Article

Metaheuristics in the Optimization of Cryptographic Boolean Functions

Isaac López-López [1], Guillermo Sosa-Gómez [2,*], Carlos Segura [1], Diego Oliva [3,4] and Omar Rojas [2]

[1] Centro de Investigación en Matemáticas A.C. (CIMAT). Área de Computación, Jalisco S/N, Col. Valenciana, Guanajuato 36023, Mexico; isaac.lopez@cimat.mx (I.L.-L.); carlos.segura@cimat.mx (C.S.)
[2] Facultad de Ciencias Económicas y Empresariales, Universidad Panamericana, Álvaro del Portillo 49, Zapopan, Jalisco 45010, Mexico; orojas@up.edu.mx
[3] IN3-Computer Science Department, Universitat Oberta de Catalunya, 08860 Castelldefels, Spain; diego.oliva@cucei.udg.mx
[4] Depto. de Ciencias Computacionales, Universidad de Guadalajara, CUCEI, Av. Revolución 1500, Guadalajara C.P. 44430, Jalisco, Mexico
* Correspondence: gsosag@up.edu.mx; Tel.: +52-33-1368-2200

Received: 3 September 2020; Accepted: 16 September 2020; Published: 21 September 2020

Abstract: Generating Boolean Functions (BFs) with high nonlinearity is a complex task that is usually addresses through algebraic constructions. Metaheuristics have also been applied extensively to this task. However, metaheuristics have not been able to attain so good results as the algebraic techniques. This paper proposes a novel diversity-aware metaheuristic that is able to excel. This proposal includes the design of a novel cost function that combines several information from the Walsh Hadamard Transform (WHT) and a replacement strategy that promotes a gradual change from exploration to exploitation as well as the formation of clusters of solutions with the aim of allowing intensification steps at each iteration. The combination of a high entropy in the population and a lower entropy inside clusters allows a proper balance between exploration and exploitation. This is the first memetic algorithm that is able to generate 10-variable BFs of similar quality than algebraic methods. Experimental results and comparisons provide evidence of the high performance of the proposed optimization mechanism for the generation of high quality BFs.

Keywords: boolean function; metaheuristics; nonlinearity; cryptography; hadamard transform; entropy

1. Introduction

In several cryptographic systems and in the use of noisy channels, Boolean Functions (BFs) play an important role. Specifically, BFs are used in the internal operation of many cryptographic algorithms, and it is known that in order to avoid the success of cryptanalysis on such systems, increasing the degree of nonlinearity of BFs is utterly important. The research in Cryptographic Boolean Functions (CBFs) has increased significantly in the last few decades. Particularly, the cryptographic community has been widely working in the problem of generating BFs with good cryptographic properties. One of the first authors to study such topic showed the important implication of the feature known as *correlation immunity* using algebraic procedures [1]. CBFs are currently used in symmetric-key cryptography both in block and stream ciphers [2]. In both of these types of ciphers, the only nonlinear elements are usually the BFs applied in the stream ciphers and the vectorial BFs (better known as substitution boxes or s-boxes) in the block ciphers. Without the use of BFs in stream ciphers, or s-boxes in block ciphers, it would be trivial to break the cryptographic systems. As a result, designing proper BFs is a crucial step.

Several efforts have been devoted to finding proper CBFs to avoid cryptanalytic attacks [3]. Thus, cryptographers study the desired properties for CBFs. In particular, some desired features are high nonlinearity, balancedness, algebraic degree and low autocorrelation. Attaining balanced BFs with high nonlinearity is one of the most challenging tasks [3] and it is particularly important because they hinder the application of linear and differential cryptanalysis.

Currently, many ways to generate CBFs with proper features have been designed. The three main approaches that are currently used are [4] algebraic constructions, random generation and metaheuristic constructions. Algebraic constructions use mathematical procedures based on algebraic properties to create BFs with good cryptographic properties. Random generation is easy and fast, but the resulting BFs usually have suboptimal properties. Finally, metaheuristic constructions are heuristic techniques that have attained quite promising BFs and are easier to design than algebraic methods. However, up to now, metaheuristic techniques have not been able to attain so good results as the algebraic techniques. For instance, attending to the nonlinearity for balanced CBFs, metaheuristics attain CBFs with lower nonlinearity than the best-known CBFs that have been generated with algebraic constructions. Note that in practice, it is first necessary to have methods that allow obtaining BFs with high nonlinearity and then, as is the trend in the literature, compromise nonlinearity to increase resistance to DPA attacks, Side Channel, and so forth [5]. Alternatively, the different features might be tackled simultaneously with multi-objective optimizers (MO). In fact, in the related literature there exist different implementations following this principle [6]. However, in our work we focus only on the nonlinearity so single-objective optimizers are applied.

