Function Induction, Gene Expression, And Evolutionary Representation Construction

Hillol Kargupta
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164

Kakali Sarkar
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131

Abstract

Different portions of the DNA, the primary information career of a living organism, are evaluated in different cells through the process of gene expression (DNA→mRNA→Protein). Such distributed evaluation of the fitness is possible only when its distributed representation using a set of basis functions is available in a living body. This paper argues that unless the evolution was provided with such a representation, we have every reason to believe that there must be an efficient mechanism to construct such a distributed representation. This paper considers functionally complete Walsh basis functions and shows that efficient polynomial-time computation of the Walsh representation (WR) is possible for problems with bounded non-linearity. It also offers a highly efficient algorithm $O(2^k r)$ to compute the WR for problems with non-negative Walsh coefficients, where r is the total number of non-zero terms in its WR.

1 INTRODUCTION

The last fifty years of this century witnessed the gradual development of different search algorithms that use motivations from natural evolution. The Genetic algorithms (GAs) [7], *evolutionstrategie* [17], evolutionary programming [3], and genetic programming [11] are some examples that found many successful applications in search, optimization, and machine learning. Despite all the success stories of these different approaches and half-a-century of research, there still remain many unanswered questions. Scalability of the evolutionary search for adaptive organisms is one among those many mysteries. The fundamental mechanism behind the relatively quick evolution (only about a couple of billions of years compared to the huge evolutionary search space)[8] of amazing forms of life is yet to be explained in the light of what we know about the theory of learning, adaptation, and optimization.

This paper takes a modest step in unraveling this mystery. It explores evolutionary search in the light of our existing understanding about inductive construction of functions from data, in short, function induction. It argues that the distributed evaluation of the fitness of a DNA through the gene expression (the transformation of DNA to Protein through the formation of mRNA) demands the existence of a scalable, efficient evolutionary mechanism for inducing functions in a distributed representation. Moreover, it demonstrates the feasibility of this argument by identifying a polynomial time algorithm to do so for problems with bounded non-linearity using Walsh analysis.

Section 2 discusses the role of function induction in learning, adaptation, and optimization. Section 3 discusses the distributed fitness evaluation through the gene expression process and argues the need for an efficient function induction mechanism in our model of evolutionary computation. Section 4 briefly overviews the Walsh basis representation. Section 5 suggests a possible evolutionary mechanism to compute Walsh representation in polynomial time. Section 6 offers a very efficient algorithm for computing the Walsh representation for functions with non-negative Walsh coefficients. Section 7 concludes this paper.

2 FUNCTION INDUCTION IN LEARNING, ADAPTATION, AND OPTIMIZATION

Function induction plays an important role in machine learning, adaptation, and non-enumerative black-box optimization. In function induction the goal is to learn a function $\hat{f}: X^n \rightarrow Y$ from the data set $\Omega = \{(\mathbf{x}_{(1)}, \mathbf{y}_{(1)}), (\mathbf{x}_{(2)}, \mathbf{y}_{(2)}), \cdots (\mathbf{x}_{(k)}, \mathbf{y}_{(k)})\}$ generated by underlying function $f: X^n \rightarrow Y$, such that the $\hat{f}$ approximates f. Any member of the domain $\mathbf{x} = x_1, x_2, \cdots x_\ell$ is an ℓ-tuple and x_j-s correspond to individual feature variables of the domain.

Empirical machine learning [14] is directly based on the function induction. Given a set of observed behaviors, the goal of empirical machine learning is to learn a function that closely approximates those behaviors, which is essentially the function induction problem itself when the behavior is viewed as a function from the domain of different situations.

Function induction also plays a critical role in natural and artificial adaptation. The word adaptation literally means "to fit to" (*ad* + *aptare*). However, in general we call a system adaptive when it can exhibit appropriate behavioral dynamics in response to changes in an uncertain environment. Inductive function learning has a close relation with adaptation [7]. Adaptation in uncertain environments typically requires learning probabilistic behavior, learning to predict uncertain future, learning strategies to outsmart competitors, learning to co-operate, and others. All these learning problems typically involve empirical function induction from observed data.

Optimization in the absence of sufficient prior domain knowledge to guide search directions is often called black-box optimization (BBO). Typical BBO algorithms like evolutionary algorithms, simulated annealing [10] sample the search space and make inductive decisions in order to explore the promising parts of the search space. The inductive decisions are fundamentally based on function induction. Here again, the optimization algorithm needs to guess intelligently about the landscape in either a local or a global sense from the samples taken from the search domain. This requires function induction. Clearly, the success of inductive BBO fundamentally depends on the success of function induction.

