Optimisation, block designs and No Free Lunch theorems

Evan J. Griffiths, Pekka Orponen

Laboratory for Theoretical Computer Science, P. O. Box 5400
FI-02015 Helsinki University of Technology, Finland

Abstract

We study the precise conditions under which all optimisation strategies for a given family of finite functions yield the same expected maximisation performance, when averaged over a uniform distribution of the functions. In the case of bounded-length searches in a family of Boolean functions, we provide tight connections between such "No Free Lunch" conditions and the structure of t-designs and t-wise balanced designs for arbitrary values t. As a corollary, we obtain a nontrivial family of n-variate Boolean functions that satisfies the "No Free Lunch" condition with respect to searches of length $\Omega(n^{1/2}/\log^{1/2} n)$. Modifying the construction, we also obtain nontrivial "No Free Lunch" families of functions with large ranges.

Key words: Theory of computation, combinatorial problems, optimisation, No Free Lunch theorems, combinatorial designs

1 Introduction

In their seminal paper [20], Wolpert and Macready proved that when averaged over the uniform distribution on the family of all functions $f : X \to Y$, where X and Y are finite sets, all optimisation strategies have exactly the same expected performance. This Wolpert-Macready "No Free Lunch" (briefly, NFL) theorem has given rise to much controversy, but has turned out to be unexpectedly difficult to refute, in the sense of a comprehensive proof that with respect to some natural nonuniform distribution of functions, some practically interesting optimisation methods such as variants of local search [1] would

[1] Research supported by grants 81120/2000 (EG) and 204156/2003 (PO) from the Academy of Finland. *Authors' e-mail addresses:* `griffiths@member.ams.org` , `orponen@tcs.hut.fi`

perform better than, say, simple random sampling of the search space X. Partial results in this direction are derived in [3,6,7,17].

It was pointed out in [15] that to prove an NFL theorem it suffices for the family of functions under consideration to be closed under permutations of the domain X (briefly, c.u.p.). In [16] this result was strengthened to show that the c.u.p. condition in fact characterises precisely those function families that satisfy a "Strong NFL" property, requiring that all optimisation algorithms have the same expected payoff with respect to *any* performance measure computable from the execution trace of an algorithm. Some consequences of this characterisation were derived in [9], and the c.u.p. condition has been generalised to versions applicable also to nonuniform distributions of test functions in [8,17]. [2]

In this paper we take a careful look into the NFL conditions specifically for function *maximisation* using bounded-length searches. After formulating the proper definitions and presenting some examples we focus in Section 3 on maximising searches of length t in families of Boolean functions. As we shall see, in this setting the NFL condition is *not* equivalent to the simple c.u.p. property, but is rather intricately connected to notions of design theory [2,4,13]. In Section 4, we characterise Boolean function families that satisfy the NFL condition with respect to searches of length t in terms of *t-wise balanced designs.* Applying a deep result of Teirlinck [18], we then obtain for every n a nontrivial non-c.u.p. family of n-variate Boolean functions that is NFL with respect to maximising searches of length $\Omega(n^{1/2}/\log^{1/2} n)$. In Section 5 this construction is extended to cover also nonbinary functions. Section 6 concludes the paper with some general remarks and a list of open questions.

2 Background

Our general setting for optimisation problems comprises a fixed finite domain X and finite range Y, with functions $f : X \to Y$ belonging to a finite family F. An *algorithm* A is a deterministic mechanism that initially chooses a start point x_1 in the domain X, and is provided with the value $f(x_1)$ of the unknown function at that point. After A is provided with each function value $f(x_i)$ it chooses a new domain element x_{i+1} based on its known domain-range pairs $(x_j, f(x_j))$, $1 \leq j \leq i$. We are concerned exclusively with algorithms that never repeat a domain element and always provide a choice of x_{i+1} up to $i + 1 = |X|$.

[2] An alternate viewpoint on the c.u.p. condition and its relevance to (anti-)NFL arguments is presented in [11]. See also [5] for multiobjective optimisation settings where the NFL results arguably do not hold.

