definition 5. Search algorithm $\mathcal{A}$ is efficient if the domain of $\mathcal{A}(f)$ is equal to $I(\lfloor\mathcal{X}\rfloor)$ for all $f \in \mathcal{Y}^{\mathcal{X}}$, where $I(\alpha)$ denote the set of ordinal numbers less than α, and $\lfloor S \rfloor$ denotes the smallest ordinal α such that $\bar{\alpha} = |S|$.

Efficient algorithms are, intuitively, ones that, when applied to any function, cover any point in the search space $\mathcal{X}$ in at most $|\mathcal{X}|$ steps (which is trivially true for finite $\mathcal{X}$). One example of a non-efficient algorithm is given in [**24**]:

> Let $\mathcal{X}$ be the positive integers and consider $\mathcal{A}$ which explores $1, 3, 5, \ldots$ before moving on to $2, 4, 6, \ldots$ unless $f(1) = 1$ in which case $\mathcal{A}$ enumerates $1, 2, 3, \ldots$

Notice that there are functions for which this algorithm spends an infinite amount of steps and still has not visited 2.

A word of caution: When the cardinality of $\mathcal{X}$ is larger than countable, efficiency of an algorithm is not synonymous with the algorithm visiting every point of the search space $\mathcal{X}$ in finite time.

The requirement of permutation closure in Theorem 4 has been seen as a potential drawback [**15, 16**], since the number of benchmarks that are permutation closed is

$$2^{\binom{|\mathcal{X}|+|\mathcal{Y}|-1}{|\mathcal{X}|}}$$

which is small when compared to the number of possible benchmarks $2^{|\mathcal{Y}^{\mathcal{X}}|}$.

Igel & Toussaint [**16**] attempted to generalize Theorem 4 in such a way that the average performance is weighed over an special probability distribution over a benchmark. The so-called *Non-Uniform Sharpened NFL* theorem is, however, a corollary of the Sharpened NFL [**24**].

To alleviate the need for permutation closed benchmarks, Whitley and Rowe [**32**] shifted from all possible algorithms to only a subset of algorithms having equal performance.

Benchmark $\mathcal{F}$ is *focused* with respect to a set $\mathfrak{A}$ of algorithms, if and only if $\mathcal{A}(\mathcal{F})$ is independent of $\mathcal{A} \in \mathfrak{A}$. In other words, all algorithms in $\mathfrak{A}$ have the same set of performance vectors over the benchmark. Notice the parallel between the notion of $\mathcal{A}(\mathcal{F})$ being independent of $\mathcal{A}$ for all algorithms used in the Sharpened NFL and this notion of independence for a *particular* set of algorithms $\mathfrak{A}$ in focused benchmarks. In fact, we can rephrase Theorem 4 as:

> Benchmark $\mathcal{F}$ is focused with respect to the set of all algorithms if and only if it is permutation closed.

As we will see later, the concept of a focused benchmark is tied to a group of permutations in an analogous way to the concept of permutation closure of benchmarks.

For each pair of algorithms $\mathcal{A}$ and $\mathcal{B}$, $\tilde{\mathcal{A}}^{-1}\tilde{\mathcal{B}}$ induces a permutation of $\mathcal{Y}^{\mathcal{X}}$. This happens because for any algorithm $\mathcal{A}$, $\tilde{\mathcal{A}}$ is a bijection between the set of all target functions and the set of all performance vectors. These permutations induced by pairs of algorithms need not form a group, but one may consider the group they generate. This is precisely what the following definition does.

Definition 6. For any set $\mathfrak{A}$ of algorithms, let $\mathcal{G}_{\mathfrak{A}}$ be the subgroup of $\mathcal{Y}^{\mathcal{X}}!$ generated by

$$\{\tilde{\mathcal{A}}^{-1}\tilde{\mathcal{B}} \,:\, \mathcal{A}, \mathcal{B} \in \mathfrak{A}\}$$

This leads to the Focused NFL theorem.

Theorem 7. *(Focused NFL) Benchmark $\mathcal{F}$ is focused with respect to a set $\mathfrak{A}$ of algorithms if and only if it is closed with respect to $\mathcal{G}_{\mathfrak{A}}$.*

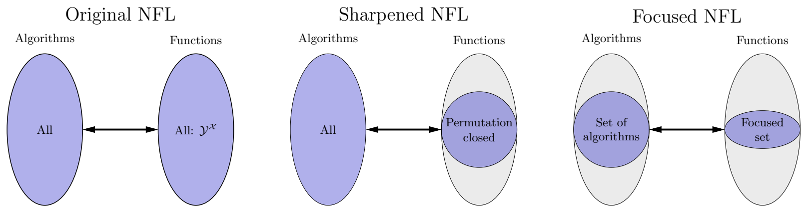

FIGURE 3.1. Schematic representation of the progressive specificity of NFL theorems. The Original NFL was concerned with all algorithms having equal performance over all functions. The Sharpened NFL strengthens the requirement from all functions to permutation closed benchmarks. Finally, Focused NFL deals with subsets of algorithms having equal performance over focused benchmarks.