Learning a function from data is indeed a common problem in many other domains like data mining, software engineering, pattern recognition, signal processing. Like all these problem domains, the emergence of adaptive living organisms through evolutionary search critically depends on function induction. Natural evolution can be viewed as a search for evolving systems that can adapt and survive the competition of natural selection. If this perspective is correct then the living organisms must have an efficient mechanism to learn functions from observed data. Indeed, there exist many facts that corroborate this observation. The role of neural networks in our brain is now widely recognized to be an important mechanism for learning functions from observed data [18]. This paper suggests that there may be an alternate mechanism for function induction in the evolutionary operators of living organisms. The following section argues this possibility.

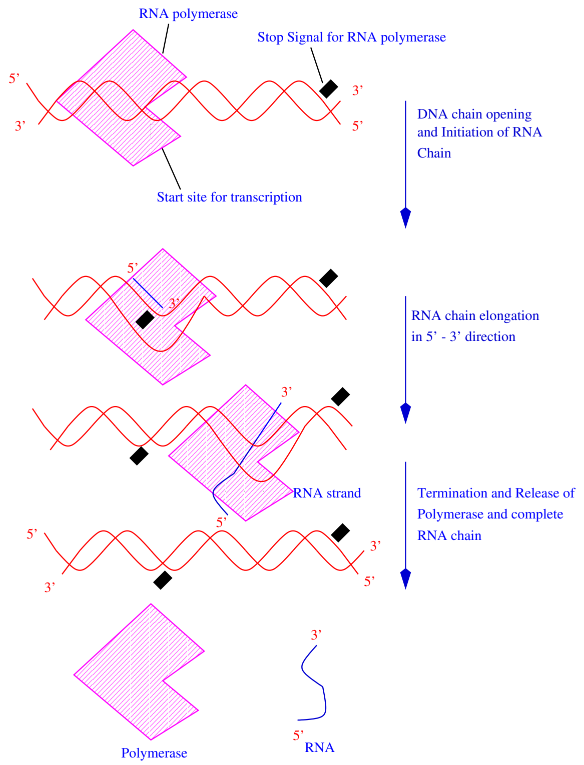

Figure 1: Transcription.

3 FITNESS EVALUATION THROUGH GENE EXPRESSION

Evaluation of the fitness of an organism in nature is a fairly interesting process. In order to fully appreciate this process we need to understand the biology to some extent. The following discussion tries to accomplish this.

DNA is the primary carrier of the evolutionary information that is transmitted from one generation to another. DNA molecules consist of two long complementary chains held together by base pairs. DNA consists of four kinds of bases joined to a sugar-phosphate backbone. The four bases in DNA are *adenine* (A), *guanine* (G), *thymine* (T) and *cytosine* (C). Chromosomes are made of DNA *double helices*. Bases in DNA helices obey the *complementary base pairing rule*. T and G pair with A and C respectively. In other words, if the base at a particular position of a helix is T then the corresponding base in the other helix should be A.

Evaluation of the fitness of a DNA takes place through a process called gene expression. Expression of genetic information coded in DNA requires construction of the mRNA sequence, followed by that of proteins. The main steps are,

- transcription: formation of mRNA (messenger ribonucleic acid) from DNA
- translation: formation of protein from mRNA
- protein folding

In a particular cell, transcription produces the mRNA from a small portion of the DNA. The mRNA defines another level of representation of the genetic information. It consists of four types of bases joined to a ribose-sugar-phosphodiester backbone. The four bases are *adenine* (A), *uracil* (U), *guanine* (G), and *cytosine* (C). All the bases defining the mRNA are same as those in DNA sequences, except that T is replaced by U. As in Figure 1, the mRNA is produced from the DNA by RNA Polymerase and the regulatory proteins following the *complementary base-pairing rules* similar to those in DNA. The RNA Polymerase initiates the transcription at a place of the DNA marked by the *promoter* region (*start site*). It splits the DNA double helix and continue generating the mRNA using one of the DNA strands as a template. The RNA Polymerase stops when it finds a termination signal sequence (*stop site*)in the DNA strand. Note that only a small portion of the DNA strand is transcribed and different cells may transcribe different regions of the DNA for producing proteins.