The *execution trace* of algorithm A on function f is the sequence of domain-range pairs A produces on f. The corresponding *range trace* (called "performance vector" in [16]), denoted $\eta(A, f)$, consists of just the respective range elements. A *performance measure* P is any function mapping range traces to real numbers.

Definition 1 *The (expected)* ***performance*** P_A *of algorithm* A *with respect to measure* P *over the family of functions* F *is defined as:*

$$P_A = \frac{1}{|F|} \sum_{f \in F} P(\eta(A, f)).$$

Wolpert and Macready [20] defined a notion of No Free Lunch which we can re-state in the current formalism:

Definition 2 *A family of functions* F *has the* ***Strong No Free Lunch*** *(briefly, SNFL) property if every pair of algorithms* A, B *have the same performance* $P_A = P_B$ *over* F, *with respect to any performance measure* P *and any trace length* $1 \leq t \leq |X|$.

In other words, when averaged over all functions $f \in F$, no algorithm performs better than any other.

Theorem 1 (Wolpert and Macready [20]) *The family of all functions from a (finite) domain* X *to a (finite) range* Y *has the SNFL property.*

The SNFL condition has recently been characterised precisely:

Definition 3 *A family of functions* F *is* ***closed under permutations (c.u.p.)*** *if for every* $f \in F$ *and every permutation* σ *of* X, *the function* g *defined by* $g(x) = f(\sigma(x))$ *is also a member of* F.

Theorem 2 (Schumacher, Vose, Whitley [16]) *A family of functions has the SNFL property if and only if it is closed under permutations.*

3 Uniform sets of functions

For the rest of this paper, we assume that our domains Y are subsets of natural numbers (or some other linearly ordered set), and restrict our attention to the maximisation performance measure.

Definition 4 *The (expected)* ***maximisation performance*** *of algorithm* A

on function family F is defined as:

$$M_A = \frac{1}{|F|} \sum_{f \in F} \max\{\tilde{\eta}(A, f))\},$$

where $\tilde{\eta}(A, f)$ is the set of elements contained in the range trace of A on f.

Definition 5 *A family of functions F is t-**uniform**, denoted U_t, if all algorithms working for t steps produce the same maximisation performance $M_A = m_{F,t}$, where the constant $m_{F,t}$ depends only on F and t.*

It is easy to see that closure under permutations is not necessary for an NFL (or U_t) condition to hold with respect to maximisation performance. Consider, e.g. the family of three binary-valued functions on a four element domain, with function values 0,1,1,1 arranged in any three of the four possible ways on the domain. This family is clearly not c.u.p., yet maximisation in two steps is achieved equally well by all algorithms; thus the family is U_2. (Note, however, that it is not U_1.) For more interesting examples, consider the following two constructions.

Example 1 *The family F of functions is obtained by taking an 8×8 Sylvester-type Hadamard matrix [13, p. 202], and deleting the first row and first column (which consist entirely of ones).*

X	f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	*0*	*1*	*0*	*1*	*0*	*1*	*0*
2	*1*	*0*	*0*	*1*	*1*	*0*	*0*
3	*0*	*0*	*1*	*1*	*0*	*0*	*1*
4	*1*	*1*	*1*	*0*	*0*	*0*	*0*
5	*0*	*1*	*0*	*0*	*1*	*0*	*1*
6	*1*	*0*	*0*	*0*	*0*	*1*	*1*
7	*0*	*0*	*1*	*0*	*1*	*1*	*0*

It is easy to verify that this family F is U_1. Furthermore, since in the underlying Hadamard matrix any two rows agree and disagree at equally many locations, the family is U_2. More specifically, for any two domain elements, the number of functions in Fwhich yield zero at both points is the same – in this case two. However, F is not U_3, as the strategy of choosing domain elements 5, 6 and 7 may produce all zeros, whereas choosing 1, 2 and 3 always guarantees a one.

