No Free Lunch Theorems for Optimization

David H. Wolpert and William G. Macready

***Abstract*—A framework is developed to explore the connection between effective optimization algorithms and the problems they are solving. A number of "no free lunch" (NFL) theorems are presented which establish that for any algorithm, any elevated performance over one class of problems is offset by performance over another class. These theorems result in a geometric interpretation of what it means for an algorithm to be well suited to an optimization problem. Applications of the NFL theorems to information-theoretic aspects of optimization and benchmark measures of performance are also presented. Other issues addressed include time-varying optimization problems and *a priori* "head-to-head" minimax distinctions between optimization algorithms, distinctions that result despite the NFL theorems' enforcing of a type of uniformity over all algorithms.**

***Index Terms*— Evolutionary algorithms, information theory, optimization.**

I. Introduction

THE past few decades have seen an increased interest in general-purpose "black-box" optimization algorithms that exploit limited knowledge concerning the optimization problem on which they are run. In large part these algorithms have drawn inspiration from optimization processes that occur in nature. In particular, the two most popular black-box optimization strategies, evolutionary algorithms [1]–[3] and simulated annealing [4], mimic processes in natural selection and statistical mechanics, respectively.

In light of this interest in general-purpose optimization algorithms, it has become important to understand the relationship between how well an algorithm a performs and the optimization problem f on which it is run. In this paper we present a formal analysis that contributes toward such an understanding by addressing questions like the following: given the abundance of black-box optimization algorithms and of optimization problems, how can we best match algorithms to problems (i.e., how best can we relax the black-box nature of the algorithms and have them exploit some knowledge concerning the optimization problem)? In particular, while serious optimization practitioners almost always perform such matching, it is usually on a heuristic basis; can such matching be formally analyzed? More generally, what is the underlying mathematical "skeleton" of optimization theory before the "flesh" of the probability distributions of a particular context and set of optimization problems are imposed? What can information theory and Bayesian analysis contribute to an understanding of these issues? How *a priori* generalizable are the performance results of a certain algorithm on a certain class of problems to its performance on other classes of problems? How should we even measure such generalization? How should we assess the performance of algorithms on problems so that we may programmatically compare those algorithms?

Broadly speaking, we take two approaches to these questions. First, we investigate what *a priori* restrictions there are on the performance of one or more algorithms as one runs over the set of all optimization problems. Our second approach is to instead focus on a particular problem and consider the effects of running over all algorithms. In the current paper we present results from both types of analyses but concentrate largely on the first approach. The reader is referred to the companion paper [5] for more types of analysis involving the second approach.

We begin in Section II by introducing the necessary notation. Also discussed in this section is the model of computation we adopt, its limitations, and the reasons we chose it.

One might expect that there are pairs of search algorithms A and B such that A performs better than B on average, even if B sometimes outperforms A. As an example, one might expect that hill climbing usually outperforms hill descending if one's goal is to find a maximum of the cost function. One might also expect it would outperform a random search in such a context.

One of the main results of this paper is that such expectations are incorrect. We prove two "no free lunch" (NFL) theorems in Section III that demonstrate this and more generally illuminate the connection between algorithms and problems. Roughly speaking, we show that for both static and time-dependent optimization problems, the average performance of any pair of algorithms across all possible problems is identical. This means in particular that if some algorithm a_1's performance is superior to that of another algorithm a_2 over some set of optimization problems, then the reverse must be true over the set of all other optimization problems. (The reader is urged to read this section carefully for a precise statement of these theorems.) This is true even if one of the algorithms is random; any algorithm a_1 performs worse than randomly just as readily (over the set of all optimization problems) as it performs better than randomly. Possible objections to these results are addressed in Sections III-A and III-B.

In Section IV we present a geometric interpretation of the NFL theorems. In particular, we show that an algorithm's average performance is determined by how "aligned" it is with the underlying probability distribution over optimization problems on which it is run. This section is critical for an

Manuscript received August 15, 1996; revised December 30, 1996. This work was supported by the Santa Fe Institute and TXN Inc.

D. H. Wolpert is with IBM Almaden Research Center, San Jose, CA 95120-6099 USA.

W. G. Macready was with Santa Fe Institute, Santa Fe, NM 87501 USA. He is now with IBM Almaden Research Center, San Jose, CA 95120-6099 USA.

