Minimal Walsh Structure and Ordinal Linkage of Monotonicity-Invariant Function Classes on Bit Strings

Lee A. Christie
IDEAS Research Institute
Robert Gordon University
Aberdeen, United Kingdom
l.a.christie4@rgu.ac.uk

John A. W. McCall
IDEAS Research Institute
Robert Gordon University
Aberdeen, United Kingdom
j.mccall@rgu.ac.uk

David P. Lonie
School of Computing Science
and Digital Media
Robert Gordon University
Aberdeen, United Kingdom
d.p.lonie@rgu.ac.uk

ABSTRACT

Problem structure, or linkage, refers to the interaction between variables in a black-box fitness function. Discovering structure is a feature of a range of algorithms, including estimation of distribution algorithms (EDAs) and perturbation methods (PMs). The complexity of structure has traditionally been used as a broad measure of problem difficulty, as the computational complexity relates directly to the complexity of structure. The EDA literature describes necessary and unnecessary interactions in terms of the relationship between problem structure and the structure of probabilistic graphical models discovered by the EDA. In this paper we introduce a classification of problems based on monotonicity invariance. We observe that the minimal problem structures for these classes often reveal that significant proportions of detected structures are unnecessary. We perform a complete classification of all functions on 3 bits. We consider non-monotonicity linkage discovery using perturbation methods and derive a concept of directed ordinal linkage associated to optimization schedules. The resulting refined classification factored out by relabeling, shows a hierarchy of nine directed ordinal linkage classes for all 3-bit functions. We show that this classification allows precise analysis of computational complexity and parallelizability and conclude with a number of suggestions for future work.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms; Design; Performance; Theory

Keywords

Estimation of Distribution Algorithms; Linkage Learning; Monotonicity; Ordinal Selection; Perturbation Methods

1. INTRODUCTION

There has been a long standing interest in understanding the role of *structure* in black-box function optimization such as estimation of distribution algorithms (EDAs). Structure, or *linkage*, refers to the interactions between variables in a function. Traditionally this was thought of as building blocks, which are potentially disrupted by mutation and recombination operators [7]. There are several approaches to overcoming the problem of disruption of structure [3]:

1. Evolving representations/operators
2. Probabilistic modeling (EDAs)
3. Perturbation methods (PMs)

In this paper, we consider the latter two approaches, due to their explicit models for variable linkage. Any analysis of problem structure runs across the issue that there are infinitely-many functions for even a fixed number of variables. Under many ordinal-selection–based algorithms, many functions are equivalent as they rank solutions equivalently. Negligible differences in value may be considered irrelevant to optimization.

EDAs build a probability distribution of high-quality solutions from a population and sample that distribution to find other high-quality solutions. Many EDAs use ordinal selection, such as tournament selection, which uses comparison between fitness values as opposed to selection based on absolute fitness values. Such approaches are invariant under monotone transformations of the fitness functions, even though the model-building stages of the EDA may not be. Perturbation methods (PMs) make small changes to candidate solutions in a controlled way and calculate the effect of changes to multiple variables to directly infer which variables interact. PMs partition the variables into typically non-overlapping subsets of mutually-interacting variables. Many PMs use *non-monotonicity detection*, that is detecting whether the setting of one variable affects the optimal setting of another, as opposed to whether the effect of both variables is a linear sum of the effect of each.

We define *equivalence classes* of monotonicity invariant functions as sets of all functions that rank solutions identically. To fully explore a subset of all possible functions, the functions used in this paper are restricted to length 3 bit strings, that is functions of the form $f : \{0,1\}^3 \to \mathbb{R}$. In this domain, we can explore the different classes of functions possible to gain theoretical insight into approaches which may be extended to higher-dimensional function spaces.

2. BACKGROUND

2.1 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs) are an offshoot of genetic algorithms. EDAs construct a population of candidate solutions and model the distribution to sample more high quality solutions. They are able to overcome the problem of building-block disruption faced by genetic algorithms, by explicitly modeling the interactions between variables. Early EDAs use a *univariate* model, examples include PBIL [1] and UMDA [10]. In *bivariate* and *multivariate* EDAs the model is often a probabilistic graphical model (PGM). The underlying PGM aims to estimate the linkage between variables.

EDAs can be classified as *univariate*, *bivariate*, or *multivariate*, according to the complexity of the probabilistic model built [14]. In addition, the same terms are also used to classify the complexity of the functions based on the complexity of interactions between the variables. Hence, it is desirable that the model built by an EDA mirrors the underlying structure of the function. Probabilistic graphical models have been used to create EDAs with directed structure (e.g. Bayesian networks [12]) or undirected structure (e.g. Markov random fields [16]).

