Structural Coherence of Problem and Algorithm: An Analysis for EDAs on all 2-bit and 3-bit Problems

Alexander E.I. Brownlee
Division of Computing Science and Mathematics
University of Stirling
Stirling, UK
sbr@cs.stir.ac.uk

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

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

***Abstract*—Metaheuristics assume some kind of coherence between decision and objective spaces. Estimation of Distribution algorithms approach this by constructing an explicit probabilistic model of high fitness solutions, the structure of which is intended to reflect the structure of the problem. In this context, "structure" means the dependencies or interactions between problem variables in a probabilistic graphical model. There are many approaches to discovering these dependencies, and existing work has already shown that often these approaches discover "unnecessary" elements of structure - that is, elements which are not needed to correctly rank solutions. This work performs an exhaustive analysis of all 2 and 3 bit problems, grouped into classes based on mononotic invariance. It is shown in [1] that each class has a minimal Walsh structure that can be used to solve the problem. We compare the structure discovered by different structure learning approaches to the minimal Walsh structure for each class, with summaries of which interactions are (in)correctly identified. Our analysis reveals a large number of symmetries that may be used to simplify problem solving. We show that negative selection can result in improved coherence between discovered and necessary structure, and conclude with some directions for a general programme of study building on this work.**

I. Introduction

Metaheuristics in general are motivated by an ability to search the decision space of a problem to achieve a goal in the objective spaces. Various concepts of *structure* exist in the literature for attempting to describe aspects of algorithm behaviour on different problems. Structure can be viewed from the perspective of the problem and the algorithm. *Problem structure* means some information about elements of solutions or subsets of solutions knowledge of which is equivalent to knowing the fitness function and can be used in principal to determine the optimal solution. For example, in the Onemax problem, we know that solutions can be grouped into sets with the same number of ones in each solution and that the more ones there are the better the fitness. We can consider solutions that have similar fitness as being near to one another from the point of view of the problem structure. *Algorithm structure* means some knowledge about the way an algorithm explores a search space by transitioning from a current (population of) solution(s) to a successor (population of) solution(s). This will be determined by the operators an algorithm uses. For example a typical mutation operator induces a neighbourhood structure on a search space and the algorithm can use this to transition from solutions it is currently considering to neighbours of those solutions.

It is important that an appropriate coherence is achieved between the problem structure and the algorithm structure to move efficiently around the search space. Here, by *coherence* we mean that the algorithm structure is such that for small subsets of the search space, it is often possible to find better solutions than those currently under consideration by using the algorithm operators. That is to say that subsets which are nearby from the point of view of algorithm structure are also nearby from the point of view of problem structure. This is referred to as the proximal optimality principle [2]. The work in this paper offers insights into this concept that are important for researchers developing EDAs for new problems. The use of problem-specific models will ensure that the algorithm is well-matched to the problem and better able to solve it efficiently.

There is an increasing interest in approaches that explicitly make use of the problem structure to increase the search efficiency, for example [3]–[5]. In the case of Estimation of Distribution Algorithms (EDAs) [6]–[8] there is a particularly close relationship between problem structure and algorithm structure. The EDA makes a model of the distribution of fitness in the current population and samples that with a bias to producing the most fit solutions. The model is usually based on value combinations of the solution variables. Very often this is a probabilistic graphical model (PGM). In this situation it is also possible to make a PGM intrinsic to the problem in the sense that solution fitness is monotonic with solution probability. We call this the problem model. As these are both PGMs, when we refer to the structure of these models we are talking about the structure of the PGM (or associated hypergraph). The EDA will estimate the problem model, representing it in what we will call the algorithm model. Thus coherence here means the extent to which the algorithm model is concordant with the problem model. That is, if two solutions are ranked one way by the problem model, they should also be ranked that way by the algorithm model. This is supported by existing proof in [9] that the relationship between the model of an EDA and that of the problem strongly influence performance. Usually the algorithm model (at any iteration) will be a heavily factorised approximation of the problem model intended to generate solutions of higher value. For the purposes of optimisation, the algorithm model may not need all of the structure contained in the problem model.

An open question is: how well do current approaches to

978-1-4799-7492-4/15/$31.00

learning structure detect the "essential" structure? That is, the elements of problem structure that allow the algorithm to distinguish the relative ranking of solutions. For discrete problems there are two potential sources of error in learning structures: missing important dependencies and including erroneous (spurious) dependencies [10]. Also, effort spent in learning non-spurious but inessential structure is wasted and may be quite costly. Recent works including [11], [12] have explored how such errors influence the scalability and efficiency of EDAs, the latter showing that spurious dependencies decrease performance. A much earlier work with EDAs based on directed models [13] showed that dependencies generated by selection may genuinely reflect such a distribution or may simply be due to a sampling bias imposed by the selection. It is useful, then, to determine how well existing approaches to structure learning discover the essential structure of a problem. In [14], this issue was considered empirically. In this work we take a systematic approach.

In the special case of binary problems, the problem model can always be expressed using Walsh coefficients. The non-zero Walsh coefficients determine the problem model. Similarly the algorithm model, which assigns a non-zero probability to each solution, can also be expressed as a Walsh sum. We can therefore talk about the problem and algorithm structure in the same terms and compare them precisely. In particular we can use this to describe concepts of essential and non-essential structure. Essential problem structure is the minimal set of Walsh coefficients required to fully rank the solutions of the problem.

