Two Broad Classes of Functions for Which a No Free Lunch Result Does Not Hold

Matthew J. Streeter

Genetic Programming, Inc.
Mountain View, California
mjs@tmolp.com

Abstract. We identify classes of functions for which a No Free Lunch result does and does not hold, with particular emphasis on the relationship between No Free Lunch and problem description length. We show that a NFL result does not apply to a set of functions when the description length of the functions is sufficiently bounded. We consider sets of functions with non-uniform associated probability distributions, and show that a NFL result does not hold if the probabilities are assigned according either to description length or to a Solomonoff-Levin distribution. We close with a discussion of the conditions under which NFL can apply to sets containing an infinite number of functions.

1 Introduction

The No Free Lunch theorems [3,8,12] loosely state that all search algorithms have the same performance when averaged over all possible functions. Given this fact, an important question for anyone interested in the design of black-box search algorithms is whether a No Free Lunch result holds for the subset of all possible problems that represents actual problems of interest. In this paper we approach this question from two angles: Bayesian learning and description length. In Sect. 2 we show that a No Free Lunch result applies only when a certain form of Bayesian learning is impossible, and suggest that the possibility of this form of Bayesian learning may be characteristic of interesting problems. Sections 3 and 4 focus on the relationship between No Free Lunch and problem description length. Section 5 discusses circumstances under which No Free Lunch applies to infinite sets. Section 6 discusses limitations of this work. Section 7 is the conclusion. The remainder of this section introduces terminology that will be used throughout this paper.

1.1 Terminology

Search Algorithm Framework. Our framework is essentially that given in [10]. Let X be a finite set of points in a search space, and let Y be a finite set of cost values assigned to the points in X by a cost function $f:X\rightarrow Y$. For simplicity, we assume in this paper that the elements of X and Y are integers. Define a trace of size m to be a sequence of pairs $T_m \equiv \langle(x_0,y_0), (x_1, y_1), ..., (x_{m-1},y_{m-1})\rangle$ where for $0 \leq i \leq m$-1, $x_i \in X$ and $y_i \in Y$. Adopt the notation: $T_m^{\ x} \equiv \langle x_0, x_1, ..., x_{m-1}\rangle$; $T_m^{\ y} \equiv \langle y_0, y_1, ..., y_{m-1}\rangle$.

Define a search algorithm $A:T,X\rightarrow X$ to be a function that takes as input a trace T and a domain X, and returns as output a point x_i where $x_i \in X$ and $x_i \notin T^x$. Under this

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1418–1430, 2003.

formulation, we consider only non-retracing, deterministic black-box search algorithms (hereafter, "search algorithms"). However, the conclusions we will draw about such search algorithms can be extended to stochastic, potentially retracing algorithms using the arguments made by Wolpert and Macready [12] to show that the No Free Lunch theorem applies to such algorithms.

A search algorithm A operating on cost function $f{:}X\rightarrow Y$ is evaluated according to the following steps:

1. Let $T=T_0$, where $T_0 \equiv \langle\rangle$ is the empty trace.
2. Evaluate $A(T,X)$ to obtain a point $x_i \notin T^x$.
3. Evaluate $f(x_i)$, and append the pair $(x_i, f(x_i))$ to T.
4. As long as there exists some x_j such that $x_j \in X$ and $x_j \notin T^x$, return to step 2.

Let $T_m(A,f)$ denote the length m trace generated by algorithm A and function f (i.e. the trace T obtained after executing steps 2-3 m times). We define $V_m(A,f) \equiv (T_m(A,f))^y$ to be the length *m performance vector* generated by A and f. When the subscript is omitted, the performance vector will be assumed to have length $|X|$ (i.e., $V(A,f) \equiv V_{|X|}(A,f)$). We let $P(v,A)$ denote the probability of obtaining performance vector v when running algorithm A against a cost function chosen at random under P.

No Free Lunch Results. Define a *performance measure* $M{:}V\rightarrow\Re$ to be a function that takes a performance vector as input and produces a non-negative real number as output. Let F be a set of functions, and let P be a probability distribution over F. We define the overall performance, $M_O(A)$, of a search algorithm A as:

$$M_O(A) \equiv \sum_{f \in F} P(f) M(V(A,f)).$$

We say that a *No Free Lunch result* applies to the pair (F,P) iff., for any performance measure M and any pair of search algorithms A and B, $M_O(A) = M_O(B)$. In other words, a No Free Lunch result applies to (F,P) iff. the overall performance with respect to any performance measure M is independent of the chosen search algorithm. This is essentially the definition of a No Free Lunch result used by Wolpert and Macready [12]. Schumacher [10] has proposed a related definition of a No Free Lunch result that applies when P is uniform (and is equivalent to our definition in this case). In some cases we shall omit P and simply say that a No Free Lunch result applies to a set of functions F. By this we shall mean that a No Free Lunch result applies to (F,P), where P is a uniform probability distribution over F.

