Predicting Epistasis Directly from Mathematical Models

Robert B. Heckendorn (heckendo@cs.colostate.edu)*
and Darrell Whitley (whitley@cs.colostate.edu)
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523 USA
(970) 484-9903

April 10, 1998

Abstract

Classically epistasis is either computed exactly by Walsh coefficients or estimated by sampling. Exact computation is usually of theoretical interest since the computation typically grows exponentially with the number of bits in the domain. Given an evaluation function, epistasis also can be estimated by sampling. However this approach gives us little insight into the origin of the epistasis and is prone to sampling error.

This paper presents theorems establishing the bounds of epistasis for problems that can be stated as mathematical expressions. This leads to substantial computational savings for bounding the difficulty of a problem. Furthermore, working with these theorems in a mathematical context, one can gain insight into the mathematical origins of epistasis and how a problem's epistasis might be reduced. We present several new measures for epistasis and give empirical evidence and examples to demonstrate the application of the theorems. In particular, we show that some functions display "parity" such that by picking a well-defined representation, all Walsh coefficients of either odd or even index become zero, thereby reducing the nonlinearity of the function.

Keywords

Epistasis, Function Order, GA-Hardness, Genetic Algorithm, Parameter Decoding, Predicting Complexity, Walsh Analysis, Walsh Functions, Walsh Sums

*This research was supported by NSF grants IRI-9312748 and IRI-9503366 to Darrell Whitley at Colorado State University.

1 Introduction

Davidor (1991) and Reeves and Wright (1993, 1994) have argued that understanding the distribution and level of epistasis is often an indicator of the difficulty of an optimization problem. Goldberg, Korb, and Deb(1989) point out the importance of building blocks and their size. Epistasis is a direct measure of limits of building block size. Classically epistasis is either computed exactly by Walsh coefficients as in Goldberg (1989) or estimated by sampling as in Dai and Sinha (1990) or in Davidor (1991). Exact computation is mostly of theoretical interest since the computation takes longer than to find the actual optimum values of the function by exhaustive search. Estimating epistasis by sampling an evaluation function gives us little insight into the origin of the epistasis and may miss computationally significant regions of the domain.

In many cases, an optimization problem is not just a black box. For many practical and classical test problems a function is first derived from a mathematical model. If the problem is to be optimized by a genetic algorithm then a representation is chosen mapping a chromosomal bitstring to the parameter vector for the model. In these cases, the restriction that the problem be expressible as a discrete sampling of a polynomial is very useful information and can tell us a lot about the epistasis of the problem without explicitly calculating Walsh coefficients.

This paper presents theorems establishing the bounds of epistasis for some problems and representations that can be stated as mathematical expressions. Computing the limits of epistasis directly from the mathematical expressions for a problem can lead to a substantial computational savings in estimating problem difficulty. Furthermore, working with these theorems in a mathematical context, one can gain insight into the mathematical origins of epistasis and how a problem's epistasis might be reduced. In the process, we present several new measures for epistasis and give empirical evidence and examples to demonstrate the application of the theorems.

Walsh sums are introduced. These measure the level of interaction for each specific number of bits of interaction. This allows us to conveniently define a measure of the maximal level interaction. In the section that follows we use this measure in a series of theorems that predict the maximal level of bit interaction based on the mathematical expressions for the model we want to optimize. These theorems include theorems on representation and problem encoding. A section showing graphic examples of consequences of these theorems emphasizes their predictive power. We also show how controlling problem representation might then be used to reduce problem epistasis.

We begin in section 2 by presenting an introduction to Walsh functions. Also presented is a convenient summary of our notation and definitions of various measures based on Walsh functions. In section 3 we rigorously show how the upper bound on epistasis can be predicted for mathematical models that are polynomials. Section 4 shows that functions can have **parity** when interpreted in Walsh space: a functions that displays parity has non-zero Walsh coefficients only on those Walsh terms whose index is odd (or even). Since this implies that at least half of the Walsh coefficients for a function are zero, this also implies that the number of Walsh coefficients that contribute to schema averages are also reduced on average by half. Since all but very low order schema averages depend on an approximately equal number of odd and even number of Walsh coefficients, the reduction in complexity that results for functions that display "parity" is relatively uniform and pervasive across the functions. Empirical tests suggest that just randomly setting half of the Walsh coefficients of a function to zero does not produce the sample reduction in complexity that is produced by creating functions with parity. Section 5 deals with representation problem, i.e. how the encoding of the problem domain of the function as a bitstring affects the epistasis in predicatable ways.

For readers interested in the practical implications of this research, we conclude in section 6 with empirical results demonstrating the prediction of epistasis from mathematical formulas. For special classes of functions, how parameters are encoded determines whether these functions display

parity, and thus, whether all Walsh coefficients that have an odd (or even) index are zero. For the cosine function, for example, "argument centering" can cause a function to display parity, and thus reduce the sources of nonlinearity. Empirical evidence suggests that picking a representation that displays parity in the resulting evaluation function also results in improved performance for a genetic algorithm. In particular, results are presented for Griewangk's function, which includes a second order polynomial combined with a cosine function as the main source of nonlinearity.

The results of this paper provide a basis for a better understanding of how epistasis develops in problems that originate in a mathematical model. They clear up some misconceptions by showing that Walsh coefficients that are zero occur more frequently than we might at first imagine. Finally, they provide us with some practical tools that may allow us predict what size building blocks to look for and give us some clues on how representation may inadvertently affect problem epistasis. Our theoretical and empirical results suggest that for large significant classes of problems that can display parity, such as those composed of odd or even ordered polynomials and transcendental functions such as sine and cosine, we may have more control over epistasis than has been previously thought.

2 Background

We begin by presenting the notations and definitions we will be using throughout the paper.

2.1 The Encoding of Functions for Genetic Algorithms

Consider a function which is a K parameter real valued mathematical model of the problem, $f_{model} : \mathcal{R}^K \rightarrow \mathcal{R}$, for which we wish to find the maximum or minimum value. When using a genetic algorithm as the optimizer, the problem must be transformed into a new function f_{ga} that is more amenable to computer analysis. We define f_{ga} as follows:

$$f_{ga} = f_{model}(f_{decode}) : \mathcal{B}^L \rightarrow \mathcal{R}$$

where $\mathcal{B}^L$ is an L dimensional bitspace and $f_{decode} : \mathcal{B}^L \rightarrow \mathcal{R}^K$ is the decoding function that maps the chromosomal bitspace of the genetic algorithm to the real valued parameters of the function model. Note the fact that this definition uses real parameters does not preclude us from using only integer parameters or other subsets.

2.2 Notation

The rest of the paper will use these notational conventions:

- Let $\mathcal{Z}_k$ represent the nonnegative integers mod$k$ and $\mathcal{Z}_0^+$ represent the positive integers plus zero.
- Ranges for summations are sometimes specified by a predicate using a colon. For example: $\sum_{k\,:\,i=k \wedge m}$ reads as the sum over all k such that $i = k \wedge m$. The choices of possible k can be inferred from context.
- The modular floor function of n and k is denoted $\lfloor n \rfloor_k$. This returns the largest integer less than or equal to n that is divisible by k. For example $\lfloor 5 \rfloor_3 = 3$ and $\lfloor 6 \rfloor_3 = 6$.
- Let $\mathcal{B} = \{0, 1\}$ and $\mathcal{Y} = \{1, -1\}$. Let the function $Y : \mathcal{B} \rightarrow \mathcal{Y}$ perform the mapping: $0 \rightarrow 1, \quad 1 \rightarrow -1$. Note that 1 does not map to 1.

- Let a **string** of length L be an ordered list from the set $\mathcal{B}$. The bits in a string are ordered $b_{L-1}, b_{L-2}, ...b_2, b_1, b_0$ and belong to $\mathcal{B}^L$. For example: `011001` is from $\mathcal{B}^6$ and has $b_0 = 1$. A string of all `1`'s will be denoted by $\vec{1}$.
- Let $[i]$ denote extracting the i^{th} bit. So if x =`1111011` then $x[2] = 0$. Extracting the bits i through j is denoted $x[i, j]$. For example x[2, 5] = `1110`.
- $i \subseteq j$ where $i, j \in \mathcal{B}^L$ reads as i **is contained in** j. That is wherever there is a `1` in i there is a `1` in j or, said another way, $i \wedge \overline{j} = 0$.
- Nonnegative integers will be used interchangeably with binary for representing strings in $\mathcal{B}^L$. e.g. $\vec{1}$ may also be referred to as $2^L - 1$. In converting a string to an integer, b_0 is the least significant bit.
- The **bit count** function $bc(i)$ returns the number of `1`'s in i. For example $bc($`001011`$) = 3$ and $bc(15) = 4$. This is also referred to as a **unitation** function.
- A **hyperplane** or **schema** is represented by one of the 3^L strings of `0`'s, `1`'s and `*`'s where the `0`'s and `1`'s are in the **fixed bit positions** and the `*`'s represent either a `0` or a `1` in the **variable bit positions**. An example hyperplane h for strings in $\mathcal{B}^7$ might be `**1101*`.
- A hyperplane with C fixed bit positions represents a hyperplane of **order** C and defines a set of $2^{(L-C)}$ strings where all possible replacements of the `*`'s have been defined. For hyperplane h the order of h is denoted by $o(h)$ and the number of strings in h is denoted $|h|$.

2.3 Walsh Polynomials

In order to measure the degree of interaction between bits in a function, $f : \mathcal{B}^L \rightarrow \mathcal{R}$, it is helpful to break the function down into a linear combination of the interactions. For an L bit function there are 2^L possible interactions. Several techniques are available in the literature for doing this including the experimental design techniques of Reeves and Wright (1993) and Walsh polynomials as explained in Bethke (1981) and in Goldberg (1989). We chose the classic Walsh polynomials because of their powerful and useful symmetries.

Any function $f : \mathcal{B}^L \rightarrow \mathcal{R}$ can be broken down into its **Walsh polynomial** as follows:

$$f(x) = \sum_{i=0}^{2^L-1} w_i \psi_i(x)$$

where $\psi_i(x)$ is the i^{th} **Walsh function** of x and w_i is the i^{th} **Walsh coefficient**. For the L bit function described above, 2^L Walsh functions are used and 2^L Walsh coefficients are used, although some may be zero.

The most important feature of a Walsh polynomial is that any function f can be uniquely decomposed into a linear weighted sum of Walsh functions where the weights are the real valued Walsh coefficients. Walsh functions are defined as functions, $\psi : \mathcal{B}^L \times \mathcal{B}^L \rightarrow \{1, -1\}$, that do a bit by bit parity check between two bit strings. If they share an even number of `1` bits in the same position, the function returns a 1, otherwise it returns a -1. This can be expressed in logic notation as:

$$\psi_j(x) = Y(\bigoplus_{i=0}^{L-1}(x[i] \wedge j[i])) \qquad x, j \in \mathcal{B}^L$$

where $\oplus$ denotes exclusive-or. Notice the use of the Y function (see the notation section) to map the result from $\mathcal{B}$ space to $\mathcal{Y}$ space. For example: $\psi_{01110}(11010)$ can be computed from above by bitwize anding of `01110` and `11010` yielding `01010`. The parity of this string is computed using exclusive-or giving 0 and the Y function maps this to $+1$. The function ψ is often written with a nonbinary subscript. In this example ψ_{01110} would be written ψ_{14} which would be referred to as the 14^{th} Walsh function.

