2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Bounding the Number of Favorable Functions in Stochastic Search

George D. Montañez
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
gmontane@cs.cmu.edu

***Abstract*—According to the No Free Lunch theorems for search, when uniformly averaged over all possible search functions, every search algorithm has identical search performance for a wide variety of common performance metrics [1], [2], [3], [4]. Differences in performance can arise, however, between two algorithms when performance is measured over non-closed under permutation sets of functions, such as sets consisting of a single function. Using uniform random sampling with replacement as a baseline, we ask how many functions exist such that a search algorithm has better expected performance than random sampling. We define *favorable functions* as those that allow an algorithm to locate a search target with higher probability than uniform random sampling with replacement, and we bound the proportion of favorable functions for stochastic search methods, including genetic algorithms. Using active information [5] as our divergence measure, we demonstrate that no more than 2^{-b} of all functions are favorable by b or more bits, for $b \geq 2$ and reasonably sized search spaces ($n \geq 19$). Thus, the proportion of functions for which an algorithm performs relatively well by a moderate degree is strictly bounded. Our results can be viewed as statement of information conservation [6], [7], [1], [8], [5], since identifying a favorable function of b or more bits requires at least b bits of information, under the conditions given.**

I. Introduction

Over the past three decades, research into conservation of information laws [6], [7], [9], [1], [8], [5], including the No Free Lunch theorems [1], [4], have helped illuminate fundamental constraints on algorithmic search and supervised machine learning. Early work in supervised learning by Mitchell [6] demonstrated that *unbiased* learners, namely those that could represent and learn any target function, were useless and could be no more accurate at predicting target labels than random guessing. Thus, biases were necessary for successful inductive learning. Schaffer later proposed a conservation law for generalization performance [7], demonstrating that when generalization performance was summed over all possible target concepts, every algorithm performed equally well. Wolpert and Macready extended these results in supervised learning [4] and to search and optimization [1]. Wolpert's extension of the No Free Lunch theorem for supervised learning demonstrated that not only were biases necessary for learning (agreeing with Mitchell [6]), but biases also needed to be correct, namely, consisting of assumptions that aligned with the actual distribution of target functions in the problem domain via a non-Euclidean inner product between two vectors, one representing the distribution of problems in the real world and the other representing the inductive bias of the learner [4]. Other researchers such as Culberson [2], Schumacher *et al.* [3], Whitley [10], Droste *et al.* [11], English [12], [8], and Dembski *et al.* [5] have continued this line of inquiry, proving additional results concerning conservation of information in algorithmic processes. We extend these results by demonstrating a conservation of information result for the proportion of favorable functions in stochastic search.

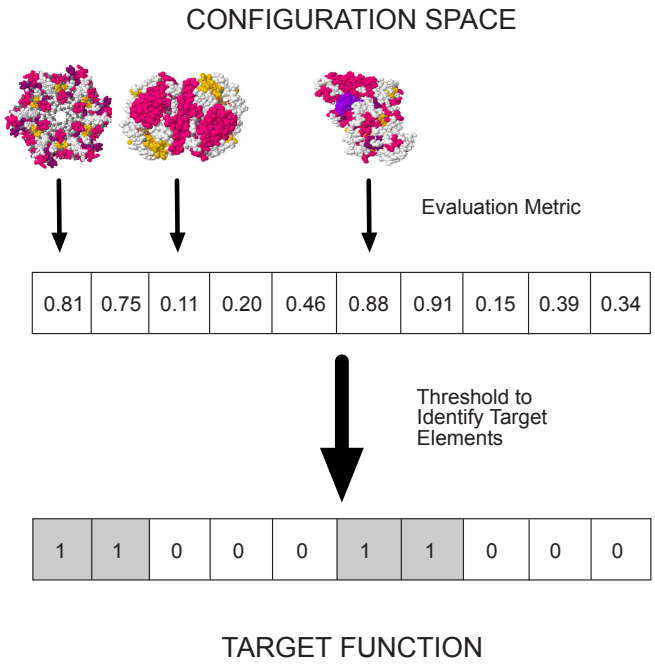

Fig. 1. Mapping a configuration space to a target function. Elements from the configuration space Ω are evaluated according to some metric (such as binding affinity) and the numerical evaluations are thresholded to produce the final target function, a binary string of length $|\Omega|$, where ones in the string indicate that those elements $\omega \in \Omega$ are in the target.

A. Search Problem

We consider a common search problem, where a configuration space Ω is searched to find some element that meets or exceeds a given threshold, such as searching protein space to find a configuration that catalyzes a specific reaction with some desired alacrity [13], [14] or binds to a target site with sufficient strength [15], [16]. Once an evaluation metric and threshold are chosen, this action partitions the configuration space into two sets, namely, those that satisfy the threshold under the evaluation metric and those that do not. We can

represent this partitioning as a binary string [8] of length $|\Omega|$, which we denote as a *target function*. The search problem becomes one of sampling the configuration space to locate an element such that it is mapped to a one in the target function, i.e., it is contained in the target set. Figure 1 illustrates the mapping from a configuration space to a target function.