In previous work [1] we defined rank equivalence classes of monotonicity invariant functions as sets of all functions that rank a set of solutions identically. Restricted to problems with 2 and 3 binary variables, we are able to exhaustively evaluate all rank equivalence classes. Our intention is to provide theoretical insight into learning structure that may be built on for the more general cases of larger problems and those with alternative encodings, similar to the approaches taken by [9], [13]. Similar to this paper, the latter of these works considers classification of problems by ranking class, but in the context of local optima resulting from a neighbourhood system based on Hamming distance. The complementary work also provides some insight into the relationship between problem structure and EDA behaviour. For each rank equivalence class, we look at common approaches to structure learning in EDAs based on Chi-square independence tests using various selection operators. The bivariate interactions learned by these are compared with the interactions discovered by the Linkage Detection Algorithm [15] and the interactions present in the minimal Walsh structure.

We draw two major conclusions. Firstly, for some rank equivalence classes, poor choice of population size is likely to result in the detection of inessential structure, or failure to detect essential structure. Secondly, there are patterns within the rank equivalence classes that could be used to improve the efficiency of search. There are distinct groups of problems within rank equivalence classes, for which the probability of detecting an interaction is equal. These can be mapped on to each other by relabelling solution variables. Furthermore, there are groups of problems for which the probability of detecting an interaction can be increased by using negative selection.

The rest of the paper is structured as follows. Section II introduces the concept of Walsh structures and minimal structures. Section III discusses several approaches to structure learning in the literature. Sections IV and V give the results of our study on 2 and 3 bit problem classes, with our analysis. Finally in VI we draw our conclusions and suggest the direction for future work.

II. Walsh Structures

For problems with a bit string encoding, $\Omega = \{0,1\}^n$ is the search space. $f(x)$ is the fitness function and $X = (X_1, \ldots, X_n)$ is the variable vector. $X_i = x_i$ denotes the event that variable X_i takes value x_i, and $x = x_1 \ldots x_n$ denotes a joint configuration of X. Any fitness function on bit strings can be rewritten using the Walsh-Hadamard transform, and the non-zero Walsh coefficients represent the problem structure [16]. As in [5], the full set of Walsh functions up to order-n (order-3 in the present case), are considered. Under this transform, the $f(x)$ is expressed as a sum of Walsh coefficients α_k multiplied by Walsh functions $W_k(x)$ as shown in Equations 1 to 3. k iterates over all possible subsets of X.

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

where

$$W_{\varnothing}(x) = 1 \quad \forall\, x \tag{2}$$

$$W_k(x) = \prod_{i \in k} X_i \begin{cases} 1 & \text{if} \quad x_i = 1 \\ -1 & \text{if} \quad x_i = 0 \end{cases} \tag{3}$$

In [1] it was shown that a column vector of Walsh coefficients could be computed using the Hadamard matrix. The constant term $\alpha_{\varnothing}$ is never necessary to maintain the ordering of function values as it is added to every output. For many problems, there are multiple problem structures that apply the same ranking over all solutions. For a given problem, those structures with the lowest number of non-zero Walsh coefficients are referred to as the *minimal structures*. We argue that minimal problem structure must be detected for efficient solution of the problem, and that detection of only the minimal structure further increases efficiency. It has already been established that spurious dependencies can decrease efficiency [11], [12] and that omission of parts of problem structure from the algorithm structure leads to reduced ability to rank solutions [5]. In EDAs, reduced structure means fewer model parameters to estimate, leading to reduced model-building effort.

The Walsh coefficients for 2-bit problem classes are $\alpha_{\{\varnothing\}}$, $\alpha_{\{0\}}$, $\alpha_{\{1\}}$ and $\alpha_{\{01\}}$. Ignoring the constant term, there are three coefficients, one for each of the variables and one for the interaction between the variables. Figure 1 shows the 8 possible combinations of these components.

A problem class is defined simply by listing the ranks of the solutions. We adopt the convention used in [1] and write for the function class of f: $C_f = [r_1, r_2, ..., r_{2^n}]$ where r_i

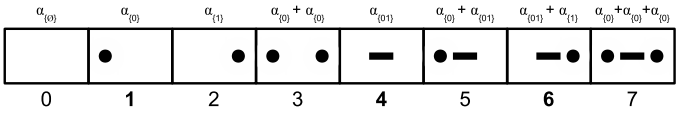

Fig. 1. Walsh structural components for 2-bit problems

is the rank of the ith solution. (The solutions are ordered as binary integers.)

A. Benchmarks

In our analysis, we also refer to the classes which relate to two well-known benchmark functions. We now describe these.

Onemax is one of the simplest and most commonly used univariate benchmarks. The fitness is simply the number of bits with the value 1, expressed formally in (4).

$$f(x) = \sum_{i=1}^{n} x_i \tag{4}$$

For 3 bits, the equivalent problem class is $C_{OneMax} = [0, 1, 1, 2, 1, 2, 2, 3]$.

BinVal [17] is also a univariate function, where the fitness is the sum of bits in the string, each weighted by their position (treating the bit string as a binary value). Formally:

$$f(x) = \sum_{i=1}^{n} 2^i x_i \tag{5}$$

For 3 bits, the equivalent problem class is $C_{BinVal} = [0, 1, 3, 4, 5, 6, 7]$.