The mRNA acts as the template for protein synthesis. A protein is defined by a sequence of *amino acids*, joined by peptide bonds. The mRNA is transported to the cell cytoplasm for producing protein in the ribosome. There exists a unique set of rules that defines the correspondence between nucleotide triplets (known as codons) and the amino acids in proteins. This is known as the *genetic code*. Each codon is comprised of three adjacent nucleotides in a DNA chain and it produces a unique amino acid. Amino acid sequence defines a new representation of the information coded in mRNA.

The final level of representation of genetic information is defined by the three dimensional structure of folded proteins. Although amino acid sequences fundamentally define proteins, formation of the three dimensional structure of proteins involves a complex process, often called *protein folding*. This process involves interaction between multiple amino acid subsequences.

Since proteins play a key role in the performance of a living organism, we can view the organism's fitness as a direct function of the different proteins generated from the DNA. Since different proteins are generated at different cells from different portions of the DNA, fitness evaluation in natural gene expression appears to take place in a distributed fashion. In other words, the fitness evaluation seems to be decomposed into different sub-function evaluations. Such distributed fitness evaluation is possible under either of the two following conditions:

1. the distributed representation of the fitness function was available to evolutionary mechanism a priori;
2. the distributed representation of the function is inductively constructed from the sample DNAs and their respective fitnesses, observed in nature.

In the following section we shall adopt the latter possibility and explore it in the context of our basic understanding about function induction.

4 FUNCTION INDUCTION AND DISTRIBUTED REPRESENTATION

Any function can be represented in a distributed, decomposed fashion using an appropriate set of basis functions. Let Ξ be a set of basis functions. Ξ does not necessarily have to be finite. Let us index the basis functions in Ξ and denote the k-th basis function in Ξ by Ψ_k. Let I be the set of all such indices of the basis functions. A function $f(\mathbf{x})$ can be represented as,

$$f(\mathbf{x}) = \sum_{k \in I} w_k \Psi_k(\mathbf{x}) \tag{1}$$

Where $\Psi_k(\mathbf{x})$ denotes the k-th basis function and w_k denotes the corresponding coefficient. The objective of function induction can be viewed as the task to generate a function, $\hat{f}(\mathbf{x}) = \sum_{k \in \hat{I}} \hat{w}_k \Psi_k(\mathbf{x})$, that approximates $f(\mathbf{x})$ from a given data set; $\hat{I}$ denotes a subset of I; $\hat{w}_k$ denotes the approximate estimation of the coefficients w_k. For a given basis representation, the underlying inductive task can be viewed as the problem to compute the non-zero, significant (not negligible) coefficients, $\hat{w}_k$-s. Let us illustrate this using a simple example. Consider a function of boolean variables, x_1, x_2, and x_3. We can write,

$$f(x_1, x_2, x_3) \quad = \quad w_0 + w_1\Psi_1(x_1) + w_2\Psi_2(x_2) +$$

$$w_3\Psi_3(x_3) + w_4\Psi_4(x_1x_2) +$$
$$w_5\Psi_5(x_1x_3) + w_6\Psi_6(x_2x_3) +$$
$$w_7\Psi_7(x_1, x_2, x_3)$$

Where, w_i-s are constant coefficients; $\Psi_i(.)$-s are monomial basis functions like Walsh functions [1] to be defined later. At this point all we need to note is that we can write f as a linear sum of a set of basis functions, weighted by the corresponding basis coefficients. In other words, computation of $f(x_1, x_2, x_3)$ can be performed by computing different sub-functions that require only a subset of the feature set x_1, x_2, x_3. Although computation of $\Psi_7(x_1, x_2, x_3)$ requires information about all the three variables, if the value of w_7 is close to zero then we can decompose the computation of $f(x_1, x_2, x_3)$ into different components that require only partial information about the feature set.

Distributed evaluation of fitness function is possible only when we have its representation using a set of basis functions and their corresponding coefficients. The distributed fitness evaluation in gene expression is no exception. The efficacy of a certain portion of the DNA depends on the 3-dimensional structure of the protein it produces; since the shape of proteins depends on different physical factors such as energy, bond properties and others, the set of basis functions used in nature are likely to be functions of these physical factors. However, the exact mathematical structure of these basis functions is not yet known. Therefore, in the following discussion we choose to work with Walsh basis functions, since it is functionally complete over the space of all boolean strings and equivalent to other choices of basis functions in this space. The following discussion offers a brief overview of Walsh representation.

