A Tractable Walsh Analysis of SAT and its Implications for Genetic Algorithms

Soraya Rana **Robert B. Heckendorn** **Darrell Whitley**
email:{rana,heckendo,whitley}@cs.colostate.edu

Abstract

Walsh Transforms measure all sources of nonlinear interactions for functions that have a bit representation. There can be exponentially many nonlinear interactions and exactly computing all Walsh coefficients is usually intractable for non-trivial functions. In this paper we will show that SAT problems evaluated as MAXSAT functions have a highly restricted set of nonzero Walsh coefficients and those coefficients can be computed in linear time with respect to the number of clauses. This analysis suggests why standard simple genetic algorithms should perform poorly on MAXSAT problems.

Introduction

Boolean Satisfiability (SAT) was the first problem proven to be NP-Complete. SAT problems consist of literals, defined as variables or negated variables, that are combined together using **and** ($\wedge$) and **or** ($\vee$). Typically, SAT problems are presented in conjunctive normal form which groups literals together into disjunctive clauses. A SAT problem is considered solved when an instantiation of variables is found such that the formula is true or it can be proven that no such instantiation exists. However, a function that provides only a 1 or 0 answer presents little information to guide a blackbox optimization algorithm. As a result, SAT problems are often expressed as MAXSAT problems where the goal is to maximize the number of satisfied clauses in the formula.

We prove that when SAT problems are evaluated as MAXSAT problems, it is possible to exactly compute the linear and nonlinear bit interactions of the function using Walsh analysis. Most importantly, this analysis can be done in linear time with the number of clauses, where the number of clauses is usually a multiple of the number of variables. Since MAXSAT is also NP-Complete, the proof that Walsh analysis can be performed in linear time is somewhat surprising. This means that if $P \neq NP$ then knowing the exact linear and nonlinear interactions of a function cannot be sufficient for inferring the global optimum.

Finally, schema averages, which are theoretically used by genetic algorithms to guide search, can be computed directly from the Walsh coefficients of a function. The limitations that exist on the Walsh coefficients for MAXSAT problems place limitations on the possible schema averages. We examine what these limitations reveal about the expected behavior of genetic algorithms in the MAXSAT problem domain.

Walsh Analysis

The **Walsh transform** is a discrete analog to the Fourier transform. It can be used to measure the bitwise nonlinearity that exists in functions whose domain is a bit representation. This nonlinearity is an important, but by no means the sole, feature in determining problem difficulty for stochastic search algorithms(Goldberg 1989a; 1989b; Reeves & Wright 1993).

Every real valued function f over an L-bit string, denoted $f : \mathcal{B}^L \rightarrow \mathcal{R}$, can be expressed as a Walsh polynomial. In this case, the role of the sines and cosines in Fourier transforms is played by an orthogonal basis of 2^L **Walsh functions** denoted by ψ_j, where $0 \leq j \leq 2^L - 1$ with each Walsh function being $\psi_j : \mathcal{B}^L \rightarrow \{-1, 1\}$. The function f can then be expressed as a weighted sum over the Walsh functions:

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

where the real valued weights w_j are called **Walsh coefficients**.

Operations on indices such as j act on the standard binary representation of j. A simple way to define a Walsh function is using a bitwise-AND of the function index and its argument. Let $bc(j)$ be a count of the number of 1 bits in string j then:

$$\psi_j(x) = (-1)^{bc(j \wedge x)}$$

Thus, if $bc(j \wedge x)$ is odd, then $\psi_j(x) = -1$ and if $bc(j \wedge x)$ is even, then $\psi_j(x) = 1$.

[0] Copyright ©1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

The 2^L Walsh coefficients can be computed by a Walsh transform:

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

The Walsh transform, denoted W, acts as an invertible function analogous to a Fourier transform.

$$w_j = W(f(x)) \qquad f(x) = W^{-1}(w_j)$$

The calculation of Walsh coefficients can also be thought of in terms of matrix multiplication. Let $\vec{f}$ be a column vector of 2^L elements where the i^{th} element is the value of the evaluation function $f(i)$. Similarly define a column vector $\vec{w}$ for the Walsh coefficients. If M is a $2^L \times 2^L$ matrix where $M_{i,j} = \psi_j(i)$ then:

$$\vec{w} = \frac{1}{2^L} \vec{f}^{\,T} M$$

An important property of Walsh coefficients is that w_j measures the contribution to the evaluation function by the interaction of the bits indicated by the positions of the 1's in j. Thus, w_{0001} measures the linear contribution to the evaluation function associated with bit b_0, while w_{0101} measures the nonlinear interaction between bits b_0 and b_2. (Bits are numbered right to left starting at 0.) In the next section we will use this decomposition of functions into Walsh coefficients to analyze the nonlinear structure of SAT problems.