The minimal Walsh structures for Onemax and BinVal with 2 and 3 bits (actually extending to n bits) have no interactions.

III. Structure learning approaches

There are several approaches taken to learning structure for undirected networks in EDAs. Typically EDAs learn structure by sampling the population. A common approach for learning undirected probabilistic graphical models [14], [18]–[23] is that a selection operator chooses a subset of solutions and statistical independence tests such as joint entropy, mutual information or Chi-square [24] are used to test the significance of interactions between variables among the selected set. This will find the bivariate interactions: a deterministic clique-finding algorithm such as Bron and Kerbosch [25] can then locate maximal cliques representing higher order interactions. Common selection operators for this purpose are tournament selection and truncation selection, detailed further below. Poor configuration of the selection operator can lead to missing key interactions, or falsely detecting unnecessary ones, which both impact on performance [11], [12], [14].

In this paper, the statistical independence test used is Chi-square. This is a comparison between the joint distribution of a pair of variables and the product of their marginal distributions. The test may also be extended to test higher order interactions by testing for conditional probabilities, but that is not covered in this study. For a selected population, summed over all possible values for x_i and x_j in a population of size S:

$$\chi^2_{i,j} = \sum_{x_i, x_j \in 0,1} \frac{(Sp(x_i, x_j) - Sp(x_i)p(x_j))^2}{Sp(x_i)p(x_j)} \tag{6}$$

where $p(x_i, x_j)$ is probability that x_i and x_j take a particular combination of values together (the observed value) and $p(x_i)$ and $p(x_j)$ is the probability that x_i and x_j take those values independently (making $p(x_i)p(x_j)$ the expected value). With a value of 3.84, the variables are said to be 95% independent [26]. Rearranging this equation for S allows us to determine the minimum population size S_{min} which would result in a dependency being detected was calculated (that is, the population size for which the Chi-Square score would be over 3.84). $S_{min} = \infty$ means that the variables will always be determined to be completely independent.

An alternative method of structure learning is linkage detection via probing or perturbation, used by the Linkage Detection Algorithm [15]. This was specifically designed to detect the structure components corresponding to non-zero Walsh coefficients. In the 2 bit case, starting with solution 00, the bits are flipped to 1 individually, then together. If the change in fitness due to flipping them together is different to the total change observed when flipping them individually, then a dependency is regarded to be present. Selection is not required for LDA.

LDA can be expanded to higher levels of interaction but its complexity grows rapidly with the order of interaction being tested. In fact, the number of possible interactions of order k in a set of n variables is given in (7).

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!} \tag{7}$$

A. Selection operators

The selection operators considered for combination with Chi-square in this study are variants of tournament and truncation selection.

Truncation selection is often used in EDAs. The experiments covered three variants of the operator, originally described in [27]. In the unaltered form of truncation selection [28], the population is sorted by descending fitness rank, and the highest $p * N$ ranking solutions are selected, where $0 < p \leq 1$ and N is the population size. A solution is selected once only, and the lowest-ranking individuals cannot be selected. The original truncation selection is referred to here as *top selection*, as it selects the top ranking solutions.

Several studies [27], [29]–[33] have shown that including low-fitness solutions can be beneficial to model building and optimisation. Further to this, [14] found that selecting the lowest-ranking solutions resulted in discovery of a similar number of false interactions as selecting the same proportion of top-ranking solutions. To explore this further, we include two variants of truncation selection that explicitly include low-ranking solutions: *bottom* selection and *top+bottom selection*. Bottom selection replicates top selection, but selects the lowest

$p * N$ ranking members of the population. Top+bottom selection selects the highest $(p/2) * N$ and lowest $(p/2) * N$ ranking solutions.

Tournament selection (with replacement) takes s solutions chosen at random from the population, and selects the one with the highest fitness. It is often used by evolutionary algorithms as it is quite simple but also allows explicit control on selection pressure to be made by choice of the tournament size s. Furthermore, with tournament selection it is possible that very low-ranking solutions will be selected. We also consider an inverse tournament selection, which always chooses the solution with the lowest fitness in the tournament.

Both operators are well-defined and it is possible to predict their behaviour in a perfect system. So, given a uniformly distributed initial population, we can determine the proportions of each solution present in the selected set. A similar approach for analysis of directed probabilistic graphical models was taken in [13]. In the 2-bit case, for both selection operators, we assume that each of the four solutions comprises exactly 25% of the initial population. For tournament selection, we look at all possible tournaments (16 for 2 bits, 64 for 3 bits), determining the winner of each. The winners are combined into a pool, representing the selected set of size S. For a class where the 4 solutions are equally ranked, the selected set will be comprised of 25% each of the 4 solutions. For a class where three solutions are equal, but one is higher ranked, then the higher ranked solution will be 44% of the selected set, and the others 19% each. For truncation selection, the fraction of the population to be selected is an algorithm parameter. When selecting the top 25% of the population: if all solutions are equally ranked the selected set will be divided equally between each of the solutions. In the 2-bit case, if one solution is ranked above all the others, it will take 100% of the selected set.

