Nonlinearity, Hyperplane Ranking and the Simple Genetic Algorithm

R. B. Heckendorn, D. Whitley, and S. Rana
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
{heckendo,whitley,rana}@cs.colostate.edu

Abstract

Several metrics are used in empirical studies to explore the mechanisms of convergence of genetic algorithms. The ϕ metric is designed to measure the consistency of an arbitrary ranking of hyperplanes in a partition with respect to a target string. Walsh coefficients can be calculated for small functions in order to characterize sources of linear and nonlinear interactions. A simple deception measure is also developed to look closely at the effects of increasing nonlinearity of functions. Correlations between the ϕ metric and deception measure are discussed and relationships between ϕ and convergence behavior of a simple genetic algorithm are studied over large sets of functions with varying degrees of nonlinearity.

1 Introduction

A recurring theme in Holland's 1975 book is that genetic algorithms process schemata. One of the ways they do that is by ranking schemata in the population according to their usefulness. Whitley et al. [4] developed a metric ϕ which can be used to measure the consistency of an ordered set of hyperplanes in a partition with respect to a specified target string. This metric makes it possible to measure the consistency of the ranking of hyperplanes in all partitions of the search space with respect to a target string when the hyperplanes are 1) sorted with respect to their static average fitness and 2) sorted by proportional representation in the population of an actual genetic algorithm.

Whitley et al. [4] show that during the first few generations of a genetic algorithm the dynamic ranking of the hyperplanes in any partition, measured in terms of their proportional representation in the population, is highly correlated with the average hyperplane fitness

measured over all strings in each hyperplane. Furthermore, as the number of generations of the genetic algorithm increases, schemata in a population are not dynamically ranked according to their observed fitness; rather, empirical results suggested that the proportional representation of schemata in the population rank schemata with respect to their consistency with the string or set of strings that come to dominate the population.

This work provided indirect evidence that genetic algorithms use information about numerous partitions implicitly and in parallel. However, the degree to which a genetic algorithm is able to ultimately exploit this information is limited by the degree of consistency in the ranking that inherently exists in the function.

In this paper we extend the work of Whitley et al. by presenting evidence to support the hypothesis that the degree of nonlinearity of a function has a direct impact on the degree to which a function displays a consistent static ranking and deception. To this end we introduce the ϕ *spectrum* metric. This is a measure of ϕ with respect to all points in the search space using the static ranking of hyperplanes based on the fitness function.

Instead of using randomly generated functions (as Whitley et al. [4]), we look at functions with differing degrees of nonlinearity created using Walsh coefficients [1, 2] and measure the distribution of the nonlinearity with a new measure called the *Walsh sum*.

With these analysis tools we go on to show that the static metric, ϕ_{sum}, is a strong indicator of dynamical convergence behavior. In particular we show how the ϕ *spectrum* indicates which strings are most likely to survive in the population of a simple genetic algorithm. In fact, on average, the ϕ_{sum} of a function computed with respect to the point of convergence tends to be higher than the ϕ_{sum} associated with the global optimum.

2 Measures of Order

2.1 Basic Notation

Let L be the length of the binary strings which we are processing. A string of all 1's will be denoted by $\vec{1}$. A **schema** is one of the 3^L strings of 0's, 1's and $*$'s where a 0 or 1 occupy each **fixed bit position** and the $*$'s represent either a 0 or a 1 in the **variable bit positions**. A schema represents a **hyperplane** which is a set of strings with the same, possibly null, set of fixed bit positions. A schema with C fixed bit positions represents a hyperplane of **order** C containing 2^{L-C} strings. A **partition** is a set of competing hyperplanes specified by a string with bits b in positions of competition and $*$'s in all other positions. A partition with C b's and $(L-C)$ $*$'s is of **order** C and defines a set of 2^C hyperplanes.

A **ranking** of a partition is an ordering of all hyperplanes in the partition by a **ranking function** R. An **evaluation function**, $f(x)$, is a function that returns the fitness of a string x. A ranking function is often an extension of an evaluation function to include not just strings but hyperplanes, where the fitness returned for a hyperplane is the average of the fitnesses of all the strings in that hyperplane and is denoted by μ.

Hyperplanes in a partition can, for example, be ranked in terms of their average fitness μ. This is called a **static ranking**, since it doesn't change during the evolution of population in a genetic algorithm. Hyperplanes in a partition can also be ranked in terms of their proportional representation in a population at time t, denoted by $P(\xi, t)$ for hyperplane ξ. This is a **dynamic ranking** since it changes from one generation to the next.

2.2 Consistency

Using the ranking function $R = \mu$, all hyperplanes within a particular partition can be ranked according to their average fitness. This partition is said to have a **consistent static ranking** if there exists a **target string**, τ, such that when all the hyperplanes in the partition are sorted with respect to their average fitness, μ, each hyperplane closer in Hamming distance to the target string will have a greater fitness than any hyperplane farther away. Note that the target string can be any string in the space. For example, let ranking function $R(\xi) = \mu_\xi$, be the average fitness of the strings in hyperplane ξ. Assume the target string is $\vec{1}$. Consider the following fitness relationships for the partitions $bb**$ and $*bb*$ which have a consistent static ranking:

$$\mu_{11**} > \{\mu_{10**}, \mu_{01**}\} > \mu_{00**} \quad \text{and} \quad \mu_{*11*} > \{\mu_{*10*}, \mu_{*01*}\} > \mu_{*00*}$$

Notice that a given partition may support several choices for the target string. For example, the second partition could fully support 0110 as the optimal string as well as 1111. However, the first partition can't fully support 0110 since the fitness of $11**$ is better than $01**$.

Given the definition of consistency for a partition, we can use that definition to define consistency for a function. We say that a **function** has a **consistent static ranking** if all partitions can be consistently statically ranked with respect to the *same* target string. For example, if a fitness function for a string counts the number of 1 bits in the string, the maximum μ is at the string $\vec{1}$. Each partition is consistently ranked with respect to the same target string, $\vec{1}$. Therefore, the function is consistently statically ranked.

When functions are consistent, however, it is easily proved that they can only be consistent about the global optimum.

Theorem 1 *Functions can only be consistently ranked when the target string is the global optimum.*

Proof: Assume to the contrary that there exists a target string τ not equal to the global optimum. Since the function is consistent, all partitions are consistent with respect to the same target string τ. In particular the partition of all hyperplanes is consistent with respect to τ. This implies that τ is the global optimum which contradicts our assumption. Therefore the theorem is proven.**QED**

It can also be similarly proven that order $N-1$ hyperplanes can also only be consistent with the global optimum. More often than not, nonlinear functions are not consistent. A common conflict occurs when two partitions share fixed bit positions but the hyperplanes may be ranked such that the highest ranked hyperplane in one partition contains conflicting bits with the highest ranked hyperplane in the other partition. The following is an example of two partitions having an inconsistent static ranking no matter what target string is chosen:

$$\mu_{11**} > \{\mu_{10**}, \mu_{01**}\} > \mu_{00**} \quad \text{and} \quad \mu_{*00*} > \{\mu_{*10*}, \mu_{*01*}\} > \mu_{*11*}$$

If genetic algorithms are sampling hyperplanes according to their relative fitness then this inconsistent static ranking should have an impact on the performance of the genetic algorithm and its dynamical behavior. Note that if a simple genetic algorithm is applied to the example function above, as the population converges, the second bit position cannot be both a 1 and a 0; one or the other must come to dominate the population.

