Maximal Nonlinearity in Balanced Boolean Functions with Even Number of Inputs, Revisited

***Abstract*—The problem of obtaining maximal nonlinearity in Boolean functions is well researched, both from the cryptographic and evolutionary computation side. However, the results are still not conclusive enough to be able to show how good a heuristic approach is when tackling this problem. In this paper, we investigate how to obtain the maximal possible nonlinearity in balanced Boolean functions, but we also analyze how difficult is the problem itself. In order to do so, we conduct experiments with Estimation of Distribution algorithms as well as the fitness landscape analysis and the deception analysis. Our results indicate that the first difficulties arise from the weak fitness function and somewhat inappropriate representation of solutions coupled with a huge search space. The fitness landscape analysis does not reveal any significant differences that could justify the assumed jump in difficulties when going from Boolean functions with 6 inputs to those with 8 inputs. Finally, we show that this problem is not order-1 deceptive.**

I. Introduction

Evolutionary algorithms represent a well used paradigm to solve many real world problems. As such, one can also find their role in the field of **cryptography**. Cryptography is a science of secret writing where the goal is to hide the meaning of a message [1]. Naturally, secrecy (confidentiality) is not the only goal, but is the easiest to grasp when considering it on a more abstract level. Some other goals that are usual in cryptography are authentication, data integrity or message integrity [1]. To fulfill those goals one has on his disposal cryptographic algorithms, more commonly known as **ciphers**.

Cryptography is a huge area and there exist a number of taxonomies where a common one is on the basis on how the key is used [1]. There, we talk about **symmetric key cryptography** when all parties use the same key, and **public key cryptography** when there is a public and a private key. Next, symmetric key cryptography can be further divided into **block ciphers** and **stream ciphers**. Block ciphers operate on blocks of fixed length of data with an unvarying transformation that is specified by the key. Most of the stream ciphers encrypt message bits by adding encryption bits modulo two.

Well designed ciphers are made in a way to be able to resist cryptanalytic attacks, e.g. to possess some cryptographic properties that make them strong. To resist against one on the most well researched kind of attacks - linear cryptanalysis [2], one requires that a cipher possess enough nonlinearity. That nonlinearity may come from various sources like multiplication or addition operation modulo some value or from Boolean functions. In fact, Boolean functions (more precisely, vectorial Boolean functions also known as S-boxes) are the most common way to add nonlinearity to ciphers [3]. S-boxes are dominantly used in block ciphers, while Boolean functions (or S-boxes with the number of outputs strictly smaller than the number of inputs) are used in stream ciphers. Although Boolean functions lost some of their appeal as a cryptographic primitive in today's practice, they still represent an interesting research area.

To obtain Boolean functions with good enough properties to be used in practice, one can use either random search, heuristics, algebraic construction, or a combination of the aforesaid techniques. Here we are interested in heuristic techniques, and more precisely evolutionary computation (EC). In Section III, we present a number of related works, but for now it suffices to say that there is a plethora of works examining the successfulness of EC techniques when evolving Boolean functions. Furthermore, one standard goal in existing papers is to evolve balanced Boolean functions with maximal nonlinearity (and usually with some additional properties). Since we have the same goal here, we first give a rough motivation for our research. As explained in Section II, finding maximal nonlinearity is not an easy problem; moreover, e.g. for balanced Boolean functions with eight inputs, theory says that the best possible nonlinearity is 118, but in practice we know of only nonlinearity 116 (obtained both with heuristics and algebraic constructions). Is finding a function with nonlinearity 118 important? In our opinion it is not, since the gain in the security would be minimal and furthermore, today it is considered as a minimum for a Boolean function to have 13 inputs to be secure [4]. However, one noteworthy goal would be either to find such a function or to show there are no such functions. Unfortunately, heuristic methods are in general not able to give us an answer on whether a particular solution exists, but they are able to produce solutions that could be (but are not necessary) good enough. Note that despite the theory, there exists a dose of doubt on whether such functions with nonlinearity equal to 118 really exist.

In this paper, although we look for maximal nonlinearity, we do not consider that as our end goal. Rather, through that investigation we try to analyze whether this problem is difficult for EC. Our reasoning is that if we can show the problem is not difficult for EC methods and yet no one is able to find such functions, that could serve as a strong indication that such functions do not exist. Conversely, if the problem is difficult, then it makes more plausible there are such functions, and we are just missing some breakthrough in the currently used approaches to find them. Naturally, it is quite obvious to see that the problem is not easy in any case since the search space for a Boolean functions with n inputs equals 2^{2^n}, which for Boolean functions with

8 inputs equals 2^{256}. To make our investigation stronger, we do not limit our attention only to Boolean functions with eight inputs. The smaller sizes are used to better understand the problem and the larger sizes are used as a sanity check for the problem difficulty, i.e. if it is easy to find maximal nonlinearity there, then it should be also easy to find it for eight inputs. Finally, we note that the Boolean functions we consider here are not appropriate for real world usage in cryptography since we do not consider all properties a Boolean function should possess to be used in cryptography.