Publisher Item Identifier S 1089-778X(97)03422-X.

understanding of how the NFL results are consistent with the well-accepted fact that many search algorithms that do not take into account knowledge concerning the cost function work well in practice.

Section V-A demonstrates that the NFL theorems allow one to answer a number of what would otherwise seem to be intractable questions. The implications of these answers for measures of algorithm performance and of how best to compare optimization algorithms are explored in Section V-B.

In Section VI we discuss some of the ways in which, despite the NFL theorems, algorithms can have *a priori* distinctions that hold even if nothing is specified concerning the optimization problems. In particular, we show that there can be "head-to-head" minimax distinctions between a pair of algorithms, i.e., that when considering one function at a time, a pair of algorithms may be distinguishable, even if they are not when one looks over all functions.

In Section VII we present an introduction to the alternative approach to the formal analysis of optimization in which problems are held fixed and one looks at properties across the space of algorithms. Since these results hold in general, they hold for any and all optimization problems and thus are independent of the types of problems one is more or less likely to encounter in the real world. In particular, these results show that there is no *a priori* justification for using a search algorithm's observed behavior to date on a particular cost function to predict its future behavior on that function. In fact when choosing between algorithms based on their observed performance it does not suffice to make an assumption about the cost function; some (currently poorly understood) assumptions are also being made about how the algorithms in question are related to each other and to the cost function. In addition to presenting results not found in [5], this section serves as an introduction to the perspective adopted in [5].

We conclude in Section VIII with a brief discussion, a summary of results, and a short list of open problems.

We have confined all proofs to appendixes to facilitate the flow of the paper. A more detailed, and substantially longer, version of this paper, a version that also analyzes some issues not addressed in this paper, can be found in [6].

II. Preliminaries

We restrict attention to combinatorial optimization in which the search space

searched points, our definition of an algorithm includes all common black-box optimization techniques like simulated annealing and evolutionary algorithms. (Techniques like branch and bound [9] are not included since they rely explicitly on the cost structure of partial solutions.)

As defined above, a search algorithm is deterministic; every sample maps to a unique new point. Of course, essentially, all algorithms implemented on computers are deterministic,[1] and in this our definition is not restrictive. Nonetheless, it is worth noting that all of our results are extensible to nondeterministic algorithms, where the new point is chosen stochastically from the set of unvisited points. This point is returned to later.

Under the oracle-based model of computation any measure of the performance of an algorithm after m iterations is a function of the sample d_m^y. Such performance measures will be indicated by $\Phi(d_m^y)$. As an example, if we are trying to find a minimum of f, then a reasonable measure of the performance of a might be the value of the lowest $\mathcal{Y}$ value in d_m^y: $\Phi(d_m^y) = \min_i\{d_m^y(i) : i = 1 \cdots m\}$. Note that measures of performance based on factors other than d_m^y (e.g., wall clock time) are outside the scope of our results.

We shall cast all of our results in terms of probability theory. We do so for three reasons. First, it allows simple generalization of our results to stochastic algorithms. Second, even when the setting is deterministic, probability theory provides a simple consistent framework in which to carry out proofs. The third reason for using probability theory is perhaps the most interesting. A crucial factor in the probabilistic framework is the distribution $P(f) = P(f(x_1), \cdots, f(x_{|\mathcal{X}|}))$. This distribution, defined over $\mathcal{F}$, gives the probability that each $f \in \mathcal{F}$ is the actual optimization problem at hand. An approach based on this distribution has the immediate advantage that often knowledge of a problem is statistical in nature and this information may be easily encodable in $P(f)$. For example, Markov or Gibbs random field descriptions [10] of families of optimization problems express $P(f)$ exactly.

Exploiting $P(f)$, however, also has advantages even when we are presented with a single uniquely specified cost function. One such advantage is the fact that although it may be fully specified, many aspects of the cost function are *effectively* unknown (e.g., we certainly do not know the extrema of the function). It is in many ways most appropriate to have this effective ignorance reflected in the analysis as a probability distribution. More generally, optimization practitioners usually act as though the cost function is partially unknown, in that the same algorithm is used for all cost functions in a class of such functions (e.g., in the class of all traveling salesman problems having certain characteristics). In so doing, the practitioner implicitly acknowledges that distinctions between the cost functions in that class are irrelevant or at least unexploitable. In this sense, even though we are presented with a single particular problem from that class, we act as though we are presented with a probability distribution over cost functions, a distribution that is nonzero only for members of that class of cost functions. $P(f)$ is thus a prior specification of the class of the optimization problem at hand, with different classes of problems corresponding to different choices of what algorithms we will use, and giving rise to different distributions $P(f)$.

