# Beyond No Free Lunch: Realistic Algorithms for Arbitrary Problem Classes

James A. R. Marshall, Thomas G. Hinton

*Abstract*—In this paper we present a simple and general new No Free Lunch-like result that applies to revisiting algorithms searching arbitrary problem sets. We begin by unifying the assumptions of closure under permutation and non-revisiting algorithms. We then propose a new approach to reasoning about search algorithm performance, treating search algorithms as stochastic processes and thereby admitting revisiting; for this approach we need only make a simple assumption that search algorithms are applied for optimisation (*i.e.* maximisation or minimisation), rather than considering arbitrary performance measures. As a consequence of a proof that non-revisiting enumeration has the best possible expected performance for arbitrary distributions over arbitrary problem sets, this allows us to show that better than enumeration performance by some algorithm on some problem set predicts nothing about performance on a second problem set when nothing is known about the relation between the two. Finally we note that enumeration typically has excellent time and space complexities. Our approach is both simple and powerful; our new No Free Lunch-like result is stronger than the originals both in its predictions and, apart from not applying to arbitrary performance measures but only to those reflecting the typical goal of search, weaker than the originals in its assumptions. It is also directly relevant to research into real-world search algorithms.

## I. INTRODUCTION

The various No Free Lunch theorems are important theoretical results, indicating that no 'black-box' problem solver can be expected to achieve better than random performance without any information about the problem it is to be applied to. Such results have been proved for off training-set generalisation in supervised learning [11], and for search and optimisation [6], [12]. It is known that a crucial assumption of the No Free Lunch theorems is that a (block) uniform distribution holds over the set of problems, or objective functions, under consideration and that this set be closed-under-permuation (c.u.p.) [4], [5]. The c.u.p. requirement means that any permutation of any objective function in the problem set under consideration, through changing the mapping between elements of the search space and objective values, results in another objective function that is also a member of that problem set. However, it has been shown that realistic problem sets are highly unlikely to satisfy the c.u.p. assumption [5]. As a result, a large field of increasingly sophisticated work has developed seeking to prove that 'Almost No Free Lunch' results that are nearly as strong as the original No Free Lunch results, typically based on

James Marshall is with the Department of Computer Science, University of Bristol, Bristol, BS8 1UB, United Kingdom (email: James.Marshall@bristol.ac.uk ).

Thomas Hinton is with the Department of Computer Science, University of Bristol, Bristol, BS8 1UB, United Kingdom (email:hinton@cs.bris.ac.uk).

information theoretic considerations, still apply in realistic problem scenarios (*e.g.* [1], [2], [10]).

In this paper we take a simplifying step back in an attempt to cut the increasingly complicated 'Gordian knot' that (Almost) No Free Lunch research has come to represent. Our approach is simple and intuitive, yet allows general results easily to be arrived at. We begin by examining the implications of the search version of the No Free Lunch theorem for real-world search algorithms, showing that revisiting algorithms also break the permutation closure condition required for the No Free Lunch theorem to hold. Thus allowing realistic, revisiting algorithms means there can be some best algorithm; the pertinent question then is whether we can identify this best algorithm, and it turns out that we can indeed. To answer this question we go on to present a novel analysis of search algorithms as stochastic processes, enabling us to present a No Free Lunch-like result that holds for arbitrary sets of problems, and for realistic algorithms that do not avoid revisiting points in the search space. Specifically we show that random enumeration has the best expected performance of all search algorithms applied to optimisation problems, for any distribution over any problem set. The implication of this is that empirical demonstration of superior search performance (relative to enumeration) for some algorithm on some problem set still predicts inferior performance on some second problem set if we know nothing about its relationship with the first. We thus 'cut the Gordian knot' by simplifying the assumptions underlying the No Free Lunch theorem, and show why violations of its assumptions are unimportant for real-world search algorithms and problems.