Regarding the generation of highly nonlinear CBFs, one of the main difficulties is that the search space $\mathcal{F}_n$ is immensely large: $|\mathcal{F}_n| = 2^{2^n}$. In fact, for problems with more than five variables, it is not possible to do an exhaustive search. Furthermore, as n increases, not only the search space grows, but also the computational cost of calculating the various important properties increases. Specifically the calculus of the nonlinearity is based on the usage of the Walsh Hadamard Transform (WHT), so there are more BFs and estimating the quality of each BF is more costly. This article focuses on the n-variable BF problem, with $n = 8, 10$. The main efforts are concentrated in further improving the highest nonlinearity achievable by metaheuristics for these n-variable BF problems. Note that some researchers consider that balanced BFs for 8 variables with nonlinearity equal to 118 do not exist, so the best-known BF might already be known [7]. This function has been searched for at least since 30 years ago but its existence has not been proved [8]. A similar situation appears for the case of 10-variable BFs with nonlinearity equal to 494 [7]. In any case, the main purpose of this research is to reduce the gap between algebraic methods and metaheuristic approaches [8]. Notice that some novel general purpose strategies are designed so the advances put forth in this paper might be applied in additional fields of optimization.

Diversity-aware population based metaheuristics have provided important benefits in recent years. Particularly, they have been able to generate new best-known solutions in several combinatorial optimization problem [9]. This paper studies the hypothesis that the state of the art in the generation of balanced CBFs with high nonlinearity can be improved further with proper diversity-aware memetic algorithms. To the best of our knowledge, our proposal is the first population-based approach that considers the diversity in an explicit way in order to generate balanced CBFs with high nonlinearity. The main novelty appears in the replacement phase, where concepts related to clustering and diversity are applied. Additionally, two new cost functions that guide the search towards proper regions are devised. Regarding the diversity management, two strategies that involve different ways of controlling the diversity are applied. The first one, which was successfully devised for the Graph Partitioning Problem [10], enforces a large contribution to diversity for every member of the population, whereas the second one, which is a novel proposal, considers the creation of clusters, meaning that some close members are maintained in the population. The lower entropy induced by the formation of clusters [11] allows the promotion of a larger degree of intensification, which is a key to success. Finally, a novel population initialization method is also proposed which considers some basic algebraic constructions with the aim of speeding up the attainment of high quality BFs. Concerning the results, we remark that our proposals reduce the gap between algebraic and heuristic constructions. In fact, with the methods proposed in this work, the results obtained by algebraic constructions for 8 and 10 variables have been matched.

This work is organized as follows. Section 2 formally defines our problem and introduces the main required concepts. Section 3 provides a review of metaheuristics applied to the generation of CBFs as well as other optimizers that guided our design. Trajectory-based and population-based proposals are described and analyzed is Sections 4 and 5, respectively. Parameterization studies and comparisons against the best-known solutions are included. Finally, conclusions and some lines of future work are given in Section 6.

2. Problem Statement: Formal Definitions

The problem addressed in this paper is the generation of balanced CBFs with high nonlinearity. This section formally defines the concepts and algorithms required to fully understand the problem.

2.1. Boolean Functions and Representations

The set $\{0, 1\}$ is the most often endowed with the structure of a field (denoted by $\mathbb{F}_2$) and the set $\mathbb{F}_2^n$ of all binary vectors of length n is viewed as an $\mathbb{F}_2$ vector space. The null vector of $\mathbb{F}_2^n$ is $\mathbf{0}$. $\mathbb{F}_2^n$ is endowed with a field structure to form the well-known Galois Field $\mathbf{GF}(2^n)$ [12]. Given these basic definitions, a n-variable BF is a function

$$f : \mathbb{F}_2^n \to \mathbb{F}_2. \tag{1}$$

It is usually written as $f(\mathbf{x}) = f(x_1, x_2, ..., x_n)$, where $\mathbf{x}$ is the shorthand writing of vector $(x_1, x_2, ..., x_n)$. Note that when considering n variables, there are $|\mathcal{F}_n| = 2^{2^n}$ different Boolean functions.