Walsh Analysis of MAXSAT Problems

Computing a single Walsh coefficient of a function usually requires the enumeration of the entire function space. In general, this makes Walsh analysis impractical for problems of nontrivial size. However, we will show that the Walsh coefficients for MAXSAT problems can be generated directly from the clause information without evaluating any points in the search space. In fact, all the Walsh coefficients for a MAXSAT problem can be computed in $O(2^K C)$ time, where K is the number of unique variables in the largest clause and C is the number of clauses. Assuming K is a bounded constant, the time required to compute the Walsh coefficients is the same complexity as the time required to simply write down the SAT expression.

We generate the Walsh coefficients for MAXSAT problems by first generating the Walsh coefficients for a single clause. We then combine the Walsh information for the various clauses.

Walsh Coefficients for a Single Clause

Consider the example function consisting of a single clause $f(x) = \neg x_2 \vee x_1 \vee x_0$ where value of x_0 is given by the least significant bit in the string x and x_1 the next least significant bit, etc. Each function can only be false when the assignment of bit values causes every literal to be false. Since this can happen only in one way, the vector representing the function values $f(x)$ is composed of all 1's except for a single 0.

The explicit calculation of the Walsh coefficients for the example function is obtained using the matrix form of the Walsh transform. The computation $\vec{f}^{\,T} M$ is shown in equation 1. Notice the single zero in the function vector for assignment $x_2 = 1, x_1 = 0, x_0 = 0$.

$$\vec{w} = \frac{1}{8} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{1}$$

The first column of M is $\psi_0(x)$ and by definition of ψ_0 must always equal to 1. This means that the value of w_0 corresponds to the average fitness of all strings. The remaining coefficients are a summation of ± 1 terms with a single term zeroed out. This leads to the following lemma for computing the Walsh coefficients for a single disjunctive clause.

Lemma 1

Let f, $f : \mathcal{B}^K \rightarrow \mathcal{B}^1$ be a general disjunctive function corresponding to a single clause over K unique variables. Let $neg(f)$ return a K-bit string with 1 bits indicating which variables in the clause are negated. Then the Walsh coefficients for f are:

$$w_j = \begin{cases} \frac{2^K-1}{2^K} & \text{if } j = 0 \\ -\frac{1}{2^K}\psi_j(neg(f)) & \text{if } j \neq 0 \end{cases}$$

Proof:

For any function that is a single disjunctive clause over K unique variables there is exactly one setting of the variables that will result in the clause being false, i.e. $f(i) = 0$. This means that there will always be $2^K - 1$ function values that evaluate to 1 and only one that can be 0. Note that

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

and since by definition $\forall x : \psi_0(x) = 1$, then the calculation of w_0 can be reduced to the constant:

$$w_0 = \frac{2^K - 1}{2^K}$$

Now consider how to compute w_j for $j > 0$. Let z be the string such that $f(z) = 0$. We know that all other strings will evaluate to 1. Therefore, we can simplify the expression for w_j:

$$w_j = \frac{1}{2^K} \sum_{i=0}^{2^K-1} f(i)\psi_j(i) = \frac{1}{2^K}\left(\left[\sum_{i=0}^{2^K-1} \psi_j(i)\right] - \psi_j(z)\right)$$

A general property of a Walsh transform for K-bit functions is that

$$\sum_{i=0}^{2^K-1} \psi_j(i) = 0 \qquad \forall j \neq 0.$$

Therefore

$$w_j = -\frac{1}{2^K}\psi_j(z) \qquad \forall j \neq 0 \qquad (2)$$

The value of z can be determined from the literals in the clause. Let $neg(f)$ return a K-bit number indicating which variables in clause are negated. For instance, $neg(\neg x_2 \vee x_1 \vee x_0) = 100$. Notice that $z = neg(f)$. Using this function, we can rewrite the equation for the Walsh coefficients.

$$w_j = -\frac{1}{2^K}\psi_j(neg(f)) \qquad \forall j \neq 0 \qquad (3)$$

□

Tautological clauses may be removed or handled as a special case. All tautological clauses result in Walsh coefficients of 0 except for w_0 which is 1.