Set-Theoretic Terminology. A multiset is a set that may contain multiple copies of a given element (e.g. {0,0,1,0}). A *generalized permutation* is an ordered listing of the elements in a multiset. We define $G(Y_{mult})$ to be the number of generalized permutations of some multiset Y_{mult}. Where $f{:}X\rightarrow Y$ is a function, we let $Y_{mult}(f)$ denote the multiset containing one copy of $f(x_i)$ for each $x_i \in X$. $Y_{mult}(f)$ differs from the range of f in that $Y_{mult}(f)$ may contain more than one copy of a given $f(x_i) \in Y$.

Let $f{:}X\rightarrow Y$ be a function and let $\sigma{:}X\rightarrow X$ be a permutation. We define the permutation σf of f to be the function $\sigma f(x)=f(\sigma^{-1}(x))$. We say that a set of functions F is closed under permutation iff., for any $f \in F$ and any σ, $\sigma f \in F$. By the words "set of functions", we will mean a finite set of functions having a common, finite domain, unless otherwise specified.

1.2 The No Free Lunch Theorems

The following is essentially the No Free Lunch theorem proved by Wolpert and Macready [12].

NFL: Let X and Y be finite sets, and let $F \equiv Y^X$ be the set of all functions having domain X and codomain Y. Let P be a uniform distribution over F. A No Free Lunch result applies to (F,P).

Schumacher [10] has provided a sharpened version of this theorem that gives both necessary and sufficient conditions when P is uniform. We refer to this as NFLP.

NFLP: Let F be a set of functions, and let P be a uniform probability distribution. A No Free Lunch result applies to (F,P) iff. F is closed under permutation.

Though Schumacher uses a different definition of No Free Lunch result than the one used in this paper, it can easily be shown that the two definitions are equivalent when P is uniform. The relationship between No Free Lunch and permutations of functions has also been studied by Whitley [11].

2 No Free Lunch and Bayesian Learning

In this section we investigate the relationship between No Free Lunch and Bayesian learning. We show that a No Free Lunch result applies to a set of functions if and only if a certain form of Bayesian learning is not possible. We then argue that only very weak assumptions about the set of "interesting" functions are necessary in order to guarantee that this type of Bayesian learning is possible for interesting functions.

Let F be a set of functions, and let f be a function chosen at random from F under some probability distribution P. Let S be a set of points in the search space that have already been visited (i.e., that are known to be part of f). Let x_i be a point in the search space that has not yet been visited. Let $P(f(x_i)=y \mid S)$ denote the conditional probability, given our knowledge that f contains the points in S, that $f(x_i)=y$. The task of the search algorithm is to choose the value of i (subject to the constraint that x_i has not yet been visited). Clearly if $P(f(x_i)=y \mid S)$ is independent of x_i for all y, the decision made by the search algorithm has no influence on the next y-value that is obtained. Thus, intuitively, one might expect that all search algorithms would perform identically if the value of $P(f(x_i)=y \mid S)$ is independent of i for all S and y. What we establish in this section is that this independence is both a necessary and sufficient condition for a No Free Lunch result to apply to (F,P). We first prove the following Lemma.

Lemma 2.1. Let F be a set of functions, and let P be a probability distribution over F. A No Free Lunch result holds for the pair (F,P) iff., for any performance vector v and any search algorithms A and B, $P(v,A) = P(v,B)$.

Proof: (IF) Trivial. (ONLY IF) Suppose that $P(v,A) \neq P(v,B)$ for some v, A, and B. In this case we can define M to be a performance measure that assigns a value of 1 to v while assigning a value of 0 to all other performance vectors. We will thus have $M_O(A)=P(v,A)$ whereas $M_O(B)=P(v,B)$, so $M_O(A) \neq M_O(B)$. □

Theorem 2.2. Let F be a set of functions, and let P be a probability distribution over F. Let f be a function selected at random from F under P, and let $S=\{(x_{f,0},y_{f,0}), (x_{f,1},y_{f,1}), ..., (x_{f,n-1},y_{f,n-1})\}$ denote a (possibly empty) set of n pairs that are known to belong to f. Let X_S denote the domain of S, and let $P(f(x_i)=y \mid S)$ denote the conditional probability that $f(x_i)=y$, given our knowledge of S. Let $x_i,x_j \notin X_S$ be two points in the search space whose cost values are not yet known. Then a No Free Lunch result applies to (F,P) iff., for any S and y, the equation

$$P(f(x_i)=y \mid S) = P(f(x_j)=y \mid S) \tag{2.1}$$

holds for all x_i,x_j.