In this problem setting, an algorithm must locate some element of a target set and there is, as a result, a natural equivalence between target sets and target functions. Specifically, the ones of the target function correspond to target elements and the zeros correspond to elements not within the target set. We define *favorable functions* as those which allow a search algorithm to locate an element of the target set with higher expected per-query probability than uniform random sampling with replacement. Our analysis makes use of the equivalence between target sets and target functions to prove results in terms of target sets, and later restate the results in terms of target functions. Thus, both terms are used interchangeably in this study.

B. Choosing Target Functions for a Fixed Search Algorithm

Previous research [17] has examined the information costs for selecting a search algorithm that satisfies a minimum performance threshold given a fixed target function and search space, and found these costs to meet or exceed the amount of information gained by using such an algorithm. Here we consider the related case of identifying what proportion of possible target functions allow a fixed algorithm to satisfy a given performance threshold. More concretely, given a fixed search algorithm, how many target functions allow the algorithm to locate an element in the target set with probability greater than or equal to that of uniform random sampling? Our results indicate that locating a target function for which the search algorithm succeeds with relatively high probability (namely, a function that effectively reduces the search space by b bits, with $b \geq 2$) requires at least b bits of information. Hence, information is conserved in target choice, for $b \geq 2$. Unless one can (correctly) bias the space of possible target functions in favor of the algorithm in question, then searching the space of target functions for a good function remains a provably difficult problem; finding an appropriate target function for a fixed algorithm is as hard as finding a good algorithm for a fixed target function. Generating b bits of active information in either case requires b bits of information, whether selecting a target set or search algorithm.

The No Free Lunch theorems stipulate that, uniformly averaged over any closed under permutation set of functions, no algorithm has better average performance than uniform random sampling [3]. Assuming one has a set of target functions not closed under permutation, thus allowing for performance differences among algorithms, the next obvious question to ask is: how many functions, at most, are a "good fit" for a given algorithm? More precisely, given this algorithm, what proportion of target functions are favorable? That question motivates the present study and we find that this proportion is strictly bounded, being inversely related to the degree of favorability.

In addition, we also investigate the maximum amount of (active) information that can be produced by a fixed search algorithm when the search agent is given the freedom to choose a target function favorably suited to their specific algorithm.

II. Active Information

Active information [5] is a method of quantifying improvement in a search over a baseline search method, such as uniform random sampling with replacement. We use this divergence measure since it allows us to quantify gains in search performance in terms of information (bits). This, in turn, allows us to characterize precisely the proportion of favorable functions in a space of possible functions in relation to the number b bits desired.

Following Dembski *et al.* [5], let p be the probability of success for a single query taken uniformly at random over search space Ω, where success is defined as selecting an element from Ω belonging to a target set T, and let q represent the (usually unknown) expected per-query probability of success for some search algorithm under the same problem setting. The *endogenous information* is then defined as $I_\Omega = -\log_2(p)$, which represents the difficulty of the original search problem, in bits. It can be thought of as the difficulty, under uniform random sampling, of selecting one particular binary string from a search space of all binary strings of length $-\log_2(p)$. The *active information* I_+ is defined as [5]

$$I_+ = -\log_2\left(\frac{p}{q}\right). \tag{1}$$

Active information measures the difference of performance in bits, providing a geometric interpretation of effectively reducing (or increasing) the size of the underlying search space, as illustrated in Figure 2. For a concrete example, consider the search for a single, particular 16 bit binary string within the space of all 16 bit strings. This search has a baseline uniform random sampling probability of success $p = \frac{1}{2^{16}}$ for a single query. Assume an improved search method exists that raises the probability of success to $q = \frac{16}{2^{16}}$. The resulting active information, $I_+ = 4$ bits, represents the reduction in the size of the search problem: the probability of success q is equivalent to the probability of drawing uniformly at random a specific 12 bit string from the space of all 12 bit binary strings. The 4 bits of active information reduce the original 16 bit search problem to an equivalent 12 bit search problem.

Given the definition of active information, we briefly consider a difficulty that can arise when using such a measure. When considering a probability of success $q = 0$, the active information becomes undefined (due to division by zero, or by taking $I_+ = \log_2(\frac{q}{p})$, becomes $-\infty$). In this paper, we limit consideration to algorithms with probability of success $q > 0$. This presents no loss of generality for search algorithms with uniformly randomly sampled initial populations, since uniformly sampling to create an initial population will produce a nonzero probability of locating any element in a target set.