Example 2 *Consider the set of all functions f_i from Example 1 and their complements $1 - f_i$:*

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	..
0	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	..
1	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*1*	..
0	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*1*	..
1	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	..
0	*1*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	..
1	*0*	*0*	*0*	*0*	*1*	*1*	*0*	*1*	..
0	*0*	*1*	*0*	*1*	*1*	*0*	*1*	*1*	..

This set of functions is U_3, as well as U_1 and U_2, even though the two families of 7 functions of which it is composed are separately U_2 but not U_3.[3]

4 Binary valued functions and block designs

We now develop a precise characterisation of families of binary valued functions which exhibit the property U_t, for all $t \geq 1$.

Definition 6 *An* ***oblivious search*** *is an algorithm which does not use information about function values already obtained when choosing its next domain element.*

We denote by obU_t the property of function families being uniform for oblivious searches of length t. Naturally U_t implies obU_t. We use U_t^* to denote the conjunction of the properties $U_1, U_2, \ldots, U_t$, and similarly obU_t^* for the conjunction of all obU_s, $s \leq t$.

Proposition 3 *For binary valued functions and all $t \geq 1$, $obU_t = U_t$.*

Proof Any non-oblivious algorithm A can be emulated by an oblivious algorithm B which does not look at function values obtained but merely follows the A-strategy *assuming* that all function values obtained are zero. These two strategies will not diverge in their choice until after the first 1 is obtained, so $M_A = M_B$. $\square$

Corollary 4 $obU_t^* = U_t^*$.

[3] This is due to the general fact that the union of a "regular 2-wise balanced design" and its complement is a "regular 3-wise balanced design", an observation attributed in [10] to R. C. Mullin [14]. These terms are defined in the next section.

Thus, in order to show that an NFL result concerning maximisation of binary valued functions holds, it suffices to consider oblivious searches, i.e. preselections of t domain elements. We shall be using the following basic notions and results from design theory (cf. [2,4,12,13]).

Definition 7 *A t-**design** consists of a set X of **points**, and a set B of subsets of X called **blocks**, satisfying the following conditions: (i) each block contains the same number k of points, and (ii) for any set T of t points from X the number of blocks containing T is a constant, denoted λ.*

Denoting $v = |X|$, the pair (X, B) as defined above is then a t-design on v points with block size k and index λ, or a t-(v, k, λ) design. We require $v \geq k \geq t \geq 0$ and $\lambda \geq 1$. A *trivial* t-design consists of all the t-subsets of a given point set X. The following is a standard result [13, p. 219]:

Proposition 5 *For all $s \leq t$, a t-(v, k, λ) design is also an s-(v, k, λ_s) design, where $\lambda_s = \lambda\binom{v-s}{t-s}/\binom{k-s}{t-s}$.*

There is a generalisation to the situation where blocks are not necessarily the same size, using a finite set of positive integers K rather than a single integer k.

Definition 8 *A t-**wise balanced design** (tBD) of type $t - (v, K, \lambda)$ is a pair (X, B) with $v = |X|$, such that the size of every block is in K and every t-element subset of X is contained in exactly λ blocks.*

In the case where $|K| > 1$ a tBD is not always an sBD for parameters $s < t$, suggesting the following notion:

Definition 9 (Kageyama [10]) *A **regular** t-wise balanced design is a tBD which is also an sBD for all $s < t$.*

To connect maximisation problems on binary valued functions to design theory we represent a family F of binary functions as a bipartite graph in the following way. Elements of X are nodes on the left, functions in F are nodes on the right, and an edge (x, f) is present if and only if $f(x) = 0$. By representing the zeros of each function this graph represents all of F.

Correspondingly, a t-design or a tBD yields a bipartite graph by representing the domain elements (v of them) as nodes on the left, the blocks as nodes on the right, and having a connection left to right iff the relevant point is in the specified block. This graph immediately converts to a collection of functions F as described above, with $|X| = v$ and $|F|$ being the number of blocks.