IV. 2 Bit Classes

We now consider the relationship between the minimal Walsh structure and the structure obtained by the structure learning approaches discussed in the previous section. We begin with the minimal case for bit strings showing possible dependency between variables: 2-bit classes. There are an infinite number of such functions, but if we consider only the rank of solutions (the goal of optimisation being to move towards highly-ranked solutions), there are a finite number of possible rankings. Different to the notation in [1], we define the rank of a solution x, for a function f, as the number of distinct fitness values in the population lower than $f(x)$. In [1], there was some consideration of the separations (there referred to as *deltas*) between the ranks. In the case that the separations between all ranks were ranks equal, the minimal structure was discovered. Here, we use the ranks as the raw fitness values for the probes in LDA. If the ranks are equally spaced, LDA discovers the minimal structure for all problem classes. Otherwise, it discovers one of the other structures. The notation reflects this equal spacing of the ranks.

For classes with 2 bits $f : \{0,1\}^2 \rightarrow \mathbb{R}$, there are four possible solutions 00, 01, 10, 11. There are 75 distinct rank equivalence classes, considering permutations to be distinct. This is small enough that an exhaustive evaluation of all rank-equivalence classes C for 2-bit classes can be performed.

TABLE I. Categories of rank-equivalence class

Category	Description	Ranks	Num. ranks	Num. classes
1	All solutions equal rank	0,0,0,0	1	1
2	3 equal rank, 1 higher	0,0,0,1	2	4
3	3 equal rank, 1 lower	0,1,1,1	2	4
4	2 pairs equally-ranked	0,0,1,1	2	6
5	1 pair equal, 2 distinct and higher	0,0,1,2	3	12
6	1 pair equal, 1 higher, 1 lower	0,1,1,2	3	12
7	1 pair equal, 2 distinct and lower	0,1,2,2	3	12
8	All solutions distinctly ranked	0,1,2,3	4	24

These can be grouped into the categories in Table I. The categories reflect the number of ranks, with the number of solutions at each rank. A textual description of each category is also given, with an list of the ranks shared by all classes in the category, sorted ascending.

The purpose of this experiment is to determine the likelihood that the various structure learning approaches will detect the essential structure for the problem. This is measured in terms of the selected population size required for the structure learning approaches to detect an interaction.

Cat: The category number from Table I.
C: A unique number to identify the rank-equivalence class.
Ranks: The order in which the solutions 00, 01, 10, 11 are ranked. Assuming maximisation, higher values indicate higher ranking. So 0021 indicates that 10 is ranked highest, followed by 11, and 00, 01 are the joint-lowest ranked solutions.
Structure: The Walsh structures that fully determine ranking for this problem class (left-most is the minimal Walsh structure), in the same format as Figure 1.
Tournament: The minimum selected population size required to make the interaction significant ($\chi^2 = 3.84$) when using two variants of tournament selection, in which the higher-ranked (Best) solution is always selected and in which the lower-ranked (Worst) solution is always selected.
Top: The minimum selected population size required to make the interaction significant ($\chi^2 = 3.84$) when using top truncation selection of the highest-ranked 0.25, 0.33 and 0.5 of the population.
Bottom: As for {Top}, but for bottom truncation selection where the lowest-rankest solutions are selected from the population.
T+B: As for {Top}, but for top+bottom truncation selection where the $n/2$ lowest and highest-ranked solutions are selected.
LDA: The Linkage Detection Algorithm finds interaction when starting with solution 00: (Y)es / (N)o.

There are several things to note from these results. In terms of the selection operators, truncation selection can detect structure but because of the homogenous nature of the selected populations it often fails to detect an interaction where the other approaches succeed. Essentially detection of the interaction is all-or-nothing above fairly small selected population sizes. The selected set does not have enough variation, so the range of Chi-square values seen with tournament selection is not repeated. Top and bottom selections work equally well over all classes, but within groups 2 and 3 the results for top selection are the complement of those for bottom selection.

TABLE II. RESULTS FOR ALL 75 2-BIT RANK EQUIVALENT CLASSES. COLUMN HEADINGS ARE DESCRIBED IN THE TEXT.