III. Bounding Active Information

We proceed towards our goal of bounding the proportion of favorable functions for a search algorithm by first considering bounds on the amount of active information that can be produced by a search algorithm on a given function. The present study will use genetic algorithms as an ongoing

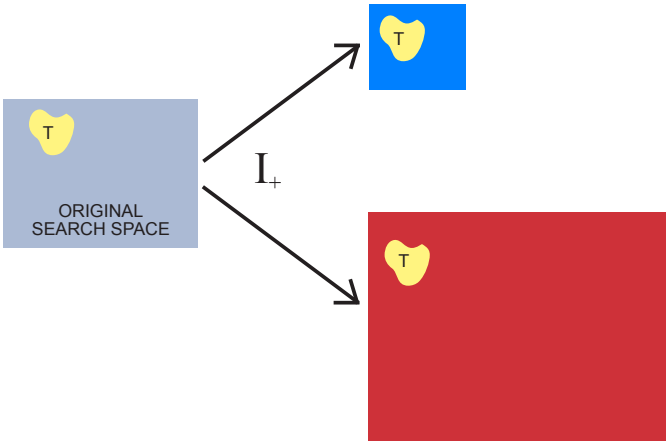

Fig. 2. Geometric interpretation of active information. Positive active information results in an improved search that is equivalent to performing uniform random sampling on a smaller search space (top-right), for a target (denoted by T) of the same size. Negative active information has the opposite effect, enlarging the effective search space (bottom-right).

example, but the bounds derived apply to a richer variety of search algorithms. We, therefore, begin our discussion of active information bounds by considering a general black-box search algorithm that chooses a probability distribution over the search space Ω at time t and uses it to select elements for evaluation. Using the definition provided in [18], a black-box algorithm is defined as follows:

Definition 1. (Black-Box Algorithm)

1) Choose some probability distribution π on $\{0,1\}^n$ and produce a random search point $x_1 \in \Omega$ according to π. Compute $f(x_1)$, where f is the fitness evaluation function.
2) In step t, stop if the considered stopping criterion is fulfilled. Otherwise, depending on $I(t) = (x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1}))$, choose some probability distribution $\pi_{I(t)}$ on Ω and produce a random search point $x_t \in \Omega$ according to $\pi_{I(t)}$. Compute $f(x_t)$.

Since this definition is sufficiently general it applies to genetic algorithms in particular, if we consider each individual of the population a queried point and allow the fitness map to serve as the fitness evaluation function. Thus, any optimal bounds derived for general black-box algorithms also hold for typical genetic algorithms.

A. Optimal Black-Box Search

Let us consider a single search for elements in a target set T within the search space Ω. Assuming that the initial probability distribution π acts as an oracle and provides complete target location information by placing all probability mass on an element within the target, we can locate an element in T in one query. This is the best performance possible, since an algorithm cannot locate the target without sampling at least one point in Ω.

Given the single-query uniform random sampling probability of success equal to p and an assisted probability of success $q = 1$, we find that the maximal amount of active information is

$$\begin{aligned} I_{+_{max}} &= -\log_2\left(\frac{p}{q}\right) \\ &= -\log_2\left(\frac{p}{1}\right) \\ &= I_\Omega. \end{aligned} \tag{2}$$

Therefore, the maximum active information produced by a black-box algorithm on a search for elements in Ω is equal to the endogenous information of the original search. Since this is the maximum active information achievable by *any* black-box search algorithm, this quantity also places an upper bound on the active information produced by a genetic algorithm.

Intuitively, this bound is expected given the geometric interpretation of active information. Given an n-bit search problem, each bit of information provided reduces the search space size by half. This process can continue until there are no more elements in the search space, forcing an end to the halving process. Thus, if there are $|\Omega|$ items in a search space, we can halve the space at most $\log_2|\Omega|$ times, with each halving operation providing one bit of active information. Therefore, we would expect a maximum of $\log_2|\Omega|$ bits of active information and find that our bound does indeed match this quantity.

Having derived the maximum active information achievable by any black-box search algorithm, we now investigate how much active information can be achieved through the choice of target function, holding the search algorithm fixed.

IV. Active Information in Target Choice

Montañez [19] noted the ability to store information in a genetic algorithm through selection of a fitness map, much like information is stored for later transmission in a flash memory device through the selection of bit patterns. The genetic algorithm search process acts as a noisy communication channel, selecting a message (fitness map) at one end that results in another message (population outcome) appearing on the receiving end with some high probability. Because different fitness maps result in different population outcomes, one can select (or design) a fitness map for a particular desired outcome or target.

We will now determine the effect of holding the algorithm constant while allowing the target set to vary. Doing so allows us to quantify the amount of active information that arises from choice of target function in isolation. We find that up to $O(\log|\Omega|)$ bits of active information can be produced through choice of target function alone, given a fixed algorithm (and fixed fitness map, when considering a genetic algorithm). In Section V, we will also demonstrate that selecting a target function producing b or more bits of active information requires at least b bits of information for $b \geq 2$ and $|\Omega| \geq 19$, mirroring the results obtained by Dembski *et al.* [5] for selection of search algorithms. Overall, our results demonstrate that choice of target function can act as an additional source of active information for stochastic search, but one that requires prior information to exploit effectively.

A. Maximum Active Information in Target Choice

Consider a fixed algorithm that performs a search on space Ω, such as a genetic algorithm with a fixed fitness map. We define the endogenous probability of success for a search as $p = \frac{|T|}{|\Omega|}$, where T denotes a non-empty target set.