2.2 Walsh-Hadamard Transform

EDAs such as the DEUM algorithm [16] use a Markov random field (MRF) model based on Walsh coefficients. This is an appropriate representation because any fitness function on bit strings may be rewritten using the Walsh-Hadamard transform, and the non-zero Walsh coefficients are indicative of the structure of the function [6]. Under this transform, the function is expressed as a sum of Walsh coefficients (α_k) multiplied by Walsh functions ($W_k(x)$) as given by Eqn. 1, 2. Here, k iterates over all possible subsets of L.

$$f(x) = \sum_{k \subseteq L} \alpha_k W_k(x) \tag{1}$$

where

$$\begin{aligned} L &= \{0, \cdots, \ell - 1\} \\ W_\emptyset(x) &= 1 \\ W_k(x) &= \prod_{i \in k} \begin{cases} 1 & \text{if } x_i = 1 \\ -1 & \text{if } x_i = 0 \end{cases} \end{aligned} \tag{2}$$

A column vector of Walsh coefficients ($\vec{\alpha}$) may be computed using the Hadamard matrix as given by Eqn. 3. A faster method for Walsh-Hadamard transform is given by Fino and Algazi [5], however, the simple method was sufficient for the experiments within this paper.

$$\vec{\alpha} = \frac{1}{2^\ell} H_\ell \vec{f} \tag{3}$$

where H_ℓ is the Hadamard matrix of order-ℓ:

$$\begin{aligned} H_0 &= \begin{bmatrix} 1 \end{bmatrix} \\ H_1 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ H_\ell &= \begin{bmatrix} H_{\ell-1} & H_{\ell-1} \\ H_{\ell-1} & -H_{\ell-1} \end{bmatrix} = H_1 \otimes H_{\ell-1} \end{aligned} \tag{4}$$

and $\vec{f}$ is a reverse binary-ordered column vector of fitnesses:

$$\vec{f} = \begin{bmatrix} f(111\ldots1) \\ \vdots \\ f(100\ldots0) \\ f(000\ldots0) \end{bmatrix} \tag{5}$$

The Walsh-Hadamard transform produces a column vector of Walsh coefficients $\vec{\alpha}$ i.e.

$$\vec{\alpha} = \begin{bmatrix} \alpha_\emptyset \\ \alpha_{\{0\}} \\ \alpha_{\{\ \ 1\}} \\ \alpha_{\{0,1\}} \\ \alpha_{\{\ \ \ \ 2\}} \\ \alpha_{\{0,\ \ 2\}} \\ \alpha_{\{\ \ 1,2\}} \\ \alpha_{\{0,1,2\}} \\ \vdots \\ \alpha_{\{0,1,\ldots,\ell-1\}} \end{bmatrix} \tag{6}$$

To maintain the ordering of function values, the constant term $\alpha_\emptyset$ is never necessary, as it is an offset added to every output. Setting this term to zero does not affect the ranking or the separation between function values. This can be achieved by calculating the mean fitness and subtracting the mean from each output. We refer to this process as normalization. There are $2^\ell - 1$ possible elements of structure for an ℓ-bit function, e.g. 7 coefficients for a 3-bit problem. These are represented by Figure 1. Each structure element has been labelled with a power of two (shown below Figure 1 in hexadecimal) allowing all possible combinations to be uniquely enumerated as shown in Figure 2.

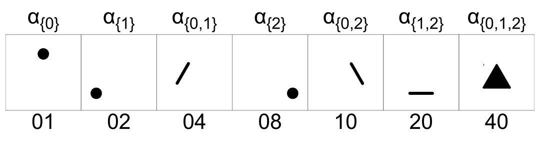

Figure 1: The parts of Walsh structure of a 3-bit function.

If we are interested in whether each coefficient is zero (not present in the structure) or non-zero (present in the structure), there are $2^{(2^\ell - 1)}$ possible structures (or 128 for a 3-bit function).

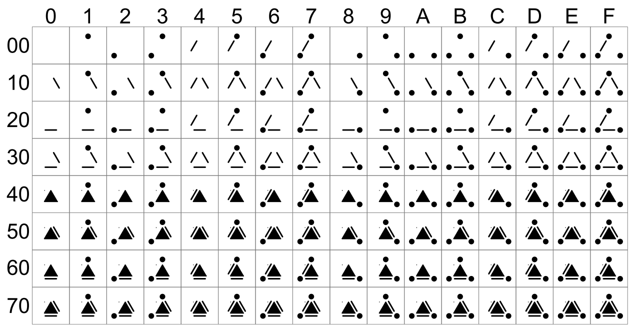

Figure 2: All 128 possible 3-bit Walsh structures after normalization.

2.3 Perturbation Methods

Like EDAs, perturbation methods (PMs) are a type of evolutionary algorithm which is able to overcome the problem of building-block disruption faced by genetic algorithms. In contrast to EDAs, PMs work by evaluating similar candidates with small changes (perturbations) and observe fitness difference $\Delta f_i(x)$:

$$\begin{aligned} \Delta f_i(x) &= f(x[i \to 1]) - f(x[i \to 0]) \quad (7) \\ x[i \to b] &= [x_0, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_{\ell-1}] \end{aligned}$$

The fitness difference is the change in fitness caused by changing the value of a variable – to detect whether two variables are independent or interdependent.