Walsh functions [1] are orthogonal functions that found applications in many different fields such as signal processing, image analysis, and others. Like Fourier, Laplace, and other transformations, Walsh functions are often used to represent the representation in a convenient form. Application of Walsh transformation (WT) in understanding Genetic Algorithms was first noted by Bethke [2]. Further investigation of this approach can be found elsewhere [4, 5, 6, 13, 16]. Traditionally, the Walsh functions are used for representing real valued functions of binary variables. However, they can be easily extended to higher cardinality representation, as shown elsewhere [15]. Although the main arguments of the following discussion can be extended for higher cardinality representations, in this paper we shall restrict ourselves to binary variables.

The Walsh basis set is comprised of 2^ℓ Walsh functions, where each basis function is defined as follows:

$$\psi_{\mathbf{j}}(\mathbf{x}) = (-1)^{(\mathbf{x}.\mathbf{j})} \tag{2}$$

Where $\mathbf{j}$ and $\mathbf{x}$ are binary strings of length ℓ. In other words $\mathbf{j} = j_1, j_2, \cdots j_\ell$, $\mathbf{x} = x_1, x_2, \cdots x_\ell$ and $\mathbf{j}, \mathbf{x} \in \{0,1\}^\ell$. $\psi_{\mathbf{j}}(\mathbf{x})$ can either be 1 or -1. The string $\mathbf{j}$ is called a *partition*. The *order* of partition $\mathbf{j}$ is the number of 1-s in $\mathbf{j}$. Since a Walsh function depends on some x_i only when $j_i = 1$. Therefore a partition can also be viewed as a representation of a certain subset of x_i-s; every unique partition corresponds to a unique subset of x_i-s. If a partition $\mathbf{j}$ has exactly α number of 1-s then we say partition is of order α since the corresponding Walsh function is a function of only those variables corresponding to the 1-s in the partition $\mathbf{j}$. A function $f(\mathbf{x})$ can be written using the Walsh basis functions as follows:

$$f(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}}\psi_{\mathbf{j}}(\mathbf{x}) \tag{3}$$

where $w_{\mathbf{j}}$ is the Walsh Coefficient (WC) corresponding to the partition $\mathbf{j}$ as defined in the following,

$$w_{\mathbf{j}} = \frac{1}{2^\ell}\sum_{\mathbf{x}} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) \tag{4}$$

We note from Equation 3 that a function can be expressed as a linear sum of the Walsh functions, each weighted by the corresponding Walsh coefficients. The Walsh coefficient $w_{\mathbf{j}}$ can be viewed as the relative contribution of the partition $\mathbf{j}$ to the function value of $f(\mathbf{x})$. Therefore, the absolute value of $w_{\mathbf{j}}$ can be used as the "significance" of the corresponding partition $\mathbf{j}$.

Partitions with non-zero Walsh coefficients reflect the underlying non-linearity of the given problem. For example, consider a function $f(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) + f_2(x_3, x_4)$. In the Walsh representation the values of w_{1111}, w_{1110} and any such other Walsh coefficient corresponding to partitions involving a pair of variables, one each from the two linearly decomposable partitions, are zero. Walsh functions can be used for representing any functions of boolean variables. If a function has non-linearity then its Walsh representation will reflect that and the function can be evaluated by computing the Walsh functions corresponding to the non-zero Walsh coefficients. If every variable in a function non-linearly interacts with every other variable then in the general case the Walsh representation will have all the 2^ℓ terms. On the other hand if at most some k variables non-linearly interact with each other, all the Walsh coefficients corresponding to partitions with more than k number of 1-s will be zero. This class of problems will be called problems with order-k

non-linearity. Detection of such non-linearity among the genes is traditionally called *linkage learning* in the genetic algorithms literature. A perspective of linkage learning using Walsh representation has been proposed elsewhere [9, 19].

If the distributed fitness evaluation of the DNA through gene expression is a result of evolutionary search, then evolution must have a mechanism for constructing such a distributed representation of the fitness function. Walsh basis functions offer one way to decompose and distribute the fitness evaluation. However, construction of such a representation in a relatively small period of time (in the evolutionary scale) is unlikely to happen unless it is fundamentally possible to do so in rigorous computational ground. The following section investigates the issue of polynomial time computation of Walsh representation and offers algorithms for that.