The original Focused NFL theorem was expressed as an implication in [**32**]. Here we opt for its most general expression as given in [**10**].

The implications of Focused NFL merit some discussion. It characterizes all possible benchmarks over which arbitrary collections of algorithms must necessarily have equal performance. In principle, one could imagine starting with a benchmark $\mathcal{F}$ and an algorithm $\mathcal{A}$ (as is customary in performance evaluation) and ask if there exists any other algorithm $\mathcal{B}$ with the same performance as $\mathcal{A}$ (i.e. if $\mathcal{A}(\mathcal{F}) = \mathcal{B}(\mathcal{F})$). As it turns out, such a $\mathcal{B}$ exists if and only if there is some non-trivial permutation under which $\mathcal{F}$ is closed. Focused NFL tells us that, in general, we cannot do better.

Figure 3.1 depicts the transition from considering all algorithms and all target functions in the Original NFL, to permutation closed benchmarks in Sharpened NFL, and finally to focused sets of functions for particular sets of algorithms in Focused NFL.

In real world applications, algorithms are seldom run until they have exhausted the search space. For instance, the search space might be vast, and the practical reason for using a black-box search algorithms is precisely to avoid enumerating all possible solutions.

Consider now algorithms restricted to $\gamma \in \mathbb{N}$ steps. The projection π_γ of a sequence $x = \langle x_0, x_1, \ldots\rangle$ to their first γ elements is simply given by

$$\pi_\gamma(x) = \langle x_0, \ldots, x_{\gamma-1}\rangle.$$

Following [**10**], let us abbreviate $\pi_\gamma \circ \tilde{\mathcal{A}}$ by $\tilde{\mathcal{A}}_\gamma$. In this way, applying algorithm $\mathcal{A}$ to function f for γ steps yields the performance vector $\tilde{\mathcal{A}}_\gamma(f)$. Notice that, even though $\tilde{\mathcal{A}}$ is a bijection between functions and performance vectors, $\tilde{\mathcal{A}}_\gamma$ is no-longer 1–1. Because of this, we will consider multisets of performance vectors instead of sets. A multiset extends concept of a set by allowing elements to appear more than once and is denoted by double brackets $\{\!\{\ \}\!\}$. We extend π_γ to multisets by

$$\pi_\gamma S \;=\; \{\!\{\pi_\gamma(s) : s \in S\}\!\}.$$

A benchmark $\mathcal{F}$ is *focused* with respect to a set $\mathfrak{A}$ of algorithms and integer γ if and only if the multiset

$$\tilde{\mathcal{A}}_\gamma(\mathcal{F}) = \{\!\{\tilde{\mathcal{A}}_\gamma(f) \,:\, f \in \mathcal{F}\}\!\}$$

is independent of $\mathcal{A} \in \mathfrak{A}$. Similarly to the case of Focused NFL above, we define the set of permutations $G_A \subset \mathcal{G}_{\mathfrak{A}}$ defined for $\mathcal{A} \in \mathfrak{A}$ by

$$G_{\mathcal{A}} = \{\tilde{\mathcal{A}}^{-1}\tilde{\mathcal{B}} : \mathcal{B} \in \mathfrak{A}\}$$

The following theorem, due to Whitley, *et al.* [**32**] (and later refined in [**10**]) generalizes the Focused NFL to a number of steps.

THEOREM 8. *(γ-step Focused NFL) The benchmark $\mathcal{F}$ is focused with respect to a collection $\mathfrak{A}$ of algorithms and integer γ if and only if for some $\mathcal{A} \in \mathfrak{A}$ and all $\alpha \in G_A$,*

$$\tilde{\mathcal{A}}_\gamma(\mathcal{F}) \;=\; \tilde{\mathcal{A}}_\gamma(\alpha\mathcal{F})$$

Moreover, if the above holds for some $\mathcal{A} \in \mathfrak{A}$ then it holds for all $\mathcal{A} \in \mathfrak{A}$.

In contrast to Focused NFL, in the γ-step Focused NFL (γFNFL) theorem above, $G_{\mathcal{A}}$ is not necessarily the group $\mathcal{G}_{\mathfrak{A}}$. Indeed, in general it is not even a group. Closure over $G_{\mathcal{A}}$ is enough to guarantee equal performance when restricted to γ steps, whereas closure with respect to the full group $\mathcal{G}_{\mathcal{A}}$ was necessary in Theorem 7.