The standard criterion for detecting interdependent variables is a non-linearity check [11]. This is a Boolean expression denoted by $\mathcal{L}(i,j)$ (Eqn. 8), and is a symmetric relation (Eqn. 9).

$$\begin{aligned} \mathcal{L}(i,j) \Leftrightarrow &(\Delta f_j(x[i \to 1]) \quad (8) \\ &\neq \Delta f_j(x[i \to 0])) \\ \mathcal{L}(j,i) \Leftrightarrow &\mathcal{L}(i,j) \quad (9) \end{aligned}$$

A definition of variable linkage is non-linearity and non-monotonicity detection [11] given by Eqn. 10.

$$\begin{aligned} \mathcal{L}(i,j) &\Leftrightarrow \forall x (\neg P_{ij}(x) \vee M_{ij}(x)) \quad (10) \\ P_{ij}(x) &= \{\Delta f_i(x) > 0 \wedge \Delta f_j(x) > 0\} \\ M_{ij}(x) &= \{\Delta f_{ij}(x) > \Delta f_i(x) \\ &\quad \wedge \Delta f_{ij}(x) > \Delta f_j(x)\} \end{aligned}$$

The motivation behind non-monotonicity detection is that learning interactions, that only affect the magnitude of a variable's effect (and not the optimal setting), is not necessary when the objective is to locate a function optimum. This is therefore another way to define which structure is necessary.

Non-monotonicity detection checks that when a change in either variable increases fitness, the result of changing both also increases fitness. When using non-monotonicity detection there can be higher-order interactions, i.e. hierarchical interactions between groups [19].

2.4 Linkage Partition

Two variables, i and j, are considered to be in the same *linkage group* if they are interdependent, or transitively dependent through a chain of interdependent variables. In this sense, the linkage groups are connected components of an undirected graph formed by pairs of interdependent variables. A linkage group is a set of linked variables, with the kth linkage group is represented by $\gamma_k = \{X_i, X_j, \ldots\}$.

The *linkage partition* is the set of linkage groups in a problem, which are each non-overlapping: $\Gamma = \{\gamma_0, \gamma_1, \ldots . \gamma_{n-1}\}$. The variables in γ_k are arranged into a *substring* (s_k). A function is considered to be an additively separable function (ASF) [13] if the linkage partition consists of more than one linkage group and therefore may be rewritten as a sum of *subfunctions* on substrings as in Eqn. 11.

$$f(x) = g_0(s_0) + g_1(s_1) + \ldots + g_{n-1}(s_{n-1}) \quad (11)$$

Because the effect on fitness of the change of one variable is only affected by the values of the variables in the same linkage group, each linkage group may be optimized without regard to the other linkage groups. The global optimum may be therefore be found by exhaustive evaluation of each linkage group in $O(2^k)$ time where k is the maximum size of any one linkage group. This is because all 2^k combinations of variables in the same linkage group must be tried to conclusively identify the optimum value for any. An ASF with small linkage group can be optimized relatively efficiently if the linkage can be discovered efficiently. It has been shown by Streeter [17] that learning the linkage of an ASF adds only an $\ell \, ln(\ell)$ term, meaning an ASF can be optimized efficiently in $O(\ell \, ln(\ell) \, 2^k)$ time.

3. BINARY BENCHMARK FUNCTIONS

In this section we consider well-known benchmark functions of bit strings, and discuss the structure of the 3-bit instances.

3.1 Ones and BinVal

The *Ones* (or MaxOnes/OneMax) function is a commonly-cited univariate benchmark function which is the sum of the bits in the bit string. (Eqn. 12)

$$\text{Ones}(x) = \sum_{i=0}^{\ell-1} x_i \quad (12)$$

The *BinVal* function [4] is an exponential univariate function which sums the bits in the bit string, weighted according to their position in the string. (Eqn. 13)

$$\text{BinVal}(x) = \sum_{i=0}^{\ell-1} 2^i x_i \quad (13)$$

The Walsh coefficients of the 3-bit Ones and BinVal functions are given by Table 1. The BinVal function has increasing Walsh coefficients because it is an *exponential function*, whereas the Ones function has constant Walsh coefficients, though, both are univariate functions.

	Ones	BinVal
$\alpha_\emptyset$	1.5	3.5
$\alpha_{\{0\}}$	0.5	0.5
$\alpha_{\{1\}}$	0.5	1.0
$\alpha_{\{2\}}$	0.5	2.0

Table 1: Walsh coefficients of the 3-bit Ones and BinVal functions.

Both the Ones function and BinVal function are univariate, hence have no interactions between the variables, thus the linkage partition is $\Gamma = \{\{0\}, \{1\}, \{2\}\}$.

3.2 Checkerboard (1-D)

The *Checkerboard* (1-D) function (Eqn. 14) is a 1-dimensional variation on a checkerboard function, which rewards for assigning opposite values to adjacent alleles [2] [9].

$$Checkerboard\,(x) = \sum_{i=0}^{\ell-2} g\,(x_i, x_{i+1}) \tag{14}$$

$$\text{where } g\,(y,z) = \begin{cases} 0 & \text{if } y = z \\ 1 & \text{if } y \neq z \end{cases}$$

The Walsh coefficients of the 3-bit Checkerboard (1-D) function are given by Table 2.

	Checkerboard
$\alpha_{\emptyset}$	1.0
$\alpha_{\{0,1\}}$	-0.5
$\alpha_{\{1,2\}}$	-0.5

Table 2: Walsh coefficients of the 3-bit Checkerboard (1-D) function.

Despite the relative simplicity of the Checkerboard (1-D) function, there is a linear chain of each variable connecting to the next. Thus, this is not additively separable. The linkage partition is $\Gamma = \{\{0,1,2\}\}$.