From the definition of active information, we see that the maximum active information occurs when p is minimized and q is maximized ($= 1$). The probability p is minimized when $|T| = 1$, so we find

$$\begin{aligned} I_{+_{max}} &= -\log_2\left(\frac{p}{q}\right) \\ &= -\log_2\left(\frac{\frac{1}{|\Omega|}}{1}\right) \\ &= \log_2|\Omega| \end{aligned} \tag{3}$$

which is the maximum active information produced by any black-box search algorithm (see previous section) and matches the boundary case given by Dembski *et al.* [5]. Therefore, given a fixed search algorithm, up to $\log_2|\Omega|$ bits of active information can be generated during a search on Ω, under the condition that at least one element $\omega \in \Omega$ is, with probability 1, guaranteed to appear in the output population, allowing us to set $T = \{\omega\}$ and fulfill the conditions outlined.

These findings indicate a level of freedom available to programmers when target choice is left open to be be defined after selection of the search algorithm. Varying the target set by choosing a target function allows up to $\log_2|\Omega|$ bits of active information to be generated by the search algorithm, holding all other settings unchanged. However, it should be noted that one often has more freedom to alter search algorithms than to choose target sets.

V. Maximum Number of Favorable Functions

Having answered the question of how much information can be gained by suitable choice of target function, we now turn to our preliminary question, namely, how many favorable functions can exist for a given search algorithm? Given the constraints imposed by the No Free Lunch theorems, we do not expect every function to be favorable for a fixed search algorithm, since every search algorithm, no matter how sophisticated or unsophisticated, has a domain of applicability where it achieves best in-class performance. Although there are cases when *every* algorithm has identical performance (and all algorithms thus achieve "best" performance), there cannot exist an algorithm that is uniformly dominated by another algorithm over all functions. At best, it can be dominated for a majority of functions, but must necessarily dominate the other algorithm for the remaining functions [20]. Our concern is how many target functions allow a given search algorithm to dominate uniform random sampling to a certain extent, measured as b bits of performance gain. We state our main result in the form of a theorem.

Theorem 1. *For $b \geq 2$ and reasonably sized search spaces (of size $n \geq 19$), the number of functions (target sets) providing b or more bits of active information in a finite set S_2 of all possible fixed-length target functions of size n is less than or equal to $\frac{|S_2|}{2^b}$. The probability of finding any such target set in S_2 under uniform random sampling is, therefore, less than or equal to 2^{-b}.*

Proof: Since there are $2^{|\Omega|}$ subsets possible over Ω, $|S_2| = 2^{|\Omega|}$. Let τ denote the set of target sets in S_2 that provide b or more bits of active information. Formally

$$\tau = \{T \in S_2 : I_+ \geq b\} \tag{4}$$

By Lemma 1 (see Appendix), τ is bounded above in size by set

$$\tau' = \left\{T \in S_2 : |T| \leq \frac{|\Omega|}{2^b}\right\} \tag{5}$$

since any target set providing greater than b bits of active information must also have a size less than or equal to $\frac{|\Omega|}{2^b}$.

Although we restrict ourselves to $b \geq 2$, when $b = 0$ the theorem holds trivially, since $\frac{|S_2|}{2^0} = |S_2|$. Furthermore, for $b > \log_2|\Omega| = I_{+_{max}}$, we have that the number of target sets producing greater than the maximum possible active information is (by definition) zero. We will examine the case of $|\Omega| \geq 19$ and $2 \leq b \leq \log_2|\Omega|$, leaving the case of marginal performance gains ($0 < b < 2$) or small search space size ($|\Omega| < 19$) for the Appendix (see Additional Lemmata and Experiments) and for possible future research.

Since there are exactly $\sum_{k=0}^{\lfloor \frac{|\Omega|}{2^b} \rfloor} \binom{|\Omega|}{k}$ unique subsets of Ω with $\frac{|\Omega|}{2^b}$ or fewer elements, by application of Lemma 3 (see Appendix) we find

$$\begin{aligned} \frac{|\tau'|}{|S_2|} &= \frac{\sum_{k=0}^{\lfloor \frac{|\Omega|}{2^b} \rfloor} \binom{|\Omega|}{k}}{2^{|\Omega|}} \\ &\leq \frac{2^{|\Omega|-b}}{2^{|\Omega|}} \\ &= \frac{1}{2^b}. \end{aligned} \tag{6}$$

Rearranging,

$$|\tau'| \leq \frac{|S_2|}{2^b}$$

and since $|\tau| \leq |\tau'|$, we obtain

$$|\tau| \leq \frac{|S_2|}{2^b}. \tag{7}$$

■

Thus, we see that if S_2 is the set of all possible target sets on Ω, namely the power set of Ω with size $|S_2| = 2^{|\Omega|}$, the number of target sets producing b or more bits of active information is less than or equal to $\frac{|S_2|}{2^b}$. Therefore, selecting a target function (uniformly from among the set of possible target functions) that effectively reduces the size of the search space by b bits requires at least b bits of information, and thus information is conserved when $b \geq 2$. Furthermore, this result quantifies a lower bound on the difficulty of finding an applicable problem domain (i.e., set of target functions for which an algorithm performs better than random sampling) for any given search algorithm.