Given our choice to use probability theory, the performance of an algorithm a iterated m times on a cost function f is measured with $P(d_m^y|f, m, a)$. This is the conditional probability of obtaining a particular sample d_m under the stated conditions. From $P(d_m^y|f, m, a)$ performance measures $\Phi(d_m^y)$ can be found easily.

In the next section we analyze $P(d_m^y|f, m, a)$ and in particular how it varies with the algorithm a. Before proceeding with that analysis, however, it is worth briefly noting that there are other formal approaches to the issues investigated in this paper. Perhaps the most prominent of these is the field of computational complexity. Unlike the approach taken in this paper, computational complexity largely ignores the statistical nature of search and concentrates instead on computational issues. Much, though by no means all, of computational complexity is concerned with physically unrealizable computational devices (e.g., Turing machines) and the worst-case resource usage required to find optimal solutions. In contrast, the analysis in this paper does not concern itself with the computational engine used by the search algorithm, but rather concentrates exclusively on the underlying statistical nature of the search problem. The current probabilistic approach is complimentary to computational complexity. Future work involves combining our analysis of the statistical nature of search with practical concerns for computational resources.

III. The NFL Theorems

In this section we analyze the connection between algorithms and cost functions. We have dubbed the associated results NFL theorems because they demonstrate that if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems. Additionally, the name emphasizes a parallel with similar results in supervised learning [11], [12].

The precise question addressed in this section is: "How does the set of problems $F_1 \subset \mathcal{F}$ for which algorithm a_1 performs better than algorithm a_2 compare to the set $F_2 \subset \mathcal{F}$ for which the reverse is true?" To address this question we compare the sum over all f of $P(d_m^y|f, m, a_1)$ to the sum over all f of $P(d_m^y|f, m, a_2)$. This comparison constitutes a major result of this paper: $P(d_m^y|f, m, a)$ is independent of a when averaged over all cost functions.

Theorem 1: For any pair of algorithms a_1 and a_2

$$\sum_f P(d_m^y|f, m, a_1) = \sum_f P(d_m^y|f, m, a_2).$$

A proof of this result is found in Appendix A. An immediate corollary of this result is that for any performance measure $\Phi(d_m^y)$, the average over all f of P

[1] In particular, note that pseudorandom number generators are deterministic given a seed.

offset by its performance on the remaining problems; that is the only way that all algorithms can have the same f-averaged performance.

A result analogous to Theorem 1 holds for a class of time-dependent cost functions. The time-dependent functions we consider begin with an initial cost function f_1 that is present at the sampling of the first $\mathcal{X}$ value. Before the beginning of each subsequent iteration of the optimization algorithm, the cost function is deformed to a new function, as specified by a mapping $T : \mathcal{F} \times \mathcal{N} \rightarrow \mathcal{F}$.[2] We indicate this mapping with the notation T_i. So the function present during the ith iteration is f

Since it is certainly true that any class of problems faced by a practitioner will not have a flat prior, what are the practical implications of the NFL theorems when viewed as a statement concerning an algorithm's performance for nonfixed f? This question is taken up in greater detail in Section IV but we offer a few comments here.

First, if the practitioner has knowledge of problem characteristics but does not incorporate them into the optimization algorithm, then $P(f)$ is effectively uniform. (Recall that $P(f)$ can be viewed as a statement concerning the practitioner's choice of optimization algorithms.) In such a case, the NFL theorems establish that there are no formal assurances that the algorithm chosen will be at all effective.

Second, while most classes of problems will certainly have some structure which, if known, might be exploitable, the simple existence of that structure does not justify choice of a particular algorithm; that structure must be known and reflected directly in the choice of algorithm to serve as such a justification. In other words, the simple existence of structure per se, absent a specification of that structure, cannot provide a basis for preferring one algorithm over another. Formally, this is established by the existence of NFL-type theorems in which rather than average over specific cost functions f, one averages over specific "kinds of structure," i.e., theorems in which one averages $P(d_m^y \mid m,$