4. RANK-EQUIVALENCE CLASSES

There are an infinite number of ℓ-bit functions, however, if we consider functions which rank solutions in the same way to be equivalent, there are a finite number of such equivalence classes of functions for a given ℓ.

We define the *rank* of a solution x, in the domain of function f, denoted by $R_f\,(x)$ as the number of solutions with a strictly lower fitness value, given by Eqn. 15. Listing the ranks for a function f gives the class C_f as in Eqn. 16. Two functions, f and g, are said to be *rank-equivalent* if $C_f = C_g$.

$$R_f\,(x) = |\{y : y \in f_{\text{domain}} \wedge f\,(y) < f\,(x)\}| \tag{15}$$

$$C_f = [R_f\,(1\ldots1), \ldots, R_f\,(1\ldots0), R_f\,(0\ldots0)] \tag{16}$$

The transpose of C_f gives a *canonical* example fitness vector (Eqn. 5) in class C_f. For the 3-bit functions $f : \{0,1\}^3 \to \mathbb{R}$, there are 545 835 distinct rank-equivalence classes, considering permutations as distinct. The number of classes for each number of distinct ranks is given by Table 3. For a given number of distinct ranks n and a given bit string length ℓ, the number of classes, $c\,(n,\ell)$ is:

$$c\,(n,\ell) = n^{\left(2^{\ell}\right)} - \sum_{k=1}^{n-1} c\,(k,\ell) \cdot {}_nC_k \tag{17}$$

Here, ${}_nC_k$ denotes the binomial coefficient, n choose k.

Num. Ranks (n)	Num. Deltas	Num. Classes (c)
1	0	1
2	1	254
3	2	5 796
4	3	40 824
5	4	126 000
6	5	191 520
7	6	141 120
8	7	40 320
	Total	545 835

Table 3: The number of rank-equivalence classes for each number of distinct ranks for 3-bit functions.

5. NECESSARY WALSH STRUCTURE

There are existing notions of *unnecessary* structure. Some parts of model structure have been called unnecessary if they are undesirable with the aim of minimizing model complexity [8] and maximizing mixing [18] or if the structure discovered by the probabilistic graphical model is surplus to accurate modeling of the fitness function [15]. In our work, we argue that structure is unnecessary when the same ordering of solutions could be maintained without it.

Take as an example, the class of the Ones function, $C_{\text{Ones}} = [7,4,4,1,4,1,1,0]$. An instance of C_{Ones} is specified by the function minimum ($f_{\min}$) and 3 positive *delta* values i.e. $\delta_0, \delta_1, \delta_2 > 0$ as shown in Table 4 and illustrated in Figure 3. The specific case of the Ones function is defined by $f_{\min} = 0$, $\delta_0 = 1$, $\delta_1 = 1$, and $\delta_2 = 1$.

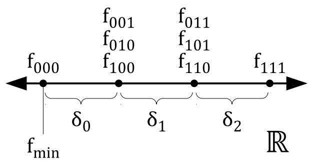

Figure 3: Illustration of the fitness values for the 3-bit class $C_{\text{Ones}} = [7,4,4,1,4,1,1,0]$ on the real number line, separated by delta values.

x	$f\,(x)$	Rank Expansion	$R_f\,(x)$
111	3	$f_{\min} + \delta_0 + \delta_1 + \delta_2$	7
011	2	$f_{\min} + \delta_0 + \delta_1$	4
101	2	$f_{\min} + \delta_0 + \delta_1$	4
001	1	$f_{\min} + \delta_0$	1
110	2	$f_{\min} + \delta_0 + \delta_1$	4
010	1	$f_{\min} + \delta_0$	1
100	1	$f_{\min} + \delta_0$	1
000	0	$f_{\min}$	0

Table 4: Fitness values, delta expansion, and ranks of the 3-bit ones function.

Performing the Walsh-Hadamard transform on the algebraic rank expansion (Table 4) with the arithmetic mean subtracted from each fitness gives a general, normalized set of Walsh coefficients in terms of delta expansion. In this class, they are given by Eqn. 18, 19, 20.

$$\alpha_{\{0\}} = \alpha_{\{1\}} = \alpha_{\{2\}} = \tfrac{1}{8}\left(\delta_0 + 2\delta_1 + \delta_2\right) \tag{18}$$

$$\alpha_{\{0,1\}} = \alpha_{\{0,2\}} = \alpha_{\{1,2\}} = \tfrac{1}{8}\left(\delta_2 - \delta_0\right) \tag{19}$$

$$\alpha_{\{0,1,2\}} = \tfrac{1}{8}\left(\delta_0 + \delta_2 - 2\delta_1\right) \tag{20}$$

As all δ are strictly positive, the univariate terms must be positive (hence, non-zero). The remaining possible structures are shown in Figure 4.

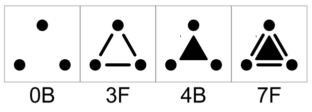

Figure 4: Minimal Walsh structure (0B) and all other possible Walsh structures of $C_{\text{ones}} = [7,4,4,1,4,1,1,0]$.

In the general case, structure 7F will be produced. On the condition $2\delta_1 = \delta_0 + \delta_2$, the trivariate term becomes zero, producing structure 3F. On the condition $\delta_0 = \delta_2$, the bivariate terms become zero, producing structure 4B. If both of the above conditions hold, then $\delta_0 = \delta_1 = \delta_2$, then the univariate structure 0B is produced, as is the case with the canonical Ones function. We conclude that this univariate structure, OB, is this minimum structure for this class.