Cat	C	Ranks	Structure	Tournament		Top			Bottom			T+B			LDA
				Best	Worst	0.25	0.33	0.5	0.25	0.33	0.5	0.25	0.33	0.5	
1	0	0000		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
2	1	0001		96	35	∞	61	∞	15	15	15	15	15	15	Y
	2	0010		96	35	∞	61	∞	15	15	15	15	15	15	Y
	3	0100		96	35	∞	61	∞	15	15	15	15	15	15	Y
	4	1000		96	35	∞	61	∞	15	15	15	15	15	15	Y
3	5	1110		35	96	15	15	15	∞	61	∞	15	15	15	Y
	6	1101		35	96	15	15	15	∞	61	∞	15	15	15	Y
	7	1011		35	96	15	15	15	∞	61	∞	15	15	15	Y
	8	0111		35	96	15	15	15	∞	61	∞	15	15	15	Y
4	9	0011		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	10	0101		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	11	1001		15	15	4	4	4	4	4	4	4	4	4	Y
	12	0110		15	15	4	4	4	4	4	4	4	4	4	Y
	13	1010		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	14	1100		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
5	15	0012		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	16	0021		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	17	0201		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	18	2001		16	14	∞	4	4	4	4	4	4	4	4	Y
	19	0102		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	20	0120		16	14	∞	4	4	4	4	4	4	4	4	Y
	21	0210		16	14	∞	4	4	4	4	4	4	4	4	Y
	22	2010		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	23	1002		16	14	∞	4	4	4	4	4	4	4	4	Y
	24	1020		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	25	1200		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	26	2100		726	81	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
6	27	1102		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	28	1120		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	29	1210		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	30	2110		143	143	∞	61	15	∞	61	15	∞	∞	∞	N
	31	1012		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	32	1021		143	143	∞	61	15	∞	61	15	∞	∞	∞	N
	33	1201		143	143	∞	61	15	∞	61	15	∞	∞	∞	N
	34	2101		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	35	0112		143	143	∞	61	15	∞	61	15	∞	∞	∞	N
	36	0121		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	37	0211		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
	38	2011		23	23	∞	10	15	∞	10	15	∞	∞	∞	Y
7	39	2201		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	40	2210		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	41	2120		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	42	1220		14	16	4	4	4	∞	4	4	4	4	4	Y
	43	2021		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	44	2012		14	16	4	4	4	∞	4	4	4	4	4	Y
	45	2102		14	16	4	4	4	∞	4	4	4	4	4	Y
	46	1202		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	47	0221		14	16	4	4	4	∞	4	4	4	4	4	Y
	48	0212		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	49	0122		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	50	1022		81	726	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
8	51	0123		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	52	0132		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	53	0213		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	54	0231		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	55	0321		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	56	0312		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	57	1023		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	58	1032		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	59	1203		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	60	1230		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	61	1320		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	62	1302		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	63	2103		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	64	2130		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	65	2013		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	66	2031		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	67	2301		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	68	2310		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	69	3120		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	70	3102		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y
	71	3210		173	173	∞	∞	∞	∞	∞	∞	∞	∞	∞	N
	72	3201		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	73	3021		46	46	∞	∞	∞	∞	∞	∞	∞	∞	∞	Y
	74	3012		14	14	∞	4	4	∞	4	4	∞	∞	∞	Y

This goes some way towards explaining results from [14] that shows top selection and bottom selection discovering similar numbers of unnecessary interactions. The implication is that for some classes, the use of negative selection is more likely to discover essential structure than conventional selection.

The tournament selections also work equally well over all classes. The results for the two operators are the complement of each other within groups 2 and 3, and groups 5 and 7. With a selected population size of 100 (within the common range of population sizes in EAs), in almost all cases (in)dependence of the variables is correctly detected. With too large a selected population size (or too small a threshold) interactions will be detected for classes 30,32,33,35 (all of which are relabellings of Onemax) and classes 51,53,58,62,66,67,69,71 (all of which are BinVal variants). The minimal Walsh structure for each of these classes excludes the interaction term, although the full Walsh structure (structure 7) is also applicable for these classes. In this sense, the detection of an interaction is *correct*, but is *unnecessary*. The operators fail to detect many interactions in category 5 and 7 unless the selected population size exceeds 726, though for each class the interaction is always detected by using one of tournament or inverse tournament selection.

LDA finds structures that perfectly match the Walsh decomposition. This ought to be expected because LDA is specifically designed to determine the non-zero Walsh coefficients [15], [34]. LDA was shown to perfectly discover bivariate structure for the Ising problem in [23].

All structure learning approaches agreed that there was no interaction in 5 classes: 0,9,10,13,14. The Walsh structure for these problem classes always excludes the interaction term, or more accurately the coefficient for this term is zero. For other classes, the Walsh structure 7, all possible terms being non-zero, applied to the class in addition to any minimal structure.

The reader might ask whether the given Walsh problem structures are truly minimal. For class 1, if the x_i can take values $\{0, 1\}$ rather than the $W_i(x)$ values $\{-1, 1\}$, then this function could be $f(x) = x_0x_1$. Alternatively, we could say that we simply need to know that the rank is determined by knowing that: $x_0 == x_1$ and $x_0 == 1$, or that $x_0 == x_1$ and $x_1 == 1$ or that $x_0 == 1$ and $x_1 == 1$. So in a broad sense an optimisation algorithm which treats the variables in this way will need only 1 or 2 terms to model this ranking (although it will need more terms on other problems that have a simpler minimal Walsh structure). In practice, this relies on making terms evaluate to 0 by allowing $x_i = 0$ instead of relying only on the Walsh coefficients being 0 to eliminate terms from the structure. This is simply hiding necessary structure rather than reducing it. This applies to at least classes 1-8, 15-26 and 39-50.

It is also worth noting the symmetries present in the results. Categories 2 and 3 are the same, but one is $-f(x)$ of the other. The same applies to categories 5 and 7. Likewise, there are matching pairs within category 8. However, the Chi-square values and minimum selected population sizes are different. This is corrected by selecting the low-ranked solutions (this is the same as using conventional selection with $-f(x)$). Furthermore, within each category there are equivalent classes which simply have the variables relabelled. For each set of equivalent classes the Chi-square values are the same.