In the following theorems we consider this as the standard correspondence for translating between Boolean functions and block designs.

Theorem 6 *A family of (binary-valued) functions F satisfies U_t, if and only if its bipartite graph representation corresponds to a t-wise balanced design.*

Proof If we perform an oblivious search of length t, i.e. select any set of t points, the t-uniformity of F guarantees that a (uniformly) randomly selected function is zero at all t points with a constant probability, say ρ. The functions of F can be partitioned into a finite number of classes depending on how many zeros they have; let the different numbers of zeros be $k_1 < \ldots < k_n$. Now if we set $K = \{k_1, \ldots, k_n\}$, and notice that every set of t points on the left of the bipartite graph are joined in common to $\lambda = |F|\rho$ blocks (functions) on the right, we have a tBD of type $t - (|X|, K, \lambda)$. The converse claim is similarly obvious. $\Box$

Corollary 7 *From a $t - (v, k, \lambda)$ design, $k \geq t$, we obtain a U_t^* family of functions.*

Proof A t-design is a t-wise balanced design, and hence yields a U_t family of functions. Since a $t - (v, k, \lambda)$ design is also an $s - (v, k, \lambda_s)$ design for all $s \leq t$, the result follows. $\Box$

We now summarise the above results, abusing the notation somewhat to let U_t denote not only the t-uniform property but also the set of bipartite graphs which have this property.

Corollary 8 *t-designs $\subsetneq$ regular tBDs $= U_t^* = obU_t^* \subsetneq$ t-wise BDs $= U_t = obU_t$.*

The inclusions in Corollary 8 are proper: Example 2 presents a family of functions that is U_t^* for $t = 3$ but is not a t-design, and the following construction yields for any $t \geq 2$ a tBD that is not U_t^* [4, p. 485].

Example 3 *Fix an integer $n > t$. Let X be the set $\{1, 2, ...n, \infty\}$ and let $B = \{\{1, 2, \ldots, n\}\} \cup \{S \cup \{\infty\} : S$ is a $(t-1)$-subset of $\{1, \ldots, n\}\}$. This is a tBD, with $K = \{t, n\}$ and $\lambda = 1$, which is not an sBD for any s, $0 < s < t$.*

The theory of what kinds of t-wise balanced designs, regular tBDs, and tBDs exist is extensive and complex, and by no means complete. By a fundamental result of L. Teirlinck [18,19], it is known that non-trivial t-designs exist for all t. One consequence of his work can be stated as follows.

Theorem 9 (Teirlinck) *For all positive integers t there exists a non-trivial t-design with block size $(t + 1)$ and number of points $v = 2t + (t + 1)!^{(2t+1)}$*

Thus for any search length t, there exists a nontrivial U_t^* family of Boolean functions on $n = O(\log v) = O(t^2 \log t)$ variables. Conversely, the search length t as a function of n is $\Omega(n^{1/2}/\log^{1/2} n)$. The number of functions in this family equals the number of blocks in the respective design, $b = \lambda\binom{v}{t}/\binom{k}{t}$, which in this case can be bounded as $(t!)^{2t^2+O(t)} = 2^{O(t^3 \log t)}$. Note that the total number of Boolean functions on n variables is $2^{2^n} = 2^{2^{O(t^2 \log t)}}$ in this case.

5 Extension to arbitrary ranges

It is possible to see the behaviour of uniform binary valued functions reflected in ranges of arbitrary size by dilating the function values 1 to some large constant c, and then judiciously replacing the 0 function values with various combinations of values from $[0, c-1]$. For example, based on a $2k \times 2k$ Sylvester-type Hadamard matrix, one can map every function value 1 to $(k+1)$, and replace the k zeros in each function with all possible permutations of the values $1, 2, \ldots, k$. This yields a non-c.u.p. U_2^* family of functions with range $\{1, 2, \ldots, k+1\}$, at the cost of increasing the size of the family from $(2k-1)$ to $(2k-1) \cdot k!$.