### A. The Sharpened No Free Lunch Theorem

We begin by informally summarising the Sharpened No Free Lunch theorem. No Free Lunch arguments typically consider a search space $\mathcal{X}$, a set of possible objective values $\mathcal{Y}$, and objective functions of the form $f : \mathcal{X} \mapsto \mathcal{Y}$. The set of all possible objective functions is then denoted $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$. The original No Free Lunch theorems prove that, across all possible objective functions (or problems) $\mathcal{F}$, all non-revisiting algorithms have equivalent performance under an arbitrary performance measure [12], [6]. Algorithms are defined here as pseudo-random processes choosing previously unvisited points to visit based on the quality of prior visited points. Subsequently it was shown that the No Free Lunch theorem only holds if the set of objective functions under consideration is closed under permutation (c.u.p.).

The resulting theorem [8], [4] can be paraphrased as:

*Theorem 1 (Sharpened No Free Lunch Theorem):* All

non-revisiting algorithms have equivalent performance over a set of objective functions $F$, under some arbitrary performance measure, iff $F$ is closed under permutation.

Theorem 1 has subsequently been extended to also hold for 'block-uniform' distributions over subsets that are c.u.p.[5] [7]. This condition for the No Free Lunch theorem to hold is satisfied by the original requirement in [12] of a uniform distribution over $\mathcal{F}$, because the uniform distribution is a special case of a block-uniform distribution. Subsequently it has been shown that permutation closure is very unlikely to be satisfied in realistic scenarios [5], leading many researchers to conclude that No Free Lunch results have little consequence for practical applications of search algorithms.

## II. MAIN RESULTS

### A. Revisiting Algorithms

We now examine one of the fundamental assumptions in the proof of the No Free Lunch result, that the algorithms used are non-revisiting in the solution space. That is, the algorithms visit every point in the solution space exactly once. It has been remarked previously that such algorithms are impractical due to the time and space cost of storing and querying the set of visited points [6], although this is not strictly correct. It has also been suggested that revisiting algorithms, in the form of algorithms having a degree of redundancy in their solution encoding, have much to recommend them [3].

If we relax the assumption of non-revisiting algorithms, we can demonstrate the following result.

*Theorem 2 (Revisiting Breaks Permutation Closure):* A revisiting search over a given search space under a given c.u.p. set of objective functions can be formally expressed as a non-revisiting search over some larger search space under a set of objective functions that is not c.u.p., for any set containing only objective functions mapping to more than one element.

*Proof:* Searching under a c.u.p. set of objective functions $F$ with an algorithm $A$ which revisits some points $x_i \in \mathcal{X}$ is equivalent to a non-revisiting search of a new set of extended objective functions $F'$ on a larger space $\mathcal{X}'$, which contains $i$ additional points $x_i'$ under the constraint that $\forall i : f(x_i) = f(x_i')$.

For $F'$ to be closed-under-permutation it must definitely be larger than $F$, as it contains functions with the same codomain (because the search is presumed to be eventually exhaustive) on the larger domain $\mathcal{X}'$, and increasing the size of the domain necessarily increases the number of possible permutations. However, by construction $|F| = |F'|$, and so $F'$ cannot be closed-under-permutation. $\square$

Note that this theorem holds regardless of the degree of revisiting, so algorithms which revisit all points in the search space with equal frequency also effectively break permutation closure of the set of objective functions (c.f. [13]). We discuss the implications of this theorem for previous No Free Lunch results on encoding redundancy [3] in section

II-C. For now we briefly note the following obvious but simplifying result:

*Corollary 1 (Free Lunches):* All violations of the No Free Lunch theorems can be expressed as non-block-uniform distributions over problem subsets that are closed under permutation.