BFs can be represented in several ways. Some of the most typical are the algebraic forms, the *binary truth table* and the *polarity truth table*. The *binary truth table* is the lexicographically ordered *vector* of all outputs of the BF f. The binary truth table has length 2^n. The lexicographical order of the binary vectors $\mathbf{x}$ follows a particular order when going through all the possible values in $\mathbb{F}_2^n$. Let $k_{\mathbf{x}}$ denote the integer representation of $\mathbf{x} = (x_1, x_2, ..., x_n)$, that is,

$$k_{\mathbf{x}} = to\_int(\mathbf{x}) = \sum_{i=1}^{n} x_i 2^{n-i}. \tag{2}$$

We can see the function *to_int* as the function that maps a vector $\mathbf{x}$ from $\mathbb{F}_2^n$ to an integer $k_{\mathbf{x}} \in \mathbb{N}_n = \{0, \cdots, 2^n - 1\}$. When the integer $k_{\mathbf{x}}$ takes all the values from 0 incrementally to $2^n - 1$, then the corresponding binary vector $\mathbf{x}$ goes through all the elements in $\mathbb{F}_2^n$. This order is the lexicographical one. Linked to this representation is the *polarity truth table* or *polar form*. Let f denote a BF, then $\hat{f}$ is used to define the *polarity truth table*. In this case $\hat{f}(\mathbf{x}) \in \{-1, 1\}$ and each element of $\hat{f}$ is obtained as $\hat{f}(\mathbf{x}) = (-1)^{f(\mathbf{x})}$.

Affine BFs on $\mathcal{F}_n$ are those that can be expressed as $L_{\mathbf{w,c}}$:

$$L_{\mathbf{w},c}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \oplus c = w_1 x_1 \oplus \cdots \oplus w_n x_n \oplus c, \tag{3}$$

where $\mathbf{w} \in \mathbb{F}_2^n$ and $c \in \mathbb{F}_2$. If $c = 0$, then $L_{\mathbf{w},0}$ ($L_{\mathbf{w}}$) is a linear BF. The sets of n-variable affine and linear BFs are denoted by $\mathcal{A}_n$ and $\mathcal{L}_n$, respectively. They are important because they are involved in the calculus of the nonlinearity. The number of n-variable affine and linear BFs are

$$|\mathcal{A}_n| = 2^{n+1}, \tag{4}$$

$$|\mathcal{L}_n| = 2^n. \tag{5}$$

Additional important concepts are the *Hamming Weight* and the *Hamming Distance*. The Hamming Weight of a BF f is defined as the numbers of $1's$ in its truth table and is denoted by $w_H(f)$:

$$w_H(f) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x}). \tag{6}$$

This definition is also true for vectors $\mathbf{x} \in \mathbb{F}_2^n$.

Let $f(\mathbf{x})$, $g(\mathbf{x})$ be two n-variable BFs, the *Hamming distance* between $f(\mathbf{x})$ and $g(\mathbf{x})$, denoted by $d_H(f,g)$, is the number of coordinates with different values in their truth tables. It can be written as

$$d_H(f,g) = w_H(f \oplus g) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x}) \oplus g(\mathbf{x}). \tag{7}$$

2.2. Walsh Hadamard Transform

The *Walsh Hadamard Transform* (WHT) can be seen as another kind of n-variable BF representation that is useful to calculate relevant cryptographic properties of BFs, such as the nonlinearity. Let f be a n-variable BF, the WHT of f is the function $W_n : \mathbb{F}_2^n \to \mathbb{Z}$ defined by

$$W_n(f)(\mathbf{w}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus L_{\mathbf{w}}(\mathbf{x})}, \qquad \mathbf{w} \in \mathbb{F}_2^n. \tag{8}$$

WHT can also be calculated by using the polarity truth tables of f and $L_{\mathbf{w}}$

$$W_n(\hat{f})(\mathbf{w}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} \hat{f}(\mathbf{x}) \hat{L}_{\mathbf{w}}(\mathbf{x}), \qquad \mathbf{w} \in \mathbb{F}_2^n. \tag{9}$$

The WHT of f measures the correlation between f and each $L_{\mathbf{w}}$. According to Millan [3], the correlation with $L_{\mathbf{w}}$ is given by

$$c(f, L_{\mathbf{w}}) = \frac{|W_n(f)(\mathbf{w})|}{2^n}. \tag{10}$$

The correlation is a real number $0 \leq c(f, L_{\mathbf{w}}) \leq 1$, that represents the degree of similarity between f and $L_{\mathbf{w}}$. A correlation value equal to zero indicates that f and $L_{\mathbf{w}}$ are completely uncorrelated; a value equal to 1 means a perfect correlation between f and $L_{\mathbf{w}}$.