The three most important intuitive properties to remember about Walsh functions are:

- **Symmetry** - The values of x and j can be interchanged without effecting the value of the function. i.e. $\psi_i(j) = \psi_j(i)$.
- **Parity** - The exclusive-or in the Walsh Function computes the parity of the results of the and's.
- **Masking** - The value of j (or x) can be thought of as a mask to select the bits in x (or j) for which we are interested in taking the parity measure.

The following two theorems are obvious from the Walsh function definition and we state them without proof.

Theorem 1 (Exclusive-or Product Theorem)

$$\psi_i(x)\psi_j(x) = \psi_{i \oplus j}(x)$$

Theorem 2 (Conjunctive Compression Theorem)

$$\psi_i(x) = \psi_{i \wedge x}(x)$$

The Walsh coefficients can be computed as follows:

$$w_j = \frac{1}{2^L} \sum_{x=0}^{2^L - 1} f(x)\psi_j(x)$$

Notice the formula for computing a Walsh coefficient is very similar to that for computing the function values from the Walsh coefficients. One need only exchange the values of the Walsh coefficients and function values and multiply by a constant in the case of computing the Walsh coefficients. It turns out that value of the summation in both expressions can be quickly computed using an analog of the Fast Fourier Transform as explained in Lechner (1971) or Goldberg (1989).

2.4 Coverage, Walsh Sums, and Function Order

In order to apply Walsh functions to the problem of epistasis we need to group the Walsh coefficients into a more manageable measure.

One of the simplist single value measures is **coverage** which is the ratio of the number of nonzero Walsh coefficients to the total number of Walsh coefficients. This indicates with a value between 0 and 1 the number of possible *contributing* bit interactions over the total number available.

A second and more informative measure returns a vector. Let a **Walsh sum** be denoted by W_k where:

$$W_k = \sum_{i : bc(i) = k} |w_i|$$

and $bc(i)$ is the bit count of i. Thus W_k is the sum of the **absolute value** of all of the Walsh coefficients for bit patterns i with exactly k bits set to 1. W_k is the k^{th} **order Walsh sum.**
For example:

$$\begin{aligned} W_0 &= |w_0| \\ W_1 &= |w_1| + |w_2| + |w_4| + |w_8| + \ldots + |w_{2^{(L-1)}}| \\ W_2 &= |w_3| + |w_5| + |w_6| + |w_9| + \ldots + |w_{3*2^{(L-2)}}| \\ &\vdots \\ W_L &= |w_{2^L-1}| \end{aligned}$$

Notice that for an L bit function there are $L+1$ Walsh sums and that W_k is the sum of $\binom{L}{k}$ Walsh coefficients. Since the absolute value of the Walsh coefficients is used, any nonzero Walsh coefficient will force the corresponding Walsh sum to be nonzero. Therefore the Walsh sum W_k is zero only when all order k Walsh coefficients are zero and is a measure of the magnitude of the nonzero Walsh coefficients of order k.

It is theorized by Holland (1975) and others that it is the hyperplanes that compete for dominance in a genetic algorithm population. The **hyperplane fitness** for a hyperplane h is the average of the value of the function over all strings in the hyperplane:

$$\frac{1}{|h|}\sum_{x \in h} f(x)$$

In order to prove theorems about averages of hyperplanes we need two functions α and β that are defined on a hyperplane as per Goldberg (1989):

$$\alpha(h)[i] = \begin{cases} \texttt{0} & \text{if } h[i] = \texttt{*} \\ \texttt{1} & \text{if } h[i] = \texttt{0} \text{ or } \texttt{1} \end{cases} \qquad \beta(h)[i] = \begin{cases} \texttt{0} & \text{if } h[i] = \texttt{*} \text{ or } \texttt{0} \\ \texttt{1} & \text{if } h[i] = \texttt{1} \end{cases}$$

For example for the hyperplane $h = \texttt{**1101*}$: $\alpha(h) = \texttt{0011110}$ and $\beta(h) = \texttt{0011010}$.

Theorem 3

The hyperplane fitness of hyperplane h can be computed using only those Walsh coefficients w_j where $bc(j) \leq o(h)$

Proof:

From the Hyperplane Averaging Theorem (see Appendix) we know that:

$$\frac{1}{|h|}\sum_{x \in h} f(x) = \sum_{j \subseteq \alpha(h)} w_j \psi_j(\beta(h))$$

For all $j \subseteq \alpha(h)$ we see that $bc(j) \leq bc(\alpha(h))$. But also know that $bc(\alpha(h)) = o(h)$. Therefore only w_j where $bc(j) \leq o(h)$ are used to compute the hyperplane average. □

The previous theorem was stated in that rather awkward way to segue into following theorem which makes an important point about Walsh sums.

Theorem 4

W_j is the sum of the w_i that represent j bit interactions.

Proof:

The w_i used to compute W_j are exactly the w_i that occur in the Hyperplane Averaging Theorem to compute the hyperplane averages for order j hyperplanes but not any lesser order hyperplanes. This follows immediately from above.

Therefore, the w_i used to compute W_j are the Walsh coefficients sufficient to express the increased order of interaction in order j hyperplanes over order $j-1$ hyperplanes. From this we conclude that W_j is a measure of the magnitude of order j bit interactions i.e. it measures the effects on the function value that can be attributed solely to j bit interactions.

□

Let the **order of a function**, denoted $\Omega(f)$, be defined as the largest i such that $W_i \neq 0$. In the special case where all W_i are 0, that is $f(x) = 0$, $\Omega(f) = 0$. Intuitively, the order of a function is the size of the largest set of interdependent bits. So $\Omega(f)$ is a measure of the level of epistasis of f and W_i measures the magnitude of the i-bit interdependence.

Another useful linear decomposition of f is by Spectral functions. f can be decomposed into the sum of at most $L+1$ spectral functions S_i as

$$f(x) = \sum_{i=0}^{2^L-1} w_i \psi_i(x) = \sum_{i=0}^{L} S_i(x)$$

where

$$S_i(x) = \sum_{j:bc(j)=i} w_j \psi_j(x)$$

The notation in the sum above means the sum over all indices with exactly i bits set to 1. This sum is called the **Spectral Decomposition** of f.

Note that the use of Spectral function is really nothing more than a "grouping" of the Walsh coefficients in $f(x)$ such that $S_i(x)$ represents the part of function f whose Walsh coefficients have bit count i and thus contribute to just the i bit interactions. Furthermore, the only possible nonzero Walsh sum for S_i is W_i and that W_i for S_i from the Spectral Decomposition of f is the same as the W_i for f itself. Spectral functions enable us to more easily express ideas about functions with limited degrees of interaction. In particular, since Spectral Decomposition isolates Walsh coefficients according to parity, it is also a useful way to decompose functions that display parity.

3 Function Order from Mathematical Models

If we know the mathematical expression for function f do we know anything about $\Omega(f)$? The answer is yes. In this section we develop a set of theorems that aid in predicting bit interaction based on the mathematical expression for the function.

Theorem 5

If $f(x) = c$ a constant then $\Omega(f) = 0$

Proof:

If $f(x) = c$ then by the Balanced Sum Theorem (See Appendix) only w_0 can be nonzero. If $c = 0$ then this is the special case of $\Omega(f) = 0$. Therefore $\Omega(f) = 0$.

□

Theorem 6

Let f be an L bit function with $f(x) = x$ then $\Omega(f) = 1$

This seems relatively intuitive and some of the implications of this theorem have been previously discussed in the literature (e.g. Goldberg 1989); however, the direct proof is somewhat complex and is given in the appendix for completeness.

The next theorem shows how scaling and translation effect the value of Ω.

Theorem 7 (Linear Composition Theorem)

If C is a nonzero constant and D is any constant.

$$\Omega(Cf + D) = \Omega(f)$$

Proof:

$$\begin{aligned} \text{Let } \ g(x) = Cf(x) + D \ &= C(\textstyle\sum_{i=0}^{2^L-1} w_i\psi_i(x)) + D \\ &= (\textstyle\sum_{i=0}^{2^L-1}(Cw_i)\psi_i(x)) + D \\ &= \textstyle\sum_{i=1}^{2^L-1}(Cw_i)\psi_i(x) + (Cw_0)\psi_0(x) + D \\ &= \textstyle\sum_{i=1}^{2^L-1}(Cw_i)\psi_i(x) + (Cw_0 + D)\psi_0(x) \end{aligned}$$

Denote the k^{th} Walsh coefficient of g above as w_k^g. Therefore the Walsh coefficients for g are:

$$w_j^g = \begin{cases} Cw_j & \text{if } \ j > 0 \\ Cw_j + D & \text{if } \ j = 0 \end{cases}$$

Since by the definition of $\Omega(f)$, if $bc(k) > \Omega(f)$ then $w_k = 0$ and therefore, we know that $w_k^g = 0$ for the same k.

□

The next two theorems consider the merging of functions with binary operators.

Theorem 8

For any two L bit functions f and g

$$\Omega(f + g) \leq \max(\Omega(f), \Omega(g))$$

Proof:

Let w^f be the Walsh coefficients for the function f and similarly for g. Then

$$f(x) = \textstyle\sum_{i=0}^{2^L-1} w_i^f\psi_i(x) \qquad \text{and} \qquad g(x) = \textstyle\sum_{i=0}^{2^L-1} w_i^g\psi_i(x)$$

$$\begin{aligned} \text{Therefore } \ f(x) + g(x) \ &= \textstyle\sum_{i=0}^{2^L-1} w_i^f\psi_i(x) + \sum_{i=0}^{2^L-1} w_i^g\psi_i(x) \\ &= \textstyle\sum_{i=0}^{2^L-1}(w_i^f + w_i^g)\psi_i(x) \\ &= \textstyle\sum_{i=0}^{2^L-1} w_i^{f+g}\psi_i(x) \end{aligned}$$

where w^{f+g} denotes Walsh coefficients of $f + g$. From the above it is clear that

$$w_i^{f+g} = w_i^f + w_i^g$$

Since both w_i^f and w_i^g are zero if $bc(i) > \max(\Omega(f), \Omega(g))$ then $(w_i^f + w_i^g) = 0$ as well. Therefore

$$\Omega(f + g) \leq \max(\Omega(f), \Omega(g))$$

□

Theorem 9 (Product Coefficient Theorem)

Let w^f be the Walsh coefficients of f and w^g the Walsh coefficients of g. Then the Walsh coefficients of the product fg, denoted by w^{fg}, are:

$$w_k^{fg} = \sum_{j=0}^{2^L-1} w_j^f w_{k\oplus j}^g$$

Proof:

Let f and g be defined by their Walsh expansions as before.