*Proof:* For uniform distributions over c.u.p. sets this follows directly from theorem 2 and the observation that non-c.u.p. sets are special cases of non-uniform distributions over c.u.p. sets. The extension to block-uniform distributions is by applying theorem 2 to each c.u.p. subset. $\square$

It is interesting to note that while violation of permutation closure leads to search algorithms potentially having different performance, this does not tell us to what extent algorithms might differ in performance. If induced performance differences are small, then searching for a superior algorithm for some problem set could be like looking for a 'needle in a haystack'. Here we present a first approach to quantifying the extent to which revisiting algorithms differ in performance, according to the amount of revisiting they allow. That violation of permutation closure leads to algorithms potentially having different performance has been known since [8], [4]. However, the strength of the Sharpened NFL theorem makes its contrapositive correspondingly weak; if $F$ is not c.u.p., then there exists at least one performance measure under which at least one pair of algorithms have differing performance on $F$. This leads to the relevant question of how likely two randomly selected algorithms are to have different performance for a given non-c.u.p. problem set, which has not previously been addressed. If most algorithms have identical performance on a given problem set, then looking for an algorithm with superior performance may be like looking for a needle in a haystack. If, however, almost all algorithms have different performance then empirical comparison of different algorithms becomes sensible. In fact the probability of two arbitrarily chosen algorithms having a different set of traces over some problem set, and hence potentially different performance under some appropriate performance measure, grows super-polynomially with increased frequency of revisiting, and so we can make a stronger claim than the contrapositive of the Sharpened No Free Lunch theorem provides: when revisiting is allowed, performance differences can be expected. The proof for this claim follows:

*Theorem 3:* Let $A$ be an algorithm which performs a revisiting search on a space $\mathcal{X}$ and revisits $r$ points before exhausting $\mathcal{X}$. The probability that $A$ and a similar algorithm $B$ are indistinguishable in performance decreases super-polynomially as the amount of revisiting increases, except where all elements of the search space are mapped to the same objective value.

*Proof:* Let $F'^{*}$ be the permutation closure of $F'$, and $F = f^{*}$ where $f^{*}$ is the permutation orbit of function $f$. Let $\mathcal{C} \subseteq \mathcal{Y}$ be the codomain of $f$, and let $\lambda_i$ be the number of points in $\mathcal{X}$ which $f$ maps to $i \in \mathcal{C}$ ($\lambda$ is the

*histogram* of $f$ [5]). Searching a set of functions takes each function to a trace, so the size of an algorithm's trace set is $|F'| = |F| = \frac{|\mathcal{X}|!}{\prod_{i\in\mathcal{C}}\lambda_i!}$. Since $F'$ is not c.u.p. this set of traces will be a small fraction of the traces for $F'^*$; we must make some assumptions about the behaviour of our algorithm to find $|F'^*|$. Consider the case where the algorithm revisits points assigned to the most common objective value, $j \in \mathcal{C}$, $r$ times during the search – the algorithm may do this by, for example, enumerating $\mathcal{X}$ until it knows $j$ and then revisiting a point mapped to it $r$ times – this simplistic revisiting algorithm is easy to analyse, and upper bounds the probability of distinguishability by ensuring that every function in $F'$ has the same histogram and minimising the size of its permutation closure. Knowing that all the functions in $F'$ have the same histogram, constructed by distributing the $r$ revisits amongst the $i$ bins of $\lambda$, the upper bound follows from the observation that $\prod_i \lambda_i'!$ is maximised by allocating all $r$ revisits to the largest bin in $\lambda$ (this claim can be shown by a simple induction).

Since every function in $F'$ has the same histogram, in which $\forall x \neq j : \lambda_x' = \lambda_x$ and $\lambda_j' = \lambda_j + r$, $|F'^*| = \frac{(|\mathcal{X}|+r)!}{\prod_{i\in\mathcal{C}}(\lambda_i+[i=j]\cdot r)!}$. The ratio $p = \frac{|F'|}{|F'^*|}$ is then the fraction of the set of all possible traces which a given algorithm will produce. Observing that a uniformly randomly chosen algorithm searching a given objective function gives a uniform distribution over the traces for that function, we see that for an algorithm $A$ searching $F'$ an arbitrarily chosen algorithm $B$ shares a trace with $A$ with probability $p$, and so $p$ bounds from above the probability that $A$ and a random $B$ are indistinguishable.