A. Motivation and Contributions

The empirical investigation of the Boolean functions using EDAs aims to determine whether the combination of low order building blocks (in EDAs these building blocks are encapsulated in the factorization determined by the probabilistic model) allows the algorithm to find better solutions of the problem and furthermore, we evaluate whether the representation of higher order dependencies in probabilistic models makes a significant impact in the quality of the solutions found by EDAs. With those experiments, we aim to give answer to the following questions:

- Whether probabilistic modeling of the search space by means of estimation of distribution algorithms can be an efficient approximation to solve the problem at hand.
- Is there any influence of the order and type of the interactions that the model is able represent in the results of the EDAs?
- Does the highly symmetric representation used to encode the problem influence the behavior of EDAs, and if yes, how?
- Which fitness function is more appropriate to solve this problem using EDAs?
- Can the reported symmetric properties of the space be used to make a more intelligent (or informed) use of probabilistic modeling?

As already indicated in the previous section, we do not limit ourselves only to experimental results with EDAs, but also we persevere to give more insight into the problem difficulty. Therefore, first we conduct a fitness landscape analysis in an effort to recognize similarities among landscapes for various dimensions of Boolean functions. Here, the motivation stems from the expectation that if landscapes are similar then there is no reason that one dimension is more difficult to solve than some other (except the obvious reason which is the exponential growth of the search space).

The notion of building blocks that are combined to create optimal solutions as the result of the application of genetic operators is central to genetic algorithms (GAs). Deception analysis is very related to the idea that for some problems the information available from the fitness of low order schema is misleading. In these situations the combination of partial subsolutions of high fitness will eventually lead the GA away from the optimum. The deception analysis uses recent theoretical results [5] about the symmetry structure of Boolean function. We show that according to the recent definition of deception proposed by Whitley [6], Boolean functions are not 1-order deceptive.

B. Outline

The paper is organized in the following way. It consists of three main parts; in Section V we work with the estimation of distribution algorithms in an effort to estimate the hardness of this problem when stepping away from more commonly used evolutionary algorithms. In Section VI, we present a fitness landscape analysis of the search space of balanced Boolean functions, and in Section VII we analyze whether this problem is deceptive. However, first we introduce in Section II necessary details about cryptographic properties ans representation of Boolean functions. Next, in Section IV, we present fitness function we use in our experiments. Finally, we end with short conclusions in Section VIII.

II. Theoretical Background

In this section, we present the notation we follow, as well as the basic information about Boolean functions representations and cryptographic properties of interest. Let $n, m \in \mathbb{N}$. Next, the set of all n-tuples of the elements in the field $\mathbb{F}_2$ is denoted as $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements. The set $\mathbb{F}_2^n$ represents all binary vectors of length n[4]. An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ where Boolean functions represent $m = 1$ case. The inner product of vectors $\vec{a}$ and $\vec{b}$ is denoted as $\vec{a} \cdot \vec{b}$ and it equals $\vec{a} \cdot \vec{b} = \oplus_{i=1}^{n} a_i b_i$, with "$\oplus$" being addition modulo two. The Hamming weight (HW) of a vector $\vec{a}$, where $\vec{a} \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector.

A. Boolean Function Representations

We are interested in two unique representations of Boolean functions. A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a **truth table** (TT), which is a vector $(f(\vec{0}), ..., f(\vec{1}))$ that contains the function values of f, ordered lexicographically [4]. This represent the natural encoding of evolutionary algorithms' solutions as a string of bits.

The second unique representation of a Boolean function is the **Walsh-Hadamard transform** W_f where we are interested in it since through it we calculate relevant cryptographic properties of Boolean functions. The Walsh-Hadamard transform of a Boolean function f measures the correlation between $f(\vec{x})$ and the linear function $\vec{a} \cdot \vec{x}$ [4]:

$$W_f(\vec{a}) = \sum_{\vec{x} \in \mathbb{F}_2^n} (-1)^{f(\vec{x}) \oplus \vec{a} \cdot \vec{x}}. \quad (1)$$

B. Boolean Function Properties

In the truth table representation, a Boolean function with n inputs is balanced if its Hamming weight equals 2^{n-1}. Alternatively, A Boolean function f is **balanced** if the Walsh-Hadamard spectrum of a vector $\vec{0}$ equals zero [7], [8]:

$$W_f(\vec{0}) = 0. \quad (2)$$

The nonlinearity N_f of a Boolean function f is the minimum Hamming distance between the function f and all affine functions [4]. The **nonlinearity** N_f of a Boolean function f expressed in terms of the Walsh-Hadamard coefficients equals [4]:

$$N_f = 2^{n-1} - \frac{1}{2} max_{\vec{a} \in \mathbb{F}_2^n} |W_f(\vec{a})|. \quad (3)$$

If a Boolean function f has the correlation immunity property of order t, an even number of inputs n, and $k \leq \frac{n}{2} - 1$ then its nonlinearity N_f has an upper bound as follows [9]:

$$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1} - 2^k, \quad (4)$$

where k equals $t+1$ if f is balanced or has Hamming weight divisible by 2^{t+1} and k equals t otherwise.