Proof: (ONLY IF) By way of contradiction, suppose that a No Free Lunch result applies to (F,P), but that $P(f(x_i)=y \mid S) \neq P(f(x_j)=y \mid S)$ for some x_i,x_j. Let F_S denote the subset of F containing functions that are consistent with S. Let $X_{LEX} = \langle x_0,x_1, ..., x_{|X|}\rangle$ denote some fixed (perhaps lexicographic) ordering of the elements of X.

Let A and B be two search algorithms that each unconditionally evaluate the n points $x_{f,0}$, $x_{f,1}$, ..., $x_{f,n-1}$ as the first n steps of their execution. If the cost values obtained by evaluating these n points are not exactly the cost values in S, then both A and B continue to evaluate the remaining points in the order specified by X_{LEX}. If the observed cost values *are* those in S, then A chooses x_i as the next point whereas B chooses x_j. From this point onward, A and B both evaluate the remaining points in the order specified by X_{LEX}.

Let V_{PRE} denote the set of performance vectors that begin with the prefix $\langle y_{f,0}, y_{f,1}, ..., y_{f,n-1}, y\rangle$, and let $P_{VPRE}(a)$ denote the probability of obtaining a performance vector that is a member of V_{PRE} using search algorithm a (run against a function drawn at random from F under P). Let $P_S(a)$ denote the probability of obtaining the n points in S as the result of the first n evaluations of a search algorithm a. We have:

$$P_{VPRE}(A) = P(f(x_i)=y \mid S)*P_S(A) \text{ and } P_{VPRE}(B) = P(f(x_j)=y \mid S)*P_S(B).$$

Because A and B behave identically for the first n steps of their execution, it is clear that $P_S(A)=P_S(B)$. It cannot be the case that $P_S(A)=0$, because A obtains the points in S when running against f. Thus the fact that $P_S(A)=P_S(B)$, in combination with the assumption that $\mathrm{P}(f(x_i)=y \mid S) \neq \mathrm{P}(f(x_j)=y \mid S)$, establishes that $P_{VPRE}(A) \neq P_{VPRE}(B)$. There must therefore be some $v \in V_{PRE}$ that satisfies the equation $P(v,A) \neq P(v,B)$. By Lemma 2.1, this establishes that a No Free Lunch result does not apply to (F,P), which is a contradiction.

(IF) If S is the empty set, then equation 2.1 becomes $P(f(x_i)=y) = P(f(x_j)=y)$, so the initial choice made by the search algorithm cannot effect the probability of obtaining any particular performance vector v_1 (of length 1). Assume that the search algorithm cannot affect the probability of obtaining a performance vector v_n (of length n), and let S denote the first n pairs observed by some search algorithm. The equation $P(f(x_i)=y \mid S) = P(f(x_j)=y \mid S)$ guarantees that the choice of i made by the search algorithm cannot affect the probability of obtaining any particular value y on the next evaluation. Therefore the performance vector v_{n+1} of length n+1 obtained on the next evaluation will also be independent of the search algorithm. Thus, by induction, the probability of obtaining any particular performance vector is independent of the search algorithm, which by Lemma 2.1 establishes that a No Free Lunch result holds for (F,P). Note that this second half of our proof is similar to the derivation of the No Free Lunch theorem itself [12]. □

2.1 Discussion

Equation 2.1 provides a necessary and sufficient condition for a No Free Lunch result to hold for (F,P). We now examine some of the consequences this equation. Suppose equation 2.1 holds for some (F,P), where F is a set of functions and P is a probability distribution. Letting S be the empty set, we have $\mathrm{P}(f(x_i)=y) = \mathrm{P}(f(x_j)=y)$ for all x_i,x_j. Since the probability that $f(x_i)$ is equal to any particular value y is independent of i, the expected fitness of x_i is also independent of i, so $\mathbf{E}[f(x_i)]=\mathbf{E}[f(x_j)]$ for all x_i,x_j. Thus, by linearity of expectations,

$$\mathbf{E}[|f(x_i)-f(x_j)|] = \mathbf{E}[|f(x_k)-f(x_l)|] \tag{2.2}$$

for all x_i, x_j, x_k, x_l.

Equation 2.2 is of particular relevance for genetic algorithms. Suppose the x_i are chromosomes, and the cost values $f(x_i)$ are their fitness values. Equation 2.2 tells us that even if x_i and x_j have 98% of their genetic material in common, while x_k and x_l have only 2% of their genetic material in common, we are to make no assumption that the fitness of x_i and x_j is likely to be closer than that of x_k and x_l. As another illustration of the consequences of equation 2.2, suppose that $k=i$, that x_j is an individual obtained by randomly mutating one gene of x_i, and that x_l is an individual obtained by randomly mutating all the genes of x_i. Equation 2.2 tells us that the expected effect of this point mutation on fitness is the same as the expected affect on fitness of replacing x_i with a chromosome generated at random. In short, equation 2.2 expresses the assumption that there is no correlation between genotypic similarity and similarity of fitness.