Focused benchmarks, characterized in Theorem 8, exhibit some interesting cyclic properties. Given algorithm $\mathcal{A}$ and integer $0 \leq \gamma \leq |\mathcal{X}|$, define the equivalence relation $\equiv$ on $\mathcal{Y}^{\mathcal{X}}$ by

$$f \equiv f' \iff \tilde{\mathcal{A}}_\gamma(f) = \tilde{\mathcal{A}}_\gamma(f')$$

that is, two functions are considered equivalent at γ steps under algorithm $\mathcal{A}$, if they produce the exact same performance vector.

DEFINITION 9. A benchmark $\mathcal{F}$ is *cyclic* with respect to algorithm $\mathcal{A}$, integer $0 \leq \gamma \leq |\mathcal{X}|$, and permutation $\alpha \in \mathcal{Y}^{\mathcal{X}}!$ if and only if for some positive integer ℓ,

$$\mathcal{F} \;=\; \{f_0, ..., f_{\ell-1}\} \;\text{ where } \alpha\left(f_{i \mod \ell}\right) \equiv f_{i+1 \mod \ell}$$

This leads to the following decomposition theorem by Duéñez-Guzmán & Vose [**10**]: every focused benchmark can be decomposed into a disjoint union of cyclic benchmarks.

THEOREM 10. *Benchmark $\mathcal{F}$ is focused with respect to a set $\mathfrak{A}$ of algorithms and $0 \leq \gamma \leq |\mathcal{X}|$ if and only if for some $\mathcal{A} \in \mathfrak{A}$ and all $\alpha \in G_{\mathcal{A}}$,*

$$\mathcal{F} \;=\; \bigcup_{j>0} \mathcal{F}_j$$

where the union above is disjoint, and each $\mathcal{F}_j$ (which may depend on $\mathfrak{A}$, $\mathcal{A}$, γ, and α) is cyclic with respect to $\mathcal{A}$, γ, and α.

While the decomposition is not (necessarily) unique, it is a complete characterization of focused benchmarks, like in γFNFL.

Part 2. Stochastic Algorithms

4. Theoretical background

In the previous section, we considered exclusively deterministic algorithms. Here, we show the extension of NFL theory to truly stochastic algorithms (sometimes called *randomized algorithms* [**10**]). A *stochastic algorithm* is simply a probability distribution over the space of deterministic algorithms. In any particular run on a target function f, an stochastic algorithm must necessarily produce a trace that is compatible with f. Therefore, a particular run of an stochastic algorithm on function f will produce exactly the same trace as some deterministic algorithm$\mathcal{A}$ (i.e. it behaves indistinguishably from $\mathcal{A}$).

Formally, a *stochastic algorithm* is described by a probability vector μ indexed over the set $\mathfrak{D}$ of all deterministic algorithms. We will use the probability vector μ itself to refer to the stochastic algorithm. We index the components of μ with deterministic algorithms $\mathcal{A}$. Thus, the $\mathcal{A}$-th component $\mu_{\mathcal{A}}$ is the probability the stochastic algorithm behaves like $\mathcal{A}$. In procedural terms, stochastic algorithm μ runs on f by choosing $\mathcal{A}$ with probability $\mu_{\mathcal{A}}$ and then applying algorithm $\mathcal{A}$ to f.

Stochastic algorithms are identified with elements of the simplex $\Lambda_{\mathfrak{D}}$ defined by

$$\Lambda_{\mathfrak{D}} \;=\; \{x \in \mathbb{R}^{|\mathfrak{D}|} \,:\, i \in \mathfrak{D},\ x_i \geq 0,\ \sum_i x_i = 1\}\,.$$

Note that the collection of stochastic algorithms $\Lambda_{\mathfrak{D}}$ contains the set of (deterministic) algorithms at the corners of the simplex. The *support* $\mathcal{S}_\mu$ of random algorithm μ is the set of algorithms $\mathcal{A}$ for which $\mu_{\mathcal{A}} > 0$. A deterministic algorithm μ has a unique nonzero component $\mu_{\mathcal{A}} = 1$ (we will call such support, *trivial*). In this case, we use $\mathcal{A}$ indistinguishably from μ.