VI. Discussion

Our results are significant in at least two ways. First, they demonstrate the power of agents to reduce the effective size of a search space by carefully matching target function (i.e., search problem) to their search algorithm. This process allows up to $O(\log |\Omega|)$ bits of active information to be produced solely through choice of target function, which we saw was equivalent to effectively reducing the size of the search space. Thus, even when no other changes can be made concerning the search algorithm, performance gains over uniform random sampling may be possible by identification of a favorable function for that algorithm.

Second, these results answer the natural question that arises from the No Free Lunch theorems once we learn that not all functions are favorable: how many, at most, are? We bound the maximum proportion of favorable functions for any search algorithm, and find the proportion to be $O(2^{-b})$ in b. Thus, the greater the performance gain desired, the fewer functions exist that produce at least such a performance gain. Furthermore, these performance gains are necessarily limited to $b \leq \log_2 |\Omega|$ bits. For gains b greater than two bits, we find that identifying a favorable function that meets or exceeds a desired level of performance gain requires at least b bits of prior information. For $b \geq 2$, information is conserved, whether selecting a search algorithm for a fixed target function, or selecting a target function for a fixed search algorithm.

These results have real-world application in industrial and research settings. For industry, given a search algorithm limited by resource constraints and other restrictions, performance gains may be achievable through clever manipulation of evaluation criteria (and thus, target function), allowing the operator to design a target function that is favorable for their algorithm, even when other changes to the algorithm cannot be made. For researchers studying the ability of genetic algorithms to produce information, care must be taken to not inadvertently introduce a favorable target function for the algorithm under study while failing to account for the information introduced by the researcher's action in choosing such a target function. Since choice of target function can introduce up to $\log_2 |\Omega|$ bits of additional information, scientific rigor demands that we carefully discriminate between the information introduced by our own actions and that produced by the algorithm itself.

Although we have discussed the problem in regards to the number of functions that allow a stochastic search algorithm to perform well, one can also view the problem in the reverse direction by asking how many functions exist for which uniform random sampling performs poorly. Phrased in this manner, Theorem 1 becomes a statement concerning the diminishing number of sparse target functions and becomes almost immediately obvious. Thus, our results simultaneously prove the difficulty of finding a target function for which the performance of uniform random sampling is significantly worse than for other types of stochastic search, which agrees with prior work of English [8], whose research concerning the behavior of random sampling on the typical function prefigures the results obtained here.

Finally, one must be careful to point out what these results do not say. Theorem 1 does **not** state that given a problem for which uniform random sampling performs poorly, there is no more than a 2^{-b} chance that a different search algorithm will perform better by at least b bits. What it actually says is closer to: under the size conditions specified, if we select a target function uniformly at random, there is no more than a 2^{-b} chance that a given search algorithm will perform better than uniform random sampling by at least b bits. The difference is subtle, but important. Once we condition on a specific target function of a given sparseness (i.e. once we know that random sampling performs poorly on the function in question), we are asking what is $\mathbb{P}(A_b \mid f)$, the probability that our algorithm A outperforms random sampling by at least b bits on the target function f. This is a different probabilistic question than "what is $\mathbb{P}(A_b)$?" The former question remains a separate, open and important problem.

VII. Conclusion

Given the search task of locating elements in a space that meet or exceed a given criterion (such as having sufficient binding affinity), we model this as an equivalent search through a binary string called a target function. Defining favorable functions as those which allow an algorithm to locate an element of the search target with higher expected per-query probability than uniform random sampling with replacement, we determined the maximum proportion of favorable functions for any fixed search algorithm once the level of favorability is specified. To the author's knowledge, this is the first explicit nontrivial bound derived for such a problem setting.

In addition, we found that even with a fixed fitness map and search algorithm, active information may be generated through the act of defining a target set (equivalently, choosing a target function). These results demonstrate that at least two methods exist for producing active information in a search: defining a fitness map and defining a target set. When one option is not available, the other option may be. Since positive active information is equivalent to an effective reduction in search problem size, these results point to the ability of agents to achieve performance gains for fixed algorithms through careful choice of target function. This agrees with prior research [6], [4] on the importance of correctly biasing computational methods to the problem domain, or conversely, choosing a correct problem domain (i.e. target function) for a specific computational search method.

Gaining a general understanding of search processes and leveraging them for human good continues to be a concern for researchers in fields such as computational intelligence [21], [22], [23], [24] and machine learning [25], [26], [27]. The work presented here addresses some theoretical concerns regarding the propensity of favorable functions for stochastic search, and provides nontrivial quantitative bounds in positive answer to the questions raised, which hold uniformly over all search algorithms within the problem setting outlined.