Let $P(\xi, t)$ denote the proportional representation of hyperplane ξ at generation t in a genetic algorithm. If a function has a consistent static ranking then this should have implications for a dynamic measurements based on $P(\xi, t)$. The orderings

$$P(11**, t) > P(00**, t) \text{ and } P(*00*, t) > P(*11*, t)$$

cannot both be true as the population reaches convergence. However, the orderings

$$P(11**, t) > P(00**, t) \text{ and } P(*11*, t) > P(*00*, t)$$

can both be true as the population reaches convergence. If static hyperplane relationships impact dynamical sampling of the hyperplanes, then a consistency metric should be a useful tool for characterizing functions and how they are processed by a simple genetic algorithm.

To support the comparison of the dynamic ranking of hyperplanes with static ranking, we say that a **partition** has a **consistent dynamic ranking** at time t if there exists a target string, such that when all the hyperplanes in the partition are sorted with respect to their proportional representation, $R(\xi) = P(\xi, t)$, each hyperplane closer in Hamming distance to the target string will have a greater representation than any hyperplane farther away. It is clear that a consistent dynamic ranking of a function is defined similarly to its static analog but with R being a dynamic measure.

2.3 Degree of Consistency for a Partition and the ϕ Metric

The notion of consistency is really a binary valued variable – either a partition or function is consistent or it is not. However, it is useful to quantify the amount of consistency inherent in a partition or function. This section describes the ϕ metric (as proposed by Whitley et. al. [4]) which can be used to measure the *degree of consistency* of a partition. The next section will expand the definition of ϕ to define the *degree of consistency* for a function.

The degree of consistency for an order k partition π containing hyperplanes $\xi_1, \xi_2, ...\xi_{2^k}$ is defined as follows[1]

$$\phi(R, \pi, \tau) = \sum_{i=1}^{2^k} \sum_{j=1}^{2^k} [\text{Pred}(R(\xi_i) > R(\xi_j)) \text{ Pred}(M(\xi_i, \tau) > M(\xi_j, \tau))] \ (M(\xi_i, \tau) - M(\xi_j, \tau))$$

where R is a ranking function of a hyperplane, π is a k order partition, τ is a target string[2] and Pred(*expression*) returns a 1 if the expression is true and 0 if it is false. $M(\xi, \tau)$ is a **match count function** that measures the number of bits that match between the target string τ and hyperplane ξ in the fixed bit positions. For example, $M(*1100*, 110100) = 2$. M can be thought of as an inverse Hamming distance function. In using ϕ we often assume $\tau = \vec{1}$. In these cases we denote $M(\xi, \vec{1})$ as $M(\xi)$ and $\phi(R, \pi, \vec{1})$ as $\phi(R, \pi)$.

The metric ϕ can be thought of as a summation of the difference between the match counts of all ordered pairs of hyperplanes in which the ordering of the pair by match count and by ranking agree in one direction. In short the two predicates decide if the weighting factor

[1] This definition is functionally the same as in Whitley et al. but expressed differently to more clearly show the conditions under which the value of ϕ is increased.

[2] The function to which this is applied is left out of the argument list to reduce the notational load.

determined by the difference of the match counts should be added to the grand total or not. We don't add in the difference of match count functions in the other direction, namely,

$$\text{Pred}(R(\xi_i) < R(\xi_j))\ \text{Pred}(M(\xi_i, \tau) < M(\xi_j, \tau))$$

since this would be a duplicate of the case where hyperplanes i and j are interchanged[3].

The resulting ϕ function has its largest value for a partition when a sort of all of the hyperplanes in the partition by increasing R results in the hyperplanes being sorted by increasing M. Thus the function has a larger value when the hyperplanes in π are more consistently ranked. ϕ is zero when the sort by ranking function results in the hyperplanes being sorted by decreasing M.

For the specific case of $R = \mu$ (denoted R_{stat}), we say ϕ measures the **degree of static ranking** for a partition. When $R = P(\xi, t)$ from a genetic algorithm population (denoted R_{dyn}), we say ϕ measures the **degree of dynamic ranking** for a partition at time t.

The maximum possible ϕ value for a partition π of order k can be computed [4] as:

$$\phi^{max}(\pi) = \sum_{q=1}^{k} \binom{k}{q} \sum_{i=0}^{q-1} \binom{k}{i} (q - i) \quad \text{where} \quad k = order(\pi)$$

Notice that ϕ^{max} does not take τ or R as arguments since the target string and ranking function are irrelevant to computation of the maximum possible ϕ. ϕ^{max} allows ϕ to be normalized by dividing each count so that the resulting measure for each partition is between 0 and 1.

$$\widehat{\phi}(R, \pi, \tau) = \phi(R, \pi, \tau)/\phi^{max}(\pi) \qquad \in [0, 1]$$

Table 1 shows an example computation of $\phi(R, \pi)$ for two different ranking functions R_1 and R_2 and the partition $\pi =$*bbb****. The definition of the ranking functions are not specified, but their values are given in the table. The table is sorted by R from greatest to least to aid hand verification of the results at the bottom of the table.

R_2 provides a good example of the computation of ϕ. The two sums in the formula for ϕ examine all possible ordered pairs. If we take the first two hyperplanes for example. We evaluate the product of the two predicates as:

$$\text{Pred}(R(111{*}{*}{*}) > R(100{*}{*}{*}))\ \text{Pred}(M(111{*}{*}{*}, \vec{1}) > M(100{*}{*}{*}, \vec{1}))$$

which becomes the product $\text{Pred}(21.0 > 13.0)\ \text{Pred}(3 > 1)$. Since both predicates are true their product returns 1. This is multiplied by

$$(M(111{*}{*}{*}, \vec{1}) - M(100{*}{*}{*}, \vec{1}))$$

giving a value of 2 which is added to the sum. The other terms are similarly computed to produce the final calculation, $\phi(R_2, \pi) = 20$, which is normalized using $\phi_{max} = 30$ to yield $\widehat{\phi}(R_2, \pi) = 0.66\overline{6}$.

$\phi(R, \pi, \tau)$ exhibits an interesting property for $R = R_{dyn}$ in a simple genetic algorithm with no mutation. As the population converges, a single individual eventually becomes the only

[3] In an actual implementation there is much room for taking advantage of symmetry and other tricks to improve the performance of this calculation.

ξ_i	$R_1(\xi_i)$	$M(\xi_i)$	ξ_i	$R_2(\xi_i)$	$M(\xi_i)$
111***	19.0	3	111***	21.0	3
110***	17.0	2	100***	13.0	1
101***	13.0	2	010***	8.0	1
011***	11.0	2	101***	5.0	2
001***	7.0	1	001***	3.0	1
010***	5.0	1	011***	2.0	2
100***	3.0	1	000***	1.0	0
000***	2.0	0	110***	1.0	2
$\phi(R_1,\pi)=30$			$\phi(R_2,\pi)=20$		
$\widehat{\phi}(R_1,\pi)=1.0$			$\widehat{\phi}(R_2,\pi)=0.66\overline{6}$		

Table 1: Computations of ϕ for two Example Rankings.

the member of the population. This means that $Pred(R(\xi_i) > R(\xi_j))$ is true for only one value of i in each partition while the remaining hyperplanes are *tied*. Ties occur quite often using R_{dyn} but rarely using R_{stat}. The method for handling ties is to rank the tied hyperplanes in reverse order in terms of Hamming distance from τ. This is done so that ties contribute nothing to the value of ϕ. As a population converges, the increasing number of ties for the dynamic ranking results in a decrease in the attainable value of ϕ. For this reason our experiments limited the observation of ϕ to the first 20 generations.