Some manipulation gives that $p = \frac{|\mathcal{X}|!}{(|\mathcal{X}|+r)!} \cdot \frac{(\lambda_j+r)!}{\lambda_j!}$, from which we can show $\ln p = \sum_{i=1}^{r} \ln \frac{\lambda_j+i}{|\mathcal{X}|+i}$. We can upper-bound this with an integral, which gives us

$$p(r) = \frac{|F'|}{|F'^*|} < \frac{(r-1+\lambda_j)^{(r-1+\lambda_j)} \times |\mathcal{X}|^{|\mathcal{X}|}}{\lambda_j^{\lambda_j} \times (r-1+|\mathcal{X}|)^{(r-1+|\mathcal{X}|)}},$$

from which we can see that $1/p(r)$ is at least superpolynomial in $r$ (excepting the trivial case where $\lambda_j = |X|$). □

Thus the probability that two algorithms selected uniformly at random are distinguishable increases very quickly as the extent to which the algorithms revisit increases. The expected number of points needing to be evaluated before a difference is found, and hence the computational complexity of detecting differences in algorithms' performance, is an interesting question that is outside the scope of the current paper. Furthermore, this approach will still not enable us to reason about *relative* performance of algorithms.

Returning to the main argument of the paper, we will next propose an alternative way of reasoning about revisiting algorithms and their expected performance over arbitrary sets of possible objective functions. This approach will indeed allow us to reason about algorithms' relative performance.

## B. Realistic Performance Measures and Random Search Algorithms

Proving No Free Lunch for arbitrary performance measures is a powerful result, based on an unrealistic assumption; that search algorithms are described by exhaustive non-revisiting enumerations of the search space and, equivalently that the set of objective functions considered be closed under permutation.

For the remainder of this paper we change our viewpoint from regarding search algorithms as deterministic exhaustive, possibly repeating, searches, to regarding them as stochastic processes. We also change from considering arbitrary performance measures, to considering performance measures representative of the goals of search and optimisation. These two changes in perspective will enable us to reason about the expected performance of realistic search algorithms, and make concrete recommendations to the practitioner.

First we define a *sensible* performance measure. As mentioned above, showing that No Free Lunch does not hold does not enable us to reason usefully about relative performance of different algorithms. We therefore define a sensible performance measure that reflects the goal of search and optimisation algorithms: finding good points in the search space. Concentrating on a particular performance measure will allow us to make definitive observations and recommendations on the relative performance of different search algorithms. This would not be possible if we allowed ourselves arbitrary performance measures (since an arbitrary performance measure is not *required* to distinguish between algorithms simply because some performance measure *could* do so).

*Definition 1 (Sensible performance measures):* Let $S(A, f, n)$ be the set of distinct values of objective function $f$ observed by algorithm $A$ after $n$ points have been evaluated. Then a *sensible performance measure* $M$, which we wish to maximise, defines the performance of $A$ on $f$ after $n$ points using only $S(A, f, n)$, with the additional constraint that $S_1 \subseteq S_2 \implies M(S_1) \leq M(S_2)$

Such measures capture the basic criterion against which search algorithms are assessed: how good are the solutions they generate? At first, it may appear that this performance measure discards the other important criterion of search algorithm performance: how long does it take to generate good solutions? However, as we shall see time is implicit in this performance measure, as we shall make comparisons of searches having equal length.

We now make the following No Free Lunch-like statement for revisiting algorithms.

*Proposition 1 (Maximising Sensible Performance):* For any distribution over any set of objective functions $F$, a randomly chosen enumeration $A$ can be expected to equal or outperform a randomly chosen non-minimally-revisiting algorithm $B$, under any sensible performance measure $M$ at any time $t$ in the search.

*Proof:* For any $t \geq |\mathcal{X}|$ it is clear that a minimally

revisiting algorithm has maximal performance as the full codomain of the objective will have been observed and, from definition 1, $M$ is required to depend only on the part of the codomain which has been seen.