$$\begin{aligned} f(x)g(x) &= (\textstyle\sum_{j=0}^{2^L-1} w_j^f \psi_j(x))(\sum_{k=0}^{2^L-1} w_k^g \psi_k(x)) \\ &= \textstyle\sum_{j=0}^{2^L-1} \sum_{k=0}^{2^L-1} w_j^f w_k^g \psi_k(x)\psi_j(x) \\ &= \textstyle\sum_{j=0}^{2^L-1} \sum_{k=0}^{2^L-1} w_j^f w_k^g \psi_{k\oplus j}(x) \qquad \text{(By Theorem 1)} \end{aligned}$$

Let $k' = j \oplus k$ then $k = k' \oplus j$ and:

$$\begin{aligned} &= \textstyle\sum_{j=0}^{2^L-1} \sum_{k'=0}^{2^L-1} w_j^f w_{k'\oplus j}^g \psi_{k'\oplus j\oplus j}(x) \\ &= \textstyle\sum_{j=0}^{2^L-1} \sum_{k'=0}^{2^L-1} w_j^f w_{k'\oplus j}^g \psi_{k'}(x) \\ &= \textstyle\sum_{k'=0}^{2^L-1} (\sum_{j=0}^{2^L-1} w_j^f w_{k'\oplus j}^g) \psi_{k'}(x) \end{aligned}$$

Therefore

$$w_{k'}^{fg} = \sum_{j=0}^{2^L-1} w_j^f w_{k'\oplus j}^g$$

□

As a side note, the Product Coefficient Theorem can be interpreted in terms of a convolution as follows: the Walsh coefficient of the product of two functions is equal to the convolution of the Walsh coefficients of the separate functions. By **convolution** we mean the finite convolution based on the restriction that exclusive-or of the subscripts of the Walsh coefficients being summed is equal to the subscript of the Walsh coefficient of the product of the functions. A further discussion of this interpretation can be found in Lechner (1971) and in Manela and Campbell (1992).

The Product Coefficient Theorem has a couple of useful consequences as we see in the next two theorems.

Theorem 10 (Parity Theorem)

Let f and g be two functions with $W_{n_f}^f$ being the only nonzero Walsh sum for f and $W_{n_g}^g$ being the only nonzero Walsh sum for g, then $f * g$ can only have nonzero Walsh sums at $W_{n_f+n_g-2m}^{fg}$ for m a nonnegative integer.

Proof:

From the Product Coefficient Theorem $w_{k'}^{fg}$ can only be nonzero when both $w_j^f \neq 0$ and $w_{k'\oplus j}^g \neq 0$ for some value of j. This, by our premise, occurs when $bc(j) = n_f$ and $bc(k' \oplus j) = n_g$. It is easy to see that the allowable values for $bc(k')$ that fit these constraints cannot have more that $n_g + n_f$ bits. In fact mutual cancellation of bits in $k' \oplus j$ means that $bc(k') = n_f + n_g - 2m$.

Another way to look at this is to make a change of variable for any given j as follows. Let $k = j \oplus k'$. Then our requirement for nonzeroness

$$w_{k'}^{fg} \neq 0 \quad \text{if} \quad bc(j) = n_f \quad \text{and} \quad bc(k' \oplus j) = n_g$$

becomes

$$w^{fg}_{k\oplus j} \neq 0 \quad \text{if} \quad bc(j) = n_f \quad \text{and} \quad bc(k) = n_g$$

Notice that the index of the Walsh coefficient is the exclusive-or of a n_f bit string and a n_g bit string which must be a $n_f + n_g - 2m$ bit string where m is a nonnegative integer. Therefore the only nonzero Walsh sums are either all odd numbered or all even numbered and the theorem is proved.

□

The previous theorem gives us two important results. The first is that functions can have parity properties based on whether they have all odd or even order Walsh coefficients. We will deal with that in the next section. Secondly, we can use the theorem to put constraints on Ω for products of functions and hence polynomials, as we see in the next two theorems.

Theorem 11 (Product Theorem)
For the product of any two L bit functions f and g

$$\Omega(fg) \leq \Omega(f) + \Omega(g)$$

Proof:
From the Product Coefficient Theorem w^{fg}_k can only be nonzero when both $w^f_j \neq 0$ and $w^g_{k\oplus j} \neq 0$ for some value of j. Since $bc(j) \leq \Omega(f)$ and $bc(k \oplus j) \leq \Omega(g)$, $bc(k)$ can never have more than $\Omega(f) + \Omega(g)$ bits.

□

The previous theorems lead us logically to combine them in a general theorem covering any polynomial with nonnegative exponents.

Theorem 12 (Polynomial Complexity Theorem)
Let P_n be a polynomial with degree $n : n \geq 0$ over a bit string such that $P_n : \mathcal{B}^L \rightarrow \mathcal{R}$ then

$$\Omega(P_n) \leq n$$

Proof:
Consider x^n. Since $\Omega(x) = 1$ and by the Product Theorem $\Omega(f^k) \leq k\Omega(f)$, we find $\Omega(x^n) \leq n\Omega(x) = n$. Therefore by the Linear Composition Theorem a term of $P_n : ax^n$ has $\Omega(ax^n) \leq n$. Hence If $P_n = \sum_{k=0}^{n} a_k x^k$; $a_n \neq 0$, then

$$\Omega(P_n) \leq \max_{k=0..n} \Omega(a_k x^k) = n; \quad a_n \neq 0, k \geq 0$$

□

In this section we have shown that we can make a substantial prediction of the order of bit interaction for a simple polynomial as measured by the Walsh sum. In the next section we build some tools for working with f_{decode} and put it all together in a general prediction theorem.

4 Parity Laws and Argument Centering

The Parity Theorem implies important parity properties for functions. Let's begin with some definitions. A function which has only even ordered nonzero Walsh sums is called an **even ordered function**. A function which has only odd ordered nonzero Walsh sums is called **odd ordered function**. This property is called the **parity of the function**.

It can't be emphasized enough that these parity properties apply to functions over the finite range $\mathcal{B}^L$. For example, it will be shown later that when the domain is first mapped by specific scaling and offset, cosine, cos : $\mathcal{B}^L \rightarrow \mathcal{R}$, is an even order function. This does **not** mean that cos : $\mathcal{R} \rightarrow \mathcal{R}$ is an even order function.

The Parity Theorem yields some useful **parity laws**.

Theorem 13 (Parity Laws)

1. The product of an odd ordered function with an even ordered function is an odd ordered function.

2. The product of two odd or two even ordered functions is an even ordered function.

Proof:

Consider first the product of two even order functions. Since the functions are even order and any function in $\mathcal{B}^L$ can be decomposed into the sum of spectral functions, let

$$f(x) = \sum_{j=0}^{\lfloor L/2 \rfloor} S_{2j}^f(x) \quad \text{and} \quad g(x) = \sum_{k=0}^{\lfloor L/2 \rfloor} S_{2k}^g(x)$$

then

$$fg(x) = \sum_{j=0}^{\lfloor L/2 \rfloor} \sum_{k=0}^{\lfloor L/2 \rfloor} S_{2j}^f(x) S_{2k}^g(x)$$

From the Parity Theorem it is immediately evident that the product of any two even spectral functions S_j, S_k results in a function that is the sum of a series of all even ordered spectral functions up to $j + k$. Therefore each product of S_j and S_k in the previous equation can be replaced by a sum of even order spectral functions S_{2i}^{jk} as follows:

$$fg(x) = \sum_{j=0}^{\lfloor L/2 \rfloor} \sum_{k=0}^{\lfloor L/2 \rfloor} \sum_{i=0}^{j+k} S_{2i}^{jk}(x)$$

and hence fg is an even order function. Similar arguments hold for the other two cases. □

It turns out that by cleverly selecting the domain for a function, some functions can be forced to be even or odd ordered. This technique is called **argument centering**. This technique is based on this observation: if a domain of a function is itself a linear function over $\mathcal{B}^L$ then it has an Ω of 1. So it must be the sum of the two spectral functions S_0 and S_1. But S_0 is just the mean of all of the function values. By adjusting the mean of the linear function, which is the domain, to zero, we can force $S_0 = 0$. This leaves the function equal to the single spectral function S_1 which is an odd ordered function. If a polynomial is applied to an argument that is an odd ordered function then the Parity Laws apply throughout any products occurring in the polynomial. The result is that for a polynomial in x, odd powers of x are odd ordered functions and even powers of x are even ordered functions. So if the polynomial is all even powers or all odd powers, the parity of the polynomial can be predicted. We will see several examples of this when we "test drive" the theorems in the empirical section of the paper.

5 Function Order and Parameter Decoding

From a practical standpoint parameter decoding consists of three steps. First the subset of bits that encode each parameter are extracted from the bitstring. The substrings are then decoded into integers and finally the integers scaled to the appropriate range for each parameter of f_{model}. The first two steps are usually performed using logical operators on the bitstring representing the chromosome. The last step can be incorporated as part of the model function. Now we have:

$$f_{ga} = f_{model}(f_{decode}) = f'_{model}(f'_{decode})$$

where $f'_{model} : \mathcal{Z}_0^{+K} \rightarrow \mathcal{R}$ includes scaling of integer parameters and $f'_{decode} : \mathcal{B}^L \rightarrow \mathcal{Z}_0^{+K}$ uses logical operators to extract the encoded parameters as scalable integers from the chromosome.

In this section we construct some theorems to account for the bit interactions introduced by parameter encoding.

5.1 Basic Extraction

Extraction is often the basis of composing parameters for functions with domain $\mathcal{R}^n$ from strings in $\mathcal{B}^L$ that are processed by genetic algorithms. Let $x[n_1, n_2] : \mathcal{B}^L \times \mathcal{Z}_L \times \mathcal{Z}_L \rightarrow \mathcal{B}^L$ be the **extraction operator** which extracts bits in positions n_1 through n_2 ($n_2 \geq n_1$) from string x and places them in the least significant bit portion of the string with the remaining bit locations filled with zeros. This operation can be performed by masking and shifting. This leads us to our next general theorem which can be used to compute the Ω for a variety of logical operators.

Let the operator $\stackrel{\text{shift}}{\longrightarrow}: \mathcal{B}^L \times \mathcal{Z}_L \rightarrow \mathcal{B}^L$ be the binary right shift operator applied to the left operand, shifting it by the number of bits found in the right operand. The result is zero filled from the left. $\stackrel{\text{shift}}{\longleftarrow}$ is defined similarly. For example: if $s =$ `01010111` then $(s \stackrel{\text{shift}}{\longleftarrow} 2) =$ `01011100` and $(s \stackrel{\text{shift}}{\longrightarrow} 3) =$ `00001010`.

Theorem 14 (General Extraction Theorem)

$$\Omega(f((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s)) \leq \min(\Omega(f(x)), bc(m), L - s)$$

Where m is a mask and s is the amount to shift the result of $(x \wedge m)$.