Walsh Coefficients for MAXSAT

An L-bit MAXSAT problem can be represented as a sum of C disjunctive clauses, f_i:

$$f(x) = \sum_{i=1}^{C} f_i(x)$$

where $f, f_1, f_2, \ldots f_C : \mathcal{B}^L \rightarrow \mathcal{R}$.

Since the Walsh transform can be performed by a simple linear transformation, we see that Walsh transform of a MAXSAT problem can be treated as a sum of the Walsh transforms of the individual clauses.

$$W(f(x)) = \sum_{i=1}^{C} W(f_i(x))$$

This implies that to compute the Walsh coefficient for a MAXSAT problem we merely have to sum over the Walsh coefficients for each clause calculated using Lemma 1. While this is actually how the Walsh coefficients can be computed, the f_i functions cannot be used directly with Lemma 1. Each clause f_i is passed an L-bit string as an argument but can utilize K_i literals, where K_i is bounded by a constant. This means, that unlike in Lemma 1, not all variables will be used in each clause and our argument for the existence of only one value z such that $f_i(z) = 0$ is no longer true. The theory presented in the previous section must be modified for extraneous variables. To facilitate this we introduce two new notations.

Let $i \subseteq j$ where $i, j \in \mathcal{B}^L$ denote i **is contained in** j such that wherever there is a 1 in i there is a 1 in j. A function **pack** is designed to map functions in L-bit space to functions in M-bit space: $pack : \mathcal{B}^L \times \mathcal{B}^L \rightarrow \mathcal{B}^M$ where $M \leq L$. The function $pack(x, m)$ extracts the values of M bits from the string x according to the L-bit mask m, where $bc(m) = M$. The values are extracted at locations corresponding to a 1 in m and compressed to form an M-bit string. For example: $pack(10101, 01101) \Longrightarrow 011$.

The next theorem shows that the only nonzero Walsh coefficients, w_j, for f_i occur when $j \subseteq m_i$ where m_i is a mask selecting the variables that occur in the clause.

Theorem 1

Let f_i be a function which is a single disjunctive clause of K_i unique variables drawn from a string of L variables i.e. $f_i : \mathcal{B}^L \rightarrow \mathcal{B}^1$. Let the mask m_i be a mask indicating which variables occur in the clause. Then

$$w_j = \begin{cases} 0 & \text{if } j \not\subseteq m_i \\ \frac{2^{K_i}-1}{2^{K_i}} & \text{if } j \subseteq m_i \text{ and } j = 0 \\ -\frac{1}{2^{K_i}}\psi_j(neg(f_i)) & \text{if } j \subseteq m_i \text{ and } j \neq 0 \end{cases}$$

Proof:

If we want to compute w_j for a given j then there are two cases: either the $j \not\subseteq m_i$ or $j \subseteq m_i$.

Consider first the case where $j \not\subseteq m_i$. Select a 1 bit from j that is not in m_i. This corresponds to selecting a variable that is not used in f_i. Let the assignment for that variable be found at position p in the L-bit string x, where x is the argument to the function. Now partition the set of all 2^L possible L-bit strings into two sets S_0 and S_1. S_0 is the set of those arguments that have a 0 in position p and S_1 is the set that have a 1 in that position. There is a one-to-one, onto mapping from S_0 to S_1 by flipping the 0 at position p to a 1. Let the function $T : S_0 \rightarrow S_1$ perform this mapping. Since the value at position p has no effect on the evaluation of f_i, we know if $x \in S_0$ then $f_i(x) = f_i(T(x))$. Hence,

$$\begin{aligned} w_j &= \tfrac{1}{2^L}\textstyle\sum_{x=0}^{2^L-1} f_i(x)\psi_j(x) \\ &= \tfrac{1}{2^L}[\textstyle\sum_{x\in S_0} f_i(x)\psi_j(x) + \sum_{y\in S_1} f_i(y)\psi_j(y)] \\ &= \tfrac{1}{2^L}\textstyle\sum_{x\in S_0}[f_i(x)\psi_j(x) + f_i(T(x))\psi_j(T(x))] \\ &= \tfrac{1}{2^L}\textstyle\sum_{x\in S_0}[f_i(x)\psi_j(x) + f_i(x)\psi_j(T(x))] \\ &= \tfrac{1}{2^L}\textstyle\sum_{x\in S_0} f_i(x)[\psi_j(x) + \psi_j(T(x))] \end{aligned}$$