We can combine in this manner any family of functions F, such that all $f \in F$ have precisely k zeros, with any family of functions G on a domain of size k, producing a family of functions H of size $|H| = |F||G|$. If there exists a constant c such that for all $f \in F$, the (binary) range of f is $\{0, c\}$, and all functions in G have ranges contained in the interval $[0, c-1]$, then we call the family of functions H thus obtained a *stratified composition* of F and G.

Theorem 10 *If H is a stratified composition of F and G, F is U_t^* and G is c.u.p., then H is U_t^*.*

Proof It is implicit in the above that $t \leq k$. Let c be the maximal value for all $f \in F$, and let X be the domain of functions in F and H. Let R be the union of the ranges of all $g \in G$. Initially, regardless of the choice of algorithm A and its associated initial point $x_1 \in X$, the probability[4] of immediately obtaining a function value of c is a constant p_1 depending only on the function family H (this is because F is U_1). The probability of getting any specific value $f(x_1) = r \in R$ is independent of the choice of x_1 as G is c.u.p. (here we don't know which domain element $z_1 \in [1, k]$ from the process of stratified composition is represented by x_1, but the conclusion holds in any case). Conditioning on $f(x_1) = r_1 < c$, the probability p_2 that the second domain point x_2 satisfies $f(x_2) = c$ is again independent of A (as F is U_2), and the probability of obtaining any element $r \in R$ is the same for any A and independent of the choice of x_2 given that it doesn't yield a c.

[4] We adopt here a probabilistic point of view, considering H as a sample space from which the unknown function h is selected, according to a uniform probability distribution.

Continuing in this manner we obtain a sequence of probabilities $p_1, \ldots, p_t$. For any algorithm A the probability p_c of producing a maximum of c is a simple function of $p_1, \ldots, p_t$. We note that among the traces produced by any algorithm, those containing no value c may have any sequence of the values of R (that is, any sequence that can be produced in a search in G of length t). Thus the probability of any maximum less than c occurring is independent of the algorithm, as is the probability that c is the maximum. $\square$

In fact the condition that G is c.u.p. is unnecessarily strong for the above proof, and can be replaced by the following: [5] define a family of functions G to be *strongly* U_t^* if G is U_t^*, and also for any $s \leq t$ and for any ordered s-tuple $y_1, ..., y_s$ of elements of Y, the number of functions in G mapping x_1 to y_1, x_2 to y_2, $\ldots$, and x_s to y_s is independent of the choice of the ordered s-tuple $x_1, ..., x_s$ of elements of X. It is sufficient in a stratified composition for G to be strongly U_t^* rather than c.u.p. for Theorem 10to hold.

Iteration of stratified compositions is possible, so that a variety of non-c.u.p. U_t^* function families of functions with large ranges can be produced.

6 Conclusion and open problems

We have shown that the often-quoted correspondence between NFL and c.u.p. conditions for function families breaks down when one focuses on the maximisation performance of algorithms over bounded-length searches. At this level, the picture is much more intricate, as we have illustrated by characterising simple NFL function families in terms of t-wise balanced designs, and pointing out that consequently also highly nontrivial non-c.u.p. such families exist. Whether these Teirlinck families can also be used for practical purposes, e.g. as a basis for guaranteed "deceptive" test problems for search algorithms, remains to be seen. (Such use would require at least designing an efficient procedure to answer queries of the form "determine the value of Teirlinck function i at domain element j". It is not immediately clear from Teirlinck's construction [18,19] whether such a procedure is feasible.)

A number of fundamental problems for further research are suggested by this new connection between search and designs, including the following: (1) Characterise the families of functions which are U_t^*, but do not correspond to t-designs. (2) Give a characterisation of t-uniform families of functions of *arbitrary range.* (3) Uniform families of functions are not amenable to any kind of search method, local or global. What concepts of uniformity would arise

[5] We are grateful to an anonymous referee for pointing out an error in an earlier version of the proof, and providing us with this condition.

from considering only *local* search algorithms? (4) What can be said about *arbitrary measures* on short searches (short traces η), as opposed to the exhaustive searches underlying the Strong NFL results?