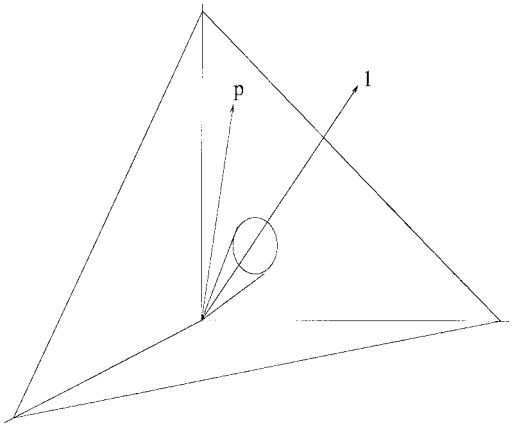

Fig. 1. Schematic view of the situation in which function space $\mathcal{F}$ is three dimensional. The uniform prior over this space, $\vec{\mathbf{1}}$, lies along the diagonal. Different algorithms a give different vectors v lying in the cone surrounding the diagonal. A particular problem is represented by its prior $\vec{\boldsymbol{p}}$ lying on the simplex. The algorithm that will perform best will be the algorithm in the cone having the largest inner product with $\vec{\boldsymbol{p}}$.

probabilities that algorithm a gives sample d_m^y on cost function f after m distinct cost evaluations) are all either zero or one, so NFL also implies that $\sum_f P$

we can choose an algorithm for which the calculation *is* tractable.

Theorem 3: For any algorithm, the fraction of cost functions that result in a particular histogram $\vec{\boldsymbol{c}} = m\vec{\boldsymbol{\alpha}}$ is

$$\rho_f(\vec{\boldsymbol{\alpha}}) = \frac{\binom{m}{c_1\, c_2 \cdots c_{|\mathcal{Y}|}} |\mathcal{Y}|^{|\mathcal{X}|-m}}{|\mathcal{Y}|^{|\mathcal{X}|}} = \frac{\binom{m}{c_1\, c_2 \cdots c_{|\mathcal{Y}|}}}{|\mathcal{Y}|^m}.$$

For large enough m, this can be approximated as

$$\rho_f(\vec{\boldsymbol{\alpha}}) \cong C(m, |\mathcal{Y}|) \frac{\exp\left[m\, S(\vec{\boldsymbol{\alpha}})\right]}{\prod_{i=1}^{|\mathcal{Y}|} \alpha_i^{1/2}}$$

where $S(\vec{\boldsymbol{\alpha}})$ is the entropy of the distribution $\vec{\boldsymbol{\alpha}}$, and $C(m, |\mathcal{Y}|)$ is a constant that does not depend on $\vec{\boldsymbol{\alpha}}$.

This theorem is derived in Appendix C. If some of the $\vec{\boldsymbol{\alpha}}_i$ are zero, the approximation still holds, only with $\mathcal{Y}$ redefined to exclude the y's corresponding to the zero-valued $\vec{\boldsymbol{\alpha}}_i$. However, $\mathcal{Y}$ is defined and the normalization constant of (3) can be found by summing over all $\vec{\boldsymbol{\alpha}}$ lying on the unit simplex [14].

A related question is the following: "For a given cost function, what is the fraction ρ_{alg} of all algorithms that give rise to a particular $\vec{\boldsymbol{c}}$?" It turns out that the only feature of f relevant for this question is the histogram of its cost values formed by looking across all $\mathcal{X}$. Specify the fractional form of this histogram by $\vec{\boldsymbol{\beta}}$ so that there are $N_i = \beta_i\, |\mathcal{X}|$ points in $\mathcal{X}$ for which $f(x)$ has the ith $\mathcal{Y}$ value.

In Appendix D it is shown that to leading order, $\rho_{\text{alg}}(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}})$ depends yet another information-theorf quantity, the Kullback–Liebler distance [15] between $\vec{}$

Gaussianly. For such cases, we can use the Gaussian nature of the distribution to facilitate our calculations. In particular, if the mean and variance of the Gaussian are μ and σ^2, respectively, then we have $\Omega(\epsilon) = \text{erfc}((\epsilon - \mu)/\sqrt{2}\sigma)/2$, where erfc is the complimentary error function.