Note that direct calculation of W_n would require about 2^{2n} operations. This is because there exists 2^n linear functions, and computing the correlation with each linear function requires 2^n operations. Fortunately there is a faster way to obtain W_n, the Fast Walsh Hadamard Transform (FWHT), which is a discrete version of the so-called Fast Fourier Transform (FFT). Since our optimizer requires the calculation of the WHT, the FWHT is used.

Parseval's Equation

The values in the WHT of f are constrained by a square sum relationship which implicitly limits the magnitude and frequency of those values, this is known as Parseval's Equation (11) [13]:

$$\sum_{\mathbf{w} \in \mathbb{F}_2^n} (W_n(f)(\mathbf{w}))^2 = 2^{2n}. \tag{11}$$

This value is constant for all n-variable BFs, that is, the values in the WHT of every BF must satisfy Parseval's equation. However, a WHT of a function can satisfy Parseval's equation and not necessarily be Boolean. For this reason, even if the WHT is required, optimization methods usually operate with encoding based on truth tables.

2.3. Cryptographic Properties of Boolean Functions and Special Functions

Boolean functions used in stream ciphers must have some required properties with the aim of making ciphers secure against some attacks. Some of the most important cryptographic properties of BFs are the algebraic degree, the balancedness, the nonlinearity and the auto correlation. In this work we consider the balancedness and nonlinearity.

The balancedness is one of the most basic of all cryptographic properties desired to be exhibited by BFs. f is said to be balanced if $w_H(f) = 2^{n-1}$. In terms of its WHT, a BF f is balanced if and only if

$$W_n(f)(\mathbf{0}) = 0. \tag{12}$$

The set of n-variable balanced BFs is denoted as $\mathcal{B}_n$ and its size is

$$|\mathcal{B}_n| = \binom{2^n}{2^{n-1}}. \tag{13}$$

It is important to remark that we use this search space along this paper, that is, our optimizer does not generate unbalanced BFs.

The nonlinearity $N_n(f)$ of f is calculated using the maximum absolute value of the WHT and represents the minimum Hamming distance between f and the affine BFs set $\mathcal{A}_n$, that is,

$$N_n(f) = \min_{A_{\mathbf{w},\mathbf{c}} \in \mathcal{A}_n} d_H\left(f, A_{\mathbf{w},\mathbf{c}}\right). \tag{14}$$

Mathematically, the relationship between the nonlinearity of a n-variable BF f and the WHT of f is given by the following equation:

$$N_n(f) = \frac{1}{2}\left(2^n - \max_{\mathbf{w}\in\mathbb{F}_2^n} |W_n(f)(\mathbf{w})|\right), \tag{15}$$

where $\max_{\mathbf{w}\in\mathbb{F}_2^n} |W_n(f)(\mathbf{w})|$ represents the maximum absolute value in the WHT. By Parseval's equation, we can obtain an upper bound of nonlinearity in the general case (when n is even), that is,

$$N_n(f) \leq 2^{n-1} - 2^{n/2-1}. \tag{16}$$

Bent BFs are a very special class of BFs. A BF $f(\mathbf{x})$ is bent if and only if

$$|W_n(f)(\mathbf{w})| = 2^{n/2},\ \forall \mathbf{w} \in \mathbb{F}_2^n. \tag{17}$$

Bent BFs are not balanced and they are not applied usually in cryptosystems. It can be noticed that bent BFs only exist when n is even. While they are not applied directly in our proposal, they inspired one of the methods applied to initialize the population in this paper.

3. Literature Review

This section reviews related papers on the application of metaheuristics to the generation of CBFs and some recent advances in population-based metaheuristics that guided the design of our proposals.

At least since two decades ago, the firsts researchers started to study the design of CBFs from the metaheuristics point of view. The first *Genetic Algorithm* (GA) that tries to maximize the nonlinearity of BFs was proposed in Reference [14]. The importance of including an intensification stage was shown by applying a smart version of a hill climbing method [3]. The superiority of memetic algorithms against pure GAs was shown. Subsequently, a novel crossover operator to find BFs with high nonlinearity was proposed [15]. This operator combines two balanced CBFs and produces a single balanced CBF.