V. 3 Bit Classes

We now move on to classes with 3 bits $f : \{0, 1\}^3 \rightarrow \mathbb{R}$, for which there are eight possible solutions 000, 001, 010, 011, 100, 101, 110, 111 and 545 835 distinct rank equivalence classes. [1] presents an analysis of the exhaustive set of rank equivalence classes for 3 bits. Here we consider the same set, comparing the minimal Walsh structures discovered with those found by the same approaches as in Section IV. For obvious reasons of space, only summary statistics are presented here.

For classes with 3 bits there are 8 components that might contribute to the rank ordering. Here we are interested in those related to interactions between variables: α_{01}, α_{02}, α_{12} and α_{012}. For each class, the structure learning approaches are deployed and the resulting structure compared with the minimal Walsh structures. The fraction (over all classes) of dependencies/independencies correctly/incorrectly discovered for each bivariate interaction are then reported. As noted in [1], there may be multiple minimal structures for 3-bit classes. Here the comparisons are made with all minimal Walsh structures for a given class and if the discovered structure perfectly matches any that result is used. LDA is run with two starting solutions: 000 and 111. The dependencies discovered by both are combined (without doing this, interactions are often missed).

Tables III and IV summarise the results over all 3-bit problem classes, with selected population sizes of 100 and 500 respectively. The specific population sizes chosen are unimportant except to illustrate that for most problem classes there is a critical population size to find dependencies that is neither zero nor infinite. Consequently, choice of population size is important with respect to detecting problem structure (this is already known) and more importantly, the critical size is influenced by the selection operator. The first column shows the structure learning approach used, subsequent columns show dependencies correctly (D corr) and incorrectly (D incorr) identified, and independencies correctly (I corr) and incorrectly (I incorr) identified. Each structure learning approach has three rows, for the dependencies a_{01}, a_{02} and a_{12} respectively.

In the raw data, the same patterns are observed as with 2-bit classes. The symmetries in selected population size required to discover dependencies within groups of classes remain. Both tournament selection operators perform equally well, and with a selected population size of 100 are more likely to miss interactions rather than introduce false ones. As population size increases, more essential interactions are found, but along with more inessential interactions. With tournament selection, a small number of classes require an infinite selected population size to discover an interaction: for most classes it is a few hundred. This again implies that increasing the selected population size (or decreasing the threshold) will result in the discovery of further interactions, regardless of the minimal Walsh structure. The structures learned using truncation selection again suffer from lack of diversity in the selected solutions, in that the threshold to discover an interaction is either a very small population or infinite. This means that changing from a selected population size of 100 to 500 has no impact on the rates of correctly / incorrectly

TABLE III. RESULTS FOR ALL 3-BIT RANK EQUIVALENT CLASSES. SELECTION OPERATOR WITH THE FRACTION OF EACH DEPENDENCY CORRECTLY OR INCORRECTLY DISCOVERED, WHEN USING A SELECTED POPULATION SIZE OF 100. ROWS ARE IN GROUPS OF 3, FOR THE INTERACTIONS 01,02 AND 12.

Selection / Algorithm	D incorr	D corr	I incorr	I corr
Tournament	0.53	0.42	0.00	0.05
	0.53	0.42	0.00	0.05
	0.52	0.42	0.00	0.05
Tournament Inv	0.53	0.42	0.00	0.05
	0.53	0.42	0.00	0.05
	0.52	0.42	0.00	0.05
Top, p0.25	0.56	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.55	0.39	0.01	0.04
Top, p0.33	0.26	0.70	0.03	0.02
	0.26	0.69	0.03	0.02
	0.25	0.69	0.03	0.02
Top, p0.5	0.28	0.67	0.03	0.02
	0.28	0.67	0.03	0.02
	0.28	0.67	0.03	0.02
Bottom, p0.25	0.56	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.55	0.39	0.01	0.04
Bottom, p0.33	0.26	0.70	0.03	0.02
	0.26	0.69	0.03	0.02
	0.25	0.69	0.03	0.02
Bottom, p0.5	0.28	0.67	0.03	0.02
	0.28	0.67	0.03	0.02
	0.28	0.67	0.03	0.02
T+B, p0.25	0.84	0.11	0.00	0.04
	0.84	0.11	0.00	0.05
	0.83	0.11	0.00	0.05
T+B, p0.33	0.60	0.36	0.01	0.04
	0.59	0.36	0.01	0.04
	0.59	0.36	0.01	0.04
T+B, p0.5	0.56	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.55	0.39	0.01	0.04
LDA	0.00	0.95	0.00	0.05
	0.00	0.95	0.00	0.05
	0.00	0.95	0.00	0.05

TABLE IV. RESULTS FOR ALL 3-BIT RANK EQUIVALENT CLASSES. SELECTION OPERATOR WITH THE FRACTION OF EACH DEPENDENCY CORRECTLY OR INCORRECTLY DISCOVERED, WHEN USING A SELECTED POPULATION SIZE OF 500. ROWS ARE IN GROUPS OF 3, FOR THE INTERACTIONS 01,02 AND 12.