Since j by construction has a 1 in bit position p and we know that $T(x) \in S_1$ has 1 more bit than $x \in S_0$ then $bc(j \wedge x) + 1 = bc(j \wedge T(x))$. Therefore, $-\psi_j(x) = \psi_j(T(x))$ and the Walsh functions cancel so that

$$w_j = 0 \quad \text{if } j \not\subseteq m_i$$

Consider the second case where $j \subseteq m_i$. We now choose a bit position p outside of the bits set in m_i and thus also outside of the bits set by j. From above:

$$w_j = \frac{1}{2^L}\sum_{x\in S_0} f_i(x)(\psi_j(x) + \psi_j(T(x)))$$

However in this case j has a 0 at position p so $bc(j \wedge x) = bc(j \wedge T(x))$ and hence $\psi_j(x) = \psi_j(T(x))$.

$$w_j = \frac{1}{2^L}\sum_{x\in S_0} 2f_i(x)\psi_j(x)$$

Since the clause in f_i has K_i elements, there will be $L - K_i$ positions not overlapping with the 1 bits in m_i. This formula can simply be repeated $L - K_i$ times to account for all positions outside of mask m_i. We can create a reduced set S_0' containing the 2^{K_i} strings with 1 bits inside the mask m_i, i.e., $x \subseteq m_i, \forall x \in S_0'$.

The simplified equation for w_j is:

$$w_j = 2^{L-K_i} \frac{1}{2^L} \sum_{x \in S_0'} f_i(x)\psi_j(x) = \frac{1}{2^{K_i}} \sum_{x \in S_0'} f_i(x)\psi_j(x) \tag{4}$$

In order to use Lemma 1 we need to compress the $\mathcal{B}^L$ space of f_i to $\mathcal{B}^{K_i}$ space by using the *pack* function. Let $f_i' : \mathcal{B}^{K_i} \rightarrow \mathcal{R}$ be the same clause as f_i but with the domain limited to just the variables in the clause. Specifically, define the values of f_i' as $f(x) = f'(pack(x, m_i)) \; \forall \; x \in S_0'$. Since $j \subseteq m_i$ we see that the following identity must hold:

$$\psi_{pack(j,m_i)}(pack(x, m_i)) = \psi_j(x) \tag{5}$$

Now we can translate equation 4 into

$$w_j = \frac{1}{2^{K_i}} \sum_{x \in S_0'} f_i'(pack(x, m_i)) \; \psi_{pack(j,m_i)}(pack(x, m_i))$$

We have now constrained the set of $x \in S_0'$ to the point where there exists only one value of x that will make f_i' zero. Thus, we have the same two cases as in Lemma 1. Either $j = 0$ in which case:

$$w_j = \frac{2^{K_i} - 1}{2^{K_i}} \quad \text{if} \;\; j \subseteq m_i \;\; \text{and} \;\; j = 0$$

or $j \neq 0$ and the reasoning of Lemma 1 can be applied. Let z be the L-bit string such that $f_i(z) = 0$.

$$w_j = -\frac{1}{2^{K_i}} \psi_{pack(j,m_i)}(pack(z, m_i))$$

The arguments to the Walsh function can now be simplified using the identity in equation 5 to

$$w_j = -\frac{1}{2^{K_i}} \psi_j(z)$$

The observations about the negation mask in equation 3 also apply here, being very careful to note that the negation mask is in $\mathcal{B}^L$. This yields:

$$w_j = -\frac{1}{2^{K_i}} \psi_j(neg(f_i)) \quad \text{if} \;\; j \subseteq m_i \;\; \text{and} \;\; j \neq 0$$