2.4 Degree of Consistency for a Function and the ϕ_{sum} Metric

As stated earlier, functions are usually not consistent. For this reason, a measure of the degree of consistency of a function can be defined. We define the **degree of consistency** for a function with respect to a target string τ as:

$$\phi_{sum}(R,\tau) = \sum_{\pi \in P} \widehat{\phi}(R,\pi,\tau), \qquad R = R_{stat}$$

where P is the set of all partitions. Thus, ϕ_{sum} is measured over 2^L partitions and ranges from 0 to $2^L - 1$. The maximum degree of consistency, $2^L - 1$, occurs only when a function is consistent.

2.5 The Deception Count Metric

Discussions of deception typically indicate whether functions are not deceptive, partially deceptive or fully deceptive. Deception occurs when a hyperplane with the best average fitness in a partition does not contain the global optimum. We introduce the notion of a **deception count** to measure the *number* of partitions that are deceptive with respect to a given target string. Note that ϕ_{sum} and the deception count are closely related, but the deception count is a coarser measure. Formally, the deception count is defined as:

$$\mathcal{D}(R,\tau) = \sum_{\pi \in P} \mathrm{NotOpt}(R,\pi,\tau)$$

where NotOpt is a function that returns 1 if the hyperplane with the highest average fitness in partition π does **not** contain the target string τ and 0 otherwise. This function ranges

in value from 0, for functions that at least have τ in the highest ranked hyperplane in every partition, to $2^L - 1$, for the case when τ appears in none of the highest ranked hyperplanes. Again, $\mathcal{D}(R, \vec{1})$ can be simplified to $\mathcal{D}(R)$ since all functions can be rotated such that the optimum occurs at $\vec{1}$.

2.6 Orders of Interaction and the Walsh Sum Metric

Nonlinearity is one of the major factors that makes search difficult. One way to look at the amount of interaction between alleles is to examine the Walsh coefficients of the function. For an L bit function there are 2^L values in the domain of the function. There are also 2^L Walsh coefficients, denoted by w_i, needed to represent the function.

For our experiments we wished to generate functions with limited nonlinearity. For this reason we created a condensed measure of the degree of interaction. Unlike the Walsh coefficients, which allow for the complete reconstruction of the function, the Walsh sums preserve only the orders of interactions. Let the **Walsh sum**, denoted by W_b, be the sum of the absolute value of the coefficients for Walsh functions with exactly b bits set to 1. This is said to be the **order** b Walsh sum. For example $W_0 = |w_0|$, where w_0 is the only Walsh coefficient with no 1 bits, and $W_1 = |w_1| + |w_2| + |w_4| + \ldots$, where each w_i has only one 1 bit set. W_2 is the sum over the $\binom{L}{2}$ Walsh coefficients for the pairwise interactions between bits, etc. This means that for an L bit function there are $L + 1$ Walsh sums.

The **order of a function**, $\Omega(f)$, is defined as the largest i such that $W_i \neq 0$. So, for example, if $\Omega(f) = 1$ then the function is bitwise linearly independent and there are no pairwise or higher order interactions.

3 Experimental Design

3.1 Generating Test Functions

Our goal was to examine the behavior of a simple genetic algorithm on functions with controlled nonlinearity. We wanted to ensure a good diversity of functions along two measures: internal consistency as measured by ϕ_{sum} and limited order of interactions as measured by Ω. We generated 6 sets of 100 8 bit test functions with $\Omega(f)$ equal to 1 through 6.

The process of generating each test function began by randomly generating the Walsh coefficients for the function in the range $[-1, 1]$. This created a set of 2^8 Walsh coefficients for each 8 bit function. If the index of the Walsh coefficient had a number of 1's greater than the intended Ω for the function then the coefficient was set to 0 to limit the degree of interaction.

For any given $\Omega(f)$, a scaling factor was devised so that the W_i would decay linearly starting at $W_0 = 1.0$ decaying to $W_{(\Omega+1)} = 0$. These factors were applied to the Walsh coefficients. Our intent was to let the Walsh sums at higher orders, and hence the nonlinearity at higher orders, decay smoothly to zero. We conjecture that many naturally occurring optimization problems may exhibit some sort of bounded nonlinearity. A second feature of the scaling factor was that it fixed the value of Walsh sums at a constant from one function to the next so that fluctuations in interaction levels are not a variable in our experiments.

b	W_b	RawSum	NumCoefs	%Nonzero
0	1	1	1	100.00
1	0.8333	-0.7458	8	100.00
2	0.6667	0.04343	28	100.00
3	0.5	-0.1441	56	100.00
4	0.3333	-0.01136	70	100.00
5	0.1667	-0.02983	56	100.00
6	0	6.787e-17	0	0.00
7	0	-1.691e-17	0	0.00
8	0	-6.939e-18	0	0.00

Table 2: Table of Walsh Sums for a Function from our Experiments with $\Omega = 5$.

From the Walsh coefficients a complete function table of 2^8 elements was generated[4] that mapped the function domain to its range. The function was then rotated in the domain space so that the maximum was at $\vec{1}$. This has no effect on the Walsh sums [work in progress] or on the value of ϕ_{sum}, since a rotation does not disturb the Hamming distance or the sets of hyperplane pairs that satisfy the predicates in the formula for ϕ.

The particular infinite population model for a genetic algorithm that we used for our experiments required that the values of the functions be nonnegative everywhere. The generated functions that had negative minimums were translated by subtracting the value of the minimum, $f(min)$, from each element in the function table. This translation is often done when using genetic algorithms and has no effect on the value of ϕ_{sum}. The value of W_0 will now be $W_0 - f(min)$ and all other W_i will remain constant. The results we present for the translated set of functions will be the same for the untranslated functions.

Table 2 is an example of a Walsh sum table for a function with $\Omega = 5$ that we used in our experiments. In the first column is the order of the Walsh sum, b. The second is the Walsh sum W_b. The third is the sum of the Walsh coefficients of order b without taking the absolute value. The last two columns are the count of the number of Walsh coefficients of order b that are nonzero and what percentage this count is of the total possible Walsh coefficients of order b. Notice in column 3 that Walsh sums of order greater than 5 are essentially zero (within round off error). Also note the linear decay of the Walsh sums.

3.2 Deception, ϕ and the Walsh Sums

The amount of deception underlying any function will affect genetic algorithm behavior. As the genetic algorithm processes hyperplanes, it will allocate a higher proportional representation to hyperplanes with higher average fitnesses. If the underlying function is deceptive, the hyperplanes with higher average fitnesses may be driving the genetic algorithm away from the global optimum. In terms of partitions and hyperplanes, deception can be defined as occurring when the highest ranked hyperplane is not the hyperplane in which the global optimum occurs.

A simple deception count can illustrate the complexity differences that occur as the nonlinearity of functions, as measured by Ω, is increased. The graphs in Figure 1 show the *average*

[4]This was done using the fast Walsh transform of Cooley and Tukey which is explained in [1].