Under (a form of) the assumption that nearby points in the search space do tend to have similar cost values, Christensen and Oppacher [3] have shown that a simple algorithm called SUBMEDIAN-SEEKER outperforms random search.

As a further consequence of equation 2.1, note that if the probability of obtaining any particular value y is independent of the choice of i, then the probability of obtaining any range of y-values is also independent of i, so that equation 2.1 implies:

$$\mathrm{P}(y_{min} \leq f(x_i) \leq y_{max} \mid S) = \mathrm{P}(y_{min} \leq f(x_j) \leq y_{max} \mid S) \tag{2.3}$$

Equation 2.3 is particularly relevant to the analysis of genetic algorithms by Holland involving schema [5]. As an illustration, suppose the search space X consists of all 32-bit chromosomes, and the set of cost values Y are interpreted as rational numbers between 0 and 1. Let $s_1 \equiv$ `ab0c????` and $s_2 \equiv$`f18a????` denote two schemata, where the chromosomes are specified in hexadecimal and where the `?` characters are 4-bit wildcards. Suppose each of these schemata are sampled at random (without replacement) 1000 times, and the observations are placed into a set S containing 2000 pairs. Let the schema s_1 have an observed fitness distribution with mean 0.7 and variance 0.1, while that of s_2 is observed to have mean 0.3 and variance 0.01. Now suppose that two additional (unique) samplings are made of s_1 and s_2, and that one is asked to bet on which sampling returns the higher value. If one enters this scenario with equation 2.3 as an assumption, one shall regard the fact that the points in s_1 and s_2 are distributed so differently as an extreme coincidence, but one will not make any extrapolations about unseen points in s_1 or s_2. Yet clearly there are a wide variety of interesting problems for which such statistical inferences are possible.

Given that equation 2.1 holds if and only if a No Free Lunch result applies to the pair (F,P), both equations 2.2 and 2.3 can be regarded as consequences of the No Free Lunch theorem. However, because equation 2.1 is both a necessary and sufficient condition for a No Free Lunch result to hold for (F,P), equations 2.2 and 2.3 can also be regarded as *assumptions* required by the No Free Lunch theorem. Thus, if one does not accept these assumptions, one has grounds for ignoring No Free Lunch.

Other restrictions on the circumstances under which a No Free Lunch result can apply have been provided by Igel and Toussaint [6], who show that a No Free Lunch does not apply to F when F includes only those functions having less than the maximum number of local optima or less than the maximum steepness (as measured w.r.t. some neighborhood structure defined on the search space). Arguments similar to the ones above have been made by Droste, Jansen, and Wegener [4].

3 Functions of Bounded Description Length

In the previous section, we defined various statistical properties that the set of functions F and probability distribution P must satisfy in order for a No Free Lunch result to apply to (F,P). We now focus on properties that the cost functions in F must have in order for a No Free Lunch result to apply, assuming that the cost functions are implemented as programs. In this section, we assume P is a uniform distribution, and that the restrictions on the admissible cost functions are couched in terms of description length. In the next section, we will consider the case where P is defined as a function of description length.

The (minimum) description length, $K_U(f)$, of a function $f{:}X\rightarrow Y$ with respect to some universal Turing machine U is the length of the shortest program that runs on U and implements f. By a program that implements f, we mean a program that produces output $f(x)$ for input $x\in X$, and that produces an error code for all inputs $x\notin X$. Description length is also known as Kolmogorov complexity [7]. The Compiler theorem [7] shows that for two universal Turing machines U and V, the difference $|K_U(f)-K_V(f)|$ is bound by some constant that depends on U and V but not on f. For this reason, it is customary to omit the subscript and simply write the description length as $K(f)$.

3.1 Relevance of Description Length

Consider running a genetic algorithm on a problem involving a 500 byte chromosome and 4 byte fitness values. A fitness function $f:X\rightarrow Y$ defines a mapping from the set of chromosomes X to the set of fitness values Y. An explicit representation of f in memory (i.e., one that simply listed all pairs $(x_i,y_i)\in f$) would require $|X|*(\lg|X|+\lg|Y|)\approx 2^{4012}$ bits of storage. Even if we allow the domain of f to be implicit (i.e., we simply list the y-values in $Y_{mult}(f)$ in some fixed order), the amount of storage required is still $|X|*\lg|Y|=2^{4005}$ bits. No amount of memory in any way approaching this quantity is available on current computer systems. Rather, evaluation of a fitness function f (assuming that f is implemented in software) would typically involve a call to compiled code occupying perhaps tens of megabytes of memory. In such a case, implementing f as a program rather than an explicit list provides a compression ratio of over 10^{1000}:1. Thus, we say that real-world fitness functions are highly compressible.