In the case when $k > \frac{n}{2} - 1$ then the nonlinearity property has the upper bound:

$$N_f \leq 2^{n-1} - 2^k. \quad (5)$$

For further information about Boolean functions and their cryptographic properties, we refer interested readers to [4].

III. Related Work

There exists a multitude of examples where heuristic techniques yielded good results in constructing Boolean functions with cryptographic properties. Therefore, one could aim to classify those works on a basis of what heuristics they use, what cryptographic properties and Boolean function sizes are objective or what representation of Boolean function is used. Here, we opted to follow the last direction and we briefly enumerate several important works. As already mentioned in Section II, there is usually a distinction between the representation that the heuristics uses to represent individuals and that fitness function uses to evaluate individuals.

Accordingly, the best researched representation (and one we also follow) of solutions is as a bitstring, i.e. truth table representation. The first paper, as far as the authors know, that explores the evolution of Boolean functions for cryptography is by Millan et al. [10]. There, the authors try to evolve Boolean functions with high nonlinearity. Millan, Clark, and Dawson further increased the strength of genetic algorithms (GAs) by combining them with the hill climbing together with a resetting step. The goal was to find highly nonlinear Boolean functions with up to 12 variables [11]. Clark and Jacob experimented with two-stage optimization to generate Boolean functions with high nonlinearity and low autocorrelation [12]. They used a combination of simulated annealing and hill climbing with a cost function motivated by Parseval's theorem.

Next, there is a line of works that uses different than the truth table representation to encode individuals, but to evaluate them, first they map solutions to the truth table representation. Picek, Jakobovic, and Golub experimented with GAs and genetic programming (GP) to find Boolean functions that possess several optimal properties [13]. As far as the authors know, this is the first GP application for evolving cryptographically suitable Boolean functions. Here, the evolved tree (genotype) is a posteriori transformed to the truth table representation for the evaluation purposes. Hrbacek and Dvorak used Cartesian genetic programming (CGP) to evolve bent Boolean functions of sizes up to 16 inputs where the authors experimented with several configurations of algorithms in order to speed up the evolution process [14]. As far as we are aware, this is the fist CGP application for evolving cryptographic Boolean functions. Picek et al. compared the effectiveness of GP and CGP algorithms when looking for highly nonlinear balanced Boolean functions with eight inputs [15]. A detailed analysis of the efficiency of a number of evolutionary algorithms and fitness functions is given in [16]. There, the authors also gave a thorough related work list.

Besides the truth table representation of solutions, we were able to find only a handful of papers that use the Walsh-Hadamard spectrum to encode individuals. Although that representation seems natural, both from the Boolean functions perspective, as well as from the evolutionary one, there are some difficulties. The main problem lies in a fact that it is not known what Walsh-Hadamard values (or in what ordering) constitute a Boolean function. More precisely, on the basis of Parseval's theorem one can infer what values could be in the Walsh-Hadamard spectrum, but it is impossible to say what the positions should be. Therefore, when generating the Walsh-Hadamard spectrum it is necessary to make an inverse transform to verify that the spectrum indeed maps to a Boolean function. Clark et al. experimented with simulated annealing in order to design Boolean functions using spectral inversion [17]. They worked in the spectral domain where the cost function punishes those solutions that are not Boolean functions. Mariot and Leporati used GAs where the genotype is a list of integer values in order to evolve plateaued Boolean functions [18]. Plateaued functions have only three values in the Walsh-Hadamard spectrum and therefore represent somewhat easier task to evolve when working with the Walsh-Hadamard representation than in the case when considering balanced Boolean functions.

Finally, Picek, McKay, Santana, and Gedeon explore how to find balanced Boolean function with eight inputs and maximal nonlinearity, but they approach this task through the analysis of symmetries one encounters in the search space [5]. They conclude that this problem exhibits a number of symmetries which makes it difficult for many evolutionary algorithms. This work also represents a motivation of a sort for our research since we continue to approach this problem from the more analytical perspective, where the main goal is to deduce possible obstacles in the evolutionary search.