More recently, it was noted that an important step in the application of metaheuristics for the generation of CBFs is the design of proper fitness or cost functions [16]. This last study was developed by applying *Simulated Annealing*. Subsequently, the proposal was extended by designing a more sophisticated search technique of Simulated Annealing called "vanilla" [17] with the aim of obtaining CBFs with some additional properties such as resilience. Authors note that the huge search space hinders the attainment of better results. Obviously, this is not a drawback specific to this technique. In order to partially avoid this drawback, some authors consider some of the advances performed with algebraic constructions. Particularly, References [18,19] proposed trajectory-based search methods that operate with *bent BFs*. They randomly adjust the bent BF to convert it on a balanced BF. In this way, they decrease the nonlinearity of a bent BF instead of increasing the nonlinearity of a randomly created BF. First, a bent BF is constructed with the method given in Reference [20]. Then, a trajectory-based heuristic is used to attain a balanced BF. These methods show really good results although not as good as more complex algebraic approaches. In fact, at the starting point of our research, these methods were the state-of-the-art in the generation of BFs with high nonlinearity by applying metaheuristics. Given the promising behavior of this method, one of the optimizers put forth in this paper applies a simple algebraic techniques in the generation of the initial population. A different alternative to deal with the so large search space in the case of maximizing the nonlinearity for balanced BFs, is to take into account the symmetries that appear in the fitness landscape when using the bit string representation [21].

Some other recent works [22] include comparisons among different metaheuristics such as GAs, Evolution Strategies (ESs) and Cartesian Genetic Programming (CGP). Among the tested approaches, CGP was the best one to generate 8-variable CBFs with high nonlinearity. In fact, it is the EA with best performance among those that do not apply the concept of bent BFs. Another contribution of this paper is to compare three different fitness functions with the aim of increasing the nonlinearity.

Finally, some notions on the difficulty of generating balanced BFs with high nonlinearity are discussed in Reference [23]. This last study is developed using *Estimation of Distribution Algorithms* (EDAs). They note some undesired properties of using the nonlinearity as fitness function.

Regarding the use of multi-objective optimization algorithms, there are important works including several metaheuristics. For example in Reference [6] the authors propose the use of a MO Genetic Programming algorithm. They used three objective functions that must be maximized: the nonlinearity, algebraic degree, and correlation immunity. In the same context, Non-dominated sorting genetic algorithm II (NSGA-II) was also implemented to construct cryptographically strong Boolean functions [24] by optimizing nonlinearity, resiliency and autocorrelation. Authors remark that the use of MO increased the computational cost.

Recent advances in metaheuristics have shown that in many combinatorial optimization problems, results are advanced further with diversity-aware memetic algorithms [25]. Particularly, relating the diversity maintained in the population to the stopping criterion and elapsed period with the aim of balancing the search from exploration to exploitation as the evolution progresses seems very promising [26]. This principle has not been used in current state-of-the-art metaheuristics for CBFs. Taking into account the difference among the best-known BFs and the ones attained by state-of-the-art metaheuristics the main aim of this paper is to advance further the development of metaheuristic construction techniques. Particularly, the paper focuses on the development of novel memetic algorithms incorporating explicit control of diversity that follow the aforementioned design principle.

4. Trajectory-Based Proposals

Since memetic algorithms attain BFs with higher nonlinearity than pure GAs [3], our diversity-aware population-based proposals are memetic approaches, meaning that a trajectory-based strategy is applied to promote intensification. One of our first steps in this research was designing and analyzing different trajectory-based proposals. This section is devoted to fully detail our trajectory-based proposal. In order to fully define our strategy, the representation of solutions, the applied cost function and the details regarding the neighborhood and selection approaches are given.