The same analysis was repeated for the class $C_{\text{BinVal}} = [7, 4, 4, 1, 4, 1, 1, 0]$), which due to higher degrees of freedom, allows for a larger number of structures, 16, which we find to be the maximum possible for a 3-bit problem. The possible structures for C_{BinVal} are shown in Figure 5.

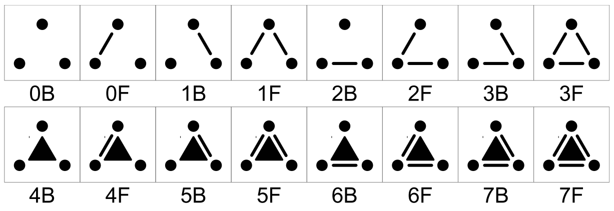

Figure 5: Minimal Walsh structure (0B) and all other possible Walsh structures of $C_{\textbf{BinVal}} = [7, 6, 5, 4, 3, 2, 1, 0]$.

For class $C_{Checkerboard} = [0, 2, 6, 2, 2, 6, 2, 0]$, we find that the bivariate terms between order-adjacent variables must be present, with an optional term between the start and end variables. No other parts of structure (univariate or trivariate terms) may be present in a function in this class. The possible structures for $C_{\text{Checkerboard}}$ are shown below in Figure 6.

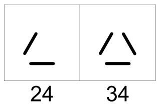

Figure 6: Minimal Walsh structure (24) and all other possible Walsh structures of $C_{\textbf{Checkerboard}} = [0, 2, 6, 2, 2, 6, 2, 0]$.

A complete analysis of all 545 835 classes of 3-bit functions was found by an automated process. By placing some restrictions on the types of functions considered, the Walsh-Hadamard transform could be computed of a well-defined large subset of functions for each class. These restrictions were to limit the image of the function to $y \in \mathbb{Z}, 0 \leq y \leq U$ and the deltas to $\delta \in \mathbb{Z}$.

The value chosen for U was 15. If a value for $U < 15$ was chosen, some structures for some classes were not detected. In the 3-bit case, a function may be constructed for any given Walsh structure where each $\alpha_I \in \{-1, 0, 1\}$. Using the expression $\vec{f} = H_3\vec{\alpha}$, the maximum value for any function value will be 7, and the minimum value will be -7 (a range of 15), therefore sampling from a set of $U = 15$ distinct integers is sufficient to generate any possible combination of negative, zero, and positive Walsh coefficients. It was confirmed experimentally, that for $U < 15$, some structures were not detected, and no additional structure were found for values $U > 15$ tried.

Table 5 shows the number of structures produced from each number of deltas. The result reveals that the Ones class has the maximum number of structures for any 3-delta class (4 structures) and that the maximum number of Walsh structures for any 3-bit class is 16. And we also see a positive correlation between number of deltas and the number of structures.

Num.	Num. Deltas							
Structs	0	1	2	3	4	5	6	7
1	1	254	2 744	6 720	4 032	-	-	-
2	-	-	3 052	16 072	23 856	10 080	-	-
3	-	-	-	11 760	30 240	21 504	8 064	-
4	-	-	-	6 272	41 664	53 760	14 784	2 688
5	-	-	-	-	12 096	41 664	34 944	5 376
6	-	-	-	-	8 064	24 864	24 192	9 408
7	-	-	-	-	5 376	18 816	16 128	2 688
8	-	-	-	-	672	15 456	25 536	8 064
9	-	-	-	-	-	-	-	-
10	-	-	-	-	-	2 688	6 720	5 376
11	-	-	-	-	-	-	-	-
12	-	-	-	-	-	2 688	9 408	4 032
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	1 344	2 688

Table 5: Number of structures produced from each number of deltas. Number of classes in each case is listed.

Analysis of the result shows that Walsh coefficients were not always simply necessary or unnecessary. For example, Figure 7 shows a case in which the trivariate structure $\alpha_{\{0,1,2\}}$, or one bivariate structure $\alpha_{\{0,2\}}$, or one univariate structure $\alpha_{\{0\}}$ may be considered unnecessary, but no more than one of those may be omitted. Another notable example is shown in Figure 8, where one bivariate structure $\alpha_{\{1,2\}}$ or the other two together (both $\alpha_{\{0,1\}}$ and $\alpha_{\{0,2\}}$) are required. This shows that from a rank-based perspective, some functions have a trade off between the presence of Walsh coefficients. Additionally, examples such as C_{ones} show that a symmetry is required to produce certain rank classes.

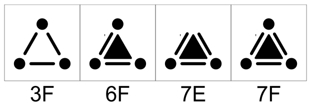

Figure 7: Three minimal Walsh structures (3F, 6F, and 7E) and the maximum structure (7F), the possible structures of the class $[5, 4, 2, 5, 5, 1, 0, 2]$.