□

An Example Computation

Table 1 shows an example of the application of this theory for a small MAXSAT function $f : \mathcal{B}^4 \rightarrow \mathcal{R}$ with $f(x) = f_1 + f_2 + f_3$ and

$$f_1 = (\neg x_2 \vee x_1 \vee x_0)$$
$$f_2 = (x_3 \vee \neg x_2 \vee x_1)$$
$$f_3 = (x_3 \vee \neg x_1 \vee \neg x_0).$$

x	w_i	$W(f_1)$	$W(f_2)$	$W(f_3)$	$W(f(x))$
0000	w_0	0.875	0.875	0.875	2.625
0001	w_1	−0.125	0	0.125	0
0010	w_2	−0.125	−0.125	0.125	−0.125
0011	w_3	−0.125	0	−0.125	−0.250
0100	w_4	0.125	0.125	0	0.250
0101	w_5	0.125	0	0	0.125
0110	w_6	0.125	0.125	0	0.250
0111	w_7	0.125	0	0	0.125
1000	w_8	0	−0.125	−0.125	−0.250
1001	w_9	0	0	0.125	0.125
1010	w_{10}	0	−0.125	0.125	0
1011	w_{11}	0	0	−0.125	−0.125
1100	w_{12}	0	0.125	0	0.125
1101	w_{13}	0	0	0	0
1110	w_{14}	0	0.125	0	0.125
1111	w_{15}	0	0	0	0

Table 1: Walsh Coefficients Broken Down by Clause.

Note that $w_0 = 2.625$, which is, as predicted, the average evaluation of f over all input strings. The linear, order-1 interactions are represented by w_1, w_2, w_4, and w_8 (each index has a single 1 bit when converted to a string). The order-2 nonlinear interactions are given by $w_3, w_5, w_6, w_9, w_{10}$ and w_{12} (each index has a pair of 1 bits when converted to a binary string). The third order nonlinear interactions are given by w_7, w_{11}, w_{13} and w_{14}. Since this is a 3-SAT problem, there can be no order 4 interactions, so $w_{15} = 0$. Other Walsh coefficients are zero due to cancellation of the constituent Walsh coefficients.

Notice that if f has L variables and C clauses the total number of possible Walsh coefficients is 2^L while an upper bound on the number of nonzero Walsh coefficients is $2^K C$ where $K = \max(K_i)$. Therefore, in general, the total number of Walsh coefficients that are zero grows exponentially as L increases.

The theorem we have presented allows us to determine which Walsh coefficients will be nonzero and compute a specific coefficient in at most $O(C)$ time. Furthermore, all Walsh coefficients can be computed in $O(2^K C)$ time because for each clause i, only the 2^{K_i} contributing coefficients need be computed. Finally, the maximum degree of bitwise nonlinearity (i.e. the maximum number of bits that interact to affect the evaluation function) is, as we would expect, limited to the number of variables in the longest clause.

Genetic Algorithms and MAXSAT

Genetic algorithms are population-based algorithms inspired by evolution. A population of strings is used to create a new population of strings through the use of **selection**, **recombination** and **mutation**. Each cycle of selection, recombination and mutation is considered one **generation**. Selection chooses individuals from a population that will be eligible for mating. Strings that are highly fit according to an evaluation function will have a better chance of being chosen for reproduction

than a less fit string. Two strings are paired together through the use of a recombination operator that divides the parent strings into several pieces and then concatenates the pieces to form two new strings that will appear in the next population. Mutation then randomly flips a small number of bits in the new offspring.

Holland (1975) explains the computational behavior of genetic algorithms by arguing that genetic algorithms compare subpartitions of the search space and then allocate more trials to the subpartitions of the search space that have the highest fitness. These subpartitions are represented by **schemata**. Schemata are L-length strings defined using a trinary alphabet $\{0,1,*\}$ yielding 3^L possible schemata. The $*$ indicates a wildcard where either a 0 or 1 are allowed. The **order** of a schema is the number of positions that contain a 0 or 1. For instance, the schema $*11*$ would contain 4 strings $\{0110, 0111, 1110, 1111\}$ and is an order-2 schema. For order-1 schemata such as $**0**$ and $**1**$, if the current population is distributed such that strings with a 0 in the third position are on average better than those strings that contain a 1 in the third position, then more samples should be allocated to the schema $**0**$ in the next population.