Given this property of real-world fitness functions, the question arises as to whether a No Free Lunch result applies to sets of functions whose definition reflects this property. Droste, Jansen, and Wegener have shown that a No Free Lunch result does not hold for certain classes of functions having highly restricted description length (e.g., functions representable by a boolean circuit of size at most 3) [4]. Schumacher has shown that a No Free Lunch result can hold for *some* sets of functions that are compressible [10] (e.g., needle-in-the-haystack functions), but this does not mean that a NFL result will apply to a subset of $F\equiv Y^X$ defined by placing a bound on description length.

The remaining theorems in this section will make use of the following Lemma.

Lemma 3.1. Let F be the permutation closure of some function $f:X\rightarrow Y$ (i.e., $F\equiv\{\sigma f, \sigma:X\rightarrow X \text{ is a permutation}\}$). Then $|F|=G(Y_{mult}(f))$, for any $f\in F$.

Proof: Because all functions in F are permutations of one another, $Y_F\equiv Y_{mult}(f)$ denotes the same multiset for any $f\in F$. The functions in F also have the same domain X. If we establish some ordering for the elements of X, then any function $f\in F$ can be uniquely specified by listing in order the y-values (among those in Y_F) that f assigns to the elements of X. Thus, there is a one-to-one correspondence between elements of F and generalized permutations of Y_F, so $|F|=G(Y_F)$. □

The following theorem defines bounds on description length that are sufficient to ensure that a No Free Lunch result does not hold for a set of functions (under a uniform probability distribution).

Theorem 3.2 ("Hashing theorem"). Let h be a hashing function, and let $F\equiv Y^X$ be a set of functions. Let F_k be the subset of F containing only those functions of description length k bits or less, where $K(h)\leq k$. Then a No Free Lunch result does not apply to F_k so long as:

$$k < \lg(G(Y_{mult}(h))+1)-1.$$

Proof: The number of halting programs of description length k bits is at most 2^k. Therefore the number of halting programs of description length k or less is at most 2^{k+1}-1, so $|F_k| \leq 2^{k+1}-1$. Let H be the set of functions containing all permutations of h. By Lemma 3.1, $|H|=G(Y_{mult}(h))$. The inequality $k<\lg(G(Y_{mult}(h))+1)-1$ can be rewritten as $2^{k+1}-1<G(Y_{mult}(h))$, so that we have $2^{k+1}-1<|H|$. This implies $|F_k|<|H|$, which means that H is not a subset of F_k. Let $h_k \in H$ denote a function not in F_k. By the definition of H, h_k must be a permutation of h. But h must be a member of F_k since we have assumed $K(h) \leq k$. Thus F_k is not closed under permutation, so by NFLP a No Free Lunch result does not apply to F_k. □

We refer to h as a "hashing function" because hashing functions (used to assign a key to a random position in a hash table) typically have small description length $K(h)$ and are designed to generate many different y-values, which maximizes $G(Y_{mult}(h))$. Thus, the theorem will tend to provide sharper upper and lower bounds when h is a hashing function. Of course, the theorem applies equally to any function h.

We now prove a special case of Theorem 3.2 that will allow us to illustrate some of its implications.

Theorem 3.3. Let X be a set containing the first $|X|$ non-negative integers, and let Y be a set containing the first $|Y|$ non-negative integers. Let h be the hashing function $h(x) = x \bmod |Y|$, and let the description length of this function be denoted by $k_{mod} \equiv K(h)$. Let F_k be the subset of Y^X containing only those functions of description length k bits or less, where $k_{mod} \leq k$. Then a No Free Lunch result does not apply to F_k so long as $k < \lg(G(Y_{mult}(h))+1)-1$, where the value of $G(Y_{mult}(h))$ is given by:

$$G(Y_{mult}(h)) = \frac{|X|!}{\left(\left\lceil \frac{|X|}{|Y|} \right\rceil !\right)^{(|X| \bmod |Y|)} \left(\left\lfloor \frac{|X|}{|Y|} \right\rfloor !\right)^{(|Y|-(|X| \bmod |Y|))}} \tag{3.1}$$

Proof: The Hashing theorem establishes that a No Free Lunch result does not apply to F_k so long as $k<\lg(G(Y_{mult}(h))+1)-1$. It remains only to establish the value of $G(Y_{mult}(h))$. Let n denote the number of distinct elements in the multiset $Y_{mult}(h)$, and let $a_0, a_1, \ldots, a_{n-1}$ be counts of the number of occurrences of each of these n elements in $Y_{mult}(h)$. $G(Y_{mult}(h))$ is given by the standard formula for counting the number of generalized permutations of a multiset [1]:

$$G(Y_{mult}(h)) = \frac{|Y_{mult}(h)|!}{(a_0!)(a_1!)\ldots(a_{n-1}!)} \tag{3.2}$$