6. DIRECTED ORDINAL LINKAGE

Linkage discovered by perturbation methods (PMs) is commonly regarded as bi-directional. i.e. $\mathcal{L}(j, i) \Leftrightarrow \mathcal{L}(i, j)$. We observe that using non-monotonicity detection perturbation, the dependence relationship between variables can in some cases be uni-directional, as illustrated in Figure 9. In this case, the optimum setting of X_0 can be inferred by sampling both values for X_0 at any point, whereas, identifying the optimal setting of X_1 depends on the value of X_0. By any previously-given definition of linkage (non-linearity or non-monotonicity), these variables are linked. We argue that this case of linkage places an ordering on the variables.

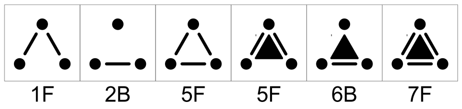

Figure 8: Two minimal Walsh structures (1F, and 2B) and four other possible structure of the class $[7, 4, 4, 2, 4, 2, 1, 0]$.

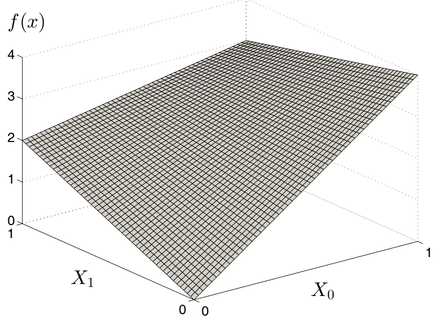

Figure 9: 2-bit fitness landscape illustrating unidirectional dependence $X_0 \to X_1$.

We provide a definition of *directed* linkage:

$$\mathcal{L}_O(i,j) \Leftrightarrow \exists\, x : sgn\left(\Delta f_j\left(x\left[i \to 1\right]\right)\right) \neq sgn\left(\Delta f_j\left(x\left[i \to 0\right]\right)\right) \tag{21}$$

The notation X_iX_j refers to interdependence:

$$X_iX_j \Leftrightarrow \mathcal{L}_O(i,j) \wedge \mathcal{L}_O(j,i) \tag{22}$$

The notation $X_i \to X_j$ refers to the unidirectional dependence j *depends on* i:

$$X_i \to X_j \Leftrightarrow \mathcal{L}_O(i,j) \wedge \neg\mathcal{L}_O(j,i) \tag{23}$$

The notation $X_i + X_j$ refers to independent variables:

$$X_i + X_j \Leftrightarrow \neg\mathcal{L}_O(i,j) \wedge \neg\mathcal{L}_O(j,i) \tag{24}$$

When referring to a general case without specifying the labelling of variables, the $\bullet$ operand may serve as a placeholder for variables, e.g. $\bullet\bullet \to \bullet$ may substitute for $X_0X_1 \to X_2$ but generalizes to any relabeling of the variables.

With this definition of directed linkage, a digraph with the same connected components as the undirected graph formed as a result of the conventional definition of linkage. Additionally, under our definition, each connected component may be regarded as one or more strongly-connected component (SCC).

Contracting each SCC to a single vertex produces a new directed acyclic graph (DAG) over the SCCs of the original graph. Assuming no higher-order dependencies, traversing this DAG with a topological ordering will give an order in which the SCCs of the original graph may be visited in which each SCC is optimized *after* its predecessors.

As with the interdependence relationships however, there can be higher-dimensional dependencies. We found experimentally that in the 3-bit case discussed in this paper, the higher-dimensional linkage can only exist where there is *no incoming edge* to a SCC of *two* vertices, thus, in these cases, it was checked whether the optimum setting of the two variables in the SCC depended on the third variable, and if so,

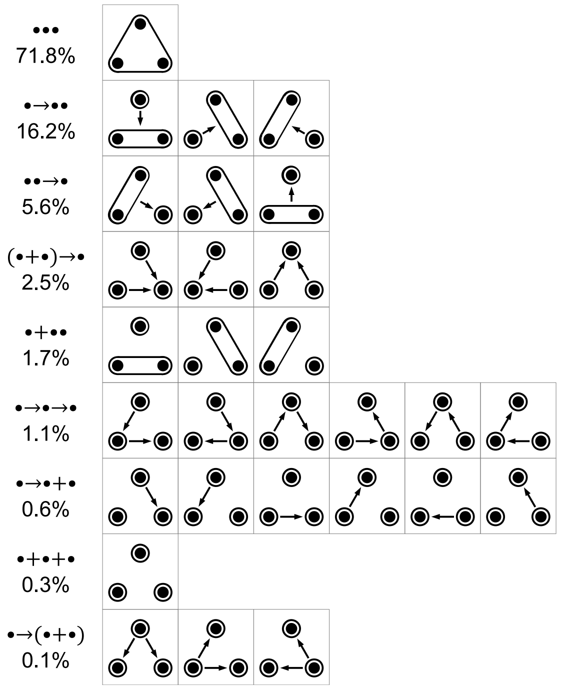

Figure 10: All 29 possible 3-bit directed linkage graphs with transitive linkages removed, grouped in to 9 cases according to relabeling invariance. The percentages show the proportion of rank-equivalence classes for different levels of structure complexity.

an incoming edge was added to correct the structure. Here, some classes in $\bullet\bullet \to \bullet$ are actually $\bullet\bullet\bullet$ and some classes in $\bullet + \bullet\bullet$ are actually $\bullet \to \bullet\bullet$.