To calculate the third performance measure, note that for fixed f and m, for any (deterministic) algorithm a, $P(\vec{\boldsymbol{c}} > \epsilon \mid f, m, a)$ is either one or zero. Therefore the fraction of algorithms that result in a $\vec{\boldsymbol{c}}$ whose minimum exceeds ϵ is given by

$$\frac{\sum_a P(\min(\vec{\boldsymbol{c}}) > \epsilon \mid f, m, a)}{\sum_a 1}.$$

Expanding in terms of $\vec{c}$, we can rewrite the numerator of this ratio as $\sum_{\vec{\boldsymbol{c}}} P(\min(\vec{\boldsymbol{c}}) > \epsilon | \vec{c}) \sum_a P(\vec{\boldsymbol{c}} \mid f, m, a)$. The ratio of this quantity to $\sum_a 1$, however, is exactly what was calculated when we evaluated measure ii) [see the beginning of the argument deriving (4)]. This establishes the following

with $\mathcal{Y}$ components z and that a_2 produces a sample with $\mathcal{Y}$ components z'.

In Appendix F, it is proven by example that this quantity need not be symmetric under interchange of z and z'.

Theorem 8: In general

$$\sum_f P_{d^y_{m,1},d^y_{m,2}}(z,z' \mid f,m,a_1,a_2) \neq \sum_f P_{d^y_{m,1},d^y_{m,2}}(z',z \mid f,m,a_1,a_2).$$

This means that under certain circumstances, even knowing only the $\mathcal{Y}$ components of the samples produced by two algorithms run on the same unknown f, we can infer something concerning which algorithm produced each population.

Now consider the quantity

$$\sum_f P_{\vec{\mathbf{c}}_1,\vec{\mathbf{c}}_2}(z,z' \mid f,m,a_1,a_2)$$

again for deterministic algorithms a_1 and a_2. This quantity is just the number of f such that it is both true that a_1 produces a histogram z and that a_2 produces a histogram z'. It too need not be symmetric under interchange of z and z' (see Appendix F). This is a stronger statement than the asymmetry of d^y's statement, since any particular histogram corresponds to multiple samples.

It would seem that neither of these two results directly implies that there are algorithms a_1 and a_2 such that for some f a_1's histogram is much better than a_2's, but for no f's is the reverse is true. To investigate this problem involves looking over all pairs of histograms (one pair for each f) such that there is the same relationship between (the performances of the algorithms, as reflected in) the histograms. Simply having an inequality between the sums presented above does not seem to directly imply that the relative performances between the associated pair of histograms is asymmetric. (To formally establish this would involve creating scenarios in which there is an inequality between the sums, but no head-to-head minimax distinctions. Such an analysis is beyond the scope of this paper.)

On the other hand, having the sums be equal does carry obvious implications for whether there are head-to-head minimax distinctions. For example, if both algorithms are deterministic, then for any particular f, $P_{d^y_{m,1},d^y_{m,2}}(z_1,z_2 \mid f,m,a_1,a_2)$ equals one for one (z_1,z_2) pair and zero for all others. In such a case, $\sum_f P_{d^y_{m,1},d^y_{m,2}}(z_1,z_2 \mid f,m,a_1,a_2)$ is just the number of f that result in the pair (z_1,z_2). So $\sum_f P_{d^y_{m,1},d^y_{m,2}}(z,z' \mid f,m,a_1,a_2) = \sum_f P_{d^y_{m,1},d^y_{m,2}}(z',z \mid f,m,a_1,a_2)$ implies that there are no head-to-head minimax distinctions between a_1 and a_2. The converse, however, does not appear to hold.[4]

[4]Consider the grid of all (z,z') pairs. Assign to each grid point the number of f that result in that grid point's (z,z') pair. Then our constraints are i) by the hypothesis that there are no head-to-head minimax distinctions, if grid point (z_1,z_2) is assigned a nonzero number, then so is (z_2,z_1) and ii) by the no-free-lunch theorem, the sum of all numbers in row z equals the sum of all numbers in column z. These two constraints do not appear to imply that the distribution of numbers is symmetric under interchange of rows and columns. Although again, like before, to formally establish this point would involve explicitly creating search scenarios in which it holds.

As a preliminary analysis of whether there can be head-to-head minimax distinctions, we can exploit the result in Appendix F, which concerns the case where $|\mathcal{X}| = |\mathcal{Y}| = 3$. First, define the following performance measures of two-element samples, $\Phi(d^y_2)$.

i) $\Phi(y_2,y_3) = \Phi(y_3,y_2) = 2$.
ii) $\Phi(y_1,y_2) = \Phi(y_2,y_1) = 0$.
iii) Φ of any other argument $= 1$.