When $t < |\mathcal{X}|$, $A$ samples $t$ points from $\mathcal{X}$ without replacement, whereas $B$ samples $t$ points from $\mathcal{X}$ with some non-zero probability of replacement (as a consequence of being non-minimally-revisiting). Consequently the expected number of distinct points sampled by $A$ is greater than the expected number sampled by $B$. Similarly the expected size of the set of objective values observed by $A$ exceeds (or equals, for functions whose codomain is of size 1) that of the set observed by $B$. Call these two sets of objective values $S_A$ and $S_B$.

$S_A$ and $S_B$ are subsets drawn randomly from the codomain of the objective, and the only effect $B$ can have on the distribution of values drawn is to reduce its range, so that $|S_B| \leq |S_A|$. For a randomly chosen set $S$ of objective values, $E(M(S))$ increases with $|S|$ (this is clear from the definition of a sensible measure), and so $E(M(S_A)) > E(M(S_B))$. This holds for any individual objective function, the extension to arbitrary distributions over arbitrary sets being by linearity of expectation. $\square$

It is sometimes remarked that No Free Lunch approaches typically ignore the time and space complexity of the algorithms, therefore it is interesting at this point to note that enumeration can typically be implemented with excellent time and space complexity; $O(\log |\mathcal{X}|)$ time for each point queried[1] and $O(\log |\mathcal{X}|)$ space overall, when solutions can be represented as finite strings. No algorithm with lower space complexity can have better performance in the NFL sense as it would necessarily revisit, as it could not have a state for each point in which it would deterministically query that point. Furthermore, although revisiting algorithms can easily be made non-revisiting, only algorithms which are specialised to the problem can optimise faster in wall-clock time. This further indicates enumeration's pre-eminent position as the general search algorithm, and motivates its use as a benchmark by practitioners.

An interesting consequence of proposition 1 is

*Corollary 2 (Random Search Performance):* Random search algorithms, those that ignore the outcome of their search in selecting subsequent search points, can differ in performance.

*Proof:* This follows directly from proposition 1 and observing that random search algorithms, as defined above, include both blind enumeration, and revisiting algorithms. $\square$

The definition of random search algorithms as being those that ignore the objective values generated during the search was proposed by [12]. Corollary 2 is interesting, as the fundamental prediction of the No Free Lunch result has

been stated as being that "if an algorithm performs better than random search on some class of problems then it must perform *worse than random search* on the remaining problems" (emphasis authors') [12]. Corollary 2 shows that the property of 'randomness' is actually not of great importance for No Free Lunch-like statements regarding realistic search algorithms.

Finally, an important result emerges as a direct consequence of proposition 1:

*Corollary 3 (Performance Prediction):* Empirically-demonstrated better-than-enumeration performance of some algorithm on some problem set predicts worse-than-enumeration performance on any other disjoint problem set whose relationship to the first is unknown.

*Proof:* For some algorithm A let us denote expected deviation from the expected performance of enumeration on a problem set $\psi$ as $E(\Delta_A(\psi))$. Then, from proposition 1, $E(\Delta_{\text{enum}}(\psi)) = 0$ for any $\psi$, and similarly for any non-minimally-revisiting algorithm $A$ whose performance on $\psi$ we know nothing about, $E(\Delta_A(\psi)) < 0$. If we have an unknown problem set $\phi$ whose relationship with set $\psi$ is unknown then, by definition, $E(\Delta_A(\phi)|\Delta_A(\psi)) = E(\Delta_A(\phi))$, which must also be negative by proposition 1. $\square$

A related but different result based on considerations of function complexity has previously been presented [1], which demonstrates that for any function where an algorithm performs well there is another function *of similar complexity* on which it performs badly. Our result is both simpler and more generally applicable, in that it makes no reference to the details of the algorithms or the complexity of the functions involved, but weaker in its predictions, in that it predicts no correlation between performances on the two problem sets, rather than a negative relation.