Functions in the same rank-equivalence class as described in section 4 are indistinguishable by non-monotonicity detection PMs and hence, in contrast to the Walsh coefficients, each function in that class will have the same structure by this description.

With the removal of redundant transitive linkages (i.e. $X_0 \to X_2$ is redundant in the case that $X_0 \to X_1$ and $X_1 \to X_2$), there are 29 possible directed structures. These are shown in Figure 10, grouped into 9 cases according to invariance of relabeling of variables.

7. OPTIMIZATION SCHEDULES

Given the 9 cases of possible directed structures invariant under relabellings, these can be regarded as optimization schedules. There is a hierarchy of general to specific schedules. e.g. the most general schedule ••• regards all variables as interdependent and hence requires that the function be optimized by exhaustive evaluation, the most specific case $\bullet + \bullet + \bullet$ regard the variables as independent (univariate problem) to be optimized separately. If a problem is solvable by the more specific case, it is also solvable (though perhaps less efficiently) by the more general case. The tree in Figure 11 shows the relationship between the 9 cases, if a problem may be solved by an instance lower down the tree, it will also be solvable by all schedules reachable by following the edges of the tree. This ordering can be constructed by observing that to gain generality, an independence rela-

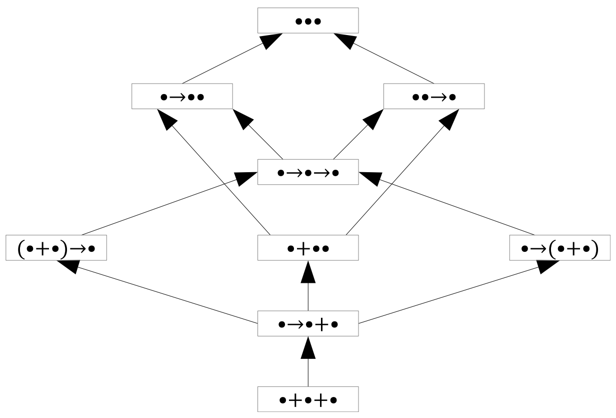

Figure 11: The relationship between the 9 ordinal linkage cases (see Figure 10). Any case may be solved processed as a case reachable by the edges in this graph.

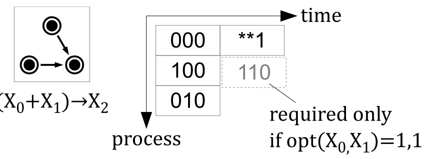

Figure 12: The 4 or 5 required function evaluations and 2 required time steps in the $(X_0 + X_1) \to X_2$ case. In the first time step, the optimum setting of X_0 and X_1 is determined, in the second time step, the optimum for the remaining variables is determined. This only requires evaluation of $f(1, 1, 0)$ if the optimum setting of the first two variables is both 1. The asterisks represent whatever optimum was determined in the previous time step.

tionship $\bullet + \bullet$ may be replaced by a dependence relationship $\bullet \to \bullet$, and a dependence relationship may be replaced with an interdependence relationship $\bullet\bullet$. The effect of these transformations is to *add* edges to the graph, allowing the schedule to be applicable to solve *more* cases.

The finesses which must be evaluated to determine a global optimum, given the structure of directed linkage is given by choosing one of the applicable schedules. The schedules differ in minimum number of function evaluations, ranging from 4 in the most specific case to 8 in the most general case.

Allowing for parallelization of evaluation, the schedules also differ in the minimum number of time steps required to achieve this minimum, ranging from 1 to 3, equaling the number of level sets in the contracted DAG. This is because the unidirectional linkage requires that the optimum for some variable(s) be determined before it is known which function evaluations will determine the optimum for some other variable(s).

As an example, Figure 12 shows the 4 or 5 required function evaluations and 2 required time steps in the $X_0 + X_1 \to X_2$ case. In the $**1$ function evaluation, the asterisks represent the optimum of the first two variables. The 5th functions evaluation, 110 is only required if the optimum setting of the first two variables is 11, otherwise the optimum of the third variable may be determined by comparing $**1$ to one of the first three evaluations. This is the only one of the 9 cases to have an evaluation which may or may not be required. Note that the function evaluations with asterisks are a result of uni-directional dependence enforcing an ordering of evaluation. After the first time step, an optimum of the first two variables may be substituted in the pattern to become 001, 101, 011, or 111, and only in the latter case, the optional 110 must be added on.

The necessary function evaluations are given in Figure 13 for one example labeling of each of the 9 cases. From this we see that the choice to move to another schedule may in some cases represent a trade-off between number of function evaluations and minimum number of time steps. This minimum depends on the number of evaluations which can be done in parallel.