In some sense genetic algorithms stochastically hill-climb in the space of schemata rather than in the space of binary strings. As with other hill-climbing search strategies, a genetic algorithm can perform poorly if there is an absence of information or if that information is misleading (deceptive). However, it is usually difficult to statically analyze large functions to determine whether or not useful schema relationships exist. Computing the exact average fitnesses of low order schemata requires exponential time for most large problems. However, we can use Walsh coefficients to efficiently compute low order schema averages.

Functions α and β are defined on a schema, h, as per Goldberg (1989a):

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

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

For example, consider $h = **01**$. Because schema h is a subset of strings, its fitness is the average fitness of all strings that belong in that subset. Rather than enumerating the subset of strings, the fitness of h can be computed using Walsh coefficients:

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

For all $j \subseteq \alpha(h)$ we see that $bc(j) \leq bc(\alpha(h))$. This implies that the highest order Walsh coefficient used in the formula has the same order as the schema.

The average fitness of schema $h = **01**$, depends only on Walsh coefficients w_0, w_4, w_8 and w_{12}.

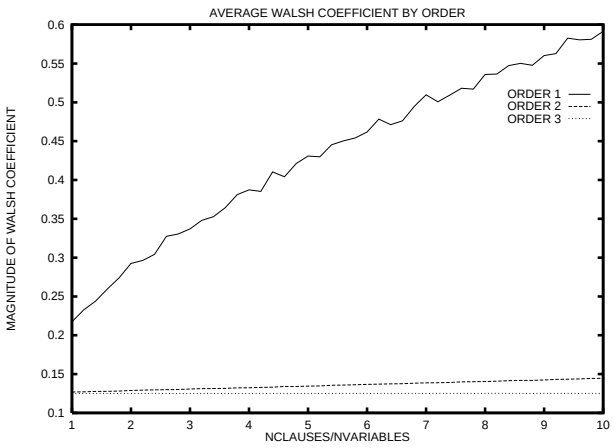

Figure 1: Average magnitudes of nonzero Walsh coefficients (excluding w_0).

Since $\alpha(**01**) = 001100$, the subset generated using α is $\{001100, 001000, 000100, 000000\}$ corresponding to the indices $\{12, 8, 4, 0\}$. The string generated by $\beta(**01**) = 000100$, so $\psi_4(000100) = -1$, $\psi_8(000100) = 1$ and $\psi_{12}(000100) = -1$. The sign of w_0 is always 1. So, $f(**01**) = w_0 - w_4 + w_8 - w_{12}$.

Schema Fitnesses for MAXSAT Problems

We have proven that the nonzero Walsh coefficients for MAXSAT problems are limited in number and maximum order. Since schema averages of varying order are generated as a summation of positive and negative Walsh coefficients, the limitations on the set of Walsh coefficients results in schema averages that will be very similar.

Consider the plot shown in Figure 1. The graph was generated using a set of randomly generated 100 variable 3-SAT problem instances. The ratio of clauses to variables ranged from 1 to 10 at increments of 0.2. Each point in the graph is the average of 30 problem instances. Each line tracks the average magnitude of all nonzero Walsh coefficients for order-1, order-2, and order-3 interactions.

Recall that any single clause in a 3-SAT problem contributes $\pm\frac{1}{2^3}$ to its nonzero Walsh coefficients. When combinations of variables co-occur in multiple clauses, the corresponding Walsh coefficient will be incremented or decremented by $\frac{1}{2^3}$. The final coefficient will either be zero or a multiple of 0.125.

Table 2 lists the number of nonzero Walsh coefficients that can occur for order-1, order-2, and order-3 interactions. In general, the maximum possible number of coefficients is $\binom{L}{k}$ for order-k $= 1, 2, 3$. However, the number of nonzero Walsh coefficients for a particular 3-SAT problem is limited. In the context of 3-SAT, the only way for 3 variables to interact is if all 3 variables occur in a clause. This means the number of nonzero order-3 Walsh coefficients is limited to the number of clauses in the problem. For any reasonable sized 3-SAT

Order	Max Possible	Max for MAX3SAT		
		100	500	1000
1	100	100	100	100
2	4950	300	1500	3000
3	161700	100	500	1000

Table 2: Upper bounds on the number of possible nonzero Walsh coefficients for an arbitrary function (Max Possible) and for a MAX3SAT problem with 100 variables and 100, 500, 1000 clauses.

problem, it is likely that the combinations of 3-variables will often be unique. Therefore the majority of the order-3 Walsh coefficients will be zero and any nonzero order-3 Walsh coefficients are likely to be ± 0.125.