IV. Fitness Function

We start from the fitness functions as used in [5] to evaluate the quality of a Boolean function:

$$\textbf{nonlin}(f) = \min_{\textbf{affine } f_1 \in 2^{2^N}} \delta(f, f_1) \quad (6)$$

$$\textbf{imbal}(f) = \text{abs}(|x : f(x) = 1| - |x : f(x) = 0|) \quad (7)$$

$$= \min_{\textbf{balanced } f_1 \in 2^{2^N}} 2 \times \delta(f, f_1)$$

$$\textbf{fit}(f) = \max_{f \in 2^{2^N}} \{\textbf{nonlin}(f) - \textbf{imbal}(f)\} \quad (8)$$

Note that the fitness function above is the same as in a number of related works, cf. [15], [16] where one punishes the imbalanced solutions. Here, δ represents Manhattan distance (Hamming distance) between two Boolean functions, i.e. between a candidate Boolean function and all affine functions. The nonlinearity $nonlin(f)$ is computed as per Eq. (3).

We note that extensive preliminary experiments with this function showed that no variant of EDA (with and without local search) was able to reach the 116 fitness value for a Boolean function size of eight inputs. It seems this is not only due to the symmetric nature of the search space but to the relatively few information about the quality of the solutions provided by the nonlinearity expression.

Indeed, in related works one can find other attempts on fitness functions that use more information in the nonlinearity part and consequently produce better results. One prominent example is from the work of Clark and Jacob [19] where they use whole Walsh-Hadamard spectrum:

$$cost(f) = \sum_{\vec{a}} ||W_f(\vec{a})| - X|^R, \quad (9)$$

with X and R being real valued parameters. We acknowledge that using the above function can result in improvements of results, but nevertheless we believe that the cost of two more additional parameters that need to be tuned is significant. As far as we are aware, there is no straightforward way to choose those parameters, nor is there some obvious scaling rule between the parameter values and the Boolean function size.

Therefore, we opted to modify fitness function in a way that adds more information about the quality of a certain solution with regards to the Walsh-Hadamard spectrum, but avoiding the drawbacks of additional parameters:

$$modified\_N_f = N_f + \frac{(2^n - freq(|W_f(\vec{a})|))}{2^n}, \quad (10)$$

where $freq$ is the number of times the maximum Walsh-Hadamard coefficient $W_f(\vec{a})$ occurs among all Walsh-Hadamard coefficients.The rationale behind this definition is allowing the optimization algorithm to reduce the number of occurrences of the Walsh-Hadamard coefficients of maximum value, in the expectation that this pressure will lead to a minimum number of repetitions, and eventually a transition to a function with a smaller maximal Wash-Hadamard coefficient. It is a more informative version of Eq. (3) because it allows to distinguish between different functions that share the same amount of nonlinearity. Since the gain due to reducing the frequency of the maximum coefficient should not be higher than the gain in increasing nonlinearity itself, we normalize the second part of our fitness function by dividing it with a value 2^n, which effectively bounds the value of the fraction to $[0, 1]$ range. Finally, to deal with the imbalanced solutions, we add the balancedness criterion penalty to the fitness function we use in this paper:

$$fitness = N_f + \frac{(2^n - freq(|W_f(\vec{a})|))}{2^n} - \textbf{imbal}(f). \quad (11)$$

V. Estimation of distribution algorithms

The main idea of Estimation of distribution algorithms (EDAs) [20], [21] is to extract patterns shared by the best solutions, represent these patterns using a probabilistic graphical model (PGM) [22], and generate new solutions from this model. In contrast to GAs, EDAs apply learning and sampling of distributions instead of classical crossover and mutation operators. Modeling the dependencies between the variables of the problem serves to efficiently orient the search to more promising areas of the search space by explicitly capturing and exploiting potential relationships between the problem variables. The pseudocode of an EDA is shown in Algorithm 1.

Algorithm 1: **Estimation of distribution algorithm**

```
1 Set t ⇐ 0. Generate N solutions randomly.
2 do {
3     Evaluate the solutions using the fitness function.
4     Select a population D_t^S of K ≤ N solutions according to a selection method.
5     Calculate a probabilistic model of D_t^S.
6     Generate N new solutions sampling from the distribution represented in the model.
7     t ⇐ t + 1
8 } until Termination criteria are met.
```

We work with positive distributions denoted by p. $p(x_I)$ denotes the marginal probability for $\mathbf{X}_I = \mathbf{x}_I$. $p(x_i \mid x_j)$ denotes the conditional probability distribution of $X_i = x_i$ given $X_j = x_j$. Three types of probabilistic graphical models are used: 1) Univariate model. 2) 1-order Markov model. 3) Tree model.

A. Probabilistic models

A probability distribution $p_\mathcal{T}(\mathbf{x})$ that is conformal with a tree is defined as:

$$p_\mathcal{T}(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|pa(x_i)), \quad (12)$$

where $Pa(X_i)$ is the parent of X_i in the tree, and $p(x_i|pa(x_i)) = p(x_i)$ when $pa(X_i) = \emptyset$, i.e. X_i is a root of the tree. We allow the existence of more than one root in the PGM (i.e. forests) although for convenience of notation we refer to the model as tree.