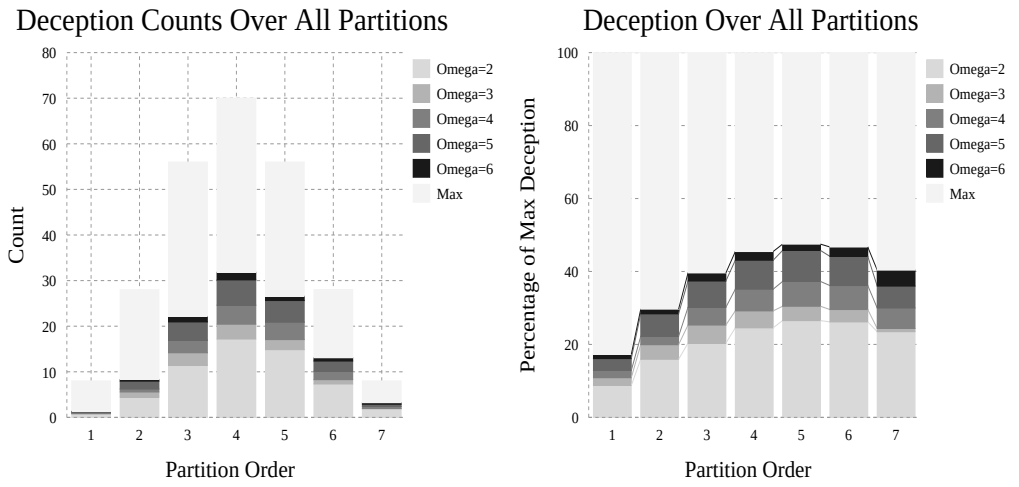

Figure 1: Sum of Deception Counts versus the Different Orders of Functions

deception count $\mathcal{D}(R_{stat}, \tau_{opt})$ about the optimum string τ_{opt} for the sets of 100 functions of each order 2 through 6 broken down by the orders of the partitions at which the deception occurs. The bars in the graphs are overlapping so that the value for each Ω is the full height of the bar from the base of the graph. The deception count is bounded by the maximum number of partitions possible for each given partition order $\binom{L}{n}$ where L is the length of the string and n is the order of the partition. The maximum for each of the partition orders is included in the graphs and is denoted by **Max**. Note that number of partitions at the various orders (and hence Max) is binomially distributed. The histogram on the left shows the increase in deception across all partition orders as the amount of nonlinearity in the functions increases. The graph on the right shows the same numbers but as a percentage of possible partitions that could be deceptive. Clearly, there is more deception occurring in higher order partitions than in lower order partitions. Functions of order 1 were not plotted since they display no deception.

The ϕ metric relates to deception in that it measures how *consistent* the ranked hyperplanes belonging to a partition are with respect to a target string (in this case, the target string is the global optimum). When deception occurs in a given partition, the ϕ for that partition will be very low indicating that the ranking of the hyperplanes in that partition are not consistent with the global optimum. In other words, as the deception increases, the ϕ decreases. The deception counts are strongly negatively correlated with the ϕ values for all 600 functions (the Pearson's r correlation varied from -0.9 to -0.85 across all six sets of test functions). Therefore, in terms of ϕ, the ϕ_{sum} will become smaller as Ω is increased indicating less *consistency* between the ranked hyperplanes and the global optimum as nonlinearity increases.

3.3 A Model of the Genetic Algorithm

The dynamic analysis was done using an executable infinite population model of a simple genetic algorithm with 1-point crossover [3, 5] to model the expected trajectory of the search. An infinite population model was chosen because it is less susceptible to truncation

Ω	Gen 1	Gen 5	Gen 10	Gen 20
1	0.9994	0.9984	0.9975	0.9223
2	0.9985	0.9748	0.9339	0.8639
3	0.9990	0.9791	0.9417	0.8622
4	0.9993	0.9870	0.9641	0.8954
5	0.9994	0.9731	0.9320	0.8525
6	0.9991	0.9747	0.9281	0.8454

Table 3: Pearson's r correlation between Static and Dynamic Ranks for functions with $\Omega = 1$ through 6 (sets of 100 functions per Ω).

effects caused by having a fixed number of individuals. Since there is no sampling taking place, the infinite population model also offers a benefit in eliminating the need for multiple trials as would be required if a finite population were used. The crossover rate was set at 0.6 and there was no mutation[5]. Initially each string had an equal representation in the population. At the end of each generation t the proportional representation of each individual was collected, allowing us to compute $\phi_{sum}(R_{dyn}(t))$ for each generation, where the dynamic ranking function $R_{dyn}(t)$ is based on $P(\xi, t)$.

In the next section we will examine how static and dynamic ranking are related and how static ranking is a strong indicator of the survivability of a string in a population and of the point of convergence for a genetic algorithm. We then compare performance of both an infinite population model and sample runs of a finite population simple genetic algorithms using various static measures of the function being optimized.

4 Results

4.1 Static versus Dynamic Ranking

Our initial experiment examines the relationship between static and dynamic ranking. In Whitley et. al. [4], it was shown that the correlation between static and dynamic ranking is highest at earlier generations. We also present similar results here; however, our results are broken out by orders of interaction using 6 sets of functions with Ω equal to 1 through 6. Figure 2 shows the static ranking ($\phi_{sum}(R_{stat})$) plotted against the dynamic ranking ($\phi_{sum}(R_{dyn}(t))$) at generations 0, 1, 10 and 20. Results were generated for sets of 100 functions for each of the 6 Ω's; however, only the first 20 of the 100 functions for each Ω were used to simplify the plots. Each of the 120 points on the graph corresponds to a single function. At generation 0, there is no dynamic ranking; that is, all hyperplane of order π tie with an equal representation. Therefore, all functions have the same $\phi_{sum}(R_{dyn}(0))$. At generation 1, however, $\phi_{sum}(R_{stat})$ and $\phi_{sum}(R_{dyn}(1))$ are very strongly correlated which is in part due to the fact that the genetic algorithm uniformly samples the space only at the first generation. As the genetic algorithm continues execution, the correlation between the static and dynamic ranks decreases as can be seen in the two lower plots corresponding to generations 10 and 20.

[5] We will study the effects of mutation in future work.