In Appendix F we show that for this scenario there exist pairs of algorithms a_1 and a_2 such that for one f a_1 generates the histogram $\{y_1,y_2\}$ and a_2 generates the histogram $\{y_2,y_3\}$, but there is no f for which the reverse occurs (i.e., there is no f such that a_1 generates the histogram $\{y_2,y_3\}$ and a_2 generates $\{y_1,y_2\}$).

So in this scenario, with our defined performance measure, there *are* minimax distinctions between a_1 and a_2. For one f the performance measures of algorithms a_1 and a_2 are, respectively, zero and two. The difference in the Φ values for the two algorithms is two for that f. There are no other f, however, for which the difference is -2. For this Φ then, algorithm a_2 is minimax superior to algorithm a_1.

It is not currently known what restrictions on $\Phi(d^y_m)$ are needed for there to be minimax distinctions between the algorithms. As an example, it may well be that for $\Phi(d^y_m) = \min_i\{d^y_m(i)\}$ there are no minimax distinctions between algorithms.

More generally, at present nothing is known about "how big a problem" these kinds of asymmetries are. All of the examples of asymmetry considered here arise when the set of X values a_1 has visited overlaps with those that a_2 has visited. Given such overlap, and certain properties of how the algorithms generated the overlap, asymmetry arises. A precise specification of those "certain properties" is not yet in hand. Nor is it known how generic they are, i.e., for what percentage of pairs of algorithms they arise. Although such issues are easy to state (see Appendix F), it is not at all clear how best to answer them.

Consider, however, the case where we are assured that, in m steps, the samples of two particular algorithms have not overlapped. Such assurances hold, for example, if we are comparing two hill-climbing algorithms that start far apart (on the scale of m) in $\mathcal{X}$. It turns out that given such assurances, there are no asymmetries between the two algorithms for m-element samples. To see this formally, go through the argument used to prove the NFL theorem, but apply that argument to the quantity $\sum_f P_{d^y_{m,1},d^y_{m,2}}(z,z' \mid f,m,a_1,a_2)$ rather than $P(\vec{\mathbf{c}} \mid f,m,a)$. Doing this establishes the following theorem.

Theorem 9: If there is no overlap between $d^x_{m,1}$ and $d^x_{m,2}$, then

$$\sum_f P_{d^y_{m,1},d^y_{m,2}}(z,z' \mid f,m,a_1,a_2) = \sum_f P_{d^y_{m,1},d^y_{m,2}}(z',z \mid f,m,a_1,a_2).$$

An immediate consequence of this theorem is that under the no-overlap conditions, the quantity $\sum_f P_{C_1,C_2}(z,z' \mid f,m,a_1,a_2)$ is symmetric under interchange of z and z', as

are all distributions determined from this one over C_1 and C_2 (e.g., the distribution over the difference between those C's extrema).

Note that with stochastic algorithms, if they give nonzero probability to all d_m^x, there is always overlap to consider. So there is always the possibility of asymmetry between algorithms if one of them is stochastic.

VII. $P(f)$-Independent Results

All work to this point has largely considered the behavior of various algorithms across a wide range of problems. In this section we introduce the kinds of results that can be obtained when we reverse roles and consider the properties of many algorithms on a *single* problem. More results of this type are found in [5]. The results of this section, although less sweeping than the NFL results, hold no matter what the real world's distribution over cost functions is.

Let a and a' be two search algorithms. Define a "choosing procedure" as a rule that examines the samples d_m and d'_m, produced by a and a', respectively, and based on those samples, decides to use either a or a' for the subsequent part of the search. As an example, one "rational" choosing procedure is to use a for the subsequent part of the search if and only if it has generated a lower cost value in its sample than has a'. Conversely we can consider an "irrational" choosing procedure that uses the algorithm that had *not* generated the sample with the lowest cost solution.

At the point that a choosing procedure takes effect, the cost function will have been sampled at $d_\cup \equiv d_m \cup d'_m$. Accordingly, if $d_{>m}$ refers to the samples of the cost function that come after using the choosing algorithm, then the user is interested in the remaining sample $d_{>m}$. As always, without loss of generality, it is assumed that the search algorithm selected by the choosing procedure does not return to any points in $d_\cup$.[5]

The following theorem, proven in Appendix G, establishes that there is no *a priori* justification for using any particular choosing procedure. Loosely speaking, no matter what the cost function, without special consideration of the algorithm at hand, simply observing how well that algorithm has done so far tells us nothing *a priori* about how well it would do if we continue to use it on the same cost function. For simplicity, in stating the result we only consider deterministic algorithms.