Proof:

$$\begin{aligned}
f((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s) &= \textstyle\sum_{i=0}^{2^L-1} w_i \psi_i((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-1}((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s)[j] \wedge i[j]) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-s-1}((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s)[j] \wedge i[j]) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-s-1}((x \wedge m)[j+s] \wedge i[j])) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=s}^{L-1}((x \wedge m)[j] \wedge i[j-s])) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-1}((x \wedge m)[j] \wedge (i \stackrel{\text{shift}}{\longleftarrow} s)[j])) \\
&\quad \text{note the reversal of the direction of shifting} \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-1}(x[j] \wedge m[j] \wedge (i \stackrel{\text{shift}}{\longleftarrow} s)[j])) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i Y(\bigoplus_{j=0}^{L-1}(x[j] \wedge (m \wedge (i \stackrel{\text{shift}}{\longleftarrow} s))[j])) \\
&= \textstyle\sum_{i=0}^{2^L-1} w_i \psi_{(m \wedge (i \stackrel{\text{shift}}{\longleftarrow} s))}(x)
\end{aligned}$$

We can see the contraction of the coefficient space from "all of i" to $(m \wedge (i \stackrel{\text{shift}}{\longleftarrow} s))$. Since the coefficient for the Walsh function $\psi_i(x)$ is by definition the i^{th} Walsh coefficient, we can use the

last expression above to compute the Walsh coefficient w'_i for the function $f((x \wedge m) \xrightarrow{\text{shift}} s)$ based on the Walsh coefficients w_k for $f(x)$.

$$w'_i = \sum_{k \,:\, i=(m \wedge (k \xleftarrow{\text{shift}} s))} w_k$$

This mapping of Walsh coefficients constrains the subscript j for **nonzero** Walsh coefficients in the following ways:

- $bc(j) \leq \Omega(f)$ since $(m \wedge (i \xleftarrow{\text{shift}} s))$ can only reduce the bit count for any nonzero Walsh coefficient.
- $bc(j) \leq bc(m)$ since m masks the whole expression.
- $bc(j) \leq L - s$ since the zero fill left shift further reduces the possible number of ones in j.

Therefore

$$\Omega(f((x \wedge m) \xrightarrow{\text{shift}} s) \leq \min(\Omega(f(x)), bc(m), L - s)$$

and the theorem follows.

□

The above theorem is more general than necessary for use with bit extraction.

Corollary 14a (Extraction Corollary)

$$\Omega(f(x[n_1, n_2])) \leq \min(\Omega(f(x)), bc(m))$$

Proof:

Since the extraction operator can be defined as:

$$x[n_1, n_2] = (x \wedge m) \xrightarrow{\text{shift}} s$$

the above theorem applies to the extraction operator when $m = \vec{1}_{n_1,n_2}$, which is a mask with bits n_1 through n_2 set to one, and $s = n_1$ (remembering that bits are numbered from 0). In the bit extraction case we see:

$$\begin{array}{ll} L & \geq n_2 + 1 \\ L - n_1 & \geq n_2 - n_1 + 1 \\ L - s & \geq bc(m) \end{array}$$

giving us:

$$\Omega(f(x[n_1, n_2])) \leq \min(\Omega(f(x)), bc(m))$$

□

Furthermore by choosing the mask and shift values appropriately we see that:

$$\Omega(f(x \xrightarrow{\text{shift}} s)) \leq \min(\Omega(f(x)), L - s)$$

and

$$\Omega(f(x \wedge m)) \leq \min(\Omega(f(x)), bc(m))$$

These relations are based on a common theme of finding a function R such that there is a function R' for which:

$$\psi_i(R(x)) = \psi_{R'(x,i)}(x) \qquad \forall x \in \mathcal{Z}_L$$

As we have seen this is certainly the case with logical function $R(x) = (x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s$ in which case $R'(x) = m \wedge (i \stackrel{\text{shift}}{\longleftarrow} s)$. A theorem about combining these logical functions is:

Theorem 15 (Logical Function Xor)
Given two functions $R, Q : \mathcal{B}^L \rightarrow \mathcal{B}^L$ and:

$$\psi_i(R(x)) = \psi_{R'(x,i)}(x) \quad \text{and} \quad \psi_i(Q(x)) = \psi_{Q'(x,i)}(x) \qquad \forall x \in \mathcal{B}^L$$

then

$$\psi_i(R(x) \oplus Q(x)) = \psi_{R'(x) \oplus Q'(x,i)}(x) \qquad \forall x \in \mathcal{B}^L$$

Proof:

$$\begin{aligned} \psi_i(R(x) \oplus Q(x)) &= \psi_i(R(x))\psi_i(Q(x)) \\ &= \psi_{R'(x,i)}(x)\psi_{Q'(x,i)}(x) \\ &= \psi_{R'(x,i) \oplus Q'(x,i)}(x) \end{aligned}$$

□

5.2 Gray Codes

To apply this theorem let's look at a common decoding technique which is to assume the extracted bits are in binary reflected Gray code. Gray code is usually defined as:

$$gray(n) = n \oplus (n \stackrel{\text{shift}}{\longrightarrow} 1)$$

In this case a **degray** function $degray : \mathcal{B}^L \rightarrow \mathcal{B}^L$ must be applied. A mathematically simple approach to degraying is through the series:

$$degray(n) = n \oplus (n \stackrel{\text{shift}}{\longrightarrow} 1) \oplus (n \stackrel{\text{shift}}{\longrightarrow} 2) \oplus ...$$

In practice n would be an extraction of bits from x performed by $(x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s$ where x is the whole chromosome. Combining extraction with the degray function we get:

$$degray(x[i,j]) = ((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s) \oplus ((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} (s+1)) \oplus ((x \wedge m) \stackrel{\text{shift}}{\longrightarrow} (s+2)) \oplus ...$$

Now using the Logical Function Xor theorem we have the next theorem.

Theorem 16 (Degraying Theorem)
Let mask m and length s be used to define an extraction of a range of bits: $x[i,j]$ where $x[i,j] = (x \wedge m) \stackrel{\text{shift}}{\longrightarrow} s$. Then

$$\Omega(f(degray(x[i,j]))) \leq bc(m)$$

Proof:

$$
\begin{aligned}
f(degray(x[i,j])) &= f(((x \wedge m) \xrightarrow{\text{shift}} s) \oplus ((x \wedge m) \xrightarrow{\text{shift}} (s+1)) \oplus ((x \wedge m) \xrightarrow{\text{shift}} (s+2)) \oplus ...) \\
&= \textstyle\sum_{k=0}^{2^L-1} w_k \psi_k(((x \wedge m) \xrightarrow{\text{shift}} s) \oplus ((x \wedge m) \xrightarrow{\text{shift}} (s+1)) \oplus ((x \wedge m) \xrightarrow{\text{shift}} (s+2)) \oplus ...) \\
&= \textstyle\sum_{k=0}^{2^L-1} w_k \psi_{(m \wedge (k \xleftarrow{\text{shift}} s)) \oplus (m \wedge (k \xleftarrow{\text{shift}} (s+1))) \oplus (m \wedge (k \xleftarrow{\text{shift}} (s+2))) \oplus ...}(x) \\
&= \textstyle\sum_{k=0}^{2^L-1} w_k \psi_{m \ \wedge \ ((k \xleftarrow{\text{shift}} s) \oplus (k \xleftarrow{\text{shift}} (s+1)) \oplus (k \xleftarrow{\text{shift}} (s+2)) \oplus ...)}(x)
\end{aligned}
$$

Proceeding as with an earlier theorem it is clear that

$$\Omega(f(degray(x[i,j]))) \leq \min(bc(m), L - s)$$

The argument that $\Omega(f(degray(x[i,j]))) \leq \Omega(f)$ doesn't hold in this case since the series of exclusive-ors may, in fact, increase the bit count (bc). The other limiters still apply.

Since this theorem is stated specifically for extraction we know $bc(m) \leq L - s$ therefore

$$\Omega(f(degray(x[i,j]))) \leq bc(m)$$

□

At this point we can define a useful function we will call *ungray*:

$$ungray(n) = n \oplus (n \xleftarrow{\text{shift}} 1) \oplus (n \xleftarrow{\text{shift}} 2) \oplus ...$$

ungray is just like *degray* except the that the shifts are in the opposite direction. This can give binary strings an infinite set of leading 1. This does not effect the ability to test two of the strings for equality or to perform operations such as logical "and" upon them.

Corollary 16a

If w'_n is the n^{th} Walsh coefficient of the function $f(degray((x \wedge m) \xrightarrow{\text{shift}} s))$ and w_k is the k^{th} Walsh coefficient of the function $f(x)$ then

$$w'_n = \sum_{k \ : \ n=(m \wedge ungray(k \xleftarrow{\text{shift}} s))} w_k$$

Proof:

The proof follows directly from the proof of the theorem above and the definition of the *ungray* function.

□

It is important to notice from the corollary that the nice property of alternating zero and nonzero Walsh sums for odd and even order functions is destroyed by application of degraying. However all is not lost as we shall see in the next theorem. It turns out that degraying has an interesting effect on odd and even order functions.

Theorem 17 (Even Order Degraying Theorem)

Let mask m and length s be used to define an extraction of a range of bits: $x[i,j]$ where $x[i,j] = (x \wedge m) \xrightarrow{\text{shift}} s$. If $f(x[i,j]) : \mathcal{B}^L \rightarrow \mathcal{R}$ is an even order function then

$$\Omega(f(degray(x[i,j])) \leq bc(m) - 1$$

Proof:

Again we use the fact that $x[i,j] = m \wedge (x \overset{\text{shift}}{\longleftarrow} s)$ for appropriately chosen mask m and shift s. If we let $F = f(x[i,j])$ and $G = f(degray(x[i,j]))$ then the we have shown in the General Extraction Theorem: if w are the Walsh coefficients of f and w^F are the Walsh coefficients for F then:

$$w_n^F = \sum_{k \,:\, n=(m \wedge (k \overset{\text{shift}}{\longleftarrow} s))} w_k$$

We also know from the corollary to the Degraying Theorem:

$$w_n^G = \sum_{k \,:\, n=(m \wedge ungray(k \overset{\text{shift}}{\longleftarrow} s))} w_k$$

Because m and s are chosen to represent an extraction of bits, m is a mask consisting of a field of contiguous 1's possibly surrounded on either or both sides by a field of zeros. Notice that $k \overset{\text{shift}}{\longleftarrow} s$ has zeros in each position where m has 0's to the right of the field of 1's. Therefore the identity $(m \wedge ungray(k \overset{\text{shift}}{\longleftarrow} s)) = ungray(m \wedge (k \overset{\text{shift}}{\longleftarrow} s))$ holds and so:

$$w_n^G = \sum_{k \,:\, n=ungray(m \wedge (k \overset{\text{shift}}{\longleftarrow} s))} w_k$$

It can be seen that the w^G are simply a remapping by the *ungray* function of the coefficients w^F.