Selection / Algorithm	D incorr	D corr	I incorr	I corr
Tournament	0.24	0.71	0.02	0.03
	0.24	0.71	0.02	0.03
	0.24	0.71	0.02	0.04
Tournament Inv	0.24	0.71	0.02	0.03
	0.24	0.71	0.02	0.03
	0.24	0.71	0.02	0.04
Top, p0.25	0.55	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.54	0.40	0.01	0.04
Top, p0.33	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
Top, p0.5	0.24	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
Bottom, p0.25	0.55	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.54	0.40	0.01	0.04
Bottom, p0.33	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
Bottom, p0.5	0.24	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
	0.23	0.72	0.03	0.02
T+B, p0.25	0.84	0.11	0.00	0.04
	0.84	0.11	0.00	0.05
	0.83	0.11	0.00	0.05
T+B, p0.33	0.56	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.55	0.40	0.01	0.04
T+B, p0.5	0.55	0.40	0.01	0.03
	0.55	0.40	0.01	0.04
	0.54	0.40	0.01	0.04
LDA	0.00	0.95	0.00	0.05
	0.00	0.95	0.00	0.05
	0.00	0.95	0.00	0.05

discovered structure. For both selected population sizes, T+B discovers more interactions correctly than top selection or bottom selection, and higher selection pressure also improves success. This mirrors the results in [14]. The important point to note here is that the different selection methods lead to more or less precise discovery of structure, in patterns that can be directly related to the problem, rather than the absolute population sizes involved. Some discussion of population size and structure discovery can be found in [14], [35].

The success of LDA depends on the starting solution, but given two starting solutions (covering both values of the bit not included in the interaction being tested) it perfectly discovers one of the minimal Walsh structures. Population size is irrelevant to LDA so the values are the same in both tables.

We now look at two specific classes that showed interesting properties with 2 bits to draw some further conclusions. For 2 bit classes, the class for Onemax was ID 35 and for BinVal was 51. These and the relabellings of these classes showed similar properties. In particular, the selected population size required to find an interaction using tournament selection was non-zero and less than infinity. Within the group of classes and their relabellings, the required selected population size was the same. The minimal Walsh structures for both classes have no interactions, so this means that with too high a selected population size or too low a Chi square threshold an unnecessary interaction will be discovered.

The selected population sizes to discover an interaction with Onemax were 327.85 for both tournament selections for all interactions a_{01}, a_{02} and a_{12}. For BinVal, these figures were 172.8, 725.76 and 3628.8 for interaction a_{01}, a_{02} and a_{12} respectively: these values were reordered depending on the specific labelling of variables made by the class. This appears to be an artefact of the different weights applied to the variables in determining the ranks. Variables 1 and 2 have a lower impact on the rank, so have a lower probability of an interaction being discovered between them.

VI. CONCLUSION

In this paper we have presented an initial study on the relationship between structures discovered by common structure learning techniques and the minimal Walsh structure for all 2 and 3 bit rank equivalence classes. Existing structure learning techniques work because they do detect minimal problem structures, in the main. However, the results here show that in some cases, even assuming no bias introduced by sampling errors or population convergence, unnecessary interactions will be detected. We have shown that there are a large number of symmetries across problem classes that may be used to simplify problem solving. Negative selection can result in improved coherence between discovered and necessary structure. This is in line with a number of other similar results for both structure learning and more general improvements to metaheuristic search [27], [29]–[33]. The

Linkage Detection Algorithm was confirmed to discover minimal Walsh structures perfectly, but with rapidly increasing cost essentially amounting to exhaustive probing over the search space to achieve such perfection.

We intend for this work to support a more general study of the relationship between the structure of problems and search algorithms. The ultimate goal is classification of search algorithms and problems by their structures, in a way that will allow us to predict performance and more efficiently explore search spaces. This will provide a very useful theoretical foundation for researchers developing search algorithms such as EDAs for new problems. The next step is clearly to extend to the more general cases of n-bit problems, and non-binary spaces. Further work will investigate structural composition and decomposition of problems aimed at understanding when complex structure can be decomposed into simpler minimal structures and under what conditions complexity is essential. We suggest that it is important that an algorithm is well-matched to the structure of the problem being solved: the algorithm structure should be *coherent* with the problem structure. This study has shown that, for bit strings, while existing approaches to discovering structure perform reasonably well they are likely to discover some parts of problem structure incorrectly, motivating further research in this area.

REFERENCES

[1] L. A. Christie, J. A. McCall, and D. P. Lonie, "Minimal walsh structure and ordinal linkage of monotonicity-invariant function classes on bit strings," *Proc. GECCO*, pp. 333–340, 2014.

[2] F. Glover and F. Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997, chapter 5: Principles, pp. 125–151.

[3] F. Chicano, D. Whitley, and A. M. Sutton, "Efficient identification of improving moves in a ball for pseudo-boolean problems," in *Proc. GECCO*. New York, NY, USA: ACM, 2014, pp. 437–444.

[4] D. Whitley and W. Chen, "Constant time steepest descent local search with lookahead for nk-landscapes and max-ksat," in *Proc. GECCO*. New York, NY, USA: ACM, 2012, pp. 1357–1364.

[5] A. Brownlee, J. McCall, and Q. Zhang, "Fitness modeling with Markov networks," *IEEE T. Evolut. Comput.*, vol. 17, no. 6, pp. 862–879, 2013.

[6] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston: Kluwer Academic Publishers, 2002.

[7] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag, 2006.

[8] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm Evol. Comput.*, vol. 1, no. 3, pp. 111 – 128, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650211000435

[9] C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano, "On the taxonomy of optimization problems under estimation of distribution algorithms," *Evol. Comput.*, vol. 21, no. 3, pp. 471–495, Sep. 2013.