Like before, we can define the action of a permutation on stochastic algorithms. Given $\sigma \in \mathcal{X}!$ and $\mu \in \Lambda_{\mathfrak{D}}$, the stochastic algorithm $\sigma\mu$ is defined by

$$(\sigma\mu)_{\mathcal{A}} \;=\; \mu_{\sigma^{-1}\mathcal{A}}$$

Intuitively, the stochastic algorithm $\sigma\mu$ applied to f, chooses algorithm $\mathcal{A}$ with probability $\mu_{\mathcal{A}}$ and then runs $\sigma\mathcal{A}$ on f. Random algorithms μ and μ' are *equivalent*, denoted by $\mu \equiv \mu'$, if and only if for all functions $f \in \mathcal{Y}^{\mathcal{X}}$ and every trace T,

$$\mathrm{Prob}\,(\mu(f) = T) = \mathrm{Prob}\,(\mu'(f) = T)$$

where the total trace $\mu(f)$ is the result of applying μ to f is equal to $\mathcal{A}(f)$ with probability $\mu_{\mathcal{A}}$. In other words, two stochastic algorithms are equivalent if the probability of generating a particular trace from a particular function is the same for both.

A performance measure m (simply called a measure) is extended to stochastic algorithms in the natural way: the performance of μ on f as measured by m is

$$m(\mu, f) \;=\; \sum_{\mathcal{A}} m(\tilde{\mathcal{A}}(f))\,\mu_{\mathcal{A}}$$

Measure m has, therefore, two interpretations. In one, it assigns real values to performance vectors, whereas in the other, it measures the expected performance of μ on f , aggregated and weighted over all deterministic algorithms. With the performance of the stochastic algorithm μ over a function defined, the expected performance of stochastic algorithm μ over benchmark $\mathcal{F}$ is

$$\mathcal{E}(\mu, \mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} m(\mu, f)$$

and is referred to as *expected average performance*. Notice that $\mathcal{E}(\cdot, \cdot)$ depends on the measure m. An important observation is that the expected average performance is a linear function of algorithms.

Proposition 11. *(Linearity) $\mathcal{E}(\cdot, \cdot)$ is linear in its first argument.*

The Duality Theorem (1) can be extended to stochastic algorithms [**10**]. This theorem shows the link between permutations of stochastic algorithms and permutations of functions in the same way the deterministic Duality Theorem did.

THEOREM 12. *(Expected Duality) For every measure m, stochastic algorithm μ, benchmark $\mathcal{F}$, and permutation $\sigma \in \mathcal{X}!$,*

$$\mathcal{E}(\sigma\mu, \mathcal{F}) = \mathcal{E}(\mu, \sigma\mathcal{F})$$

Intuitively, this theorem states that applying a permuted stochastic algorithm to a benchmark is the same, in the sense of expected performance, as applying the stochastic algorithm to a permuted benchmark.

5. No free lunch theorems

The restriction of traditional NFL theorems to deterministic algorithms might appear a shortcoming. However, even if one insists on considering truly stochastic algorithms, Proposition 11 has some immediate implications. A linear function over a convex set such as $\Lambda_{\mathfrak{D}}$ must necessarily attain its maximum and minimum at the boundary (in particular, since $\Lambda_{\mathfrak{D}}$ is a simplex, it attains them at a corner). With this, it is possible to prove that regardless of the measure m and benchmark $\mathcal{F}$ used, a non-trivial (i.e. non deterministic) stochastic algorithm will have the same performance as infinitely many other stochastic algorithms. Moreover, if m is not constant with respect to $\mathcal{F}$ and the support of μ, then there necessarily exist stochastic algorithms with larger and smaller average expected performance than μ. This is expressed formally in the following proposition [**10**].

PROPOSITION 13. *For every measure m, stochastic algorithm μ, and benchmark $\mathcal{F}$, if m is non-constant with respect to $\mathcal{F}$ and $\mathcal{S}_\mu$, then there exist infinitely many μ' and μ'' with support $\mathcal{S}_\mu$ such that*

$$\mathcal{E}(\mu', \mathcal{F}) < \mathcal{E}(\mu, \mathcal{F}) < \mathcal{E}(\mu'', \mathcal{F})$$

If m is constant with respect to $\mathcal{F}$ and $\mathcal{S}_\mu$, then $\mathcal{S}_\eta \subset \mathcal{S}_\mu \Longrightarrow \mathcal{E}(\eta, \mathcal{F}) = \mathcal{E}(\mu, \mathcal{F})$.

Given benchmark $\mathcal{F}$, the group $H(\mathcal{F})$ of *benchmark symmetries* is

$$H(\mathcal{F}) = \{\sigma \in \mathcal{X}! \, : \, \sigma\mathcal{F} = \mathcal{F}\}\,.$$

Notice that the set of benchmark symmetries is intrinsic to the benchmark $\mathcal{F}$. As a consequence of Expected Duality, for any permutation σ in $H(\mathcal{F})$ and any stochastic algorithm μ, the expected performance of μ is the same as that of $\sigma\mu$. The following theorem by Rowe & Vose [**25**] expresses this formally.