Let's look at this mapping more closely. If $d = ungray(g)$ then the p^{th} bit position in d, denoted $d[p]$ is the parity of all of the bits at that position and to the right.

$$\begin{aligned} n[p] &= ungray(g)[p] \\ &= (g \oplus (g \overset{\text{shift}}{\longleftarrow} 1) \oplus (g \overset{\text{shift}}{\longleftarrow} 2) \oplus ...)[p] \\ &= g[p] \oplus (g \overset{\text{shift}}{\longleftarrow} 1)[p] \oplus (g \overset{\text{shift}}{\longleftarrow} 2)[p] \oplus ... \\ &= g[p] \oplus g[p-1] \oplus g[p-2] \oplus ... \\ &= \bigoplus_{l=0}^{p} g[l] \end{aligned}$$

If F is an even order function then all $w_n^F = 0$ if $bc(n)$ is odd. If $bc(n)$ is odd then by our observation about the *ungray* function, $ungray(n)$ must have the high bit (left most bit of the masked region set). Hence, all $w_n^G = 0$ if the high bit of n is set and therefore

$$\Omega(f(degray(x[i,j])) \leq bc(m) - 1$$

and the theorem is proved.

□

An analogous proof can be given that shows that for odd ordered functions all of the nonzero Walsh coefficients of f are mapped to the upper half of the Walsh coefficient space, that is with the high order bit set. In this case the Ω of the function is obviously not reduced by 1 as it is for even order functions.

Corollary 17a

If $f : \mathcal{B}^L \rightarrow \mathcal{R}$ is an even order function then

$$\Omega(f(degray(x)) \leq L - 1$$

Proof:

The proof follows directly from the theorem above.

□

5.3 Generalized Decoding

The next theorem allows us to unite the extraction theorem and polynomial theorem and apply them to the composite function, f_{ga}, that we are trying to optimize. In the theorem below the coefficients of the polynomial are indexed by the exponents on the different variables in a standard fashion. An example polynomial of three variables might be:

$$P(x,y,z) = a_{111}xyz + a_{020}y^2 + a_{034}y^3z^4$$

Theorem 18 (Polynomial Composition Theorem)

Let $P_n(x_0, x_1, ...x_{n-1})$ be a polynomial in $x_0, x_1, ...x_{n-1}$ that takes $\mathcal{R}^n \rightarrow \mathcal{R}$ such that

$$P_n(x_0, x_1, ...x_{n-1}) = \sum_{all\ terms} a_{k_0k_1...k_{n-1}} x_0^{k_0} x_1^{k_1} ... x_{n-1}^{k_{n-1}}$$

with $k_i \geq 0$ and let $f_0, f_1, ...f_{n-1}$ be functions such that $f_i : \mathcal{B}^L \rightarrow \mathcal{R}$ then

$$\Omega(P_n(x_0, x_1, ...x_{n-1})) \leq \max_{all\ terms} k_0\Omega(x_0) + k_1\Omega(x_1) + ... + k_{n-1}\Omega(x_{n-1})$$

and

$$\Omega(P_n(f_0(x), f_1(x), ...f_{n-1}(x))) \leq \max_{all\ terms} k_0\Omega(f_0(x)) + k_1\Omega(f_1(x)) + ... + k_{n-1}\Omega(f_{n-1}(x))$$

Proof:

The Ω of any given term is

$$\begin{aligned}
\Omega(P_n(x_0, x_1, ...x_{n-1})) &= \Omega(\textstyle\sum_{all\ terms} a_{k_0k_1...k_{n-1}} x_0^{k_0} x_1^{k_1} ... x_{n-1}^{k_{n-1}}) \\
&\leq \max_{all\ terms} \Omega(a_{k_0k_1...k_{n-1}} x_0^{k_0} x_1^{k_1} ... x_{n-1}^{k_{n-1}}) \\
&= \max_{all\ terms} \Omega(x_0^{k_0} x_1^{k_1} ... x_{n-1}^{k_{n-1}}) \\
&\leq \max_{all\ terms} \Omega(x_0^{k_0}) + \Omega(x_1^{k_1}) + ... + \Omega(x_{n-1}^{k_{n-1}}) \\
&\leq \max_{all\ terms} k_0\Omega(x_0) + k_1\Omega(x_1) + ... + k_{n-1}\Omega(x_{n-1})
\end{aligned}$$

proving the first part of the theorem. The second part follows directly by substitution in the logic above.

□

Returning to problem of using genetic algorithms to optimize a function f_{ga}:

$$f_{ga} = f_{model}(f_{decode})$$

the decoding function can be represented as a vector of functions $\vec{f} = (f_0, f_1, ...f_{n-1})$ where $f_i : \mathcal{B}^L \rightarrow \mathcal{R}$. Therefore for a chromosome x in $\mathcal{B}^L$:

$$f_{model}(f_{decode}(x)) = P_n(f_0(x), f_1(x), ...f_{n-1}(x))$$

for model functions that are polynomials of n variables and the Polynomial Composition Theorem applies! This means given just the mathematical models and extraction functions we can derive a very good upper bound for the degree of bit interaction in the f_{ga} that we have designed to be solved. We may be able to apply this understanding to control the level of epistasis and thereby improve performance.

In the next section we will give some examples of the predictive power of the theorems. We will then use this knowledge to control the complexity of the problem in some simple ways. Finally, we will estimate the complexity of a common test function that was designed to have fixed levels of bit interactions.

6 Test Driving the Theorems

Throughout this article we have been using the term complexity to mean the epistatic component of problem difficulty. Although there are certainly easy problems with high epistasis there is a correlation between epistasis and difficulty of solving a problem with a simple genetic algorithm (see Heckendorn et al. (1997)). Of course, the difficulty of a problem can only truly be measured relative to the algorithm applied to solve the problem.

Barring recombination operators that understand about the structure of the decoding function, the genetic algorithm only sees f_{ga}. Therefore, the function that is really being solved is f_{ga} and it is the difficulty of this *composite* function that estimates the difficulty of solving the problem with a genetic algorithm. However, the difficulty of solving f_{ga} is often estimated by asking how hard is it to solve f_{model}.

The decoding function can be represented as a vector of functions $\vec{f} = (f_0, f_1, ...f_{n-1})$ where $f_i : \mathcal{B}^L \rightarrow \mathcal{R}$ and maps the bitstring to a real argument for the model function. Therefore, for a chromosome x in $\mathcal{B}^L$:

$$f_{model}(f_{decode}(x)) = P_n(f_0(x), f_1(x), ...f_{n-1}(x))$$

for model functions that are polynomials of n variables. In these cases the Polynomial Composition Theorem applies. This means given just the mathematical models and extraction functions we can derive a very good upper bound for the degree of bit interaction in the f_{ga} that we have designed to be solved. We may even be able to apply this understanding to control the level of complexity and thereby improve performance.

6.1 Some Empirical Results

In this section we will give empirical examples of the predictive power of the theorems. We will examine tables of Walsh sums W_0 through W_8 for sample functions. The first column of each table is the order of the Walsh sum. The remaining columns are the Walsh sums of that order for various functions listed at the head of each column. All functions presented in this section are evaluated using a string length of 8.

Table 1 shows that the n^{th} power of a linear function has an Ω of n. The second column is the Walsh sum for $f(x) = x$. The third column is $f(x) = x^5$. For both functions $x \in [0, 2^L - 1]$. Notice how $\Omega(f)$ is limited by the Polynomial Complexity Theorem to the maximum power of the polynomial.

Order	x	x^5
0	127.5	1.811e+11
1	127.5	4.299e+11
2	0	3.457e+11
3	0	1.088e+11
4	0	1.229e+10
5	0	3.731e+08
6	0	0
7	0	0
8	0	0

Table 1: Walsh Sums for $f(x) = x$ and x^5

In table 2 we compare the function x^5 over the noncentered domain $[0, 2^L - 1]$ to the centered domain $[-(2^{(L-1)} - .5), (2^{(L-1)} - .5)]$. We see in column 2 that without centering x^5 has nonzero Walsh sums for all orders less than 6, but with centering the even ordered Walsh sums go to zero as predicted.

Order	Without Centering	With Centering
0	1.811e+11	0
1	4.299e+11	1.923e+10
2	3.457e+11	0
3	1.088e+11	1.409e+10
4	1.229e+10	0
5	3.731e+08	3.731e+08
6	0	0
7	0	0
8	0	0

Table 2: Walsh Sums for x^5

The reason that argument centering has such a dramatic effect is that the function x or even $x[n_1, n_2]$, as we saw earlier, can be reduced to a single spectral function by argument averaging.

Keeping f_{model} restricted to a polynomial may seem limiting but actually it is quite powerful since all continuously differentiable functions can be expressed as a Taylor series expansion. For example, the Taylor series expansion about 0 for cos is:

$$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + ...$$

The Walsh sums for $\cos(x)$ for $x \in [0, \pi/2]$ are presented in table 3. By centering the argument x so that $x \in [-\pi/2, pi/2]$ the powers of x in cosine, which are all even, force the function to become even ordered as in table 3. One may not always be in position to change the domain of arguments to a function but there is often more flexibility than may seem at first.

Order	Without Centering	With Centering
0	0.6386	0.6366
1	0.512	0
2	0.1363	0.6459
3	0.01511	0
4	0.0007757	0.01548
5	1.837e-05	0
6	2.019e-07	1.634e-05
7	9.314e-10	0
8	1.444e-12	4.692e-10

Table 3: Walsh Sums for $\cos(x)$

6.2 Applying Centering to Griewangk's Function

Griewangk's function is often used as a test function for evolutionary algorithms (Whitley et al., 1995; Mülenbein and Schlierkamp-Voosen, 1993) A generalized version of the Griewangk function is:

$$G_{d,b}(\vec{x}) = 1 + \sum_{i=1}^{d} x_i^2/4000 + \prod_{i=1}^{d} \cos(x_i/\sqrt{i})$$

where $\vec{x}$ is a d dimensional vector of arguments with each x_i being b bits wide. For an encoding we will look at both the binary encoding and a Gray encoding. We assume that each x_i is created by extracting b bits from a bit vector chromosome then possibly decoding it using the *degray* function and scaling it by using some linear scaling with an offset.

Using the theory we have developed what can we tell about the Walsh coefficients from the formula and encoding alone?