Theorem 10: Let d_m and d'_m be two fixed samples of size m, that are generated when the algorithms a and a', respectively, are run on the (arbitrary) cost function at hand. Let A and B be two different choosing procedures. Let k be the number of elements in $c_{>m}$. Then

$$\sum_{a,a'} P(c_{>m} \mid f, d, d', k, a, a', A) = \sum_{a,a'} P(c_{>m} \mid f, d, d', k, a, a', B).$$

Implicit in this result is the assumption that the sum excludes those algorithms a and a' that do not result in d and d' respectively when run on f.

In the precise form it is presented above, the result may appear misleading, since it treats all samples equally, when for any given f some samples will be more likely than others. Even if one weights samples according to their probability of occurrence, however, it is still true that, on average, the choosing procedure one uses has no effect on likely $c_{>m}$. This is established by the following result, proven in Appendix H.

Theorem 11: Under the conditions given in the preceding theorem

$$\sum_{a,a'} P(c_{>m} \mid f, m, k, a, a', A) = \sum_{a,a'} P(c_{>m} \mid, f, m, k, a, a', B).$$

These results show that no assumption for $P(f)$ alone justifies using some choosing procedure as far as subsequent search is concerned. To have an intelligent choosing procedure, one must take into account not only $P(f)$ but also the search algorithms one is choosing among. This conclusion may be surprising. In particular, note that it means that there is no intrinsic advantage to using a rational choosing procedure, which continues with the better of a and a', rather than using an irrational choosing procedure which does the opposite.

These results also have interesting implications for degenerate choosing procedures $A \equiv \{\text{always use algorithm } a\}$ and $B \equiv \{\text{always use algorithm } a'\}$. As applied to this case, they mean that for fixed f_1 and f_2, if f_1 does better (on average) with the algorithms in some set $\mathcal{A}$, then f_2 does better (on average) with the algorithms in the set of all other algorithms. In particular, if for some favorite algorithms a certain "well-behaved" f results in better performance than does the random f, then that well-behaved f gives *worse than random* behavior on the set all remaining algorithms. In this sense, just as there are no universally efficacious search algorithms, there are no universally benign f which can be assured of resulting in better than random performance regardless of one's algorithm.

In fact, things may very well be worse than this. In supervised learning, there is a related result [11]. Translated into the current context, that result suggests that if one restricts sums to only be over those algorithms that are a good match to $P(f)$, then it is often the case that "stupid" choosing procedures—like the irrational procedure of choosing the algorithm with the less desirable $\vec{c}$—outperform "intelligent" ones. What the set of algorithms summed over must be in order for a rational choosing procedure to be superior to an irrational procedure is not currently known.

[5] a can know to avoid the elements *it* has seen before. However *a priori*, a has no way to avoid the elements observed by a' has (and vice-versa). Rather than have the definition of a somehow depend on the elements in $d' - d$ (and similarly for a'), we deal with this problem by defining $c_{>m}$ to be set only by those elements in $d_{>m}$ that lie outside of $d_\cup$. (This is similar to the convention we exploited above to deal with potentially retracing algorithms.) Formally, this means that the random variable $c_{>m}$ is a function of $d_\cup$ as well as of $d_{>m}$. It also means there may be fewer elements in the histogram $c_{>m}$ than there are in the sample $d_{>m}$.

VIII. Conclusions

A framework has been presented in which to compare general-purpose optimization algorithms. A number of NFL theorems were derived that demonstrate the danger of comparing algorithms by their performance on a small sample of problems. These same results also indicate the importance of incorporating problem-specific knowledge into the behavior of the algorithm. A geometric interpretation was given showing what it means for an algorithm to be well suited to solving a certain class of problems. The geometric perspective also suggests a number of measures to compare the similarity of various optimization algorithms.

More direct calculational applications of the NFL theorem were demonstrated by investigating certain information-theoretic aspects of search, as well as by developing a number of benchmark measures of algorithm performance. These benchmark measures should prove useful in practice.

We provided an analysis of the ways that algorithms can differ *a priori* despite the NFL theorems. We have also provided an introduction to a variant of the framework that focuses on the behavior of a range of algorithms on specific problems (rather than specific algorithms over a range of problems). This variant leads directly to reconsideration of many issues addressed by computational complexity, as detailed in [5].