THEOREM 14. *Given a benchmark $\mathcal{F}$, the set of algorithms $H(\mathcal{F})\mu$ has the same expected average performance over benchmark $\mathcal{F}$. Moreover if $\mathcal{F}$ is permutation closed, $\mathcal{E}(\mu, \mathcal{F}) = \mathcal{E}(\mu', \mathcal{F})$ for any stochastic algorithm μ'.*

In the theorem above, $H(\mathcal{F})\mu$ denotes the set of all the permutations of μ given by elements of $H(\mathcal{F})$ (the group of benchmark symmetries.) If benchmark $\mathcal{F}$ is permutation closed, them $H(\mathcal{F})$ is the group of all permutations $\mathcal{X}!$, and thus all stochastic algorithms have the same performance.

Benchmark symmetries were generalized in [**25**] to depend upon the measure m as well as the benchmark $\mathcal{F}$, and a result similar to the Focused NFL emerges which, likewise, depends upon m. Given benchmark $\mathcal{F}$, the collection $H_m(\mathcal{F})$ of *benchmark invariants* is

$$H_m(\mathcal{F}) \;=\; \left\{\sigma \in \mathcal{X}! \,:\, \forall \mathcal{A} \,.\, \mathcal{E}(\sigma\mathcal{A},\mathcal{F}) = \mathcal{E}(\mathcal{A},\mathcal{F})\right\}.$$

The set of benchmark invariants is intrinsic to the benchmark and the measure. The following theorem by Duéñez-Guzmán & Vose [**10**] is analogous to the previous theorem, but for benchmark invariants instead of for benchmark symmetries.

THEOREM 15. *The set of benchmark invariants $H_m(\mathcal{F})$ is a group (under composition) for every measure m and benchmark $\mathcal{F}$. For every stochastic algorithm μ, the stochastic algorithms in $H_m(\mathcal{F})\mu$ all have the same expected average performance over $\mathcal{F}$.*

This theorem can be specialized to algorithms run for γ steps. Since benchmark invariants depend on the measure m, we can consider a measure that is restricted to only the first γ components of a performance vector. Formally, let m_γ be $m \circ \pi_\gamma$ for some $0 \le \gamma \le |\mathcal{X}|$ and some measure m. For every stochastic algorithm μ, the γ-step expected average performance of the collection of algorithms $H_{m_\gamma}(\mathcal{F})\mu$ is identical over benchmark $\mathcal{F}$.

The sets of benchmark symmetries and benchmark invariants are important because they capture the inherent symmetries of a benchmark, in the sense that algorithms transformed by the permutations in them have the same average performance. The following theorem ties together the concept of benchmark invariants with the familiar concept of permutation closure of deterministic NFL theorems [**10**].

THEOREM 16. *For every measure m and benchmark $\mathcal{F}$,*

$$\sigma \in H_m(\mathcal{F}) \implies H_m(\sigma\mathcal{F}) = H_m(\mathcal{F})$$

By the Focused NFL theorem, we know that given a collection of algorithms $\mathfrak{A}$, $\mathcal{G}_{\mathfrak{A}}$ is a group such that if benchmark $\mathcal{F}$ is focused with respect to $\mathfrak{A}$, then $\mathcal{F}$ is closed with respect to the permutations in $\mathcal{G}_{\mathfrak{A}}$. Moreover, the γ-step Focused NFL theorem simplifies to the normal Focused NFL when $\gamma = |\mathcal{X}|$. In contrast, it is not necessarily true that $\mathcal{F}$ is closed with respect to the permutations in $H_m(\mathcal{F})$, even when $\gamma = |\mathcal{X}|$.

Theorem 16 tells us that for any $\sigma \in H_m(\mathcal{F})$, benchmark $\sigma\mathcal{F}$ has the same benchmark invariant as $\mathcal{F}$. This means that while the benchmark itself is not preserved under the action of $H_m(\mathcal{F})$, the benchmark invariant itself is preserved under the action of $H_m(\mathcal{F})$.

We conclude this section with two results that deal with matching the performance of a particular algorithm on a benchmark. It turns out that, for every measure, the performance of any stochastic algorithm on any benchmark can be matched in a non-trivial way.