Appendix
Additional Lemmata and Experiments

Lemma 1. *Any target set T of search space Ω, $T \subseteq \Omega$, providing b bits of active information must have a size $|T| \leq \frac{|\Omega|}{2^b}$.*

Proof: This follows from the definition of active information. Since, under the single-query interpretation of active

information, $p = \frac{|T|}{|\Omega|}$ and $0 \leq q \leq 1$, we have

$$\begin{aligned} b &= I_+ \\ &= -\log_2\left(\frac{p}{q}\right) \\ &= \log_2\left(\frac{q}{\frac{|T|}{|\Omega|}}\right) \\ &= \log_2\left(\frac{q|\Omega|}{|T|}\right) \\ |T| &= \frac{q|\Omega|}{2^b} \\ &\leq \frac{|\Omega|}{2^b}. \end{aligned} \tag{8}$$

■

Lemma 2 (Sauer-Shelah Inequality)**.** *For $d \leq n$, $\sum_{j=0}^{d}\binom{n}{j} \leq \left(\frac{en}{d}\right)^d$.*

Proof: This proof of the Sauer-Shelah inequality [28] is reproduced here for completeness.

$$\sum_{j=0}^{d}\binom{n}{j} \leq \left(\frac{n}{d}\right)^d \sum_{j=0}^{d}\binom{n}{j}\left(\frac{d}{n}\right)^j \tag{9}$$

$$\leq \left(\frac{n}{d}\right)^d \sum_{j=0}^{n}\binom{n}{j}\left(\frac{d}{n}\right)^j \tag{10}$$

$$= \left(\frac{n}{d}\right)^d \left(1+\frac{d}{n}\right)^n \tag{11}$$

$$\leq \left(\frac{en}{d}\right)^d. \tag{12}$$

■

Lemma 3. *$\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor}\binom{n}{j} \leq 2^{n-b}$ for $n \geq 19$ and $2 \leq b \leq \log_2 n$.*

Proof: We seek to prove

$$\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor}\binom{n}{j} \leq 2^{n-b}. \tag{13}$$

We present two proofs here, one for the case when $3 \leq b \leq \log_2 n$ with $n \geq 8$, and a second, slightly less detailed proof that additionally covers the case of $2 \leq b < 3$.

First, note that for $b \geq 3$

$$2b \leq 2^b - 2$$

so that

$$\begin{aligned} b &\leq 2^b - b - 2 \\ &= 2^b - (b+2) \\ &= 2^b\left(1 - \frac{b+2}{2^b}\right) \end{aligned}$$

which implies

$$\frac{b}{\left(1-\frac{b+2}{2^b}\right)} \leq 2^b.$$

Also note that $n = 2^{\log_2 n} \geq 2^b$, for $b \leq \log_2 n$. Thus,

$$\frac{b}{\left(1-\frac{b+2}{2^b}\right)} \leq n$$

which implies the following series of inequalities:

$$\begin{aligned} b &\leq n\left(1 - \frac{b+2}{2^b}\right) \\ -b &\geq n\left(\frac{b+2}{2^b} - 1\right) \\ -b &\geq n\left(\frac{b+2}{2^b}\right) - n \\ n &\geq n\left(\frac{b+2}{2^b}\right) + b. \end{aligned} \tag{14}$$

We now use inequality (14) and the Sauer-Shelah inequality [28] to prove our main result.

$$\begin{aligned} 2^n &\geq 2^{n\left(\frac{b+2}{2^b}\right)+b} \\ &= 2^b 2^{n\left(\frac{b+2}{2^b}\right)} \\ &= 2^b 2^{\frac{n}{2^b}(b+2)} \\ &= 2^b\left(2^{b+2}\right)^{\frac{n}{2^b}} \\ &= 2^b\left(2^2 2^b\right)^{\frac{n}{2^b}} \\ &> 2^b\left(e2^b\right)^{\frac{n}{2^b}} \\ &= 2^b\left(en/\left(\frac{n}{2^b}\right)\right)^{\frac{n}{2^b}} \\ &\geq 2^b \sum_{j=0}^{\frac{n}{2^b}}\binom{n}{j} \\ &\geq 2^b \sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor}\binom{n}{j}. \end{aligned}$$

where the penultimate inequality follows from the Sauer-Shelah inequality.

Thus,

$$2^n \geq 2^b \sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor}\binom{n}{j}$$

and

$$\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor}\binom{n}{j} \leq 2^{n-b}$$

proving our result for $b \geq 3$.

Second Proof: Let us now consider the case of $2 \leq b \leq \log_2 n$. We make use of a previous result in complexity theory [29], namely

$$\sum_{j=0}^{\lfloor \epsilon n \rfloor}\binom{n}{j} \leq 2^{H(\epsilon)n},$$

where $H(\epsilon) = -\epsilon\log_2\epsilon - (1-\epsilon)\log_2(1-\epsilon)$ is the binary entropy of ϵ and $0 < \epsilon \leq 1/2$. Let $\epsilon = 2^{-b}$, allowing us to use the above result for any $b \geq 1$.