5 POLYNOMIAL-TIME EVOLUTIONARY REPRESENTATION CONSTRUCTION

As we saw in the previous section (Equation 4) computation of a single Walsh Coefficient requires information about all the 2^ℓ domain members. Clearly this cannot be done in time, polynomial in ℓ. In order to make the problem tractable we are going to assume that the problem has bounded non-linearity of order-k. This is a reasonable assumption for many practical domains. This results in a sparse Walsh representation. In general, function induction is fundamentally difficult if the orthonormal representation is not sparse [12]. Even if the problem has bounded non-linearity, explicit computation of WCs using Equation 4 requires exponential time. Another possibility is to generate $m = \sum_{z=0}^{k} \binom{\ell}{z}$ different members from the domain and solve a large $(m \times m)$ linear system of equations in order to compute the m possibly non-zero terms in the Walsh representation. Although it is theoretically possible in polynomial time, it is not clear how and if at all the evolutionary search is equipped with any mechanism to do it this way. In the following discussion we explore an alternate way to compute individual WCs in an efficient and distributed manner that appears to fit naturally into the realm of population based evolutionary computation.

From Equation 3 we can write,

$$f(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x}) = \sum_{\mathbf{j}} w_{\mathbf{j}}\psi_{\mathbf{j}}(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x})$$

Let $\mathbf{i}$ be a partition and $S(\mathbf{i})$ be the set of all strings that satisfies the following conditions: (1) every member of $S(\mathbf{i})$ has same values at all positions where there is a 0 in $\mathbf{i}$ and (2) the substring defined by the positions corresponding to 1-s in $\mathbf{i}$ is unique in every member of $S(\mathbf{i})$. Let us denote the invariant values at the positions, corresponding to 0-s in $\mathbf{i}$ by $\mathbf{T}$ and call it the *template*. For example, if $\mathbf{i} = 001100$ then one possible choice of S may be $\{000000, 001000, 000100, 001100\}$. $|S(\mathbf{i})|$ is the size of the set $S(\mathbf{i})$ and $|S(\mathbf{i})| = 2^k$ when the partition $\mathbf{i}$ is of order-k. In this the template $\mathbf{T} = 00\_\_00$. Since there exists different such $S(\mathbf{i})$-s depending on the choice of $\mathbf{T}$, we may choose to use the symbol $S_{\mathbf{T}}(\mathbf{i})$ to denote the set $S(\mathbf{i})$ with respect to certain template $\mathbf{T}$, wherever needed. Now we can write,

$$\begin{aligned} \sum_{\mathbf{x}\in S_{\mathbf{T}}(\mathbf{i})} f(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x}) &= \sum_{\mathbf{j}} w_{\mathbf{j}} \sum_{\mathbf{x}\in S_{\mathbf{T}}(\mathbf{i})} \psi_{\mathbf{j}}(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x}) \\ &= \sum_{\mathbf{j}\in J(\mathbf{i})} w_{\mathbf{j}} \sum_{\mathbf{x}\in S_{\mathbf{T}}(\mathbf{i})} \psi_{\mathbf{j}\oplus\mathbf{i}}(\mathbf{x}) \end{aligned}$$

Where $J(\mathbf{i})$ denotes the set of all partitions that completely subsumes partition $\mathbf{i}$. In other words, every partition in $J(\mathbf{i})$ must have a 1 at every location where there is a 1 in $\mathbf{i}$. $\mathbf{j}\oplus\mathbf{i}$ denotes the partition defined by the boolean XOR between $\mathbf{j}$ and $\mathbf{i}$. Now let us rewrite the above equation in the following form.

$$\begin{aligned} |S_{\mathbf{T}}(\mathbf{i})|w_{\mathbf{i}} &+ \sum_{\mathbf{j}\in J'(\mathbf{i})} w_{\mathbf{j}} \sum_{\mathbf{x}\in S_{\mathbf{T}}(\mathbf{i})} \psi_{\mathbf{j}\oplus\mathbf{i}}(\mathbf{x}) \\ &= \sum_{\mathbf{x}\in S_{\mathbf{T}}(\mathbf{i})} f(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x}) \end{aligned} \qquad (5)$$

Where $J'(\mathbf{i})$ is $J(\mathbf{i})$, excluding the partition $\mathbf{i}$. Since we can choose any invariant set of values for the template, let us consider a special case where all the required values of the template are set to 0. Let us denote the $S_{\mathbf{T}}(\mathbf{i})$ corresponding to this special case template $\mathbf{T} = \mathbf{0}$ by $S_{\mathbf{0}}(\mathbf{i})$. Note that the example that we gave earlier follows this case. Equation 5 can be specialized for this case as follows.

$$\sum_{\mathbf{j}\in J(\mathbf{i})} w_{\mathbf{j}} = \frac{1}{|S_{\mathbf{0}}(\mathbf{i})|} \sum_{\mathbf{x}\in S_{\mathbf{0}}(\mathbf{i})} f(\mathbf{x})\psi_{\mathbf{i}}(\mathbf{x}) \qquad (6)$$

Note that we included the first term of Equation 5 inside the summation and therefore replaced J' by J. Equation 6 can be used to efficiently detect the significant Walsh coefficients of the function with bounded non-linearity. Recall that for functions with at most k variables non-linearly interacting with each other, all the WCs corresponding to partitions with more than k 1-s will be zero. Therefore, for all partitions $\mathbf{i}$ with more than k number of 1-s, $J'(\mathbf{i})$ will be a null set.

