Estimating the Significant Non-Linearities in the Genome Problem-Coding

Dirk Thierens

Department of Computer Science, Utrecht University
Padualaan 14, 3584 CH Utrecht, The Netherlands
Dirk.Thierens@cs.uu.nl

Abstract

Substantial insight in the genome problem-coding can be gained if we know the most important Walsh coefficients - that is, the coefficients with large value. The practical use of the Walsh transform however is severely limited by the computational cost, even when using the Fast Walsh Transform. Part of this is caused by the fact that the transform computes all Walsh coefficients, not just the most significant. Here we discuss the use in a GA context of the recently developed KM algorithm, which estimates the most significant Walsh coefficients in a computational efficient way.

1 INTRODUCTION

It is generally acknowledged that a key characteristic of simple genetic algorithms is their repeatedly sampling and recombining of substrings that show above average fitness - as compared to the current population - and that have a fairly high survival probability during the exploration phase. These substrings - or building blocks - are therefore not merely defined by the fitness function, but also by the problem-coding and the genetic explorative operators, such as crossover and mutation. Effective GA processing then can be expected when these building blocks can be assembled to ever increasingly better solutions until optimal or near-optimal solutions are found. As a result the choice of problem-coding method and genetic explorative operators - especially crossover - is very important, and theoretical methods and algorithms to investigate this issue are indispensable in the genetic algorithmist's toolkit.

One useful method concerning the problem-coding is the use of Fourier techniques - in this case the Walsh transform. The use of the Walsh decomposition in genetic algorithm research has been initiated by Bethke who applied Walsh functions to efficiently calculate the schema average fitness values (Bethke, 1981). Goldberg extended this technique using Walsh polynomials such that the analysis became an intuitively clear method to visualise the non-linear interactions in a particular function-coding combination (Goldberg, 1989a and 1989b). The Walsh transform is particularly insightful because the magnitude of the Walsh coefficients is a direct measure of the fitness contribution given by the non-linear interaction of the genes corresponding with the Walsh index set. Although of great theoretical value the practical use of the Walsh transform is limited due to its computational complexity, this despite the existence of the Fast Walsh Transform.

One observation however indicates a possible way to improve the practical applicability of the Walsh transform: since we are basically interested in those groups of genes that have a strong non-linear interaction, we only need to look at those Walsh coefficients that have a high value as compared to the rest. So our objective can be redefined as to design a computationally efficient algorithm that gives us only the large magnitude Walsh coefficients. Actually, it turns out that such an algorithm already exists in the Computational Learning community where it is usually called the KM-algorithm (Kushilevitz and Mansour, 1993).

Here in this paper, we would like to discuss the potential use of this algorithm for the genetic algorithm practitioner. In the next section we first summarise the Walsh transform, and subsequently review the KM-algorithm. Section 3 shows how KM might be used to get an insight in the genome problem-coding structure. As an example we look at some results on GA-hard problems. Finally in section 4 we discuss some further possible usage.

2 BACKGROUND

2.1 WALSH TRANSFORM

The Walsh transform - or the multidimensional discrete Fourier transform - can be used to expand each function $f : \{0,1\}^n \rightarrow \Re$ as a linear combination of the basis functions $\phi_z(x)$:

$$f(x) = \sum_{z \in \{0,1\}^n} w_z \phi_z(x)$$

where the real valued Walsh coefficients are

$$w_z = E[f(x)\phi_z(x)],$$

and the basis functions $\phi_z : \{0,1\}^n \rightarrow \{-1,+1\}$

$$\phi_z(x) = (-1)^{\Sigma_i z_i x_i}.$$

The Walsh basis functions $\phi_z(x)$ are parity functions over those subset of bits in x indicated by z. If the number of bits is even then $\phi_z(x) = +1$ else $\phi_z(x) = -1$.