Thus,

$$\begin{aligned}\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} &\leq 2^{H(2^{-b})n} \\ &= 2^{\left[-\frac{1}{2^b}\log_2 \frac{1}{2^b} - (1-\frac{1}{2^b})\log_2(1-\frac{1}{2^b})\right]n} \\ &= 2^{\left[\frac{b}{2^b} - (1-\frac{1}{2^b})\log_2(1-\frac{1}{2^b})\right]n} \\ &\leq 2^{\left[1-\frac{b}{2^b}\right]n} \quad (15) \\ &\leq 2^{\left[1-\frac{b}{n}\right]n} \quad (16) \\ &\leq 2^{n-b}\end{aligned}$$

where inequality (15) holds when $b \geq 2.79$ and inequality (16) holds when $b \leq \log_2 n$, which implies $n \geq 7$.

To prove our result for smaller values of b requires larger values of n, thus we now make use of the condition that $n \geq 19$. For $2 \leq b \leq 2.79$ and $n \geq 19$, we have

$$\begin{aligned} b &\leq 2.79 \\ &< \log_2 19 - 1.41 \\ &\leq \log_2 n - 1.41 \end{aligned}$$

which implies

$$n \geq 2^{b+1.41}.$$

Under the conditions of $n \geq 19$ and $2 \leq b \leq 2.79$, we thus have

$$\begin{aligned}\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} &\leq 2^{\left[\frac{b}{2^b} - (1-\frac{1}{2^b})\log_2(1-\frac{1}{2^b})\right]n} \\ &\leq 2^{\left[1-\frac{b}{2^{b+1.41}}\right]n} \quad (17) \\ &\leq 2^{\left[1-\frac{b}{n}\right]n} \quad (18) \\ &\leq 2^{n-b}.\end{aligned}$$

where inequality (17) holds for $b \geq 2$ and inequality (18) holds for $b \leq \log_2 n - 1.41$. ■

Lemma 4. *When* $n \geq 100$, $\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} \leq 2^{n-b}$ *for* $1.28 \leq b \leq \log_2 n$. *For* $n \geq 500$, *the bound holds for* $1.09 \leq b \leq \log_2 n$.

Proof: For $b \leq 2$ and any search space of size $n \geq 100$, we have

$$\begin{aligned} b &\leq 2 \\ &< \log_2 100 - 4.643 \\ &\leq \log_2 n - 4.643 \end{aligned}$$

which implies

$$n \geq 2^{b+4.643}.$$

Thus, by the entropy bound used in Lemma 3, we have

$$\begin{aligned}\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} &\leq 2^{\left[\frac{b}{2^b} - (1-\frac{1}{2^b})\log_2(1-\frac{1}{2^b})\right]n} \\ &\leq 2^{\left[1-\frac{b}{2^{b+4.643}}\right]n} \quad (19) \\ &\leq 2^{\left[1-\frac{b}{n}\right]n} \quad (20) \\ &\leq 2^{n-b}\end{aligned}$$

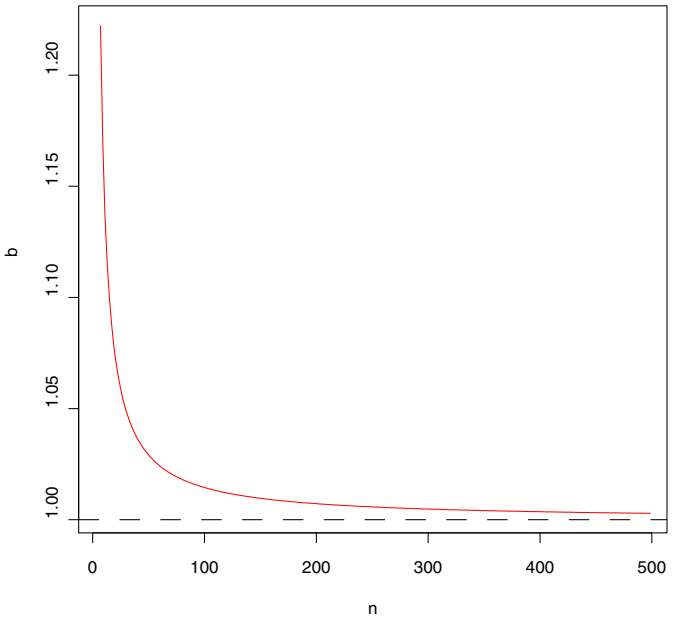

Fig. 3. Numerically identified smallest b values for which bound of Lemma 3 still holds, for $7 \leq n \leq 500$ and step size of 0.00001. Values are plotted only for odd n, as for even n the values were roughly 1 in all cases (i.e. < 1.00001).

where inequality (19) holds for $b \geq 1.28$ and inequality (20) holds when $b \leq \log_2 n - 4.643$, which is true whenever $b \leq 2$ and $n \geq 100$. By a similar argument, we find that when $n \geq 500$, the bound interval holds for $1.09 \leq b \leq \log_2 n$. ■

Numerically Testing Bound for Small b

We have proven Theorem 1 for $b \geq 2$ and $n \geq 19$, and have proven results for smaller b that hold when n is large (see Lemma 4). The question remains open as to how small one can set b and still have the bound from Lemma 3 hold, namely

$$\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} \leq 2^{n-b}, \quad (21)$$

without requiring large n. To investigate this, we systematically tested combinations of b and n values, for n from 7 to 500 and for b from $\log_2 n$ to 1, using a step size of 0.00001. For each n, we sought the smallest b for which inequality (21) remained valid. Empirically, the bound holds for all $b \geq 1.23$ tested. Figure 3 plots the minimum b value for which the bound still holds for every given n. The values rapidly approach 1 and become smaller than the b values of our proven bounds at $n \geq 25$. Thus, the results of Theorem 1 appear to hold for values as small as $n \geq 7$ and $b \geq 1.23$, once we combine our numerical observations with the additional cases proved in Lemma 4.