For any partition with the order equal or more to k, we can write,

$$w_{\mathbf{i}} = \frac{1}{|S_{\mathbf{0}}(\mathbf{i})|} \sum_{\mathbf{x} \in S_{\mathbf{0}}(\mathbf{i})} f(\mathbf{x}) \psi_{\mathbf{i}}(\mathbf{x}) \tag{7}$$

Note that the computation of $w_{\mathbf{i}}$ requires only 2^k evaluations of $f(\mathbf{x})$. The bounded non-linearity property can therefore be exploited to compute the WCs using only a small number of evaluations (2^k) compared to the usual 2^ℓ evaluations needed using the regular approach. Once we compute all the k-th order non-zero WCs using Equation 7, lower order coefficients can be computed using the known higher order coefficients and Equation 5. The main algorithmic steps of this technique can be summarized as,

1. select $k \leq \ell$, some constant that bounds the highest order non-linearity of the given problem;

2. compute the order-k WCs using Equation 7;

3. use Equation 5 and the already evaluated order-k WCs in order to compute order-$(k-1)$ WCs; continue this process iteratively for $(k-1), (k-2), \cdots$ 1-order WCs; note that no additional function evaluation is needed after the order-k WCs are computed.

This technique can find all the non-zero WCs of a problem with order-k non-linearity in polynomial time using $2^k \binom{\ell}{k}$ function evaluations. The computation of individual WCs simply requires computing the average of 2^k fitness values and a few addition and subtraction. It is also fairly distributed and it does not require solving a large system of linear equations. This technique can be made further efficient when the WCs are all non-negative. The following section discusses this.

6 SPECIAL CASE: FUNCTIONS WITH NON-NEGATIVE WALSH COEFFICIENTS

The Walsh representation construction can be made further efficient when all the WCs are non-negative. Consider Equation 6. Note that when the right hand side of Equation 6 is zero and the WCs are all non-negative, we can conclude that all the WCs in $J(\mathbf{i})$ must be individually equal to zero. Therefore Equation 6 can be used to determine whether there exists any partition with non-zero WCs that completely subsumes a given partition $\mathbf{i}$ or not. Figure 2 illustrates

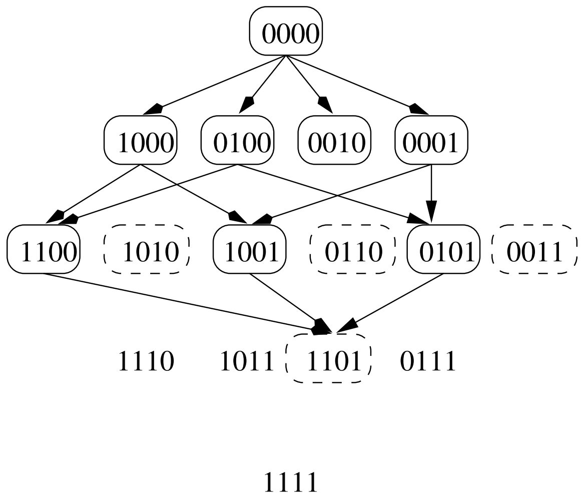

Figure 2: Efficient detection of non-negative Walsh Coefficients.

this approach. It shows the lattice of all sixteen partitions for a four-bit problem. Let us say, we have a function for which only the partitions, encircled with solid lines, have non-zero WCs. The proposed algorithm first computes the right hand side of Equation 6 for all nodes for order-1 partitions. For this problem at each of these nodes the test will result in non-zero value, indicating that the partition is subsumed by some partitions with non-zero WC. Next we move to the third level from top in Figure 2. Since all the order-2 partitions subsume some set of order-1 partitions and all the order-1 partitions returned non-zero value for our test, we need to apply our test at every node corresponding to the order-2 partitions. Our test will return non-zero values only for 1100, 1001, and 0101. Next we consider every unique order-3 partition that subsumes at least one of the partitions 1100, 1001, and 0101 but do not subsume any of the partitions 1010, 0110, and 0011. There is only one such partition and that is 1101. We again apply our test at this node and observe that test returns a zero value. Next we backtrack to the order-2 partition level and note that the outcomes of the tests that we performed earlier at each of these nodes are essentially the WC of the corresponding node since the coefficient of the partition 1101 is zero. Similarly, we continue to backtrack and compute the coefficients of every non-zero partition. This technique requires $O(2^k r)$ function evaluations, where r is the total number of non-zero WCs in its Walsh representation.