The Walsh coefficients are particularly interesting for genetic algorithm research because of the elegant way they correspond to the notion of building blocks, and because of their ease of use to calculate schema average fitnesses. The average schema fitness can be computed as the partial signed sum of those Walsh coefficients whose corresponding partition contain the schema, and the sign of a particular coefficient is positive (negative) when the number of 1's covering the positions is even (odd). For instance the average fitness of the schema $***11$ is:

$$f(***11) = w_{00000} - w_{00001} - w_{00010} + w_{00011}$$

whereas the competing schemata in the partition $***ff$ have average fitnesses:

$$f(***00) = w_{00000} + w_{00001} + w_{00010} + w_{00011}$$

$$f(***01) = w_{00000} - w_{00001} + w_{00010} - w_{00011}$$

$$f(***10) = w_{00000} + w_{00001} - w_{00010} - w_{00011}$$

The connection with building blocks becomes clear if we also calculate the schema average fitness of the lower order schemata:

$$f(*****) = w_{00000}$$

$$f(****1) = w_{00000} - w_{00001}$$

$$f(***1*) = w_{00000} - w_{00010}$$

The particular Walsh coefficient w_{00011} can thus be interpreted as the magnitude of the fitness contribution due to the non-linear interaction of the bits. At the other hand when a particular Walsh coefficient of degree k is zero this means that there is no contribution due to the non-linear interaction of the k-bits within the corresponding partition. From a GA standpoint this implies that groups of bits corresponding with significant Walsh coefficients should be processed as a whole by the recombination operator - this is, they should have a high survival probability when applying crossover.

In the next section we discuss an algorithm that searches for exactly those significant Walsh coefficients.

2.2 KM ALGORITHM

The KM-algorithm computes the large Walsh coefficients by sampling and recursively extending partitions that might contain large Walsh coefficients. The algorithm receives as input the function f and a threshold parameter θ. It outputs a set of indices whose corresponding Walsh coefficients are large - this is, at least θ.

The algorithm when called with a partition α computes a boolean test and if this is true KM will recursively invoke the extended partitions $\alpha 0$ and $\alpha 1$. The boolean test computes whether or not a large Walsh coefficient can possibly be a member of one or both extended partitions. It conceptually does this by computing the sum of squares of the Walsh coefficients in each partition. If this sum is smaller than θ^2 then certainly no single Walsh coefficient can exceed our threshold θ, and there is no need to extend this partition. Formally KM looks as follows where it is initially called with α the empty string:

SUBROUTINE Search(α)
    IF $E[f_\alpha^2] \geq \theta^2$ THEN
        IF $|\alpha| = n$ THEN return α
            ELSE Search($\alpha 0$); Search($\alpha 1$)

Through the recursive extensions the Walsh coefficients are partitioned in a binary tree as shown in figure 1. Each node represents the sum of the squares of $2^k (k : l \rightarrow 0)$ Walsh coefficients with the root node summing all coefficients ($k = l$) and the leaf nodes representing a single coefficient ($k = 0$). Starting from the root node the KM-algorithm extends a node in

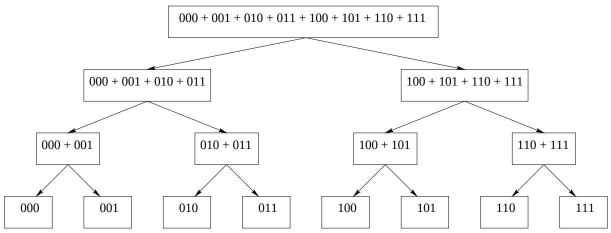

Figure 1: *Binary tree partitioning of the Walsh coefficients.*

the tree if the sum of the squares of the corresponding Walsh coefficients is larger than the square of the threshold θ until a leaf node is reached. Leaf nodes will only be visited whenever they have a significant value - this is, a value larger than the threshold θ (or actually when they or their immediate neighboring leaf node are a significant Walsh coefficient since the recursion expands each node into two child nodes during a single call).

Of course the time to calculate the sum of squares of the Walsh coefficients in a given partition, could be exponential. KM efficiently sidesteps this problem by estimating the sum of squares in the following way. Define the function $f_\alpha(x) : \{0,1\}^k \to \Re$ and $\alpha \in \{0,1\}^k$ as

$$f_\alpha(x) = \sum_{\beta \in \{0,1\}^{n-k}} w_{\alpha\beta}\phi_\beta(x),$$

or

$$f_\alpha(x) = E_{y\in\{0,1\}^k}[f(yx)\phi_\alpha(y)]$$

with $1 \le k < n$ and $x \in \{0,1\}^{n-k}$.