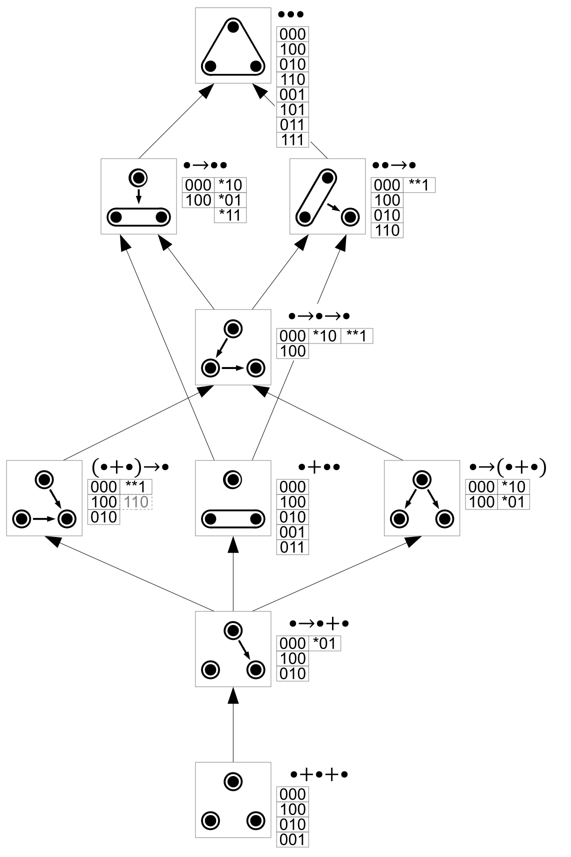

Figure 13: One example per ordinal linkage case (see Figure 10) with a linkage schedule (see Figure 12) which will locate a global optimum for the shown case. This figure is set out in the same arrangement as Figure 11

8. CONCLUSIONS

Many EDAs use linkage learning to determine which variables should be optimized together, but often construct models which use more information about linkage than is necessary. This leads to overly-complex models, which do not give an advantage over a simpler EDAs. This paper gives some theoretical insight in to the type of linkage which is strictly necessary from the point of view of a Markov network EDA or non-monotonicity-detection perturbation methods. Further, as the probability distribution modelled by *any* EDA could be represented as a Markov network, this view of unnecessary interactions may have implications for other EDAs.

In this paper we have given a method of classifying functions as being rank-equivalent, and given the number of instances in the 3-bit case for each of the possible number of different fitness levels. We have shown the minimal Walsh structure of the classes which the benchmark functions Ones, BinVal, and Checkerboard (1-D) belong to, and highlighted examples of equivalence classes which have a trade-off in necessary structure; and shown a positive correlation between number of deltas and the number of structures with the maximum at 16 for the 3-bit case. We have described how variables in non-monotonicity–detection perturbation methods can have uni-directional interactions, and enumerated all possible digraphs which may be formed in the 3-bit case. We have shown how these graphs dictate the minimum optimization schedules necessary to discover a global optimum for a given function and structure, and how the different possible optimization schedules are hierarchically related.

There are several directions for future work in this area. Modeling methods in EDAs may be adapted to look for minimal structures to avoid unnecessary computation. In bit string spaces, minimal Walsh structure may be estimated and the argument may be extended to other variable encoding. The adoption of perturbation methods for linkage detection allows consideration of choice of optimization schedule. The hierarchical classification on an unseen problem allows trade-offs to be made between computational effort / parallelizability and likelihood of optimality, which can be conditioned incrementally on evidence as the search progresses. Finally, the structure classification may be used to generate more representative benchmark problem sets.

9. REFERENCES

[1] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *ICML*, pages 38–46, 1995.

[2] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical report, DTIC Document, 1997.

[3] C.-Y. Chuang and Y.-p. Chen. Likage identification by perturbation and decision tree induction. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 357–363. IEEE, 2007.

[4] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1):51–81, 2002.

[5] B. J. Fino and V. R. Algazi. Unified matrix treatment of the fast (walsh-hadamard) transform. *Computers, IEEE Transactions on*, C-25(11):1142–1146, 1976.

[6] D. E. Goldberg. Genetic algorithms and Walsh functions: Part i, a gentle introduction. *Complex Systems*, 3(2):129–152, 1989.

[7] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5):493–530, 1989.

[8] M. Hauschild, M. Pelikan, C. F. Lima, and K. Sastry. Analyzing probabilistic models in hierarchical boa on traps and spin glasses. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 523–530. ACM, 2007.

[9] P. Larrañaga and J. A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer, 2002.

[10] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In *Proceedings of the 4th International Con-ference on Parallel Problem Solving from Nature, Lecture Notes In Computer Science*, volume 1141, pages 178–187, 1996.

[11] M. Munetomo and D. E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In *Proceedings of the genetic and evolutionary computation conference*, volume 1, pages 433–440, 1999.

[12] M. Pelikan. Bayesian optimization algorithm. In *Hierarchical Bayesian Optimization Algorithm*, pages 31–48. Springer, 2005.

[13] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary computation*, 8(3):311–340, 2000.

[14] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, 21(1):5–20, 2002.

[15] E. Radetic and M. Pelikan. Spurious dependencies and eda scalability. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 303–310. ACM, 2010.

[16] S. Shakya, A. Brownlee, J. McCall, F. Fournier, and G. Owusu. A fully multivariate deum algorithm. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 479–486. IEEE, 2009.

[17] M. J. Streeter. Upper bounds on the time and space complexity of optimizing additively separable functions. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 186–197. Springer, 2004.

[18] D. Thierens and P. A. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 617–624. ACM, 2011.

[19] M. Tsuji, M. Munetomo, and K. Akama. Metropolitan area network design using ga based on hierarchical linkage identification. In *Genetic and Evolutionary Computation–GECCO 2003*, pages 1616–1617. Springer, 2003.