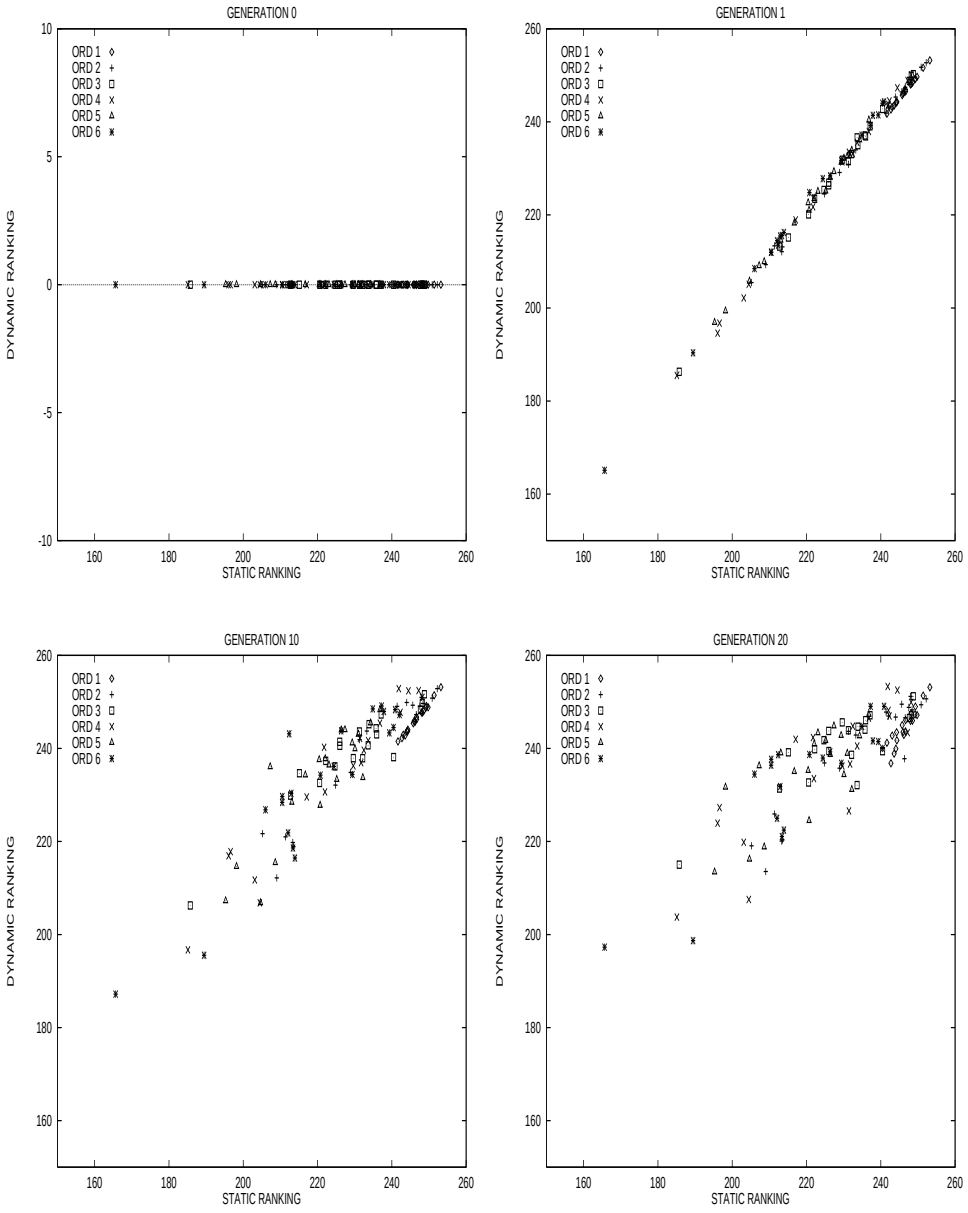

Figure 2: Static Ranking versus Dynamic Ranking for Generations 0, 1, 10 and 20.

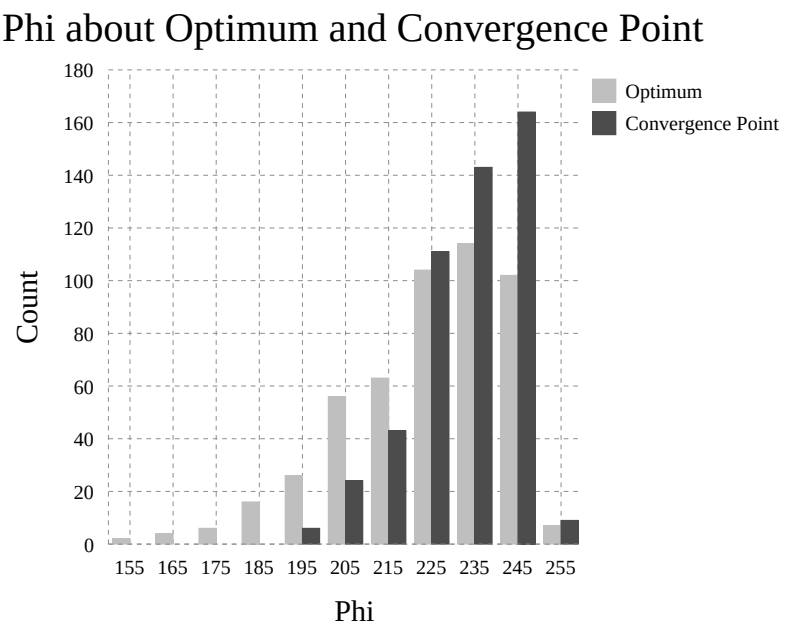

Figure 3: Count of the Number of Occurrences Grouped by Static Ranking About the Global Optimum and Point of Convergence.

Table 3 illustrates that the correlation between static and dynamic ranks also changes as we move from lower to higher orders of nonlinearity. The table was generated using the full set of 100 functions per Ω. Given the correlation can range from 1 to -1 (1 being perfectly correlated and -1 begin perfectly negatively correlated), it is clear that the static and dynamic ranks are still strongly correlated at the 20th generation. However, the trend moving from lower to higher orders of nonlinearity is that the correlation between the static and dynamic ranks decreases. We believe this behavior is due to the decreasing *consistency* in functions as Ω is increased.

4.2 Static Ranking and Point of Convergence

We have illustrated that the static ranking is strongly correlated with the dynamic ranking using functions with varying degrees of nonlinearity. However, this does not answer the question of whether or not the static ranking can be used to predict the convergence behavior of a simple genetic algorithm. Since ϕ can be computed about arbitrary points in the search space, we compare ϕ at (i.e., with respect to) the global optimum, $\phi_{sum}(R_{stat}, \tau_{opt})$ and ϕ at the point of convergence τ_C, $\phi_{sum}(R_{stat}, \tau_C)$. Using the set of 600 functions with varying Ω's, we found that in 93% of the cases where the point of convergence was some string other than the global optimum, $\phi_{sum}(R_{stat}, \tau_C)$ was higher than $\phi_{sum}(R_{stat}, \tau_{opt})$.

The histogram in Figure 3 compares the distribution of $\phi_{sum}(R_{stat})$ using the infinite population model with respect to both the global optimum and the point of convergence. Despite the fact that the global optimum was also the point of convergence in 70% of the cases, the

Ω	Average ϕ_{sum} for Convergence Point		Average ϕ_{sum} for Optimum Point	
	Mean	Std Dev	Mean	Std Dev
1	246.27	3.14	246.27	3.14
2	238.92	9.43	235.06	11.79
3	233.92	11.48	229.57	15.16
4	234.03	11.48	224.13	18.27
5	229.34	12.21	217.46	17.35
6	227.40	13.23	213.84	20.09

Table 4: Average ϕ_{sum} for Each Ω Tested.

graph clearly shows a larger $\phi_{sum}(R_{stat})$ value associated with the point of convergence than with the global optimum.

The results from the histogram can be partitioned based on the varying Ω's. Table 4 shows the average ϕ_{sum} for both the convergence point and the optimum. For **convergence point** we used the string in the population with the highest proportional representation at the end of generation 20. It is easy to see that, **on average**, for the sample functions ϕ_{sum} of the converged point is greater than or equal to the ϕ_{sum} for the optimum. The means are equal for the order 1 functions since all bitwise linearly independent functions always converge to the optimum. Note that the means of the ϕ_{sum}'s decrease as the Ω increases. This can be explained using the earlier observation that as the degree of nonlinearity is increased functions become less *consistent*. When functions become less *consistent*, the mean ϕ_{sum} for those function will be lower than that of functions that are more *consistent*. Also note the standard deviation of the ϕ_{sum} increases as the Ω increases. The increased degrees of freedom afforded functions with higher degrees of nonlinearity creates a more diverse set of functions yielding a wider possible range of attainable ϕ_{sum}'s.