Now it can be shown that

$$\begin{aligned}\sum_{\beta\in\{0,1\}^{n-k}} w^2_{\alpha\beta} &= E[f^2_\alpha(x)] \\ &= E_x E^2_y[f(yx)\phi_\alpha(y)]\end{aligned}$$

with $x \in \{0,1\}^{n-k}$ and $y \in \{0,1\}^k$.

This gives us a way to estimate the sum of squares of the Walsh coefficients within a certain partition:

1. choose m_1 random $x_i \in \{0,1\}^{n-k}$
2. choose m_2 random $y_j \in \{0,1\}^k$
3. query f on $y_j x_i$
4. compute estimate of $E[f^2_\alpha]$ by:

$$\frac{1}{m_1}\sum_{i=1}^{m_1}\left(\frac{1}{m_2}\sum_{j=1}^{m_2} f(y_j x_i)\phi_\alpha(y_j)\right)^2$$

More details regarding the KM-algorithm and some implementations issues can be found in (Mansour, 1994; Mansour & Sahar, 1995)

3 KM AND GA PROBLEM-CODING

The KM-algorithm assumes that the function whose Walsh coefficients it searches is boolean valued. From Parseval's identity we know that the sum of the squares of all Walsh coefficients is equal to the average of the squares of all fitness values, so for boolean outputs $+1$ and -1 the squared Walsh coefficients sum to 1. This property is useful in determining the threshold above which a coefficient is called significant. In principle we could also use the KM-algorithm for functions that map to real values, and estimate the average of the squared fitness values by sampling.

Here we have chosen to work with boolean valued functions so the Walsh coefficients we obtain are not those of the actual fitness function. Instead we define a boolean valued function by assigning the value $+1$ to the 50% most fit strings of a randomly generated population, and -1 to the other half. The most significant Walsh coefficients of this boolean function actually identify the groups of non-linear interacting bits that have an influence on whether or not a string has a fitness value above or below the median value in the population. Applying KM in this way identifies the important building blocks.

One advantage of estimating the coefficients of the boolean *population-top-half* concept is that the Walsh coefficient corresponding with the all zeroes index has a very low magnitude. Remember that this coefficient actually computes the average fitness and since we are basically comparing against the median value this coefficient is quite low. In most functions this coefficient is by far the most significant one as compared to the ones who detect the nonlinear interaction, which is undesirable for sampling purposes.

To illustrate our use of KM we have applied the algorithm to two GA-hard functions. Both are composed by concatenating 10 deceptive trap functions of length 4 and fitness signal 0.25. In the first function (F_{uni}) all trap functions have an equal fitness contribution, whereas in the second one (F_{lin}) the fitness values of trap functions are linearly scaled.

Call $o(x)$ the number of one-bits then the function value of our trap function is given by

$$t(x) = 0.75 - (0.25.o(x)),$$

while if all bits are equal to one ($o(x) = k$) the function value reaches its maximal value

$$t(x) = 1.$$

Calling the ith trap function in our test functions $t_i(x)$ then we have

$$F_{uni}(x) = \sum_{i=1}^{10} t_i(x)$$

and

$$F_{lin}(x) = \sum_{i=1}^{10} i.t_i(x)$$

We have chosen as threshold $\theta = 0.1$ so KM returns all Walsh coefficients with a squared value at least $(0.1)^2/2 = 0.0050$ (note that KM takes half of θ^2 in the estimation procedure as a safeguard against sampling errors).

Table 1 shows the significant Walsh coefficients and their corresponding indices for the uniformly scaled concatenated trap functions F_{uni}. Clearly the algorithm identifies all the deceptive trap functions.

Table 2 shows the significant Walsh coefficients and their corresponding indices for the linearly scaled concatenated trap functions. Now the five most highly scaled trap functions are identified, while the lower significant trap functions are below the threshold.

