How to get a Free Lunch (at no cost)

John R. Rice  
*Purdue University, jrr@cs.purdue.edu*

Naren Ramakrishnan

Report Number:  
99-014

https://docs.lib.purdue.edu/cstech/1445

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
HOW TO GET A FREE LUNCH (AT NO COST)

John R. Rice
Naren Ramakrishnan

CSD-TR #99-014
April 1999
How to get a free lunch (at no cost)

John R. Rice and Naren Ramakrishnan

March 25, 1999

Abstract

In this brief research note, we show that the conclusions presented in [20] are incorrect and contradict real world knowledge about optimization problems and algorithms. We argue for a practical viewpoint in the methodological analysis of algorithms.

1 Introduction

In [20], Wolpert and Macready present an analysis where they explore ‘no-free-lunch’ (NFL) theorems in the context of modeling the connection between optimization algorithms and problem instances. Specifically, they construct a mathematical model for which they show that the selective superiority of an algorithm $A$ for a set of problem instances is offset by a degradation of performance over another set of problem instances. In addition, a geometric model is proposed to identify how ‘aligned’ a given algorithm is to a problem instance. Interestingly, they also emphasize that their analysis allows certain types of ‘distinctions’ to exist among algorithm instances, notwithstanding the results of the NFL theorems. A similar analysis in the context of classification algorithms is presented in [16].

There has been recent widespread interest in formalizing these and similar notions. In particular, machine learning techniques have matured over the past few years and researchers have started addressing the important issue of automating problem decomposition and technique selection in the context of specific applications. The recent AAAI/ICML workshop on ‘The Methodology of Applying Machine Learning (Problem Definition, Task Decomposition and Technique Selection)’ [3] provides interesting application experiences and case studies in selecting techniques for machine learning.
One of the early results from this work has been the realization that NFL theorems make artificial assumptions that are not necessarily true in real-world situations. For instance, Rao et al. [12] show that they do not apply when all classification problems are not equally likely. Domingos [2] provides a cost model that determines the economics of 'fielding' a given algorithm and uses it to show that NFL theorems do not apply under 'realistic situations'. Our perspective here is to argue towards a practical viewpoint in such methodological research. For instance, several of the assumptions made in work such as [16, 20] contradicts knowledge of real world optimization and algorithms. Specifically,

- A uniform distribution assumption on the problem space implies that we could not solve real world problems of any size because only exhaustive search works for unstructured problems. In fact, the result in [20] could be established more easily by observing that the number of unstructured optimization problems totally dominates the number of structured problems. In mathematical analysis terms, this is expressed by saying that the structured $f$'s are of measure zero in the set of $f$'s. The inapplicability of the results in [20] to real world optimization follows by observing that all the real world optimization problems are structured.

- The authors also conclude that experimental comparisons of a particular algorithm with specific settings on a select sample problems are of limited utility from the fact 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. These statements are incorrect. While generalizing to larger spaces could be definitely misleading, experimental comparisons are becoming increasingly validated by well accepted statistical methodologies. For example, see [18]. This also implies that it is incorrect to assert that 'there is no apriori justification for a search algorithm's observed behavior to date on a particular cost function to predicts its future behavior' [20].

- The conclusion that the probability distribution $P(f)$ (over the optimization problem space) is 'effectively uniform', is grounded on the
assumption that the designer or practitioner of an optimization algorithm does not incorporate knowledge of problem characteristics into the algorithm. It is commonly observed that researchers repeatedly 'tune' and specialize their algorithms in order that they perform well on specific problem instances. Thus, this is an extremely difficult assumption to verify and even if true, does not adequately support its conclusion.

- In real world optimization, structure exists and is indeed exploited (often cleverly) by the choice of a specific algorithm. The remark that even if structure exists (but not a 'specification') is intended to suggest that some analytical or mathematical basis is required. This is incorrect and one merely needs to know that algorithm A outperforms algorithm B on a specific class and that the given \( f \) is in this class.

- Theorem 1 in [20] and the subsequent discussion that it can be extended to nonuniform priors is wrong. Counter examples to it are easily generated. Choosing a single \( f \) is one-way i.e., \( P(f^*) = 1 \). \( P(f / f \neq f^*) = 0 \). A more illustrative case study considers choosing \( f \)'s for Traveling Salesperson problems. Call this class TSP and partition it into subclasses by the number of cities \( \rightarrow C_N, N = 1, 2, \ldots \text{max} \). Choose \( f \) from class \( C_N \) with probability \( 2^{-N} \). Choose \( f \) from outside TSP with probability zero.

- The claim that real-world algorithms do not 'remember their history' and are wasteful of function evaluations is incorrect and is used, moreover, to simplify the mathematical analysis in [20]. Memory bounded algorithms [6, 15] and tabu search algorithms are becoming increasingly important in mainstream operations research. This follows the tradition set in [5] whose original motivation was to optimize the use of words in computer memory (Early computers had a few kilobytes of RAM).

- It is asserted in [20] that all algorithms implemented on computers are 'essentially deterministic'. Even though this is irrelevant to the content of this paper, it is completely untrue.
Figure 1: Schematic representation of the algorithm selection problem.

2 Algorithm Selection Systems

The first author of this paper was the first to seriously formulate and analyze the algorithm selection problem over 20 years ago [13, 14]. An abstract methodology for this problem was also presented in [14] and is reproduced in Fig. 1, where \( p \) is the problem given and \( w \) are the performance criteria. The problem \( p \) is 'represented' by the feature(s) \( f \) in the feature space. The task is to 'construct' a selection mapping \( S \) that provides a good algorithm \( A \) to solve \( p \) (where "good" is measured by \( w \)). We therefore need a 'means' to determine a 'good enough' algorithm \( A \) subject to the constraint that the \( \mu = \delta(A, p) \) value (the performance of algorithm \( A \) on \( p \)) is "optimized" (satisfies the constraints \( w \) to the 'best extent'). The 'best' selection is then the mapping that is better (in the sense of producing a better performance indicator in the performance measure space) than other possible mappings. Though this formulation caters to a wide variety of problems, methodologies have been developed with specific reference to performance evaluation in several domains such as numerical software, search algorithms, constraint satisfaction and database tuning.

In each of the above mentioned domains, algorithm selection systems (sometimes referred to as recommender systems) are becoming increasingly popular. For example, the mathematical software community has witnessed recent widespread research into this problem for several domains such as partial differential equations [19], ordinary differential equations [4] and numerical quadrature [10, 11]. The *Mathematica* software environment incorporates a dynamic 'algorithm switching' system for the solution of initial value problems. For example, when the nature of the differential equation changes from
non-stiff to stiff, the LSODE algorithm [9] dynamically switches to a different family of methods suited for these problems. For a number-theoretic example from the domain of integer factoring, see [17]. The Microsoft SQL Server software incorporates a ‘database tuning wizard’ that adopts a greedy approach [1] to perform ‘index selection’ customized to its user’s workload requirements. Such physical database tuning is one of the most complex problems in database design. Currently, this software automatically recommends a set of indices given a workload consisting of a set of SQL statements. A technique to automatically select branch-and-bound algorithms is presented in [7] and a related one to automatically specialize constraint satisfaction algorithms is outlined in [8].

References