We now establish that the right hand sides of equations 3.1 and 3.2 are equal. Because there is one element in $Y_{mult}(h)$ for every element in X, clearly $|X|=|Y_{mult}(h)|$, so that the numerators in the two expressions are equal. To see that the denominators are also equal, note that the first $(|X| \bmod |Y|)$ integers will appear the most times in $Y_{mult}(h)$. Specifically, these integers will each appear ceiling($|X|/|Y|$) times in $Y_{mult}(h)$. The remaining $(|Y|-(|X| \bmod |Y|))$ integers will appear only floor($|X|/|Y|$) times in $Y_{mult}(h)$. Letting $L=(|X| \bmod |Y|)$, $c=$ceiling($|X|/|Y|$), and $f=$floor($|X|/|Y|$), we see that the equation $a_i=c$ holds for L distinct values of i, while the equation $a_i=f$ for holds for $|Y|-L$ values of i. The value of the denominator is thus $((c)!)^L*(f!)^{|Y|-L}$, so that:

$$G(Y_{mult}(h))=\frac{|X|!}{(c!)^L(f!)^{(|Y|-L)}} \tag{3.3}$$

Expanding equation (3.3) in terms of c, f, and L establishes the theorem. □

As an illustration of Theorem 3.3, let X contain all n-bit integers ("chromosomes"), and let Y contain all m-bit integers ("fitness values"). Let F_k be the subset of Y^X containing only those functions of description length k bits or less. Theorem 3.3 ensures that a No Free Lunch result does not apply to F_k so long as $k_{mod}<k<\lg(G(Y_{mult}(h))+1)-1$, where the value of $G(Y_{mult}(h))$ is given by equation 3.1. Table 1 presents the values of these lower and upper bounds on k (which is itself an upper bound on the description length of functions in F_k) as a function of n and m. The values in the last column of the table were computed using Stirling's approximation for factorials. All table entries are in bits. Note that the values in the rightmost column of the table are the upper bounds $\lg(G(Y_h)+1)-1$, and not the numbers of generalized permuations $G(Y_h)$.

Table 1. Upper bounds on description length necessary to ensure that a No Free Lunch result does not hold

Chromosome length n	Fitness value length m	Least upper bound	Greatest upper bound $\lg(G(Y_{mult}(h))+1)-1$
16	1	k_{mod}	$6.55*10^4$
32*8	8	k_{mod}	$9.26*10^{77}$
500*8	32	k_{mod}	$4.22*10^{1205}$

The second row of the table tells us that for a No Free Lunch result to apply to F_k where $n=32$ bytes and $m=1$ byte, we must be prepared to allow F_k to contain functions of description length $9.26*10^{77}$ bits. To put this number into perspective, the number of atoms in the universe is estimated to be approximately 10^{80}. For any reasonable machine architecture, the lower bound k_{mod} will be measured in tens of bytes. Thus, if we assume that P is uniform and define F_k as above, only very weak assumptions about k are required in order to ensure that a No Free Lunch result does not apply to (F_k,P).

4 Distributions Defined by Description Length

The previous section concerned sets of functions that are defined by placing a bound on description length. In this section, we will be concerned with probability distributions that are defined in ways that are closely related to description length. We begin by introducing some complexity-theoretic terminology that is relevant to our discussion.

Let U be a special type of Turing machine that runs only self-delimiting programs (i.e., no halting program can be the prefix of another). The Solomonoff universal prior (or Solomonoff-Levin distribution), P_U, is defined as:

$$P_U(f) = \sum_{p:\, p \text{ implements } f \text{ on } U} 2^{-|p|}$$

The Solomonoff universal prior has been well-studied in theoretical computer science [7]. This distribution has the characteristic that for any computable probability distribution P_C, $P_C(f)/P_U(f)$ is bound by a constant dependent on P_C but not on f, which is the origin of the name "universal prior". It has been argued that the Solomonoff universal prior is a formalization of Occam's Razor [9]. In the context of genetic algorithms, the Solomonoff universal prior has the desirable property that it assigns high probabilities to problems that have actually been studied in the GA literature (e.g. TSP, maximum clique), while assigning low probability to most of the functions that could only be described as random.

Theorem 4.1 lays the groundwork both for the remaining theorem proved in this section and for our discussion of infinite sets in Sect. 5. Theorem 4.2 shows that a No Free Lunch result does not apply to (F,P) if P is a Solomonoff universal prior.