Much future work clearly remains. Most important is the development of practical applications of these ideas. Can the geometric viewpoint be used to construct new optimization techniques in practice? We believe the answer to be yes. At a minimum, as Markov random field models of landscapes become more wide spread, the approach embodied in this paper should find wider applicability.

Appendix A
NFL Proof for Static Cost Functions

We show that $\sum_f P(\vec{c}$

$1), f(a(d_m)))$ depends only on the f values defined over points outside d_m^x. (Recall that $a(d_m^x) \notin d_m^x$.) So we have

$$\sum_f P(d_{m+1}^y | f, m+1, a) = \sum_{d_m^x} \sum_{f(x \in d_m^x)} P(d_m | f, m, a) \sum_{f(x \notin d_m^x)} \delta(d_{m+1}^y(m+1), f(a(d_m))). \quad (6)$$

The sum $\sum_{f(x \notin d_m^x)}$ contributes a constant, $|\mathcal{Y}|^{|\mathcal{X}|-m-1}$, equal to the number of functions defined over points not in d_m^x passing through $(d_{m+1}^x(m+1), f(a(d_m)))$. So

$$\begin{aligned}\sum_f P(d_{m+1}^y | f, m+1, a) &= |\mathcal{Y}|^{|\mathcal{X}|-m-1} \sum_{f(x \in d_m^x), d_m^x} P(d_m | f, m, a) \\ &= \frac{1}{|\mathcal{Y}|} \sum_{f, d_m^x} P(d_m | f, m, a) \\ &= \frac{1}{|\mathcal{Y}|} \sum_f P(d_m^y | f, m, a).\end{aligned}$$

By hypothesis, the right-hand side of this equation is independent of a, so the left-hand side must also be. This completes the proof.

Appendix B
NFL Proof for Time-Dependent Cost Functions

In analogy with the proof of the static NFL theorem, the proof for the time-dependent case proceeds by establishing the a-independence of the sum $\sum_T P(c | f, T, m, a)$, where here c is either d_m^y or D_m^y.

To begin, replace each T in this sum with a set of cost functions, f_i, one for each iteration of the algorithm. To do this, we start with the following:

$$\begin{aligned}\sum_T P(c | f, T, m, a) &= \sum_T \sum_{d_m^x} \sum_{f_2 \cdots f_m} P(c | \vec{f}, d_m^x, T, m, a) \\ &\quad \cdot P(f_2 \cdots f_m, d_m^x \mid f_1, T, m, a) \\ &= \sum_{d_m^x} \sum_{f_2 \cdots f_m} P(\vec{c} \mid \vec{f}, d_m^x) P(d_m^x \mid f, m, a) \\ &\quad \cdot \sum_T P(f_2 \cdots f_m \mid f_1, T, m, a)\end{aligned}$$

where the sequence of cost functions,

decoupled. Nevertheless, the NFL result still holds. This is proven by expanding (7) over possible d_m^y values

$$\begin{aligned}\sum_T P(d_m^y|f,T,m,a) \propto & \sum_{d_m^x} \sum_{f_2 \cdots f_m} \sum_{d_m^y} P(d_m^y|d_m^y) \\ & \cdot P(d_m^y \mid \vec{f}, d_m^x) \\ & \cdot P(d_m^x \mid f_1 \cdots f_{m-1}, m, a) \\ = & \sum\end{aligned}$$

$\phi(a) = \pi$ is a constant, independent of π, f, and $\vec{c}$. and that ϕ is single valued will complete the proof.

Recalling that every x value in an unordered path is distinct, any unordered path π gives a set of $m!$ different ordered paths. Each such ordered path π_{ord} in turn provides a set of m successive d's (if the empty d is included) and a following x. Indicate by $d(\pi_{ord})$ this set of the first m d's provided by π_{ord}.

From any ordered path π_{ord} a "partial algorithm" can be constructed. This consists of the list of an

Using this result in (9) we find

$$\sum_f P(\min(d^y$$

Let $proc$ represent either A or B. We are interested in

$$\sum_{a,a'} P(c_{>m}|f, d_1, d_2, k, a, a', proc)$$
$$= \sum_{a_{\perp d}, a'_{\perp d'}} \sum_{a_{\subset d}, a'_{\subset d'}} \sum_{a_{\supseteq d}, a'_{\supseteq d'}} P$$