One particular case of trees are chain-shaped distributions in which each variable depends X_i on the previous variable in a given order. We call this case as the 1-order Markov model.

$$p_{MK}(\mathbf{x}) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}) \quad (13)$$

The simplest example of distributions is the univariate model, where variables are assumed to be independent. For this model the probability of a solution is the product of the univariate probabilities for all variables:

$$p_u(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) \quad (14)$$

Univariate approximations are expected to work well for functions that can be additively decomposed into functions of order one (e.g. $g(\mathbf{x}) = \sum_i x_i$). The 1-order Markov model captures only dependencies between adjacent variables, and the tree model can represent a maximum of $n-1$ bivariate dependencies. The computational cost of EDAs is mainly associated to the methods needed to learn and sample the models. The most complex EDA used in this paper is Tree-EDA which has a computational cost $O(n^2)$. Examples of EDAs that use univariate, 1-order Markov, and tree models are respectively presented in [20], [23] and [24], [25] and details on the methods used to learn and sample the models can be obtained from these references.

B. Enhancement to EDAs

Hybridization has been proposed [26] an as one of the most efficiency-enhancement technique for EDAs. We implemented different greedy search algorithms as part of the EDA. In this paper, results are presented for a local optimizer that proposes swaps between pairs of variables with different values and accept the move if the fitness is improved. A maximum number of moves was set to $\frac{2^n}{10}$.

The other enhancement added to the EDA was partial sampling. In classical EDAs, when a new solution is generated all variables are sampled from the model. The idea of partial sampling is to use a template from the selected population and sample only a subset of the variables using the learned probabilistic model. That way, some of the structure of the selected solution is kept, but information from the model is added. Partial sampling is defined by a parameter $0 < rate \leq 1$ that indicates which proportion of variables will be sampled.

C. Experimental benchmark

From preliminary experiments with the three EDAs some observations were extracted that shaped the design of the experiments described here:

- The class of probabilistic model used was not necessarily the most decisive factor in the success rate of the algorithm.
- The selection strength, determined by the truncation selection factor had a strong effect in the success of the EDAs. In particular, it seems that a very strong selection pressure was important to reach high fitness solutions.
- The sampling rate was also crucial. A low sampling rate ($r = 0.25$) led almost in every case to solutions of higher fitness.

Therefore, in an initial step we decided to investigate the influence of three factors in the behavior of the algorithms:

- Type of probabilistic model.
- Strength of selection.
- Population size.

We conducted experiments with the fitness function as given in Eq. 8 (informally called "weak" fitness) where for smaller dimensions (i.e. up to 6 inputs) EDAs succeed in reaching the global optimum, while for larger dimension (8 inputs and higher) they are stuck in local optima. By local optima here we understand all values that are lower than the currently known best values (since it is not always clear whether there exist better values). Due to the lack of space, we do not give those results here. Instead, we concentrate only on results with fitness function as in Eq. (11).

Each EDA is uniquely defined by the configuration of these three factors. All other parameters were identically set. 100 experiments were run for each EDA configuration for problems $n \in [8, 10]$. For $n = 12$, only 20 repetitions of the problems were conducted due to the high computational cost. Also, for $n = 12$, Tree-EDA was not applied for the same reasons. Experiments were run in a cluster of $752 \times 86\_64$ processing cores, 2.5 TB of RAM and 22 TB of disk. Note that we do not give results for smaller sizes of Boolean functions since the problem there is easy regardless of the choice of the fitness function.

The average fitness of the best solution achieved in each of the runs for all problem sizes are shown in Fig. 1. The color of each cell in the matrix represents the average fitness for an EDA (x axis top) with given population size (x axis bottom) and truncation selection (y axis). We give results for Boolean function dimensions a) $n = 8$; b) $n = 10$; c) $n = 12$.

In order to determine the effect of using different factorizations in the search for optimal solutions, we tested for significant differences between the three EDAs for each combination of population size and truncation selection. A multiple comparison statistical test was conducted using the best solutions reached in 100 runs. The Kruskal Wallis test was applied first, and a post-hoc test was applied afterwards to look for statistical differences between each pair of algorithms. A Bonferroni correction was added to compensate for multiple comparisons. All tests used as pvalue $\alpha = 0.01$. The test did not show statistical differences between any pair of algorithms showing that the order of dependencies between the variables included in the model is not a relevant factor for the behavior of EDAs for the Balanced function problem.

One interesting question is which is the best solution among all that have the same non-linearity value, i.e. which is the best value of $freq(|W_f(\vec{a})|)$? This value could serve as a surrogate of how close are the solutions to the next best non-linearity value (118). In our experiments, the best value of $256 - freq(|W_f(\vec{a})|)$ was 240, still far from the possible maximum which is 255. It was reached only once. The

b)

Fig. 1: EDA results for different configurations of the population size ($N \in \{50, 100, 250, 500\}$), the truncation parameter $t \in \{0.2, 0.4\}$, and different problems. a) $n = 8$; b) $n = 10$; $n = 12$.