PROPOSITION 17. *For every measure m, stochastic algorithm μ, and benchmark $\mathcal{F}$, there exist $\mathcal{F}'$ and μ' such that $\mathcal{E}(\mu,\mathcal{F}) = \mathcal{E}(\mu',\mathcal{F}')$ where $\mathcal{F}' \neq \mathcal{F}$ or μ' may be chosen arbitrarily.*

One might hope for a variant of Proposition 17 asserting that for every m, μ, and $\mathcal{F}$, there exists $\mu' \neq \mu$ for which $\mathcal{E}(\mu,\mathcal{F}) = \mathcal{E}(\mu',\mathcal{F})$ when μ is nontrivial. However, this is not the case in general. The following example was given in [**10**]:

If $\mathcal{X} = \mathcal{Y} = \{0,1\}$, then there are only two possible deterministic algorithms $\mathfrak{D} = \{\mathcal{A}, \mathcal{B}\}$ where

$$\begin{aligned}\mathcal{A}(f) &= \langle (0, f(0)), (1, f(1)) \rangle \\ \mathcal{B}(f) &= \langle (1, f(1)), (0, f(0)) \rangle\end{aligned}$$

If benchmark $\mathcal{F}$ contains only the identity function, and the performance measure is

$$m(\langle y_0, y_1 \rangle) = 2y_0 + y_1 ,$$

it then follows that $H_m(\mathcal{F})$ is the trivial group (it contains only the identity). The example above demonstrates that the following cannot be improved.

Proposition 18. *Let $|\mathfrak{D}| > 2$. For every measure m, benchmark $\mathcal{F}$, and nontrivial stochastic algorithm μ, there exist infinitely many $\mu' \neq \mu$ such that $\mathcal{E}(\mu, \mathcal{F}) = \mathcal{E}(\mu', \mathcal{F})$.*

These two results highlights how the performance of stochastic algorithms allows for even more symmetries than deterministic ones. There is no restriction of permutation closure with respect to any set or group of permutations.

6. Discussion

Several progressively more general No Free Lunch theorems have been proposed [**35, 27, 16, 32, 24, 10**]. All of these theorems improve our understanding of the applicability and limitations of black-box optimization algorithms. Understanding these limitations is relevant in two different ways. First, there is a need for a more meaningful way of comparing algorithm performance than simply analyzing the performance over a benchmark set of functions. Second, they provide a way to clearly define when algorithms are truly better than others.

The traditional approach of NFL theorems, discussed in Part 1, is the characterization of the sets of functions for which a particular set of algorithms have equal performance. For instance, the original NFL considered all target functions. Schumacher, *et al.* [**27**] later found that the set of all functions is not required to guarantee equal performance, showing that permutation closure is the relevant property. In a similar way, the result proposed by Whitley & Rowe [**32**] was sharpened and extended in [**10**] to characterize all focused sets of functions.

While the original formulation of the NFL theorem was concerned with all algorithms having the same average performance over a set of functions, later the Focused NFL [**32**] restricted its study on fixing a set of algorithms and finding sets of functions over which the algorithms in question had equal performance. In [**10**] a different approach to NFL results was also explored. Instead of trying to find the set of functions for which a set of algorithms perform equally on average, it is possible to ask under which conditions, given a set of functions and an algorithm, one can find other algorithms or sets of functions that have equal performance. Section 5 gives the main results in this direction. This result addresses, at least partially, one of the long standing criticisms to NFL theorems, namely that they only apply to benchmarks with restrictive properties.

The extensions of NFL to stochastic algorithms in Part 2 show that regardless of the benchmark function and random algorithm under consideration, there is always at least another random algorithm with the same performance as the original algorithm over that benchmark. Moreover, under arguably general conditions, given any algorithm, there are (infinitely many) other algorithms that perform better and others that perform worse. In Section 5, we discussed that the performance of an algorithm on a particular benchmark can be matched by some other algorithm on some other benchmark. Also, the maximal performance of a stochastic algorithm on a benchmark is attainable, but typically at a vertex of the simplex, thus reducing to a deterministic algorithm. Furthermore, this maximization of performance on the benchmark offers no clear indication of performance on other functions. Indeed, by the Sharpened NFL theorem, there exists another deterministic algorithm that performs at least as well in some other benchmark. The concept of benchmark invariants given at the end of Section 4, is analogous for stochastic algorithms to the concept of permutation closure for deterministic algorithms.

Future directions. There is a large amount of research done in developing and empirically improving heuristic optimization algorithms. In contrast, little effort has been devoted to understanding their theoretical limitations and overall behavior. Virtually nobody argues against the practical importance and success of heuristics. However, it is crucial to balance the empirical progress with theoretical understanding. Despite claims that everything on NFL has already been said [**30**], new and more powerful NFL theorems are still being found.

The applicability of NFL results to classes of functions considered of "real-world importance" is frequently questioned. While some progress has been recently achieved in this regard (see Section 5), there is still ample room for research in this area. Droste, *et al.* [**9**] describe what they consider realistic scenarios for black-box search. Their observations present an opportunity for further NFL research, and they can be summarized as pertaining to three broad categories: modality, separability and compressibility.