Table 5 shows for each set of functions, $\Omega = 1$ through 6, the number of times the infinite population model genetic algorithm converged to a point at a given Hamming distance after 20 generations. For instance, of the 100 fourth order functions tested, 11 converged to a point 2 bits away from the optimum point. A Hamming distance of zero, of course, indicates that functions converged to the optimum. Two observations should be made from these results. First, as Ω increases, the genetic algorithm becomes less likely to converge to the global optimum. Second, as Ω increases, the Hamming distance between the point of convergence and the global optimum increases.

4.3 The ϕ Spectrum

Comparing the convergence point and the global optimum illustrates that ϕ generally results in higher values for the convergence point than the global optimum. However, in order to determine how accurate the ϕ metric is at predicting the convergence point, the ϕ_{sum} at the convergence point needs to be compared to ϕ_{sum} at all other points in the space. We define the ϕ **spectrum** to be a vector in which each element is associated with one of the 2^L possible strings in the population. The value of each element is $\phi_{sum}(R, \tau)$ computed using the string associated with that element as the target string τ. Each element in the vector therefore represents the degree to which that string could be "supported" as a potential

Ω	Hamming Dist.					% Converged
	0	1	2	3	4	to Nonoptimum
1	100	0	0	0	0	0%
2	78	18	4	0	0	22%
3	78	19	2	1	0	22%
4	60	28	11	1	0	40%
5	56	26	15	2	1	44%
6	52	31	9	7	1	48%

Table 5: Hamming Distance from the Optimum for $\Omega = 1$ through 6.

solution by ranking R. With ϕ *spectrum* and $R = R_{stat}$, we get a much broader view of the biases that exist in the ranking information inherent in the function than with the ϕ_{sum} relative to a single target string.

The scatter-plot in Figure 4 illustrates a strong relationship between the ϕ *spectrum* and the proportional representation of strings in a simple genetic algorithm at generation 20. The points in the graphs represent each possible string τ in the search space. On the x-axis is the $\phi_{sum}(R_{stat}, \tau)$. On the y-axis is $P(\tau, 20)$, i.e. the string's proportional representation in the infinite population at generation 20. The graph shows that the strings with highest proportional representation in the population are also those strings that have the highest $\phi_{sum}(R_{stat}, \tau)$. It also drives home the point that strings with even moderate $\phi_{sum}(R_{stat}, \tau)$ values have little chance of competing for representation in the population. These graphs universally represent the behavior found for all the functions tested regardless of their order.

When we computed the ϕ *spectrum* for all 600 functions, we observed that the point of convergence did not always produce the highest $\phi_{sum}(R_{stat}, \tau)$ measures in the ϕ *spectrum*, but the point of convergence was always among the highest values in the ϕ *spectrum*. Bitwise linear functions with $\Omega = 1$ always had the convergence point (the global optimum) ranked highest in the ϕ *spectrum* and were not included in any analysis. Figure 5 shows where the convergence point ranked in the ϕ *spectrum* for the functions with $\Omega = 2$ through 6. For example, over 50 of the order 2 functions had their convergence points ranked highest in the ϕ *spectrum*. Unlike previous bar graphs, the bars are additive rather than overlapping in order to have the total height of the bar represent the total count for all 500 functions. What is clear from the graph is that the convergence points usually rank very high in the ϕ *spectrum* over all functions. In fact, 89.4% of the convergence points fell into the top 5 positions in the ϕ *spectrum*. This experiment suggests that the static metric, $\phi_{sum}(R_{stat}, \tau)$, is a strong indicator of dynamical convergence behavior.

4.4 Infinite Population versus Finite Population

The infinite population model of the simple genetic algorithm is a useful tool for studying the various metrics described here because it simplifies the problem of modeling the genetic algorithm. However, we contend that ϕ will have similar performance prediction capabilities for a finite population simple genetic algorithm as it has for the infinite population model. In Figure 6 we compare the results of the infinite population model and a finite population simple genetic algorithm run on the same functions. In both cases, the probability of crossover is 0.6, no mutation is used and they were both run to 20 generations. The left

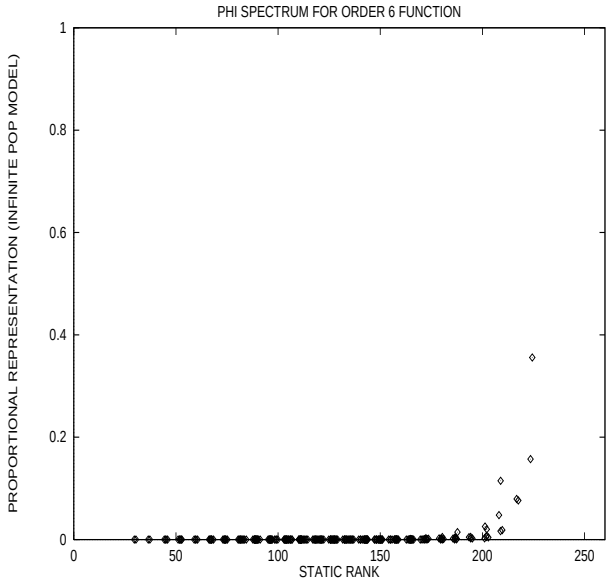

Figure 4: Static Ranking Versus Proportional Representation for all Points in the Search Space for an $\Omega = 6$ Function at Generation 20.

column is the plot of the infinite population model run on one randomly chosen function in each of two orders: $\Omega = 2$ and $\Omega = 6$. The right column corresponds to 5 runs of a finite population simple genetic algorithm with a population of 50 using the same functions used by the infinite population model. The graphs show $P(\tau, 20)$ versus $\phi_{sum}(R_{stat}, \tau)$. Note that the behavior of the finite population examples is not radically different from the infinite population model for the case where the population size was less than 20% of the total possible number of strings. This indicates that the an extrapolation from our infinite population model to moderate finite populations is not unreasonable.

5 Conclusions

The purpose of this paper has been to relate increased nonlinearity with hyperplane ranking and ultimately with the convergence of a simple genetic algorithm. To achieve this goal, we first required a new function generation technique so that functions with a controlled amount of nonlinearity could be generated. Using Walsh coefficients and Walsh Sums enabled us to generate large sets of functions with fixed degrees of nonlinearity.

The majority of this paper is dedicated to studying the ϕ metric in order to analyze the relationships between hyperplanes with the underlying belief that those relationships will affect the convergence behavior of the simple genetic algorithm. To this end, we refine the notion of consistency as introduced by Whitley et. al. [4]. We relate consistency and the ϕ metric with deception and illustrate that increasing nonlinearity results in more deception and a lower ϕ, and thus a lower degree of consistency. We empirically show that the static ϕ is strongly correlated with the dynamic ϕ which illustrates that there is a relationship

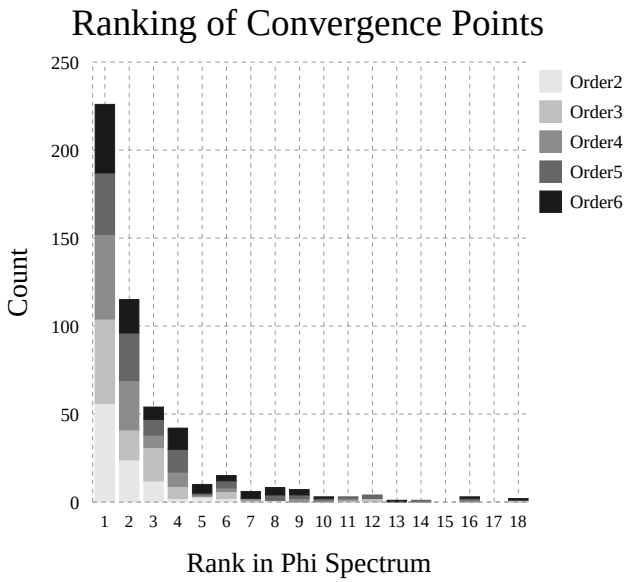

Figure 5: Cumulative Counts Grouped by Ranking in the ϕ *Spectrum* of the Convergence Point.

between average hyperplane fitness and proportional representation of a hyperplane in the execution of a simple genetic algorithm.