Fig. 2: Histogram of the fitness values above 116.

distribution of the fitness values for all solutions with fitness above 116 in all the configurations previously analyzed are shown in Fig. 2.

VI. Fitness Landscape Analysis

In this section, we give a fitness landscape analysis for balanced Boolean functions with dimensions from 4 to 12 inputs. The goal is to identify whether there are some similarities in landscapes that could point us how difficult this problem is. We start with Boolean functions with 4 inputs since in [5] authors showed that the large part of the search space belongs to the functions with the maximal nonlinearity. Accordingly, there the problem of obtaining maximal nonlinearity is easy. When going to functions with 6 inputs, authors report that it is still easy to find maximal nonlinearity functions, but now the big proportion of the solutions have suboptimal nonlinearity values. Increasing just one step more, e.g. going to 8 inputs, the trend from the 6 inputs case is amplified and now one cannot find maximal nonlinearity anymore.

Therefore, one can immediately see that this discrepancy requires more research. Indeed, if the 4 and 6 inputs cases are easy, and 8 input case is difficult, can this observation be confirmed with differences in the fitness landscape? Or, if the landscapes are similar, should that point us that there is in fact no 118 nonlinearity for functions with 8 inputs? More generally, one expects that the landscape analysis should help when designing more suited representations and fitness functions. Besides the aforementioned sizes, we investigate also 10 and 12 input cases to strengthen our observations.

When approaching the fitness landscape analysis, the first question that needs to be addresses is how to measure landscape properties. Here, we follow properties as given by Talbi [27], but due to the lack of space, we do not give description of investigated properties. To measure the distance between solutions we use a standard measure for the bitstring representation - Hamming distance (i.e. Manhattan distance) [27]. To obtain fitness landscape properties, we use the following strategy. First a random population of 100 balanced individuals (Boolean functions) is generated. Next, we run a local search algorithm that is run until the solution converges as given in Algorithm 2. The average fitness of the initial random populations and results after convergence of the local search algorithm are given in Table I.

Algorithm 2: **Local search algorithm.**

```
1  Set individual_size t ⇐ 2^n.
2  for i = 0 to individual_size
3      Flip bit at position i.
4      for j = 0 to individual_size
5          if Value at i = value at j and i ≠ j then
6              Flip bit at position j.
7              Evalaute individual.
8              if Individual improved then
9                  Keep individual.
10                 Exit loop and go to next individual.
11             else
12                 Restore individual.
```

Based on the local search results, the obtained fitness landscape values are given in Table II. Here, $d(P)$ is the population diameter and $dmm_n(P)$ is the normalized average

TABLE I: Local search results (landscape analysis).

Boolean variables	4	6	8	10	12
Initial avg. fitness	3.125	19.33	98.21	445.6	1896.7
Converged - avg.	4.871	25.07	111.5	471.6	1951.3
Converged - max.	4.875	26.75	112.984	476.999	1967

TABLE II: Fitness landscape properties.

Property	4	6	8	10	12
$d(P)$	16.0	50.0	161.0	590.0	1102.0
$dmm_n(P)$	0.250162	0.319821	0.397437	0.433907	0.464635
Δ_{dmm}	-0.000322	0.000432	-0.000173	0.000047	0.000260
$Amp(P)$	1.000796	1.066978	1.030360	1.011429	1.000000
Δ_{Amp}	0.345531	0.173949	0.071524	0.034883	0.8319
$Lmm(P)$	3.224	12.828	26.606	42.102	7.9

population distance. Δ_{dmm} represents variation after convergence, $Amp(P)$ is fitness amplitude and Δ_{Amp} its variation, and finally, $Lmm(P)$ is the number of local search steps.

From the presented results, it is not easy to reach any definitive conclusions, but some trends can be observed. First, normalized average population distance $dmm_n(P)$ points us to the concentration of the solutions in the search space. We see that as the search space grows the corresponding $dmm_n(P)$ value grows which indicates that the clustering of the solutions in a small region of search space reduces. Next, when considering distribution in the objective space, $Amp(P)$ value gives us the relative difference between the best quality of the population and the worst one. From the results we see that the difference is similar for all dimensions of interest. Furthermore, when checking the relative variation of the amplitude Δ_{Amp} between a starting random population and the final population, we see the value drops as the dimension increase, which is to be expected. However, for dimension 12 that value is much higher than for any other dimension which indicates that the difference between the starting and final solutions is much higher. Next, the length of a walk $Lmm(P)$ belongs to the correlation measures and indicates the ruggedness of the landscape. $Lmm(P)$ grows for dimensions 4 to 10, which points us that those landscapes are similar. Since we know that for dimension 4 there are only three possible fitness values (cf. [5]), it is clear that the landscape is relatively smooth (i.e. with a small number of optima). The dimensions from 6 to 10 inputs behave similarly. However, the value for problem size of 12 inputs points to a extremely rugged landscape with a large number of optima.