Table 1: Significant Walsh Coefficients: F_{uni}

w^2	Walsh index set
0.0150	00000000..................... 00000000
0.0066	 00000011
0.0082	 00000101
0.0060	 00000110
0.0067	 00000111
0.0063	 00001001
0.0077	 00001010
0.0074	 00001011
0.0074	 00001100
0.0068	 00001101
0.0076	 00001110
0.0068	 00001111
......	....
0.0055	00110000 ....
0.0062	01010000 ....
0.0072	01100000 ....
0.0069	01110000 ....
0.0073	10010000 ....
0.0081	10100000 ....
0.0070	10110000 ....
0.0068	11000000 ....
0.0067	11010000 ....
0.0075	11100000 ....
0.0076	11110000 ....

4 DISCUSSION

The Walsh transform computes all bitwise nonlinear interactions that exist in a certain fitness function given some binary problem-coding. Since there can be exponentially many sources of nonlinearity computing all Walsh coefficients exactly is in general intractable. The KM-algorithm estimates the most significant Walsh coefficients in a computational efficient way for functions that can well be approximated by a t-sparse function - this is, a function with at most t non-zero Walsh coefficients. Although one might think at first that this is a severe limitation, it is clear from the previous section that such functions are in no way by definition trivial to solve by a simple genetic algorithm. Deceptive trap functions cannot be solved efficiently by a simple GA unless one knows the linkage information - this is, the groups of bits that cause the nonlinear fitness interactions. When the bits with high Walsh coefficients are placed close together on the genome, a simple GA could easily solve the deceptive trap functions to global optimality.

However one should not conclude from this that once we know the significant Walsh coefficients of functions with a modest amount of non-zero Walsh coefficients then this should lead to an efficient GA to optimise the problem. Indeed recent work of Rana and her coworkers (Rana, Heckendorn, & Withley, 1998) showed that Boolean Satisfiability 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. Since MAXSAT is NP-Complete the GA cannot be expected to find the global optimum efficiently and reliably even when given all the information about the nonlinear fitness interactions in the problem-coding (unless of course, $P = NP$ which is extremely unlikely). Nevertheless, knowing the structure of nonlinearities of a problem should allow to design more competent GAs.

The approach could also be beneficial to investigate certain genome problem-coding issues. For instance there is some debate in the GA community whether binary integer coding or Gray coding is better for optimizing some functions. In general Gray coding seems

Table 2: Significant Walsh Coefficients: F_{lin}

w^2	Walsh index set
0.0055	00000000.................... 00000000
0.0075	00000000 ...... 0011 00000000
0.0075	00000000 ...... 0101 00000000
0.0080	00000000 ...... 0110 00000000
0.0051	00000000 ...... 0111 00000000
0.0075	00000000 ...... 1001 00000000
0.0072	00000000 ...... 1010 00000000
0.0069	00000000 ...... 1100 00000000
0.0067	00000000 ...... 1101 00000000
0.0068	00000000 ...... 1110 00000000
0.0068	00000000 ...... 1111 00000000
......	....
0.0081	00010000 ....
0.011	00100000 ....
0.016	00110000 ....
0.0088	01000000 ....
0.015	01010000 ....
0.014	01100000 ....
0.016	01110000 ....
0.0088	10000000 ....
0.017	10010000 ....
0.015	10100000 ....
0.015	10110000 ....
0.015	11000000 ....
0.013	11010000 ....
0.015	11100000 ....
0.013	11110000 ....

to be less prone to hamming cliffs, but going from binary integer coding to Gray coding does introduce other non-linearities. Using the approach outlined here it can be investigated where the most significant non-linearities are situated. We have tested this on some functions and results show that the most significant Walsh coefficients can be found in substantially different partition orders for both codings. Whether the binary integer coding or the Gray coding have fewer significant higher order non-linearities depends on the fitness function.