*C. Encoding Redundancy, Neutrality and Revisiting*

In No Free Lunch contexts, revisiting by an algorithm might occur through the decision not to keep track of search points visited so far. An additional cause for revisiting is encoding redundancy. We can represent encoding in a No Free Lunch context by considering an additional set $\mathcal{W}$ of encoded solutions, which are decoded into their corresponding points in the search space by a 'growth' function $g : \mathcal{W} \mapsto \mathcal{X}$ [6]. If this mapping is not injective, then we have a redundant encoding. Such redundancy is often found in heuristic search algorithms, and will turn a non-revisiting algorithm in the space of encoded solutions into a revisiting algorithm in the true solution space, so that the No Free Lunch theorem no longer holds. Radcliffe & Surry may not have noticed this fact, as their method of proving the No Free Lunch result relies on setting the search space $\mathcal{W}$ and solution space $\mathcal{X}$ to be the same, and the growth function $g$ to be the identity mapping. It has previously been suggested in [13] that a non-uniform (*i.e.* biased) encoding redundancy would break the No Free Lunch theorem. Here we have demonstrated that in fact non-uniformity is not the key requirement for breaking No Free Lunch, encoding redundancy alone is sufficient.

---

[1]The time taken to exhaust the search space is thus $O(|\mathcal{X}|)$, which also cannot be improved upon.

Previously Igel and Toussaint have presented an analysis of encoding redundancy, or neutrality, in No Free Lunch situations, arguing that under encoding redundancy all algorithms still have equal expected performance [3]. In fact the No Free Lunch theorem does not hold once redundancy is introduced for, as shown above, redundant and hence revisiting algorithms break permutation closure of any set of objective functions. Thus care must be taken in interpreting these results; the assumption of theorem 1 in [3] is violated and hence the subsequent claim that the time to find an optimum averaged over all functions is the same for all algorithms when redundancy is allowed is incorrect. Igel and Toussaint actually analyse the difficulty of search problems in terms of the proportion of *distinct* solutions (members of $\mathcal{X}$) mapped to *the same* global optimum (member of set $\mathcal{Y}$); this is not the same thing as considering multiple encoded solutions (members of $\mathcal{W}$) mapped to the same actual solution (member of set $\mathcal{X}$). As their analysis considers only algorithms that are non-revisiting in the solution space the No Free Lunch result is still applicable and their analysis valid, however nothing has been said about the effects of true encoding neutrality on the performance of search algorithms. When encoding redundancy is correctly defined as a non-injective mapping between a larger representation space and a smaller solution space, the stochastic search results presented here become applicable. The observation that encoding redundancy in a given problem leads to enumeration having the best expected performance may raise questions about the theoretical basis for the claimed benefits of neutrality and self-adaptation [3], which are outside the scope of this paper.

## III. Conclusions

In this paper we have shown that the requirement for the Sharpened No Free Lunch theorem to hold, that the problem set under consideration be closed under permutation, is in fact a necessary and sufficient condition for a version of the theorem, extended to also admit revisiting algorithms, to hold. It has previously been shown that the c.u.p. condition is very unlikely to be satisfied for realistic problem classes. We have shown further that realistic, revisiting algorithms also violate this condition. That the No Free Lunch theorems' assumptions are typically violated has been used many times as an argument that they can reasonably be ignored by those working with search algorithms, while a branch of research has also grown up showing conditions under which 'Almost' No Free Lunch results hold.

If the conditions of No Free Lunch are violated then some algorithm can have best expected performance, but this does not help us to find it, or to say anything about it. The main contribution of this paper is to propose a statistical analysis of search algorithms as random processes, demonstrating that algorithms that minimise the extent to which they revisit points in the search space have higher expected performance. The result of this is the demonstration that for arbitrary sets of objective functions, including those which are not closed under permutation and over which arbitrary probability distributions hold, enumeration has better expected sensible performance than any arbitrary revisiting algorithm. Note that arbitrary problem distributions obviously includes realistic problem distributions, such as the universal distribution [9]. It is also interesting to note that, through use of a string encoding, an algorithm to perform a blind enumeration of a search space will typically have excellent algorithmic time and space complexity, in $O(|\mathcal{X}|)$ and $O(\log|\mathcal{X}|)$ respectively.