The most compelling results we show are using the ϕ spectrum. These results show that the degree of consistency is (more often than not) higher for the point of convergence in the simple genetic algorithm than for the global optimum. Upon closer examination, the vast majority of the convergence points were found to occur in the top five positions in the ϕ spectrum across 600 functions. These results indicate that the ϕ metric can be a strong indicator of dynamical convergence behavior for a simple genetic algorithm. We are extending this work to include mutation in both the infinite population model and the finite population simple genetic algorithm.

This research was supported by NSF grant IRI-9503366 and by the Colorado Advanced Software Institute (CASI). Soraya Rana was supported by a National Physical Science Consortium Fellowship.

References

[1] David Goldberg. Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems*, 3:129–152, 1989.

[2] Collin Reeves and Christine Wright. An Experimental Design Perspective on Genetic Algorithms. In D. Whitley and M. Vose, editors, *FOGA - 3*, pages 7–22. Morgan Kaufmann, 1995.

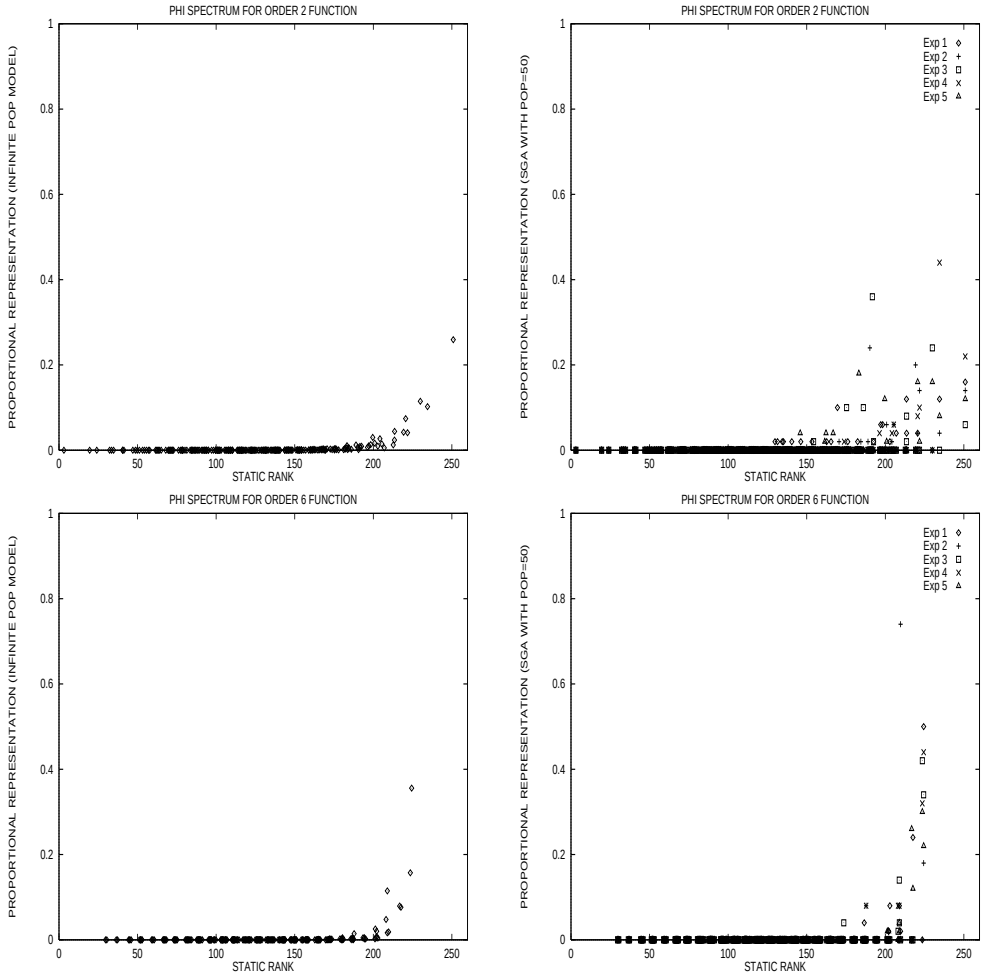

Figure 6: A Comparison of Convergence in Infinite and Finite Populations at Generation 20.

[3] M. Vose and G. Liepins. Punctuated Equilibria in Genetic Search. *Complex Systems*, 5:31.44, 1991.

[4] Darrell Whitley, Keith Mathias, and Larry Pyeatt. Hyperplane Ranking in Simple Genetic Algorithms. In L. Eshelman, editor, *Proc. of the 6th Int'l. Conf. on GAs*. Morgan Kaufmann, 1995.

[5] L. Darrell Whitley. An Executable Model of the Simple Genetic Algorithm. In L. Darrell Whitley, editor, *FOGA - 2*, pages 45–62. Morgan Kaufmann, 1993.

APPENDIX

A Theorems and Observations about Linear Functions

We now present several theorems and observations relating the ϕ metric to bitwise linear functions. We assume functions are rotated with the optimum at $\vec{1}$.

Whitley et al. [4] have previously proven the following theorem.

Theorem 2 *Given a bitwise linear function F, F has a consistent static ranking (has maximum ϕ in every hyperplane partition) if and only if the set of all strings in the search space have a consistent static ranking.*

Note that nonlinear functions, such as linearly dominated functions, may also have a consistent static ranking.

Theorem 3 *For a static ranking function R based on a bitwise linear evaluation function F with a unique optimum at $\vec{1}$, sort the bitwise weight coefficients ω_i in descending order and index the sorted set from from L to 1. The smallest possible ϕ_{sum} occurs for linear functions that have weights where $\omega_L > \sum_{i=1}^{L-1} \omega_i$, $\omega_{L-1} > \sum_{i=1}^{L-2} \omega_i$ and in general, $\omega_j > \sum_{i=1}^{j-1} \omega_i$.*

Proof: Since the function is linear, the actual position of the various bits is irrelevant. Without loss of generality we pick a prototype case that induces this particular ranking, then discuss the ϕ metric and partitions for linear functions having minimal ϕ values with respect to this prototype case. Such a function is the standard binary function that maps binary strings to integers; we will denote the resulting ranking by R_B.

The proof is inductive. The proof is constructed to apply to a set of strings, but as will be shown it also applies to subsets representing hyperplanes.

For strings of length 1, 2 and 3 one can show by exhaustive enumeration that R_B results in the minimal ϕ compared to all other ordering that are consistent with linear functions.

Assume R_B represents the ranking that results in the lowest possible ϕ for sets of strings of length L or less. We show that the R_B ranking then results in the lowest ϕ for a linear function over a set of strings of length L+1. Take the set of strings at length L sorted according to R_B. Construct the $L+1$ ordering of binary strings by making two copies of the set of the strings of length L. Construct set C1 by appending a 1 to the beginning of the strings in the first copy of the set, and construct set C2 by appending a 0 to the second copy. Since the function is linear, the rankings among members of the sets C1 and C2 must be the same as the rankings of the original set of strings of length L. To conclude the proof, we must now show that all members of C1 must be ranked higher than all members of C2 in order to minimize ϕ. This corresponds to showing that the weight ω_{L+1} associated with the new bit must larger than the sum over all the other bits.