It may be possible to extend the above algorithm to the general case where the functions can have both negative and non-negative WCs. From Equation 4 we can write,

$$w_{\mathbf{j}} = \sum_{\mathbf{x}\in\Omega^{+}(\mathbf{j})} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) + \sum_{\mathbf{x}\in\Omega^{-}(\mathbf{j})} f(\mathbf{x})\psi_{\mathbf{j}}(\mathbf{x}) \qquad (8)$$

where $\Omega^{+}(\mathbf{j})$ and $\Omega^{-}(\mathbf{j})$ define the set of all strings for which $\psi_{\mathbf{j}}(\mathbf{x})$ is 1 and -1 respectively. Clearly there exists a decomposition of any function $f(\mathbf{x}) = f_1(\mathbf{x}) - f_2(\mathbf{x})$ in such a way that both f_1 and f_2 have non-negative WCs. Note that this requires controlled generation of $\mathbf{x}$-s such the corresponding basis functions for a give j return either all positive or all negative. This simple construction shows that it is always possible to transform a given function so that all the WCs are non-negative. A simple way to do that is to translate the $f(\mathbf{0})$ value by some large enough positive constant, ρ. Since $\psi_{\mathbf{j}}(\mathbf{0}) = 1$ such translation will translate every WC by the amount $\frac{\rho}{2^{\ell}}$. If ρ is large enough then all the WCs will be non-negative. Although this makes our algorithm applicable, the problem is that this may destroy the underlying sparseness of the coefficients. In other words, coefficients that were originally zero will become non-zero because of the translation. This may give rise to an exponential value of r and as a result the algorithm may no longer remain computable in polynomial time.

An alternate approach to address this problem is to manipulate the representation in such a way that the coefficients become non-negative without destroying the sparseness. It is not clear how to do that. However, it is interesting to observe that complementary alphabet transformations in the transcription offer some control over the basis functions that may be useful.

Let $\mathbf{x}$ be a binary string and let $\mathcal{T}$ be an complementary operator that maps 1 to 0 and vice versa. Let $\mathcal{T}_{\beta}(\S)$ be the complementary transformation applied on the β-partition of $\mathbf{x}$. In other words, $\mathcal{T}_{\beta}(\S)$ complements only those positions of $\mathbf{x}$ where there is a 1 in β. Let us now consider the difference between the Walsh function $\psi_{\beta}(\mathcal{T}_{\beta}(\mathbf{x})$ and $\psi_{\beta}(\mathbf{x})$. Recall that $\psi_{\beta}(\mathbf{x})$ can be either -1 or 1. It turns out that $\psi_{\beta}(\mathcal{T}_{\beta}(\mathbf{x}) = \psi_{\beta}(\mathbf{x})$ when the number of transcribed features is even; on the other hand $\psi_{\beta}(\mathcal{T}_{\beta}(\mathbf{x}) = -\psi_{\beta}(\mathbf{x})$ when the number of transcribed features is odd. This can be understood as follows. Let g be the number of 1-s in β; g_1 and g_0 be the numbers of 1-s and 0-s in $\overline{x}$ within the partition defined by the 1-s in β. The number of ones in $\mathcal{T}_{\beta}(\mathbf{x})$ within the partition β, $g_1' = g_0 = g - g_1$. Now if both g and g_1 are even numbers then g_1' must be even. On the other hand if g and g_1 are even and odd numbers respectively then g_1' must be odd. In other words the sign of $\psi_{\beta}(\overline{x})$ does not change when $\overline{x}$ is transformed to a different string by applying $\mathcal{T}$ over an even number of bits. Consider $\psi_{0110}(1\underline{01}1) = -1$. Now if we apply $\mathcal{T}$ to the underscored bits we get $\psi_{0110}(1\underline{10}1) = -1$. A similar rationale can be developed for the case when g is odd.