Theorem 4.1. Let F be a set of functions, where each $f \in F$ has a finite (but not necessarily common) domain X_f and range Y_f, and let P be a probability distribution over F. A No Free Lunch result holds for (F,P) iff., for any $f \in F$ and any permutation g of f, $P(g)=P(f)$, where we define $P(g \notin F) \equiv 0$.

Proof: (IF) Suppose that for any $f \in F$ and any permutation g of f, $P(g)=P(f)$. The set F can thus be partitioned into subsets which are each closed under permutation and assigned uniform probability by P. By NFLP, a No Free Lunch result applies to each of these subsets. Thus, because the overall performance is simply the sum of the performance on each of these subsets, a No Free Lunch result applies to (F,P).

(ONLY IF) Suppose $P(g) \neq P(f)$ for some g that is a permutation of f. Let H denote the set of all functions that are permutations of f. Schumacher has shown that, when run against all functions in H, any search algorithm will obtain each possible performance vector exactly once [10]. Let A be a search algorithm, and let $V(A,f)$ denote the performance vector that A obtains against f. Let B be a search algorithm that, for all domains $X \neq X_f$, behaves identically to A (B can recognize such cases because the domain X is an input to the search algorithm). However, for $X=X_f$, let B be the search algorithm such that $V(B,g)=V(A,f)$. Such a B must exist because f and g are permutations of one another (and B can simply visit points in whatever order is necessary to obtain $V(A,f)$ when running against g). Furthermore, for all $h \in H$, $h \neq g$ we have $V(B,h) \neq V(A,f)$ because (as just mentioned), when running over all functions in H, B must obtain each performance vector exactly once.

Now consider the performance measure M that assigns a value of 1 to $V(A,f)$ and a value of 0 to all other performance vectors. Let X_i represent the domain of some arbitrary function i, and suppose we run both A and B on i. For $X_i \neq X_f$, A and B will be assigned the same performance because they are defined to perform identically in this case. For $Y_{mult}(i) \neq Y_{mult}(f)$, A and B will both be assigned 0 performance, because the performance vector $V(A,f)$ can only be obtained when the available multiset of y-values are those in $V(A,f)$. Thus the difference $M_O(f)-M_O(g)$ must depend only on those i for which $X_i=X_f$ and $Y_{mult}(i)=Y_{mult}(f)$ (i.e., only on those $i \in H$). When run over all $i \in H$, A will obtain $V(A,f)$ only on f while B obtains $V(A,f)$ only on g. Thus, over all $i \in H$, A will have performance $P(f)$ while B has performance $P(g)$. Thus the overall performance difference $M_O(f)-M_O(g)$ is precisely equal to $P(f)-P(g)$. Since we have hypothesized that $P(f) \neq P(g)$, it follows that $M_O(f) \neq M_O(g)$, so a No Free Lunch result does not hold for (F,P). This establishes the theorem. □

Theorem 4.1 can be seen as a generalization of NFLP simultaneously to non-uniform probability distributions, to sets of functions that do not have a common domain, and (as will be seen in Sect. 5) to infinite sets of functions. The only significant restriction we have placed on each $f \in F$ is that it have a finite domain. Thus, aside from this one restriction, Theorem 4.1 is the most general possible form of the No Free Lunch theorem.

Theorem 4.2. Let F and P be defined as in Theorem 4.1, let h be a hashing function, and let P_U be a Solomonoff universal prior. Then a No Free Lunch result does not hold for (F,P_U) so long as $K_U(h)<\lg(G(Y_{mult}(h)))$.

Proof: Let $k \equiv K_U(h)$. Because h has description length k, there must be at least one program of length k that computes h. Thus h must be assigned a probability of at least 2^{-k} under P_U. Since the sum of the probabilities P_U assigns to all functions in F must not exceed 1, there can be at most 2^k functions assigned probability $P_U(h)$. If a No Free Lunch Result holds for (F,P_U), then by Theorem 4.1 all functions that are permutations of h must be assigned probability $P_U(h)$. The number of such functions is $G(Y_{mult}(h))$. Thus a No Free Lunch result cannot hold so long as $2^k < G(Y_{mult}(h))$. □

A similar proof can be used to establish the same result for P that assign probability in a manner that is strictly decreasing w.r.t. description length.

5 No Free Lunch and Infinite Sets

The No Free Lunch theorems are commonly said to apply to "all possible functions". However, the original No Free Lunch theorems applied only to any finite set of functions $F \equiv Y^X$ having a specified finite domain X and specified finite codomain Y. In this section we discuss circumstances under which a No Free Lunch result will apply when F is infinite.