Acknowledgement

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 0946825 and the Ford Foundation Predoctoral Fellowship program. Any opinions, findings, and conclusions

or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the Ford Foundation.

The author would also like to thank Cosma Shalizi for his suggestion concerning the combinatoric results presented here.

REFERENCES

[1] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[2] J. Culberson, "On the futility of blind search: An algorithmic view of no free lunch," *Evolutionary Computation*, vol. 6, no. 2, pp. 109–127, 1998.

[3] C. Schumacher, M. Vose, and L. Whitley, "The no free lunch and problem description length," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001, pp. 565–570.

[4] D. Wolpert, "The supervised learning no-free-lunch theorems," in *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, vol. 6, no. 1, 2001, pp. 1–20.

[5] W. Dembski and R. Marks II, "Conservation of information in search: Measuring the cost of success," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, no. 5, pp. 1051 –1061, sept. 2009.

[6] T. M. Mitchell, "The need for biases in learning generalizations," Rutgers University, Tech. Rep., 1980.

[7] C. Schaffer, "A conservation law for generalization performance," in *Proceedings of the Eleventh International Machine Learning Conference*, W. W. Cohen and H. Hirsch, Eds. Rutgers University, New Brunswick, NJ, 1994, pp. 259–265.

[8] T. English, "Optimization is easy and learning is hard in the typical function," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2. IEEE, 2000, pp. 924–931.

[9] ——, "No more lunch: Analysis of sequential search," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1. IEEE, 2004, pp. 227–234.

[10] D. Whitley, "Functions as permutations: regarding no free lunch, walsh analysis and summary statistics," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 169–178.

[11] S. Droste, T. Jansen, and I. Wegener, "Optimization with randomized search heuristics–the (a)nfl theorem, realistic scenarios, and difficult functions," *Theoretical Computer Science*, vol. 287, no. 1, pp. 131–144, 2002.

[12] T. English, "Evaluation of evolutionary and genetic optimizers: No free lunch," in *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, 1996, pp. 163–169.

[13] F. Wen, N. U. Nair, and H. Zhao, "Protein engineering in designing tailored enzymes and microorganisms for biofuels production," *Current opinion in biotechnology*, vol. 20, no. 4, pp. 412–419, 2009.

[14] V. Nanda and R. L. Koder, "Designing artificial enzymes by intuition and computation," *Nature chemistry*, vol. 2, no. 1, pp. 15–24, 2009.

[15] Y. Choo and A. Klug, "Designing dna-binding proteins on the surface of filamentous phage," *Current opinion in biotechnology*, vol. 6, no. 4, pp. 431–436, 1995.

[16] R. N. McLaughlin Jr, F. J. Poelwijk, A. Raman, W. S. Gosal, and R. Ranganathan, "The spatial architecture of protein function and adaptation," *Nature*, 2012.

[17] W. Dembski and R. Marks II, "The search for a search: Measuring the information cost of higher level search," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 14, no. 5, pp. 475–486, 2010.

[18] J. He, C. Reeves, C. Witt, and X. Yao, "A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability," *Evol. Comput.*, vol. 15, no. 4, pp. 435–443, 2007.

[19] G. Montañez, "Information transmission through genetic algorithm fitness maps," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, (accepted for presentation).

[20] D. Whitley and J. Watson, "Complexity theory and the no free lunch theorem," *Search Methodologies*, pp. 317–339, 2005.

[21] T. Krink, B. Filipic, and G. Fogel, "Noisy optimization problems - a particular challenge for differential evolution?" in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, june 2004, pp. 332 – 339 Vol.1.

[22] J. Vrugt and B. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007.

[23] L. Feng, Y. Ong, I. Tsang, and A. Tan, "An evolutionary search paradigm that learns with past experiences," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[24] F. Ensan, E. Bagheri, and D. Gašević, "Evolutionary search-based test generation for software product line feature models," in *Advanced Information Systems Engineering*. Springer, 2012, pp. 613–628.

[25] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245 – 271, 1997.

[26] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, vol. 135, no. 1, pp. 1–54, 2002.

[27] M. Shimbo and T. Ishida, "Controlling the learning process of real-time heuristic search," *Artificial Intelligence*, vol. 146, no. 1, pp. 1–41, 2003.

[28] N. Sauer, "On the density of families of sets," *Journal of Combinatorial Theory, Series A*, vol. 13, no. 1, pp. 145–147, 1972.

[29] J. Flum and M. Grohe, *Parameterized complexity theory*. Springer Berlin, 2006, vol. 3, p. 427.