Let's begin by computing the maximum number of bits of interaction, Ω, for a binary encoding, assuming the arguments are centered:

$$\begin{aligned}
\Omega(G_{d,\ b}) &= \Omega(1 + \textstyle\sum_{i=1}^{d} x_i^2/4000 + \prod_{i=1}^{d} \cos(x_i/\sqrt{i})\ ) \\
&= \max(\Omega(1),\ \Omega(\textstyle\sum_{i=1}^{d} x_i^2/4000),\ \Omega(\prod_{i=1}^{d} \cos(x_i/\sqrt{i}))\ ) \\
&= \max(0,\ \max_{i=1}^{d}(\Omega(x_i^2/4000),\ \textstyle\sum_{i=1}^{d} \Omega(\cos(x_i/\sqrt{i})))\ ) \\
&= \max(0,\ \max_{i=1}^{d}(\Omega(x_i^2),\ \textstyle\sum_{i=1}^{d} \Omega(\cos(x_i)))\ )
\end{aligned}$$

At this point we can use the Corollary to the General Extraction Theorem and the fact that each x_i is created from an extraction of b bits.

$$\Omega(G_{d,\ b}) = \max(0,\ \max_{i=1}^{d}(\min(2,b),\ \textstyle\sum_{i=1}^{d} \min(\Omega(\cos(x_i)),b))\ )$$

We note that cosine is an even order function and so only has nonzero Walsh coefficients with indexes having an even number of bits. (Remember cosine is an even ordered function only when its arguments are centered.) Therefore, if b is even then $\min(\Omega(\cos(x_i)),b) = b$. If b is odd then $\min(\Omega(\cos(x_i)),b) = b-1$, since the b^{th} Walsh sum is zero and the next nonzero Walsh sum is $b-1$. Let the act of rounding down to the closest number divisible by 2 be denoted by $\lfloor b \rfloor_2$, then $\min(\Omega(\cos(x_i)),b) = \lfloor b \rfloor_2$ and so:

$$\begin{aligned}
\Omega(G_{d,\ b}) &= \max(0,\ \max_{i=1}^{d} \min(2,b),\ \textstyle\sum_{i=1}^{d} \lfloor b \rfloor_2)\ ) \\
&= \max(\min(2,b),\ d * \lfloor b \rfloor_2)\ )
\end{aligned}$$

If the function is *not centered* then the analysis simply becomes:

$$\begin{aligned}
\Omega(G_{d,\ b}) &= \max(0,\ \max_{i=1}^{d} \Omega(x_i^2),\ \textstyle\sum_{i=1}^{d} \Omega(\cos(x_i))\ ) \\
&= \max(0,\ \max_{i=1}^{d}(\min(2,b),\ \textstyle\sum_{i=1}^{d} b\ )) \\
&= \max(\min(2,b),\ d * b\ )
\end{aligned}$$

which in most cases is simply $d * b$.

Empirical confirmation of these results is presented in the first four columns of table 4. Recall that Ω is the largest i such that $W_i \neq 0$, thus the odd and even nature of the function impacts Ω. The column labeled "Dimension" refers to the variable d above and represents the number of arguments to the function. "Width" refers to the variable b above and represents the number of bits encoding each argument. Under the heading "Binary Encoding" are the two columns of Ωs.

Dimension	Width	Ω for Binary Encoding		Ω for Gray Encoding	
		Centered	Offset	Centered	Offset
1	2	2	2	1	2
1	3	2	3	2	3
1	4	4	4	3	4
1	5	4	5	4	5
1	6	6	6	5	6
1	7	6	7	6	7
1	8	8	8	7	8
1	9	8	9	8	9
1	10	10	10	9	10
3	2	6	6	3	6
3	3	6	9	6	9
3	4	12	12	9	12
3	5	12	15	12	15
3	6	18	18	15	18
5	2	10	10	5	10
5	3	10	15	10	15
5	4	20	20	15	20

Table 4: Ω for the Generalized Griewangk in both Centered and Offset Forms Using Both Binary and Gray Encodings. The effects of parity due to centering is shown to reduce Ω. Combining Gray coding with centering can further remove high order nonlinear interactions.

The first column, labeled "Centered", contains the Ωs for functions whose arguments are centered in the range $[-(2^{b-1} - .5), 2^{b-1} - .5]$. The second column, labeled "Offset", contains the Ωs for functions whose arguments are offset from center by an arbitrary nontrivial constant, $e^{-.5}$, giving a range of $[-(2^{b-1} - .5) + e^{-.5}, 2^{b-1} - .5 + e^{-.5}]$. The effect parity on the function under a binary encoding shows up as a limit on Ω such that it is rounded down to the nearest even number.

If the same function is used with a Gray encoding then a degraying step must be added by applying the *degray* function **after** extraction and before scaling and offset adjustments. The analysis is similar to the binary centered case except the Corollary to the General Extraction Theorem is replaced by the Even Ordered Degraying Theorem in the second step below.

$$\begin{aligned}\Omega(G_{d,\,b}) &= \max(0,\ \max_{i=1}^{d}(\Omega(x_i^2)),\ \textstyle\sum_{i=1}^{d}\Omega(\cos(x_i)))\\ &= \max(0,\ \max_{i=1}^{d}(\min(2,b)),\ \textstyle\sum_{i=1}^{d}(b-1))\\ &= \max(\min(2,b),\ d*(b-1))\end{aligned}$$

Empirical confirmation of these results is presented in the last two columns of table 4. Here it can be seen that the use of cosine in the Griewangk function means that using centering and a Gray encoding is guaranteed to reduce the maximum number of bits of interaction in the function by the number bits equal to the number of arguments to the function, d. Thus, Gray coding causes a further general reduction in Ω and thus removes the higher level nonlinearities that are present in a representation that is not centered.

6.2.1 Centering and Problem Difficulty

The classic Griewangk function, as presented in Whitley et al. (1995), is a 10 dimensional function with 10 bit fields, one for each argument. The arguments are scaled $[-512, 511]$ which is the same

Offset	Optimum Value	Ω	Coverage	Optima				Number Evaluations	
				Total Number		Number Global			
				Min	Max	Min	Max	Mean	Stddev
0.0	0.0558068	16	0.0625	144	144	1	1	2575.3	2307.9
-0.1	0.0472153	20	1.0	144	162	1	1	5217.3	3675.5
-0.2	0.041276	20	1.0	262	264	1	1	5476.6	3503.6
-0.3	0.0312921	20	1.0	400	401	1	1	14754.8	19161.1
-0.4	0.0103844	20	1.0	400	480	1	1	13110.3	12428.8
-0.5	0	20	1.0	810	810	1	4	10874.2	7943.5

Table 5: Results of Six Generalized Griewangk Functions From Centered to Classical

as offseting a centered argument by $-.5$.

In our next experiments we wanted to see how varying amounts of offset from center would effect the difficulty of the classical Griewangk function. Computing the statistics we wanted would require complete enumeration of the function to be sure all regions of the function were explored. For the classic Griewangk this would be a space of 2^{100} function values, which is too large to be practical.

In order to get around this limitation we reduced the space to 20 bits. This allows us to exhaustively analyze all points in the function domain. The function we choose was a 4 dimensional version of the generalized Griewangk with 5 bits in each dimension. We looked at six versions of the function. The first version is a argument centered version with each argument in a range $[-15.5, 15.5]$. The remaining five functions have an increasingly distant offsets from center of $-.1$ to $-.5$. For example an offset of $-.5$ gives a range from $[-16, 15]$ simulating, in a smaller number of bits, the classical Griewangk which contains zero in the set of possible domain values. All functions in this experiment use a Gray encoding of the arguments.

Table 5 shows the results of testing these six functions. The first column is the offset from centered arguments. The second column shows the global optimum of the discretized function. Clearly the optimum must vary given the different sampling of the domain imposed by the different offsets. The third column is the Ω determined by computing the Walsh coefficients based on the 2^{20} values in the function. The **coverage** is the ratio of the number of nonzero Walsh coefficients to the total number of Walsh coefficients. The next two columns show the total number of (local and global) minima and maxima. Minima and maxima are defined with respect to a Hamming-1 neighborhood using a Gray code representation of the function. Notice that as the offset increasingly shifts the function from centered, the total number of optima increase. The next column shows the number of global optima. This suggests as this particular function deviates from centered arguments, the number of local optima increase. The number of global optima stays constant at 1. This should have an adverse affect on the ease with which the function can be solved by stochastic means.

The last two columns show the time required to locate the global optimum of the function using a genetic algorithm. The algorithm is a steady state (nongenerational) GA similar to GENITOR; parents are selected a pair at a time and a single offspring replaces the worst in a population. Rank based selection is used and no duplicate chromosomes are allowed in the population. The numbers in the table represent the results when solved with a population of 64 individuals, a mutation rate of 0.1 and a linear rank based selection bias probability distribution function with a .1 Y-intercept at the worst individual and a 1.9 at the best. (This selects the best ranked individuals 19 times more frequently than the worst ranked with a linear interpolation for the remaining individuals.)

The number of evaluations measured in the table is the number of times a new child is generated and its fitness evaluated. The last two columns contain the mean and standard deviation of the number of evaluations of the function necessary to find the optimum value over 32 runs for each function. The results indicate that the centered function is easier than the non-centered versions of the function.

6.2.2 Other Effects of Centering

The domain of multidimensional real valued functions is often mapped to series of bit fields. This means that each argument to the real valued function can take on one of some power of 2 number of different values. In the case of Griewangk's function the centered arguments fail to cover the value zero where the true optima is. The result is that for a Griewangk function in which the domain of each argument is symmetric about zero, centering produces 2^d number of equal valued optima, where d is the dimension of the Griewangk's function.

The reason that the centered function in Table 5 does not show the predicted 2^4 optima is that all of the optima are adjacent in Hamming space and hence counted by our software as belonging to one basin of attraction. The existence of the 16 minima in one basin was empirically verified by a separate program.

6.3 The Effects of Selective Walsh Filtering

Do the performance results we see for Griewangk hold in general for other functions with either limited Ω or parity such as evenness or oddness? Are these measures a useful indicator of problem difficulty?

To explore these questions we performed a series of experiments on two classes of functions: *even order functions* and *complete functions*. In a **complete function**, all the Walsh coefficients whose order is Ω or less (i.e. has Ω or fewer bits set) are assigned a random value between -1 and $+1$. This practically guarantees all Walsh coefficients of order less than or equal to Ω are nonzero. The remaining coefficients are all zero. This generates highly irregular landscapes, but with limited orders of bit interactions. In an **even order function**, all the *even order* Walsh coefficients whose order is Ω or less (i.e. has an even number of bits less than Ω set) are assigned a random value between -1 and $+1$. This effectively zeros out whole odd order partitions. The results of our experiments are shown in Figure 1.

For each class of function we generated 32 different 16 bit functions. Each problem was then solved in 32 attempts by random populations of 64 chromosomes using the same steady state GA with rank selection as before. This was repeated for maximum Ω's from 1 to 16. The various values of Ω are given on the X-axis of the first three graphs in Figure 1.

In the coverage graph in Figure 1, both classes of functions behave as we expect and is a direct consequence of the available number of Walsh coefficients of order less than or equal to Ω. You can see the stepwize behavior in the coverage for the even order function since the number of available nonzero Walsh coefficients is the same for $\Omega = 2k$ and $\Omega = 2k+1$. This graph shows how coverage is a direct consequence of Ω.