The *modality* of a function is how many local maxima (or minima) the function has. A unimodal function has a unique maximum (minimum), while a multimodal function has many. Modality is, therefore, generally used as a proxy for optimization difficulty, an observation that is empirically supported. Some efforts to understand the relationship between black-box search and modality have been published [**33, 6**], but more research is needed.

Separability means that the optimum of a target function can be obtained by independently optimizing each of its variables. Separability is considered to indicate ease of optimization. The extent of the relevance of the class of separable functions to NFL theorems is currently unknown.

Compressibility refers to the possibility of describing a target function in a compact form instead of having to enumerate all its (x, y) pairs. Intuitively, highly structured functions can be represented succinctly, like $f(x) = x^2$, while completely unstructured ones require the enumeration of all (x, y) pairs. Formally, a function is compressible if its Kolmogorov length [**19**] is shorter than its plain representation length. Interestingly, Schumacher, *et al.* [**27**] pointed out that permutation closed benchmarks can be highly compressible, and their argument extends to focused benchmarks. Therefore, not all compressible functions violate the assumptions of NFL, but it is not known to what extent they are relevant for NFL theory.

There are, however, many other classes of functions that researchers can consider realistic. Perhaps the most difficult obstacle in finding NFL results for problems of "real-world importance" is not mathematical at all: researchers do not agree on what precisely "real-world importance" means. Without a precise definition, there is little hope for the formulation of a mathematical result.

From a purely theoretical perspective, there are still several open questions in NFL. A characterization of benchmark symmetries $H(\mathcal{F})$ and benchmark invariants $H_m(\mathcal{F})$ would be desirable. As of yet, little has been said about stochastic algorithms limited to a number of steps (but see Theorem 15). Compared to properties of benchmarks, little is known about the properties of search operators (the workhorses of algorithms). The particular way an algorithm explores the search space is precisely what differentiates it from others. More research is needed on the symmetries underlying the space of algorithms (beyond the Duality theorems), and on identifying the properties of search operators that are relevant for performance; especially for performance over a "real-world" class of target functions.

At its inception, NFL theory caused some stir in the optimization community. Some people saw it as highligting a fundamental flaw in black-box search algorithms, while other saw it as merely a mathematical curiosity. The ensuing debate lasted several years, and many researchers settled on a view of NFL as a modest cautionary tale: there are some classes of problems for which blind search is useless, but if you steer away from those, you are fine. However, there is, perhaps, a broader lesson here. Developing a branch of knowledge exclusively resting on empirical assessments is a risky enterprise. Whether NFL applies to a given set of functions and algorithms or not, critical thinking about algorithms and their performance assessment may enable us to avoid pitfalls. Showcasing an algorithm's properties on a benchmark might be appropriate sometimes, but if we are to ever truly understand what makes an algorithm *better*, we need to understand what *better* means mathematically.

Acknowledgments

The authors would like to thank Suzanne Sadedin for her very helpful comments on an early version of this manuscript.

References

1. A. Auger and O. Teytaud, *Continuous lunches are free!*, Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, GECCO-2007 (2007), 916–922.
2. ———, *Continuous lunches are free plus the design ofÂ optimal optimization algorithms*, Algorithmica **57** (2010), 121–146.
3. T. Back and H. P. Schwefel, *Evolutionary computation: an overview*, Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, IEEE, May 1996, pp. 20–29.
4. Hans-Georg Beyer and Hans-Paul Schwefel, *Evolution strategies – a comprehensive introduction*, Natural Computing **1** (2002), no. 1, 3–52.
5. D.W. Corne and J.D. Knowles, *No free lunch and free leftovers theorems for multiobjective optimisation problems*, Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 2632/2003, Springer Berlin / Heidelberg, 2003, p. 66.
6. Martin Dietzfelbinger, Jonathan E. Rowe, Ingo Wegener, and Philipp Woelfel, *Precision, local search and unimodal functions*, Algorithmica **59** (2011), no. 3, 301–322 (English).
7. P. Domingos, *How to get a free lunch: A simple cost model for machine learning applications*, Proc. AAAI98/ICML98, Workshop on the Methodology of Applying Machine Learning (1998), 1–7.
8. M. Dorigo, V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **26** (1996), no. 1, 29–41.
9. S. Droste, T. Jansen, and I. Wegener, *Perhaps not a free lunch but at least a free appetizer*, Proceedings of the First Genetic and Evolutionary Computation Conference, GECCO-1999 (1999), 833–839.