References

[1] Aarts, E., Lenstra, J. K. *Local Search in Combinatorial Optimization.* John Wiley & Sons, Chichester, 1997.

[2] Beth, T., Jungnickel, D., Lenz, H. *Design Theory, 2nd Ed.* Cambridge University Press, 1999.

[3] Christensen, S., Oppacher, F. "What can we learn from No Free Lunch? A first attempt to characterize the concept of a searchable function." *Proc. Genetic and Evolutionary Computation Conference GECCO-2001*, 1219–1226. Morgan Kaufmann, San Francisco CA, 2001.

[4] Colbourn, C., Dinitz, J. (Eds.) *The CRC Handbook of Combinatorial Designs.* CRC Press, Boca Raton FL, 1996.

[5] Corne, D., Knowles, J. "Some multiobjective optimizers are better than others." *Proc. 2003 Congress on Evolutionary Computation (CEC-2003)*, 2506–2512. IEEE, New York NY, 2003.

[6] Droste, S., Jansen, T., Wegener, I. "Perhaps not a free lunch but at least a free appetizer." *Proc. Genetic and Evolutionary Computation Conference GECCO-1999*, 833–839. Morgan Kaufmann, San Francisco CA, 1999.

[7] Droste, S., Jansen, T., Wegener, I. "Optimization with randomized search heuristics—the (A)NFL theorem, realistic scenarios, and difficult functions." *Theoretical Computer Science 287* (2002), 131–144.

[8] English, T. "No more lunch: Analysis of sequential search." *Proc. 2004 Congress on Evolutionary Computation (CEC-2004)*, 227–234. IEEE, New York NY, 2004.

[9] Igel, C., Toussaint, M. "On classes of functions for which No Free Lunch results hold." *Information Processing Letters 86* (2003), 317–321.

[10] Kageyama, S. "A property of t-wise Balanced Designs." *Ars Combinatoria 31* (1991), 237–238.

[11] Köppen, M. "No-Free-Lunch theoremss and the diversity of algorithms." *Proc. 2004 Congress on Evolutionary Computation (CEC-2004)*, 235–241. IEEE, New York NY, 2004.

[12] Kramer, E. S. "Some results on t-wise Balanced Designs *Ars Combinatoria 15* (1983), 179–192.

[13] Van Lint, J. H., Wilson, R. M. *A Course in Combinatorics, 2nd Ed.* Cambridge University Press, 2001.

[14] Mullin, R. C. “A note on self-complementary designs.” *Congressus Numerantium 10* (1974), 591–598.

[15] Radcliffe, N. J., Surry, P. D. “Fundamental limitations on search algorithms: evolutionary computing in perspective.” *Computer Science Today: Recent Trends and Developments* (ed. J. van Leeuwen), 275–291. Lecture Notes in Computer Science 1000. Springer-Verlag, Berlin, 1995.

[16] Schumacher, C., Vose, M.D., Whitley, L.D. “The No Free Lunch and problem description length.” *Proc. Genetic and Evolutionary Computation Conference GECCO-2001*, 565–570. Morgan Kaufmann, San Francisco CA, 2001.

[17] Streeter, M. J. “Two broad classes of functions for which a No Free Lunch result does not hold.” *Proc. Genetic and Evolutionary Computation Conference GECCO-2003*, 1418–1430. Morgan Kaufmann, San Francisco CA, 2003.

[18] Teirlinck, L. “Non-trivial t-designs without repeated blocks exist for all t.” *Discrete Math. 65* (1987), 301–311.

[19] Teirlinck, L. “Locally trivial t-designs and t-designs without repeated blocks.” *Discrete Math. 77* (1989), 345–356.

[20] Wolpert, D. H., Macready, W. G. “No Free Lunch theorems for optimization.” *IEEE Trans. on Evolutionary Computation 1* (1997), 67–82.