4.1. Representation of Solutions

In this paper each solution $\mathcal{S}$ is encoded using a binary representation composed of a vector with 2^n Boolean decision variables $\mathcal{S}[k]$, where n is the number of variables of the BF f and $k \in \mathcal{N}_n$. In order to introduce the relationship between the solution $\mathcal{S}$ and f, let us introduce the analogous function that maps an integer k to a binary vector $\mathbf{x} \in \mathbb{F}_2^n$, thus, $to\_bin : \mathbb{N} \to \mathbb{F}_2^n$, such that

$$\mathbf{x} = (x_1, \ldots, x_n) = to\_bin(k) = \left(\left\lfloor \frac{k}{2^{n-1}} \right\rfloor \%2, \ldots, \left\lfloor \frac{k}{2^{n-n}} \right\rfloor \%2 \right). \tag{18}$$

Now, we are ready to introduce the relationship between the solution $\mathcal{S}$ and f. Each decision variable $\mathcal{S}[k]$ is set equal to the corresponding truth table value $f(\mathbf{x})$, where $k = to\_int(\mathbf{x}), \mathbf{x} \in \mathbb{F}_2^n$ (or $\mathbf{x} = to\_bin(k), k \in \mathcal{N}_n$)

$$f(\mathbf{x}) = \mathcal{S}[k]. \tag{19}$$

Note that both our trajectory-based and population-based proposals operate with balanced BFs, so 2^{n-1} decision variables are set to 0 and 2^{n-1} decision variables are set to 1. In order to facilitate some descriptions, let us define the positions sets $\mathcal{P}_0$ and $\mathcal{P}_1$ as the set of positions whose values are 0 and 1, respectively. The sets $\mathcal{P}_0$ and $\mathcal{P}_1$ have the following properties:

$$|\mathcal{P}_0| = |\mathcal{P}_1| = 2^{n-1}, \quad \mathcal{P}_0 \cap \mathcal{P}_1 = \emptyset, \quad \mathcal{P}_0 \cup \mathcal{P}_1 = \mathcal{N}_n.$$

Each solution has associated its $\mathcal{P}_0$, $\mathcal{P}_1$ sets and an integer vector $\mathcal{W}$ of length 2^n which denotes the WHT of the BF f. Analogously to the equality (19) we have that the following holds for a vector $\mathcal{W}$ denoting the WHT of the BF f

$$W_n(f)(\mathbf{w}) = \mathcal{W}[k] \quad \text{s.t} \quad k \in \mathcal{N}_n, \quad \mathbf{w} = to\_bin(k), \tag{20}$$

where $\mathcal{W}[k]$ denotes an integer value and $W_n(f)(\mathbf{w})$ the WHT value associated to the linear function $L_{\mathbf{w}}$, which is also an integer.

4.2. Neighborhood

In order to define the neighborhood employed in this paper it is necessary to define an operator that transforms a solution $\mathcal{S}$ into another one $\mathcal{S}'$. We call this operator *Bit Swapping* (BS). BS receives a solution $\mathcal{S}$ and two positions p_0 and p_1 as inputs and returns another solution $\mathcal{S}'$, such that the decision variables $\mathcal{S}[p_0]$ and $\mathcal{S}[p_1]$ are exchanged (or complemented) and $d_H(\mathcal{S}, \mathcal{S}') = 2$, that is,

$$\mathcal{S}'[k] = \begin{cases} 1 \oplus \mathcal{S}[k] & \text{if } k \in \{p_0, p_1\} \\ \mathcal{S}[k] & \text{if } k \in \mathcal{N}_n \setminus \{p_0, p_1\}. \end{cases} \tag{21}$$

The Bit Swapping operator is defined as follows

$$BS(\mathcal{S}, p_0, p_1) := swap(\mathcal{S}[p_0], \mathcal{S}[p_1]). \tag{22}$$

Employing BS we can construct the full neighborhood for a single solution $\mathcal{S}$ as

$$N(\mathcal{S}) = \{BS(\mathcal{S}, p_0, p_1) : p_0 \in \mathcal{S}.\mathcal{P}_0 \wedge p_1 \in \mathcal{S}.\mathcal{P}_1\}. \tag{23}$$

The total amount of bit-swapping that can be performed is the neighborhood size

$$|N(\mathcal{S})| = |\mathcal{S}.\mathcal{P}_0| \times |\mathcal{S}.\mathcal{P}_1| = 2^{n-1} \times 2^{n-1} = 2^{2n-2}. \tag{24}$$

4.3. Cost Functions

In order to properly measure the performance of the algorithms developed in this paper, it is important to remark that our aim is to generate balanced BFs with high nonlinearity, that is, our objective function is the nonlinearity

$$\mathbf{F_1}(\mathcal{S}) = \frac{1}{2}\left(2^n - \max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|\right). \tag{25}$$