VII. Deception Analysis

For convenience, we use the definition of deception recently given by Whitley [6]. Let h denote a $(n-j)$-dimensional hyperplane where j variables have preassigned bit values and $\alpha(h)$ be a mask with 1 bits marking the locations where the j variables appear in the problem location and 0 elsewhere. Let $MAX(\mathbf{x}, \alpha(h))$ return the hyperplane with the best mean over all 2^j order j hyperplanes that can be defined using the $\alpha(h)$ mask.

A function is *order-j deceptive* [6] if the j bit values returned by $MAX(\mathbf{x}, \alpha(h))$ for all hyperplanes of order j are not the same as the bit values found in a string which is a global optimum. In the particular case of order-1 deception, the definition requires that the function will be deceptive if for the n order-1 hyperplanes the value of $MAX(\mathbf{x}, \alpha(h))$ will be different to the optimum.

Theorem 1. *Balanced Boolean functions are not order-*1 *deceptive.*

Proof. For $j = 1$, $MAX(\mathbf{x}, \alpha(h))$ can only return two possible values, 0 and 1, for any variable. Since all balanced Boolean functions are value-symmetric, for any variable X_i there is always a global optimum solution $\hat{\mathbf{x}}$ such that $\hat{x}_i = 0$ (respectively, $\hat{x}_i = 1$). Therefore, by definition the function cannot be order-1 deceptive. □

The value-symmetry property that holds for Boolean functions determines that order-1 schemata have equal average fitness. One open question is what is the distribution of the average fitness between higher order schemata. To investigate this issue, we evaluate the complete space of solutions for $n = 4$ and determined the frequencies for the schemata. This information is shown in Fig. 3 for $k \in \{1, 2, 3, 4\}$ where the average fitness of the schemata have been normalized to ease the visualization of the results. It can be shown in Fig. 3 that with the exception or all-zero and or-ones schemata, all other higher order schemata up to order $k = 4$ share similar average fitness. This fact seems to indicate that that the difference between the average contribution of optimal and suboptimal schemata is very narrow. An EA that uses this information can end up trapped in a local optimum or producing optimal solutions with different configurations.

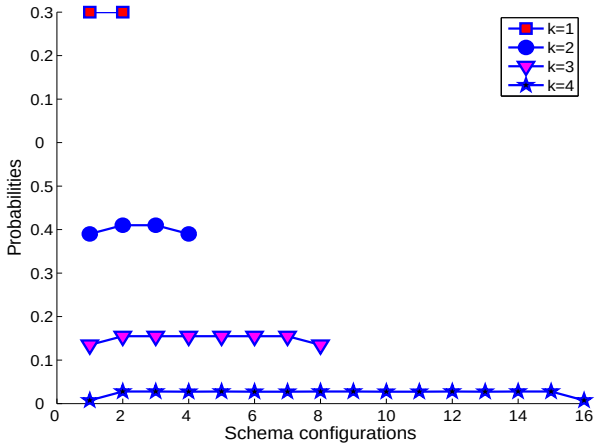

Fig. 3: Probabilitity of the schema for $k \in \{1, 2, 3, 4\}$

VIII. Conclusion

In this paper, we approach well researched problem of obtaining the maximal nonlinearity in balanced Boolean functions with varying number of inputs. First, we give experimental results with Estimation of Distribution Algorithms that

confirm a weakness in fitness function if using only maximal value of Walsh-Hadamard spectrum. However, changing the fitness function enables EDAs to reach for instance the currently best known nonlinearity value (116) for a balanced Boolean function with 8 inputs. This points us that the same change should result in the improvement of results for other EAs that use such a "weak" fitness function and bitstring representation. Besides that, we note no significant difference in relation to other evolutionary paradigms.

Next, we provide a fitness landscape analysis that gives some insights into the perceived difficulty of this problem and shows there is nothing significantly different with the landscape of balanced Boolean functions with 8 inputs when for instance comparing with functions with 6 or 10 inputs. However, we detect a number of differences for balanced Boolean functions with 12 inputs which merits further investigation that we leave for future work. Finally, we provide a proof that this problem is not order-1 deceptive, which means the difficulty should also not come from there. As the future work we plan to investigate higher orders of deception. To conclude, our investigation points us that two main sources of difficulties are the search space size and the high level of symmetries.

REFERENCES

[1] L. R. Knudsen and M. Robshaw, *The Block Cipher Companion*, ser. Information Security and Cryptography. Springer, 2011.