This paper may also help guide practice. Corollary 3 should be of great interest to those empirically investigating the performance of search algorithms. It does not say that superior performance on one problem set will necessarily result in inferior performance on another, but does say that if we have no information whatsoever about the relationship between two problem sets, performance observed to be better than enumeration on one still predicts performance worse than enumeration on the other. Of course, it is rare that one knows nothing at all about the relationship between problem sets, even if it is simply that at an intuitive level problems from the two sets seem similar in some way. Furthermore as an experimenter collects performance statistics for their algorithms on the unknown problem set, uncertainty over the algorithms' performance on that set will decrease so that they are able to predict performance on previously unseen instances with increasing confidence. Nevertheless, consider the best policy if you were asked to play the following simple game: a third-party gives you an algorithm and a set of performance data for it on some problem set, then tells you that you have a choice of whether to apply the algorithm to one problem from some other problem set, or to apply a random enumerative search. No information whatsoever is given about the new problem set, and the aim of this game is to achieve the best performance on the new unseen problem on your first attempt. Corollary 3 indicates that you should choose enumeration instead of the known algorithm. Obviously one rarely, if ever, wishes to win such a game, and applying a known algorithm to an unknown problem set has another payoff in terms of information gained, but hopefully corollary 3, the computational simplicity of enumeration, and the game outlined above will all help to convince practitioners that an algorithm's empirical performance on some problem set should be directly compared against a deterministically enumerative search started at some uniformly randomly selected point in the search space or, equivalently, a uniform random search using sampling without replacement. At present such practice is far from common.

## References

[1] S. Droste, T. Jansen, I. Wegener, Optimization with randomized search heuristics–the (A) NFL theorem, realistic scenarios, and difficult functions, Theor. Comp. Sci. 287 (1) (2002) 131–144.
[2] T. M. English, Some information theoretic results on evolutionary optimization, in: P. J. Angeline, et al. (eds.), CEC 1999, vol. 1, IEEE Service Center, 1999, pp. 788–795.
[3] C. Igel, M. Toussaint, Neutrality and self-adaptation, Nat. Comp. 2 (2) (2003) 117–132.

[4] C. Igel, M. Toussaint, On classes of functions for which no free lunch results hold, Inf. Process. Lett. 86 (6) (2003) 317–321.

[5] C. Igel, M. Toussaint, A No-Free-Lunch theorem for non-uniform distributions of target functions, J. Math. Model. Algs. 3 (4) (2005) 313–322.

[6] N. J. Radcliffe, P. D. Surry, Fundamental limitations on search algorithms: evolutionary computing in perspective, in: J. van Leeuwen (ed.), LNCS, vol. 1000, Springer, Berlin, 1995, pp. 275–291.

[7] J. Rowe, M. Vose, A. Wright, Reinterpreting no free lunch, Evol. Comp. 17 (1) (2009) 117–129.

[8] C. Schumacher, M. D. Vose, L. D. Whitley, The no free lunch and problem description length, in: L. Spector, et al. (eds.), GECCO 2001, Morgan Kaufmann, 2001, pp. 565–570.

[9] R. Solomonoff, A formal theory of inductive inference. Parts I and II, Info. and Cont. 7 (2) (1964) 224–254.

[10] D. Whitley, J. Rowe, Focused no free lunch theorems, in: C. Ryan, M. Keijzer (eds.), GECCO 2008, ACM, 2008, pp. 811–818.

[11] D. H. Wolpert, The lack of A priori distinctions between learning algorithms, Neur. Comp. 8 (7) (1996) 1341–1390.

[12] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comp. 1 (1) (1997) 67–82.

[13] J. Woodward, GA or GP? that is not the question, in: R. Sarker, et al. (eds.), CEC 2003, 2003, pp. 1056–1063.