When comparing members of C1 and C2 note that for any binary string X of length L, the string 1X must be ranked before any string of the form 0X in the overall ranking, since the contribution of bits for substring X is identical for linear functions. We now show that $\omega_{L+1} > \sum_{i=1}^{L-1} \omega_i$ is necessary to minimize ϕ; it is also sufficient since it produces a total ordering over the set of strings.

The lowest ranked string in C1 is 10^L and the highest ranked string in C0 is 01^L. Assume the weight associated with the new bit is not greater than the sum of the other weights; then the ranking of 10^L drops below 01^L which increases ϕ.

Similarly, and in general, one can prove inductively that for sets C1 and C2, there are more 1 bits in the top K strings than in the bottom $2^L - K$ strings when C1 and C0 are sorted best to worst by evaluation (we assume $K < 2^L$ since no string can drop below 00^L in rank). Thus, any string of length L+1 that starts with a 1 that moved down in rank below strings that start with 0 increases ϕ since 1X cannot move below 0X and moving strings with more 1 bits to higher ranked positions makes the overall ranking more consistent with the target $\vec{1}$ and increases ϕ.

Ignoring bit position without loss of generality, any ranking other than one consistent with R_B results in an increase in ϕ, given that the function is linear. Thus, the ranking represented by R_B results in the lowest possible ϕ for a linear function over the partition composed of strings.

Finally, we use the following theorem to show that if the the set of all strings has a minimal ϕ, every partition of a linear function also has a minimal ϕ. **QED**

Theorem 4 *Given a linear function F with the lowest possible ϕ over the set of all strings, each partition of the space has the lowest possible ϕ compared to the possible ϕ value for the same partition over all possible linear functions.*

Proof: Since the functions are linear, each partition of order-k can be treated as a linear function with string length k since the values contributed by each bit are independent. We convert each hyperplane ξ in partition π into a string, Z_ξ, by *discarding* $*$ symbols in ξ. Given the pattern established for characterizing the lowest possible ϕ value for a linear function over a set of strings, it follows that when the k relevant bits are still sorted in descending order and after reindexing the remaining bits that $w_j > \sum_{i=j}^{k} w_i$ still holds. Since every partition of order k can be converted into a linear function over strings of length k, it follows that the resulting ϕ must be minimal for that partition. **QED**

Minimal ϕ for Linear Functions

We now show how to compute the minimum possible ϕ value for each partition of a linear function whose maximum is at $\vec{1}$. We present the results without a proof, although we have verified the results empirically. Since any order-k partition of a linear function over L bits can be converted to a linear function over k bits, the result generalizes as a method to compute the worst case ϕ for all partitions.

First, we take advantage of the recursive structure of the bit patterns over the set of standard binary strings when sorted from largest to smallest. Figure 7 shows this pattern for bit strings of length 3 and shows how ϕ can be computed in a recursive fashion for this particular set of strings. The minimal ϕ for strings of length 1 is given by summing over the terms in the inner most box shown under the recursive ϕ (i.e., $\phi = 1$). The minimal ϕ for strings of length 2 is given by summing over the terms in the middle box ($\phi = 3 + 1 + 1 + 1 = 6$). Finally, the minimal ϕ for strings of length 3 is given by summing over all the terms in the outer-most box. Note these boxes follow the recursive structure of the binary encoding as shown. Thus, the recursive ϕ at length L+1 first duplicates the recursive *phi* at length L,

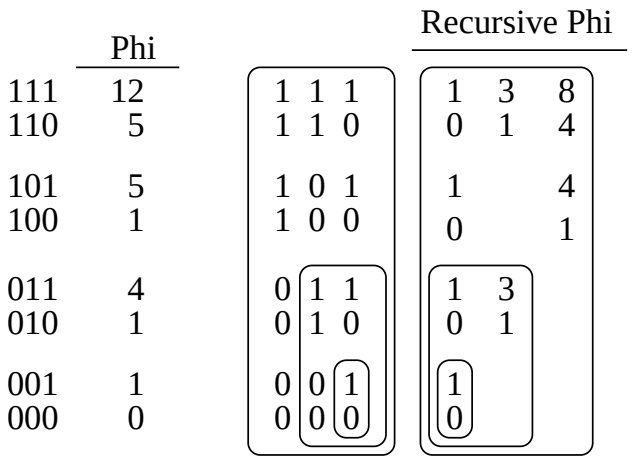

Figure 7: Recursive Phi.

then adds in additional terms. So how are these new terms computed?

The critical values can be reduced to the following nonrepeating matrix form, which is extended in this example to cases with string length 6. As will be shown, we can use the binomial distribution to get back the original repetition of values.

```
1  3  8  20  48  112  256
   1  4  12  32   80  192
      1   5  17   49  129
          1   6  23   72
              1   7  30
                  1   8
                      1
```

We will refer to this upper triangle matrix as the T-matrix. We will index T from 0 to $L-1$ (i.e., column index j corresponds to L-1). The diagonal of the matrix corresponds to the contribution associated with the recursive ϕ for strings of length 1 (or $0^{L-1}1$ compared to 0^L), thus:

$$t(i,j) = 1 \text{ when } i = j$$

The sum of the first row of the T-matrix corresponds to the contribution to ϕ associated with $\vec{1}$. Since $\vec{1}$ dominates every other string, it dominates $\frac{2^L L}{2}$ zeros which is also its contribution to ϕ. Thus, $\sum_{i=0}^{j} t(0,i) = \frac{2^L L}{2} = 2^{L-1}L$. Therefore the first (i.e., 0th) row of the matrix is given by:

$$t(0,j) = 2^j(j+1) - \sum_{i=0}^{j-1} t(0,i)$$

$$t(0,j) = 2^j(j+1) - (2^{j-1}j)$$

Empirically, we observe that for all $j > i, i > 0$ the matrix has the following properties:

$$t(i,j) = \sum_{k=0}^{j-1} t(i-1,k)$$

$$t(i,j) = \left[\sum_{k=0}^{j-2} t(i-1,k)\right] + t(i-1,j-1)$$

$$t(i,j-1) = \sum_{k=0}^{j-2} t(i-1,k)$$

$$t(i,j) = t(i,j-1) + t(i-1,j-1)$$

We can prove the correctness of this observation only for select rows of the T-matrix (those rows for which we can compute a closed form for the sum of the elements in that row), but we do not have a general proof. However, we have empirically verified the T-matrix.

Assuming the T-matrix is correct, let $F_\phi(k)$ be a recursive function that generates the minimal ϕ over a partition of strings of length k when strings are ranked using R_B.

$$F_\phi(k) = 2F_\phi(k) + \sum_{j=0}^{k-1} \binom{k-1}{j} t(i,j) \quad ; \qquad F_\phi(1) = 1$$

By induction, one can prove that

$$F_\phi(k) = \sum_{i=0}^{k} 2^{k-1-i} \left[\sum_{j=0}^{i} \binom{i}{j} t(i,j)\right].$$

Thus,

$$\phi_{sum} = \sum_{i=1}^{L} \binom{L}{i} \frac{F_\phi(i)}{Max(\phi(R,\pi),i)}$$

is the lower bound on the possible ϕ_{sum} value for a linear function of length L.