The order-2 interactions result in higher Walsh coefficient magnitudes because as the number of clauses is increased, it becomes more likely for a pair of variables will co-occur in multiple clauses. Similarly, the magnitudes of the order-1 Walsh coefficients are larger because each individual variable occurs more often as the number of clauses is increased. As a general rule, the magnitudes of the Walsh coefficients will increase as it becomes more likely that a variable or variable combinations will reoccur in the problem.

An obvious observation from Figure 1 is that the size of the nonzero Walsh coefficients are minuscule (less than 0.6) compared to w_0 which is $\frac{7}{8}C$. For the 3-SAT problems in Figure 1, the C ranged from 100 to 1000 meaning that $87.5 \leq w_0 \leq 875$. Since all schema averages include the Walsh coefficient w_0, it will take many linear and nonlinear terms to induce any noticeable difference between schema averages. Consequently, the low order schema averages will all be near w_0. Of course, there will be some variation between low order schemata; the variation that occurs will largely be due to the linear Walsh coefficients. These differences will become more pronounced in higher order schemata. However, a genetic algorithm will have difficulty when there is little information in the low order schemata or if that information is misleading.

Conclusion

MAXSAT is commonly used as an evaluation function when general optimization techniques are applied to SAT problems. We have shown that all nonzero Walsh coefficients can be directly computed for MAXSAT in linear time as a function of the number of clauses. This result allows us to conclude that: If $P \neq NP$ then knowing the exact linear and nonlinear interactions of a function cannot be sufficient for inferring the global optimum in polynomial time. Furthermore, the Walsh coefficients that are generated for such functions are limited. The degree of bitwise nonlinearity that can exist in such problems is limited by the longest clause in the formula. The number of possible nonzero nonlinear Walsh coefficients is also limited by the number of clauses in the formula.

The Walsh analysis illustrates that the class of MAXSAT problems will result in little variation in schema averages. Furthermore, any differences that occur between lower level schemata are dominated by the linear order-1 schema information. If $P \neq NP$ then this order-1 schema information (as well as order-2 and order-3 schema information) cannot reliably guide search algorithms to a global optimum for MAXSAT. The misleading schema information makes MAXSAT unsuitable for optimization by traditional genetic algorithms. However, we cannot generalize this statement to other evaluation functions or other NP-Complete problems because modifying the evaluation function will result in different Walsh coefficients.

The Walsh analysis of MAXSAT problems is also related to polynomial time approximate algorithms for NP-Complete problems. This class of algorithms, sometimes called r-approximate algorithms (Trevisan 1997), is guaranteed to locate a solution that is at least a factor r, where $0 \leq r \leq 1$, from the optimum. It has been proven (Håstad 1997) that this factor cannot be higher than $\frac{7}{8}$ for satisfiable problem instances. The Walsh analysis proves that for *any* MAX3SAT problem, the average evaluation of all possible solutions to the problem is always $\frac{7}{8}$ times the number of clauses; thus, the polynomial time approximate algorithms cannot guarantee a solution that is better than average (unless $P = NP$). The relationship between the Walsh analysis of MAXSAT and polynomial time approximate algorithms will be explored in future work.

Acknowledgments

This work was supported by NSF grant IRI-9503366 and AFOSR grant F49620-97-1-0271. Soraya Rana was also supported by a National Physical Science Consortium fellowship awarded by NASA-SSC.

References

Goldberg, D. 1989a. Genetic algorithms and walsh functions: Part i, a gentle introduction. *Complex Systems* 3:129–152.

Goldberg, D. 1989b. Genetic algorithms and walsh functions: Part ii, deception and its analysis. *Complex Systems* 3:153–171.

Håstad, J. 1997. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computation*, 1–10.

Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Reeves, C., and Wright, C. 1993. Epistasis in genetic algorithms: An experimental design perspective. In Eshelman, L., ed., *Proceedings of the Seventh International Conference on Genetic Algorithms*, 217–224. Morgan Kaufmann.

Trevisan, L. 1997. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms*.