Now consider a transformation of the representation of the domain in such a way that every $\mathbf{x}$ is represented by $\mathcal{T}_{\mathbf{1}}(\mathbf{x})$, where $\mathbf{1}$ is an ℓ-bit string of 1-s. Such representation flips the sign of the WC corresponding to every odd-order partition. However, it does not destroy the sparseness; WCs with a value of zero, remain zero. We may be able to exploit such complementation-based representation transformation techniques for computing the summation of all non-negative coefficients. However, this is only our hypothesis since this is very similar to the physical representation transformations in gene expression.

The following section concludes this paper and identifies the future work.

7 CONCLUSIONS

This paper notes that fitness evaluation in natural evolution takes place in a distributed fashion in different living cells through the process of gene expression. It argues that such distributed evaluation of the fitness is only possible when the function can be decomposed in a distributed fashion using a set of basis functions. Unless such a representation was available a priori, we have every reason to believe that there must be an efficient mechanism to compute such a distributed representation. This paper considers functionally complete Walsh basis functions and shows that efficient polynomial-time computation of the distributed representation is possible for problems with bounded nonlinearity. It also offers a highly efficient $O(2^k r)$ algorithm to compute the Walsh representation for problems with non-negative Walsh coefficients. This approach is based on the computation of the average fitness over a small set of size 2^k that appears plausible in a population based evolutionary algorithm. This approach also requires controlled generation of different samples from specific partitions, that may be accomplished using the evolutionary operators. Either transcription or mutation, and crossover may be used for such controlled sample generation. At this point our interest in the fundamental polynomial-time computation of such orthonormal representation. Further investigation is needed to identify the actual operator responsible for such controlled scheme in nature. We are also currently investigating ways to generalize the algorithm presented in Section 6 for the general class of problems with a combination of negative and non-negative Walsh coefficients.

Acknowledgments

This work was supported by the United States National Science Foundation Grant IIS-9803360.

References

[1] K. G. Beauchamp. *Applications of Walsh and Related Functions.* Academic Press, USA, 1984.

[2] A. D. Bethke. Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Efficiency of use of processing capacity. Tech. Rep. No. 197, University of Michigan, Logic of Computers Group, Ann Arbor, 1976.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, 1966.

[4] S. Forrest and M. Mitchell. The performance of genetic algorithms on Walsh polynomials: Some anomalous results and their explanation. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 182–189. Morgan Kaufmann, San Mateo, CA, 1991.

[5] D. E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3(2):129–152, 1989. (Also TCGA Report 88006).

[6] D. E. Goldberg. Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, 3(2):153–171, 1989. (Also TCGA Report 89001).

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[8] F. Hoyle and N. C. Wickramasinghe. *Evolution from Space.* Dent, London, 1981.

[9] H. Kargupta and S. Bandyopadhyay. A perspective on the foundation and evolution of the linkage learning genetic algorithms. Accepted in the Special Issue in Genetic Algorithms: The Journal of Computer Methods in Applied Mechanics and Engineering. Editors: Goldberg, D. E. and Deb, K.

[10] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] J. R. Koza. *Genetic programming: On programming computers by means of natural selection and genetics.* MIT Press, Cambridge, MA, 1992.

[12] S. Kushilevitz and Y. Mansour. Learning decision rees using fourier spectrum. In *Proc. 23rd Annual ACM Symp. on Theory of Computing*, pages 455–464, 1991.

[13] G. E. Liepins and M. D. Vose. Polynomials, basic sets, and deceptiveness in genetic algorithm s. *Complex Systems*, 5(1):45–61, 1991.

[14] R. S. Michalski. Theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, pages 323–348. Tioga Publishing Co, 1983.

[15] C. K. Oei. Walsh function analysis of genetic algorithms of nonbinary strings. Unpublished master's thesis, Urbana, 1992. University of Illinois at Urbana-Champaign, Department of Computer Science.

[16] S. Rana, R. B. Heckendron, and D. Whitley. A tractable Walsh analysis of SAT and its implications for genetic algorithms. In *Proceedings of the AAAI-98*, 1998. AAAI Press.

[17] I Rechenberg. Kybernetische lösungsansteuerung einer experimentellen forschungsaufgabe. Seminarvortrag, Hermann-Föttinger-Institut für Strömungstechnik der Technische Universität, Berlin, 1964.

[18] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1: Foundations.* MIT Press, Cambridge, Mass., 1 edition, 1986.

[19] D. Thierens. Estimating the significant nonlinearities in the genome problem-coding. To be published in the Proceedings of the Genetic and Evolutionary Computation Conference, AAAI Press, 1999.