[2] M. Matsui and A. Yamagishi, "A new method for known plaintext attack of FEAL cipher," in *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques*, ser. EUROCRYPT'92. Berlin, Heidelberg: Springer-Verlag, 1993, pp. 81–91.

[3] J. Daemen and V. Rijmen, *The Design of Rijndael*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[4] C. Carlet, "Boolean Functions for Cryptography and Error Correcting Codes," in *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 1st ed., Y. Crama and P. L. Hammer, Eds. New York, NY, USA: Cambridge University Press, 2010, pp. 257–397.

[5] S. Picek, R. I. McKay, R. Santana, and T. D. Gedeon, "Fighting the symmetries: The structure of cryptographic boolean function spaces," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 457–464.

[6] D. Whitley, "Mk landscapes, NK landscapes, MAX-kSAT: A proof that the only challenging problems are deceptive," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. ACM, 2015, pp. 927–934.

[7] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, "Propagation characteristics of Boolean functions," in *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, ser. EUROCRYPT '90. New York, NY, USA: Springer-Verlag New York, Inc., 1991, pp. 161–173.

[8] R. Forrié, "The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition," in *Advances in Cryptology - CRYPTO' 88*, ser. Lecture Notes in Computer Science, S. Goldwasser, Ed. Springer New York, 1990, vol. 403, pp. 450–468.

[9] P. Sarkar and S. Maitra, "Nonlinearity Bounds and Constructions of Resilient Boolean Functions," in *Advances in Cryptology - CRYPTO 2000*, ser. Lecture Notes in Computer Science, M. Bellare, Ed., vol. 1880. Springer Berlin Heidelberg, 2000, pp. 515–532.

[10] W. Millan, A. Clark, and E. Dawson, "An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions," in *Proceedings of the First International Conference on Information and Communication Security*, ser. ICICS '97. London, UK, UK: Springer-Verlag, 1997, pp. 149–158.

[11] ——, "Heuristic design of cryptographically strong balanced Boolean functions," in *Advances in Cryptology - EUROCRYPT '98*, 1998, pp. 489–499.

[12] J. Clark and J. Jacob, "Two-Stage Optimisation in the Design of Boolean Functions," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, E. Dawson, A. Clark, and C. Boyd, Eds. Springer Berlin Heidelberg, 2000, vol. 1841, pp. 242–254.

[13] S. Picek, D. Jakobovic, and M. Golub, "Evolving Cryptographically Sound Boolean Functions," in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '13 Companion. New York, NY, USA: ACM, 2013, pp. 191–192.

[14] R. Hrbacek and V. Dvorak, "Bent Function Synthesis by Means of Cartesian Genetic Programming," in *Parallel Problem Solving from Nature - PPSN XIII*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, Eds. Springer International Publishing, 2014, vol. 8672, pp. 414–423.

[15] S. Picek, D. Jakobovic, J. F. Miller, E. Marchiori, and L. Batina, "Evolutionary methods for the construction of cryptographic boolean functions," in *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, 2015, pp. 192–204.

[16] S. Picek, D. Jakobovic, J. F. Miller, L. Batina, and M. Cupic, "Cryptographic boolean functions: One output, many design criteria," *Applied Soft Computing*, vol. 40, pp. 635 – 653, 2016.

[17] J. A. Clark, J. Jacob, S. Maitra, and P. Stănică, "Almost Boolean functions: the design of Boolean functions by spectral inversion," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 3, Dec 2003, pp. 2173–2180 Vol.3.

[18] L. Mariot and A. Leporati, "Heuristic search by particle swarm optimization of boolean functions for cryptographic applications," in *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, 2015, pp. 1425–1426.

[19] J. Clark and J. Jacob, "Two-Stage Optimisation in the Design of Boolean Functions," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, E. Dawson, A. Clark, and C. Boyd, Eds. Springer Berlin Heidelberg, 2000, vol. 1841, pp. 242–254.

[20] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Parallel Problem Solving from Nature - PPSN IV*, ser. Lectures Notes in Computer Science, vol. 1141. Berlin: Springer, 1996, pp. 178–187.

[21] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on probabilistic graphical models in evolutionary computation," *Journal of Heuristics*, vol. 18, no. 5, pp. 795–819, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10732-012-9208-4

[22] J. Pearl, *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

[23] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*, Mozer et al., Ed., vol. 9. The MIT Press, Cambridge, 1997, pp. 424–430.

[24] S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," in *Proceedings of the 14th International Conference on Machine Learning*, D. H. Fisher, Ed., 1997, pp. 30–38.

[25] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 418–438, 2008.

[26] K. Sastry, M. Pelikan, and D. E. Goldberg, "Efficiency enhancement of estimation of distribution algorithms," in *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, ser. Studies in Computational Intelligence, M. Pelikan, K. Sastry, and E. Cantú-Paz, Eds. Springer, 2006, pp. 161–186.

[27] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.