The graph in the upper right of Figure 1 shows the average number of minima in both classes of functions by Ω. In this case a minimum is defined as either a local minimum or global minimum with minimum points that are adjacent in Hamming space being counted as a single minimum. Notice the very strong correlation between coverage and number of minima. This is especially evident for the even order functions where the number of minima displays the same stepwise behavior as the coverage. A very interesting feature is that the number of minima for complete and even order

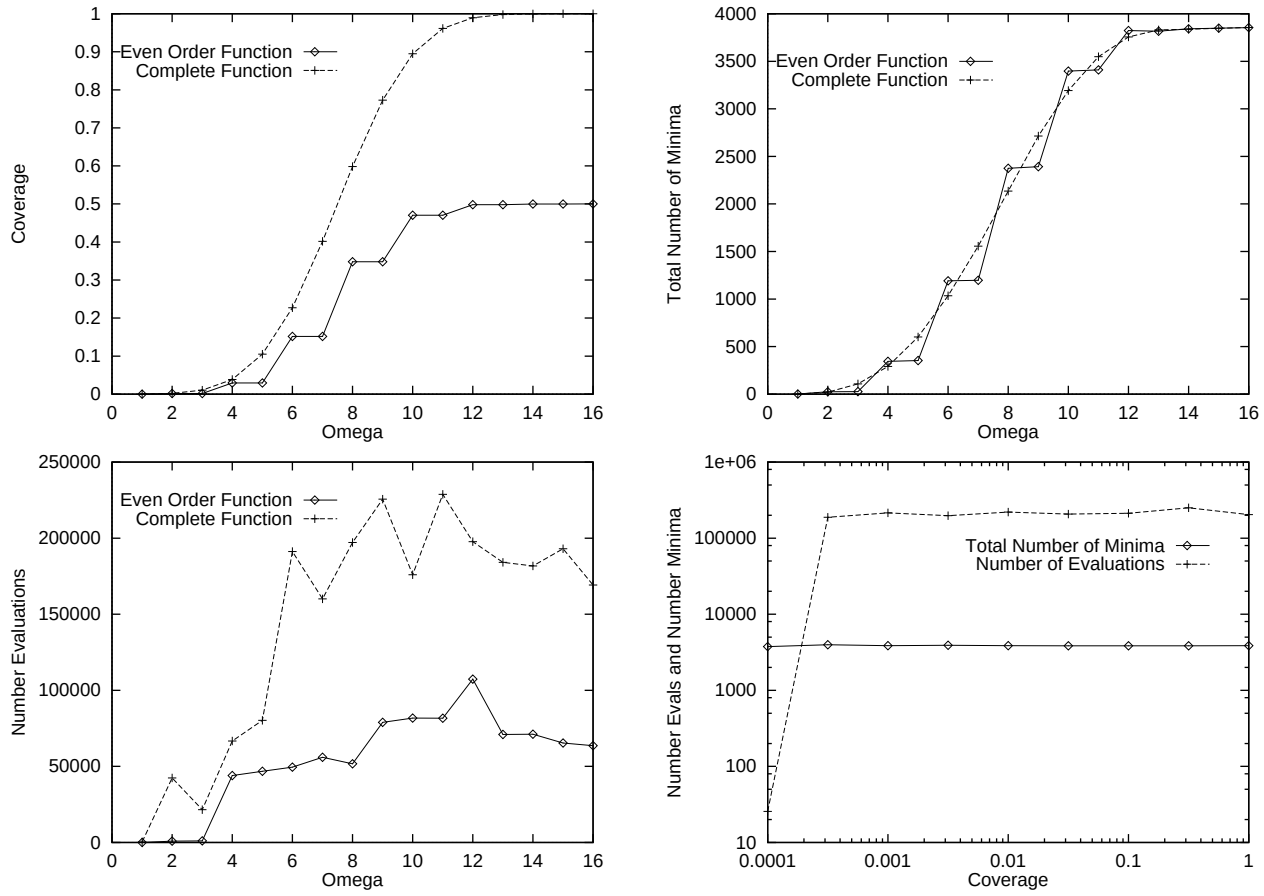

Figure 1: The first three graphs, from upper left to lower right, are for 16 bit even order and complete functions of Ω from 1 to 16 bits. The graphs show coverage, average number of minima, average number of evaluations to solution. The graph on the lower right is a comparison of number of minima and number of evaluations for problems of a given level random coverage.

functions are about the same throughout the range of Ω. That is the removal of the odd order partitions didn't seem to strongly affect the number of minima of the function. This graph shows that the number of minima is strongly related to the Ω.

The difficulty of the functions for the genetic algorithm is measured by number of function evaluations needed to find a solution and is displayed in the third graph in Figure 1. The number of evaluations seemed to remain fairly high until the Ω drops below half of the number of bits. This is probably partly due to the fact that the potential number of Walsh coefficients, all of approximately the same magnitude, forms a bell curve and so the addition of higher order Walsh coefficients from the tail of the curve has little effect on the final function value. The removal of the odd order Walsh coefficients seems to have curbed the precipitous increase in difficulty as the Ω approached the middle of the range.

These graphs suggest that the Ω may influence coverage, number of minima and problem difficulty. But is the measure of coverage alone a measure of difficulty? To examine this, we performed the same experiment ignoring the maximum number of bits of interaction, Ω, but setting the coverage to fixed values. The results are found in the graph in the lower right of Figure 1. Here coverage is on the X-axis. The coverage for each test was reduced by dividing the previous coverage by $\sqrt{10}$ forming a log scale along the X-axis. In each case, the Walsh coefficients that are set to zero are chosen randomly *without regard to parity.* Surprisingly, the number of minima remain fairly constant, as does the performance measured in number of function evaluations. This suggests that, ignoring cases of extreme depletion of nonzero Walsh coefficients, the difficulty and the number of minima are **not** related to the coverage as long as the nonzero Walsh coefficients are randomly dispersed.

We saw in the third graph of Figure 1 that throughout the range of Ω's the performance of the GA on the even order functions is far superior to the performance on complete functions. If the difficulty of the problem for functions from random Walsh coefficients is not related to the level of coverage between the two types of functions, then we must suspect that the difference is in the structure of what has been removed. This suggests that it is far more useful to remove half of the nonzero Walsh coefficients by zeroing the odd (or even) order coefficients than to randomly remove the same number of Walsh coefficients of various orders.

These are just two classes of problems and can't stand in for all classes of problems. The fact that these results support the performance observations made about Griewangk are highly suggestive of the generalizability of these results to larger classes of problems.

We believe that one of the key factors that caused this reduction in difficulty is related to the fact that whole partitions are zeroed in even order functions while few to none are zeroed my random zeroing.

A final observation is, regardless of the parity or nonparity of the function, the value of Ω seems to only be related to problem difficulty for values of Ω less than half the number of bits in the problem. Beyond this the problems in our experiments seemed to reach a plateau. In fact for our 16 bit cases even an Ω equal to 6 seemed to belong to the plateau set.

6.4 The Applicability of Centering

The analysis we have developed in this paper worked well for Griewangk's function, which is considered to be a nontrivial test function. However, Griewangk was particularly susceptible to our analysis and the use of centering techniques.

From a practical standpoint there are several concerns with applying argument centering to a problem and thereby altering the interval of the domain of the function. The first concern is that there are effects on the search space caused by rediscretizing the domain in order to center the

function. For instance, a new discretization may not contain the global optimum of the continuous function. Of course, in real world situations the global optimum is usually not known a priori so it is difficult to determine if this case applies. Another problem with rediscretizing is that a real world problem specification may require the examination of exactly the solution space given by the precentered discretization. In this case centering could be achieved by expanding the search space to a symmetric one but still containing all of the points of uncentered domain. The cost for this might be a lengthening of the search and therefore might cancel out the benefits of centering.

The second concern is that the main benefit of centering arguments is to functions that can be converted to odd or even functions. Straight forward centering can occur only to functions such as sin, cos or other functions with only even or odd exponents in their polynomial expansions.

7 Conclusions

In this paper we have shown that in the case where a model function can be expressed as a polynomial and each parameter encoding can be expressed as a polynomial applied to an extraction of bits from a bit string, the degree and magnitude of bit interactions can be predicted. This information could be used to design model and encoding functions with lower epistasis. By use of Taylor series the meaning of polynomial can be expanded to some common infinitely differentiable functions. Finally, we developed some techniques such as argument centering and odd/even series truncation that could be used to reduce bit interactions.

Work has proceeded beyond what is presented here on a variety of fronts including generation of functions of controlled complexity such as the NK Landscapes of Kaufmann (1993) and K-sat problems. Future work will look at predicting the magnitudes of Walsh coefficients, filtering of Walsh spectra, and incorporating linkage features by using Walsh spans. The relationship between various Walsh measures and GA performance is still unclear and needs to be rigorously examined. Other areas of application might be in using the predictions of maximum block size in conjunction with the Messy GAs of Goldberg, Korb, and Deb(1989).

Appendix

Some of these theorems in this appendix have been published before. Generally the proofs were given in various degrees of detail. They are provided here as a handy reference.

Theorem 19 (Balanced Sum Theorem)

$$\sum_{x=0}^{2^L-1} \psi_j(x) = \begin{cases} 2^L & \text{if } j = 0 \\ 0 & \text{if } j \neq 0 \end{cases}$$

First observe that this is a problem of counting +1's and -1's

Proof:
CASE 1: if $j = 0$ then

$$\begin{aligned} \textstyle\sum_{x=0}^{2^L-1} \psi_j(x) &= \textstyle\sum_{x=0}^{2^L-1} \psi_0(x) \\ &= \textstyle\sum_{x=0}^{2^L-1} 1 \\ &= 2^L - 1 \end{aligned}$$

CASE 2: if $j \neq 0$ then $\exists k$ such that $j[k] = 1$. We now divide the 2^L x values into 2 sets.

Let A be the set of all $\psi_j(x)$ where $x[k] = 1$.

Let B be the set of all $\psi_j(x)$ where $x[k] = 0$.

If $x \in A$ then $x \oplus 2^k \in B$. Let $y = x \oplus 2^k$. Note that for every $x \in A$ there is a unique $y \in B$ and vise versa therefore the $|A| = |B| = 2^{L-1}$.

$$\psi_j(y) \quad = Y(\bigoplus_{i=0}^{L-1}(y[i] \wedge j[i]))$$

For the k^{th} term in the above xor:

$$\begin{aligned}(y[k] \wedge j[k]) \quad &= ((x[k] \oplus 1) \wedge j[k]) \\ &\quad (x[k] \wedge j[k]) \oplus 1\end{aligned}$$

Bringing the 1 out of the xor:

$$\begin{aligned}\psi_j(x) \quad &= Y(1 \oplus (\bigoplus_{i=0}^{L-1}(x[i] \wedge j[i]))) \\ &= Y(1)Y(\bigoplus_{i=0}^{L-1}(x[i] \wedge j[i])) \\ &= -1\psi_j(y)\end{aligned}$$

Since there is a one to one onto correspondence between elements of A and elements of B and each pair is of opposite sign the sum is zero.

□

Theorem 20 (Orthogonality Theorem)

$$\sum_{x=0}^{2^L-1} \psi_i(x)\psi_j(x) = \begin{cases} 2^L & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Proof:

$$\sum_{x=0}^{2^L-1} \psi_i(x)\psi_j(x) \quad = \sum_{x=0}^{2^L-1} \psi_{i \oplus j}(x)$$

if $i = j$ then $i \oplus j = 0$
and
if $i \neq j$ then $i \oplus j \neq 0$.
Therefore by Balanced Sum theorem the theorem is proved.