We can analyze some properties of this function by taking into account that the values in the WHT $\mathcal{S}.\mathcal{W}$ are constrained by the Parseval's equation. We know that for a balanced BF all the values in its WHT are multiples of 4. This has as a consequence that the amount of different values that could take $\max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|$ is really small so the different values of $\mathbf{F_1}$ are also small. This relationship between $\max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|$ and $\mathbf{F_1}$ is clearer if we transform the original objective function (maximize the nonlinearity) into another one (minimize the maximum value in the WHT):

$$\mathbf{F_2}(\mathcal{S}) = \max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|. \tag{26}$$

Similarly to other researchers [27], we realized that using the objective function as the fitness function is not suitable to guide the search in EAs and it is necessary to define a new fitness function (in our case a cost function). One of the drawbacks seen easily from the objective function $\mathbf{F_1}$ or $\mathbf{F_2}$ is that it just takes a small amount of different values, so a lot of BFs are considered to be of equal value. However, even if they share the same nonlinearity, some of them could be used to easily improve the nonlinearity by just making some modifications, while another ones could be far from better solutions.

The most popular cost function—since it is a cost function it must be minimized—used with metaheuristics is the objective function $\mathbf{F_2}$, that will be denoted as C_1 in the following:

$$C_1(\mathcal{S}) = \mathbf{F_2}(\mathcal{S}) = \max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|. \tag{27}$$

Also, the cost function designed by Clark [27] is widely used,

$$C_2(\mathcal{S}) = \sum_{k=0}^{2^n-1} ||\mathcal{S}.\mathcal{W}[k]| - \mathcal{X}|^R. \tag{28}$$

When using C_2 every value in the WHT influences the cost function of S, rather than just the maximum one as in the cost function C_1 . The authors note "the parameters $\mathcal{X}$ and R provide freedom to experiment". Similarly to the authors, we use $R = 3$ and for the $\mathcal{X}$, the expected maximum value for the WHT when it achieves the nonlinearity upper bound.

One of the contributions of this paper is the definition of new cost functions. We propose two new cost functions in order to address the problem properly. In order to explain our cost function, lets denote the absolute values appearing in the WHT as $X_1, X_2, \cdots, X_\alpha$, such that

$$X_1 = 0 < X_2 < \cdots < X_\alpha = \max_{k\in\mathcal{N}_n} |\mathcal{S}.\mathcal{W}[k]|.$$

In addition, η_i is used to denote the amount of times that the X_i or $-X_i$ value appears in the WHT. Note that $X_1 = 0$, since $\mathcal{S}.\mathcal{W}[0] = 0$ and $\eta_1 > 0$. Based on this, we propose two new cost functions. In both we form a tuple of values

$$C_3(\mathcal{S}) = (\mathbf{F}_2(\mathcal{S}), \xi_1(\mathcal{S})), \tag{29}$$

$$C_4(\mathcal{S}) = (\mathbf{F}_2(\mathcal{S}), \xi_2(\mathcal{S})), \tag{30}$$

where the first component is the objective function $\mathbf{F}_2$ and the second components $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$—which are the novelty—help us to discern between solutions with the same nonlinearity. The second members ξ_1, ξ_2 of such cost functions are the following:

1. In the second value of C_3 we try to minimize the maximum absolute value X_α and the second maximum absolute value $X_{\alpha-1}$ in the WHT. In order to attain this aim, it takes also into account the number of appearances of these values (η_α and $\eta_{\alpha-1}$) in the WHT. This is implemented as follows:
$$\xi_1(\mathcal{S}) = (\eta_\alpha \times X_\alpha)^3 + (\eta_{\alpha-1} \times X_{\alpha-1}). \tag{31}$$
2. Let X_k be a value such that $X_{k-1} \le \mathcal{X} < X_k$, with $\mathcal{X}$ defined as in Reference [12]. In the second value of the cost function, we try to minimize the appearance of entries with absolute values greater than $\mathcal{X}$ assigning larger penalties to larger values:
$$\xi_2(\mathcal{S}) = \sum_{i=k}^{\alpha} (2 \times \eta_i \times X_i)^{i-k+1}. \tag{32}$$