Although the KM-algorithm runs in polynomial time, the number of samples it takes is quite large. As a theoretical tool this is not so much of a problem, since there we would only be interested in the results obtained - given a non-exponential time of computation. It is tempting though to ask whether estimating significant Walsh coefficients by sampling could not be used more directly to guide a genetic algorithm at run time. For this to become feasible the number of samples needed should be reduced. In its standard implementation KM is rather wasteful on the samples it takes. For instance when estimating the value of $E[f_\alpha^2(x)]$ a large number of samples are evaluated and thereafter discarded. If the obtained value is larger than the threshold θ^2 KM will recurse and compute an estimate of $E[f_{\alpha 0}^2(x)]$ and $E[f_{\alpha 1}^2(x)]$. All the discarded samples however could well be used again in the calculation of the new estimates. This simple procedure would actually reduce the amount of samples needed by half.

A second possible source of more efficient sampling could be found in the sequence at which the partitions are sampled. KM simply goes from left to right on the string, but it could be beneficial to adapt this sequence of expanding according to the salience of the partitions and corresponding schemata.

It remains to be investigated though if these and other changes could turn a sample based Walsh coefficient estimator into an algorithm efficient enough to guide GAs at run time.

In related work Kargupta and Sarkar also present a polynomial time algorithm to compute the Walsh coefficients under certain conditions (Kargupta and Sarkar, 1999). Their restricting conditions are that all Walsh coefficients above a certain order k are assumed to be zero. After computing all the order k Walsh coefficients - requiring $2^k \binom{l}{k}$ function evaluations - the remaining coefficients of order less than k can be computed without further function evaluations. An even more efficient algorithm is proposed which is applicable when in addition to the above conditions all the Walsh coefficients are also non-negative.

5 CONCLUSION

We have discussed the KM-algorithm in the context of genetic algorithms. KM allows one to estimate the most significant Walsh coefficients in a computational efficient way for a large class of functions. Knowing these coefficients gives valuable information about the sources of nonlinear fitness interactions in the problem-coding. This insight can be used for instance for evaluating a problem-coding, or for designing genetic recombination operators.

The Walsh transform has continued to be an important theoretical tool ever since Bethke's dissertation (Bethke, 1981). Over the past years the Walsh transform has also witnessed a steadily increasing interest from the Computational Learning community, which has led among others to the interesting KM-algorithm.

It has been the purpose of this paper to draw the attention of the GA community to this work. Theoretical tools that might supply the genetic algorithm practitioner with valuable insight are hard to come by, and KM or similar analysis algorithms certainly seem to have a lot of potential for helping to achieve this effort.

References

Bethke, A. D. (1981). Genetic algorithms as function optimizers. (Doctoral dissertation University of Michigan). *Dissertation Abstracts International 41(9), 3503B.*

Goldberg, D. E. (1992). Construction of higher-order deceptive functions from low-order Walsh coefficients. *Annals of Mathematics and Artificial Intelligence, Vol.5*, p.35-48.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems, 3*, p.129-152.

Goldberg, D. E. (1989b). Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems, 3*, p.153-171.

Kargupta, H., and Sarkar, K. (1999). Function Induction, Gene Expression, and Evolutionary Representation Construction. *(this volume).*

Kushilevitz, E. , and Mansour, Y. (1993). Learning Decision Trees using the Fourier Spectrum. *SIAM Journal on Computing, 3* 22(6):1331-1348.

Mansour, Y. (1994). Learning Boolean Functions via the Fourier Transform. *Advances in Neural Computation.* eds. Roychodhury, Siu, and Orlithy. Kluwer Academic Publishers.

Mansour, Y., and Sahar, S. (1995) Implementation Issues in the Fourier Transform Algorithm. *Proceedings of the Neural Information Processing Systems 8*, p.260-266.

Rana, S, and Whitley, D. (1998). Genetic Algorithm Behavior in the MAXSAT Domain. *Parallel Problem Solving from Nature - PPSN V*, Lecture Notes in Computer Science, Vol.1498, p.785-794.

Rana, S, Heckendorn, R. B., and Whitley, D. (1998). A Tractable Walsh Analysis of SAT and its Implications for Genetic Algorithms. *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, p.392-397.