□

Theorem 21 (Balanced Sum Theorem for Hyperplanes)

$$\sum_{x \in h} \psi_j(x) = \begin{cases} 0 & \text{if } j \wedge \overline{\alpha}(h) \neq 0 \\ \psi_j(\beta(h))|h| & \text{if } j \wedge \overline{\alpha}(h) = 0 \end{cases}$$

Note $j \wedge \overline{\alpha}(h) = 0$ means that no bit positions in j other than the fixed bit positions have a 1 in them.

Proof:

CASE 1: if $j \wedge \overline{\alpha}(h) = 0$

$$\begin{aligned}\sum_{x\in h}\psi_j(x) &= \sum_{x\in h}\psi_j((x\wedge\overline{\alpha}(h))\oplus(x\wedge\alpha(h)))\\ &= \sum_{x\in h}\psi_j(x\wedge\overline{\alpha}(h))\psi_j(x\wedge\alpha(h))\\ &= \sum_{x\in h}Y(\bigoplus_{i=0}^{L-1}(j\wedge(x\wedge\overline{\alpha}(h))))\psi_j(x\wedge\alpha(h))\\ &= \sum_{x\in h}Y(\bigoplus_{i=0}^{L-1}(j\wedge 0))\psi_j(x\wedge\alpha(h))\\ &= \sum_{x\in h}Y(0)\psi_j(x\wedge\alpha(h))\\ &= |h|\psi_j(x\wedge\alpha(h))\\ &= |h|\psi_j(\beta(h))\end{aligned}$$

CASE 2: if $j\wedge\overline{\alpha}(h)\neq 0$ (we will proceed as for the Balanced Sum Theorem) then $\exists k$ such that $j[k]=1$ and k is a bit position outside the fixed bit positions of h. We now divide the $2^{L-|h|}$ x values into 2 sets.

Let A be the set of all $\psi_j(x)$ where $x[k]=1$.

Let B be the set of all $\psi_j(x)$ where $x[k]=0$.

Note that if $x\in h$ then $x\oplus 2^k\in h$ since the k^{th} bit position is in the nonconstant portion of the bit strings in h.

Furthermore, if $x\in A$ then $x\oplus 2^k\in B$. Let $y=x\oplus 2^k$. Note that for every $x\in A$ there is a unique $y\in B$ and vise versa therefore the $|A|=|B|$.

The remainder of the proof proceeds as in the Balanced Sum Theorem except that A and B may be smaller sets contained entirely in a hyperplane.

$$\psi_j(y) = Y(\bigoplus_{i=0}^{L-1}(y[i]\wedge j[i]))$$

For the k^{th} term in the above xor:

$$\begin{aligned}(y[k]\wedge j[k]) &= ((x[k]\oplus 1)\wedge j[k])\\ &\quad (x[k]\wedge j[k])\oplus 1\end{aligned}$$

Bringing the 1 out of the xor:

$$\begin{aligned}\psi_j(x) &= Y(1\oplus(\bigoplus_{i=0}^{L-1}(x[i]\wedge j[i])))\\ &= Y(1)Y(\bigoplus_{i=0}^{L-1}(x[i]\wedge j[i]))\\ &= -1\psi_j(y)\end{aligned}$$

Since there is a one to one onto correspondence between elements of A and elements of B and each pair is of opposite sign the sum is zero.

□

Theorem 22 (Hyperplane Averaging Theorem)

$$\frac{1}{|h|}\sum_{x\in h}f(x)=\sum_{j\subseteq\alpha(h)}w_j\psi_j(\beta(h))$$

Proof:

$$\begin{aligned}\tfrac{1}{|h|}\textstyle\sum_{x\in h}f(x) &= \tfrac{1}{|h|}\textstyle\sum_{x\in h}\sum_{j=0}^{2^L-1}w_j\psi_j(x)\\ &= \tfrac{1}{|h|}\textstyle\sum_{j=0}^{2^L-1}w_j\sum_{x\in h}\psi_j(x)\end{aligned}$$

By the Balanced Sum Theorem for Hyperplanes the inner sum is only nonzero when $j\subseteq\alpha(h)$. Therefore

$$\begin{aligned}\tfrac{1}{|h|}\textstyle\sum_{x\in h} f(x) &= \tfrac{1}{|h|}\textstyle\sum_{j\subseteq\alpha(h)} w_j \sum_{x\in h}\psi_j(x)\\ &= \tfrac{1}{|h|}\textstyle\sum_{j\subseteq\alpha(h)} w_j|h|\psi_j(\beta(h))\\ &= \textstyle\sum_{j\subseteq\alpha(h)} w_j\psi_j(\beta(h))\end{aligned}$$

□

Theorem 23

Let f be an L bit function with $f(x) = x$ then $\Omega(f) = 1$

Proof:

This proof is done by establishing values for the Walsh coefficients of an L bit function f based on the $L-1$ bit version function of f. Let w_j^k be the j^{th} Walsh coefficient for function f over a k bit domain. Then we know that for function f in the L bit domain:

$$\begin{aligned}w_j^L &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^L-1} x\psi_j(x)\\ &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^{L-1}-1} x\psi_j(x) + \tfrac{1}{2^L}\sum_{x=2^{L-1}}^{2^L} x\psi_j(x)\\ &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^{L-1}-1} x\psi_j(x) + \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1}(x+2^{L-1})\psi_j(x+2^{L-1})\\ &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^{L-1}-1} x\psi_j(x) + \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} x\psi_j(x+2^{L-1}) + \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} 2^{L-1}\psi_j(x+2^{L-1})\\ &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^{L-1}-1} x(\psi_j(x)+\psi_j(x+2^{L-1})) + \tfrac{1}{2}\sum_{x=0}^{2^{L-1}-1}\psi_j(x+2^{L-1})\end{aligned}$$

This results in four cases based on two observations. The first observation is by the definition of ψ:

$$\psi_j(x+2^{L-1}) = \begin{cases}\psi_j(x) & \text{if } j < 2^L-1\\ -\psi_j(x) & \text{if } j \geq 2^L-1\end{cases}$$

The change in sign is caused by the inclusion of the L^{th} bit in the parity check in ψ.

The second observation is based on the first and the Balanced Sum Theorem:

$$\sum_{x=0}^{2^{L-1}-1}\psi_j(x+2^{L-1}) = \begin{cases}2^{L-1} & \text{if } j=0\\ -2^{L-1} & \text{if } j=2^{L-1}\\ 0 & otherwise\end{cases}$$

Now picking up where we left off: the observations induce four different cases:

$$\begin{aligned}w_j^L &= \begin{cases}\tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} x(\psi_j(x)+\psi_j(x)) + 2^{L-1}/2 & \text{if } j=0\\ \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} x(\psi_j(x)+\psi_j(x)) & \text{if } 0<j<2^{L-1}\\ \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} x(\psi_j(x)-\psi_j(x)) - 2^{L-1}/2 & \text{if } j=2^{L-1}\\ \tfrac{1}{2^L}\sum_{x=0}^{2^{L-1}-1} x(\psi_j(x)-\psi_j(x)) & \text{if } j>2^{L-1}\end{cases}\\ &= \begin{cases}\tfrac{1}{2^{L-1}}\sum_{x=0}^{2^{L-1}-1} x\psi_j(x) + 2^{L-2} & \text{if } j=0\\ \tfrac{1}{2^{L-1}}\sum_{x=0}^{2^{L-1}-1} x\psi_j(x) & \text{if } 0<j<2^{L-1}\\ -2^{L-2} & \text{if } j=2^{L-1}\\ 0 & \text{if } j>2^{L-1}\end{cases}\\ &= \begin{cases}w_j^{L-1}+2^{L-1} & \text{if } j=0\\ w_j^{L-1} & \text{if } 0<j<2^{L-1}\\ -2^{L-2} & \text{if } j=2^{L-1}\\ 0 & \text{if } j>2^{L-1}\end{cases}\end{aligned}$$

Hand calculation reveals that for $L=1$: $w_0^1 = \frac{1}{2}$ and $w_1^1 = -\frac{1}{2}$. Solving the recurrence above gives:

$$w_j^L = \begin{cases} (2^L - 1)/2 & \text{if } j = 0 \\ -2^{i-1} & \text{if } j = 2^i \\ 0 & \text{otherwise} \end{cases}$$

Notice that the only nonzero Walsh coefficients are for strings with $bc(j) \leq 1$ therefore $\Omega(f) = 1$. □

Bibliography

Bethke, A. D. (1981). *Genetic Algorithms as Function Optimizers.* PhD thesis, University of Michigan, Department of Computer and Communication Sciences Ann Arbor, Michigan.

Dai H. and Sinha N.K.. (1990). Robust coefficient estimation of Walsh functions *IEE Proceedings*, 137(6) 357–363.

Y. Davidor (1991) Epistasis variance: a viewpoint on GA-hardness In G.J.E. Rawlins editor, *Foundations of Genetic Algorithms (FOGA).* Morgan Kaufmann.

Goldberg D., Korb B. and Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 4:415–444.

Goldberg, D. (1989). Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems*, 3:129–152.

Heckendorn, R. B., Whitley D., Rana S. (1997). Nonlinearity, hyperplane ranking and the simple genetic algorithm. In Richard K. Belew, M. Vose, editor, *Foundations of Genetic Algorithms (FOGA) 4.* Morgan Kaufmann.

Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*, 1st ed. MIT Press.

Kauffman, S. A. (1993). *The Origins of Order.* Oxford Press.

Lechner, R. J. (1971). Harmonic Analysis of Switching Functions. In Mukhopadhyay, A., editor, *Recent Developments in Switching Theory.* Academic Press.

Manela, M. and Campbell, J. A. (1992). Harmonic Analysis, Epistasis and Genetic Algorithms. In *Parallel Problem Solving from Nature 2.* (pp 57-64). Elsevier Science Publishers.

Mülenbein, H and Schlierkamp-Voosen, D. (1993) Predictive Models for the Breeder Genetic Algorithm. *Journal of Evolutionary Computation, 1(1) pp25-49*, MIT Press.

Reeves, C. and Wright, C. (1993). Epistasis in Genetic Algorithms: An Experimental Design Perspective. In Eshelman, L., editor, *Proc. of the 6th International Conference on Genetic Algorithms*, (pp 217–224). Morgan Kaufmann.

Reeves, C. and Wright, C. (1994). An Experimental Design Perspective on Genetic Algorithms. In L. Darrell Whitley and Michael D. Vose editors, *Foundations of Genetic Algorithms (FOGA) 3.* Morgan Kaufmann.

Whitley, D, Mathias, K., Rana, S. and Dzubra, J. (1995) In Eshelman, L., editor, Building Better Test Functions. *Proc. of the 6th International Conference on Genetic Algorithms*, (pp 239–246). Morgan Kaufmann.

Whitley, D, (1989) The Genitor Algorithm and Selection Pressure: Why Rank-Base Allocation of Reproductive Trials is Best. *Proc. of the 3rd International Conference on Genetic Algorithms*, (pp 116–121).