When F may be infinite, our terminology from Sect. 1 essentially still works, with two caveats. First, we must now distinguish between search functions and search algorithms. We define a search function $A{:}T,X \rightarrow X$ in the same way we defined a search algorithm in Sect. 1.1, and we now define a search algorithm as a computable search function. Second, we must require that the values returned by a performance measure M not be unbounded (i.e., there must always exists some C_M such that $M(v)<C_M$ for all v), so that the summation required to compute the overall performance M_O will always be guaranteed to converge.

The output of the performance measure M used in second half of the proof of Theorem 4.1 is certainly bounded, since it can only take on the values of 0 and 1. The only objection one might have to the assertion that Theorem 4.1 applies to infinite sets is that the proof of this theorem describes A and B as search algorithms, whereas using our above terminology they may only be search functions. However, it turns out that both A and B can be search algorithms. Because the choice of A in the proof of the theorem was arbitrary, A may be a search algorithm by hypothesis, whereas B is a search algorithm because it emulates A in all but a finite number of cases. Thus, Theorem 4.1 applies to infinite sets of functions.

When considering a finite set of functions $F \equiv Y^X$, a No Free Lunch result applies to (F,P) so long as P is uniform. Though we would argue against such a choice of P even in this case, such a choice of P can at least be justified under the grounds of "making no assumptions". However, when F is infinite, a uniform distribution is not possible, and a No Free Lunch result will apply to (F,P) only if P satisfies the partitioning criteria given in the statement of Theorem 4.1. It can be shown that these partitioning criteria are satisfied iff., for any $f \in F$, P can be fully specified as a function of the domain X_f and the multiset $Y_{mult}(f)$ (i.e., $P(f)=P_{XY}(X_f,Y_{mult}(f))$ for all $f \in F$, for some $P_{X,Y}$). What can be the argument for assuming the real world's P is of this form?

One could of course argue that memory limitations restrict X and Y to be finite in practice, but these limitations also restrict description length. And, as we have seen in Sect. 3, only very moderate bounds on description length are required to ensure that a No Free Lunch result does not apply to Y^X.

6 Limitations

The purpose of the proofs in this paper has been identify pairs (F,P) for which a No Free Lunch result does and does not hold. We must acknowledge, however, that the mere fact that a No Free Lunch result does not apply to some pair (F,P) does not guarantee that the development of black-box algorithms for (F,P) is worthwhile. All that is guaranteed is that, if a No Free Lunch result does not hold for (F,P), some search algorithms will perform better than *some* others in terms of *some* performance measure. We have not shown that any algorithms outperform either random or exhaustive search. Also, we have not shown that the performance measures are of the kind that typically would be used in optimization. We hope to address both of these limitations in future work.

7 Summary and Conclusions

We have presented two broad classes of functions for which a No Free Lunch result does not hold: functions of bounded description length, and functions whose probability is defined according to description length. We have argued that such probability distributions are more relevant to search and optimization than are the ones required by NFL. Finally, we have identified the circumstances under which a No Free Lunch result may apply to infinite sets of functions.

References

1. V.K. Balakrishnan. *Introductory Discrete Mathematics*. New York: Dover Publications; 1991.
2. S. Christensen and F. Oppacher. What can we learn from no free lunch? A first attempt to characterize the concept of a searchable function. In *Proc. 2001 Genetic and Evolutionary Computation Conf.*, 2001, pp. 1219–1226.
3. J. Culberson. On the futility of blind search. *Evolutionary Computation*, 6(2):109–127, 1999.
4. S. Droste, T. Jansen, and I. Wegener. Perhaps Not a Free Lunch But At Least a Free Appetizer. In *Proc. 1999 Genetic and Evolutionary Computation Conf.*, 1999, pp. 833–839.
5. J. Holland. *Adaptation in Natural and Artificial Systems*. Cambridge, MA: The MIT Press; 1992.
6. C. Igel and M. Toussaint. On classes of functions for which no free lunch results hold. Los Alamos e-Print Archive cs.NE/0108011. 2001.
7. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag; 1993.
8. N.J. Radcliffe and P.D. Surry. Fundamental limits on search algorithms: Evolutionary computing in perspective. In J. van Leeuwen, ed., *Lecture Notes in Computer Science 1000*. Springer-Verlag, 1996.
9. J. Schmidhuber. Discovering solutions with low Kolmogorov complexity and high generalization capability. In *Proc. 12th Intern. Conf. on Machine Learning*, 1995, pp. 488–496.
10. C. Schumacher, M.D. Vose, and L.D. Whitley. The no free lunch and problem description length. In *Proc. 2001 Genetic and Evolutionary Computation Conf.*, 2001, pp. 565–570.
11. D. Whitley. Functions as permutations: regarding no free lunch, walsh analysis and summary statistics. In Schoenauer et al., eds., *Parallel Problem Solving from Nature 6*, 2000, pp. 169–178.
12. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 4:67–82, 1997.