[10] H. Mühlenbein and T. Mahnig, "Evolutionary optimization using graphical models," *New Gen. Comput.*, vol. 18, no. 2, pp. 157–166, 2000.

[11] C. F. Lima, F. G. Lobo, M. Pelikan, and D. E. Goldberg, "Model accuracy in the Bayesian optimization algorithm," *Soft Comput.*, vol. 15, no. 7, pp. 1351–1371, 2010.

[12] E. Radetic and M. Pelikan, "Spurious dependencies and EDA scalability," in *Proc. GECCO*, 2010, pp. 303–310.

[13] M. Pelikan, K. Sastry, and D. E. Goldberg, "Scalability of the Bayesian optimization algorithm," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 221 – 258, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888613X02000956

[14] A. E. I. Brownlee, J. A. W. McCall, and M. Pelikan, "Influence of selection on structure learning in Markov network EDAs: An empirical study," in *Proc. GECCO*. ACM Press, 2012, pp. 249–256.

[15] R. B. Heckendorn and A. H. Wright, "Efficient linkage discovery by limited probing," *Evol. Comput.*, vol. 12, no. 4, pp. 517–545, 2004.

[16] D. Goldberg, "Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction," *Complex Systems*, vol. 3, no. 2, pp. 129–152, 1989.

[17] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, no. 1-2, pp. 51–81, 2002.

[18] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," in *Advances in Soft Computing - Engineering Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry, Eds., 1999, pp. 521–535.

[19] R. Santana, "A Markov network based factorized distribution algorithm for optimization," in *Proc. of the 14th European Conf. on Machine Learning*, ser. LNAI, vol. 2837. Springer, 2003, pp. 337–348.

[20] R. Santana, P. Larrañaga, and J. A. Lozano, "Interactions and dependencies in estimation of distribution algorithms," in *Proc. IEEE CEC*, vol. 1. IEEE Press, 2-4 Sept. 2005, pp. 1418–1425.

[21] S. Shakya, R. Santana, and J. A. Lozano, "A Markovianity based optimisation algorithm," *Genet Program Evolvable Mach*, vol. 13, no. 2, pp. 159–195, Sep 2011. [Online]. Available: http://dx.doi.org/10.1007/s10710-011-9149-y

[22] S. Shakya, J. McCall, and A. Brownlee, "DEUM - distribution estimation using Markov networks," in *Markov Networks in Evolutionary Comp.*, S. Shakya and R. Santana, Eds. Springer, 2012, pp. 55–71.

[23] A. E. I. Brownlee, J. A. W. McCall, S. K. Shakya, and Q. Zhang, "Structure Learning and Optimisation in a Markov-network based Estimation of Distribution Algorithm," in *Proc. IEEE CEC*. Trondheim, Norway: IEEE Press, 2009, pp. 447–454.

[24] L. A. Marascuilo and M. McSweeney, *Nonparametric and Distribution-Free Methods for Social Sciences*. California: Brooks / Cole Publishing Company, 1977.

[25] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[26] S. Boslaugh and P. A. Watters, *Statistics: A Desktop Quick Reference*. Sebastopol, CA: O'Reilly, 2008.

[27] A. E. I. Brownlee, J. A. W. McCall, Q. Zhang, and D. Brown, "Approaches to Selection and their effect on Fitness Modeling in an Estimation of Distribution Algorithm," in *Proc. IEEE WCCI*. Hong Kong, China: IEEE Press, 2008, pp. 2621 –2628.

[28] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I. continuous parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 25–49, 1993.

[29] D. Wallin and C. Ryan, "Using over-sampling in a Bayesian classifier EDA to solve deceptive and hierarchical problems," in *Proc. IEEE CEC*, 2009, pp. 1660 – 1667.

[30] X. Li, S. Mabu, and K. Hirasawa, "Use of infeasible individuals in probabilistic model building genetic network programming," in *Proc. of the 13th Genetic and Evolutionary Comp. Conf. (GECCO 2011)*. New York, NY, USA: ACM, 2011, pp. 601–608.

[31] M. Munetomo, N. Murao, and K. Akama, "Introducing assignment functions to Bayesian optimization algorithms," *Info. Sciences*, vol. 178, no. 1, pp. 152 – 163, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025507003817

[32] Y. Hong, G. Zhu, S. Kwong, and Q. Ren, "Estimation of distribution algorithms making use of both high quality and low quality individuals," in *IEEE Int'l. Conf. on Fuzzy Systems (FUZZ-IEEE 2009)*, August 2009, pp. 1806–1813.

[33] T. Miquélez, E. Bengoetxea, and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers," *Int. J. Appl. Math. Comp.*, vol. 14, no. 3, pp. 101–115, 2004.

[34] A. H. Wright and S. Pulavarty, "On the convergence of an estimation of distribution algorithm based on linkage discovery and factorization," in *Proc. GECCO*. ACM Press, 2005, pp. 695–702.

[35] A. E. I. Brownlee, "Multivariate Markov Networks for Fitness Modelling in an Estimation of Distribution Algorithm," Ph.D. dissertation, Robert Gordon University, Aberdeen, May 2009. [Online]. Available: http://hdl.handle.net/10059/381