10. EA Duéñez Guzmán and MD Vose, *No free lunch and benchmarks*, Evolutionary Computation **In press** (2012).
11. Lawrence J. Fogel, *Intelligence through simulated evolution: forty years of evolutionary programming*, John Wiley & Sons, Inc., New York, NY, USA, 1999.
12. D. Goldberg, *Genetic algorithm in search, optimization and machine learning*, Addison-Wesley Pub. Co. ISBN: 0201157675, 1989.
13. R.L. Haupt and S.E. Haupt, *Practical genetic algorithms*, second ed., John Wiley and Sons, Inc., 2004.
14. J.H. Holland, *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI, 1975.
15. C. Igel and M. Toussaint, *On classes of functions for which no free lunch results hold*, 2001.
16. ———, *A no-free-lunch theorem for non-uniform distributions of target functions*, Journal of Mathematical Modeling and Algorithms **3(4)** (2004), 313–322.
17. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), no. 4598, 671–680.
18. Gary J. Koehler, *Conditions that obviate the no-free-lunch theorems for optimization*, INFORMS Journal on Computing **19** (2007), no. 2, 273–279.
19. Ming Li and Paul Vitányi, *An introduction to kolmogorov complexity and its applications*, Springer New York, New York, NY, 2008.
20. Malik Magdon-Ismail, *No free lunch for noise prediction*, Neural Computation **12** (2000), no. 3, 547–564.
21. M. Mitchell, *An introduction to genetic algorithms*, third ed., Bradford, 1998.
22. N. Radcliffe and P. Surry, *Fundamental limitations on search algorithms: Evolutionary computing in perspective*, Computer Science Today (Jan van Leeuwen, ed.), Lecture Notes in Computer Science, vol. 1000, Springer Berlin / Heidelberg, 1995, pp. 275–291.
23. Ingo Rechenberg, *Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution*, Werkstatt Bionik und Evolutionstechnik, vol. 1, Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart, 1994.
24. J. Rowe, M.D. Vose, and A.H. Wright, *Reinterpreting no free lunch*, Evolutionary Computation **17** (2009), 117–129.
25. J.E. Rowe and M.D. Vose, *Unbiased black box search algorithms*, Proceedings of the 13th annual conference on Genetic and evolutionary computation (Natalio Krasnogor, ed.), ACM New York, NY, USA, 2011, pp. 2035–2042.
26. C. Schumacher, *Black box search - framework and methods.*, PhD thesis, The University of Tennesee, Knoxville (2000).
27. C. Schumacher, M.D. Vose, and D. Whitley, *The no free lunch and problem description length*, Proceedings of the 3rd annual conference on Genetic and evolutionary computation, GECCO-2001 (2001), 565–570.
28. Rainer Storn and Kenneth Price, *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization **11** (1997), no. 4, 341–359.
29. M. Wardetzky, S. Mathur, F. Kalberer, and E. Grinspun, *Discrete laplace operators: No free lunch*, Eurographics Symposium on Geometry Processing (A. Belyaev and M. Garland, eds.), 2007.
30. Ingo Wegener, *Computational complexity and ec.*, Tutorial at GECCO **04** (2004).
31. D. Whitley, *Functions as permutations: Regarding no free lunch, walsh analysis and summary statistics*, Parallel Problem Solving from Nature PPSN VI (Marc Schoenauer, Kalyanmoy Deb, Günther Rudolph, Xin Yao, Evelyne Lutton, JuanJulian Merelo, and Hans-Paul Schwefel, eds.), Lecture Notes in Computer Science, vol. 1917, Springer Berlin Heidelberg, 2000, pp. 169–178.
32. D. Whitley and J. Rowe, *Focused no free lunch theorems*, Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO-2008 (2008), 811–818.
33. L. D. Whitley and J. E. Rowe, *Subthreshold-seeking local search*, Theoretical Computer Science **361** (2006), 2–17.
34. D. Wolpert and M. Macready, *No free lunch theorems for search*, Technical Report **SFI-TR-95-02-010** (1995).
35. ———, *No free lunch theorems for optimization*, IEEE Transactions on Evolutionary Computation **1(1)** (1997), 67–82.

36. David H. Wolpert, *The supervised learning no-free-lunch theorems*, In Proc. 6th Online World Conference on Soft Computing in Industrial Applications, 2001, pp. 25–42.
37. J.R. Woodward and J.R. Neil, *No free lunch, program induction and combinatorial problems*, Genetic Programming, Lecture Notes in Computer Science, vol. 2610/2003, Springer Berlin / Heidelberg, 2003, pp. 287–313.
38. Huan Xu, C. Caramanis, and S. Mannor, *Sparse algorithms are not stable: A no-free-lunch theorem*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **34** (2012), no. 1, 187–193.

Google Inc., 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
E-mail address: duenez@google.com

Google Inc., 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
E-mail address: darkmars@google.com