In order to employ the cost functions previously defined, we define the new operators $\prec$ and $\preceq$ to identify if a solution $\mathcal{S}'$ is better than a solution $\mathcal{S}$ ($\mathcal{S}' \prec \mathcal{S}$) or if a solution $\mathcal{S}'$ is better or equal than a solution $\mathcal{S}$ ($\mathcal{S}' \preceq \mathcal{S}$). It used the lexicographic comparison. For example, for C_3 , we have

$$\mathcal{S}' \prec \mathcal{S} : \mathbf{F_2}(\mathcal{S}') < \mathbf{F_2}(\mathcal{S}) \vee \left(\mathbf{F_2}(\mathcal{S}') = \mathbf{F_2}(\mathcal{S}) \wedge \xi_1(\mathcal{S}') < \xi_1(\mathcal{S})\right), \tag{33}$$
$$\mathcal{S}' \preceq \mathcal{S} : \mathbf{F_2}(\mathcal{S}') < \mathbf{F_2}(\mathcal{S}) \vee \left(\mathbf{F_2}(\mathcal{S}') = \mathbf{F_2}(\mathcal{S}) \wedge \xi_1(\mathcal{S}') \le \xi_1(\mathcal{S})\right). \tag{34}$$

4.4. Full Hill Climbing

The most straightforward trajectory-based approach is full hill climbing (FHC) [28]. In hill climbing the neighborhood is completely explored and only improvement movements are accepted. There are several strategies to select among the neighbors that improve the current solution. In stochastic hill climbing, which is the variant that we applied, it is selected randomly. Algorithm 1 illustrates the working operation of FHC. This algorithm makes use of the neighborhood (23). FHC ensures that all the neighbors of a solution S are visited just once. First one decision variable set equal to 0 is selected at random (line 5) and it is swapped with all the decision variables set equal to 1 (line 6). Then if any of the neighbors generated is better than the current solution, this is selected to replace the current solution S (line 9) and both for cycles are restarted. If no better neighbor is found and the neighborhood is fully visited, the solution is a local optimum and the algorithm stops (line 16). Another stopping criterion that might be used is the elapsed time (line 12) returning the best solution found so far.

In order to perform a comparison among the four cost functions previously detailed, they were integrated in FHC and tested for $n = 8, 10$ and it was executed 50 times with different seeds. FHC was executed until a local optimum was reached. Table 1 summarizes the results obtained for 50 independent runs. Specifically, for each cost function the minimum, maximum, mean, median and standard deviation of the attained nonlinearity is reported. Additionally, the mean execution time is shown. The best results obtained for each n are shown in boldface. We can see that the cost function C3 attains the best mean nonlinearity in both cases. If we look at the time column, we see that the cost function C3 employs less execution time to achieve a local optimum than the cost function C2.

Algorithm 1: Full Hill Climbing (FHC).

```
Input  :Initial solution $\mathcal{S}$, $T_{end}$
1  $T_{elapsed} = 0$ ;
2  LOF= False ;                                        /* Local Optimum Found */
3  while LOF= False do
       /* Check all the neighbors $\mathcal{S}'$ by fixing $p_0$ at random and swap with the
          whole set $\mathcal{P}_1$ randomly */
4      BSF= False ;                                    /* Best Solution Found */
5      foreach $p_0 \in \mathcal{S}.\mathcal{P}_0 \wedge$ BSF= False do
6          foreach $p_1 \in \mathcal{S}.\mathcal{P}_1 \wedge$ BSF= False do
7              $\mathcal{S}' = BS(\mathcal{S}, p_0, p_1)$ ;    /* Select the current $\mathcal{S}' \in N(\mathcal{S})$ */
8              if $\mathcal{S}' \prec \mathcal{S}$ then
9                  $\mathcal{S} = \mathcal{S}'$ ;
10                 BSF= True;
11             Update($T_{elapsed}$) ;
12             if $T_{elapsed} > T_{end}$ then
13                 BSF= True ;
14                 LOF= True ;
       /* If all the neighbor solutions $\mathcal{S}'$ are visited and BSF= False the
          solution $\mathcal{S}$ is a local optimum */
15     if BSF= False then
16         LOF= True ;
Output:Local Optimum $\mathcal{S}$
```

Table 1. Summary of the results attained by FHC applying various cost functions.

n	CF	Min	Max	Mean	Median	σ_{N_n}	$\bar{t}(s)$
8	C_1	108	112	110.16	110	1.503	3.4×10^{-2}
	C_2	112	116	114.12	114	0.627	1.6×10^{0}
	$\mathbf{C_3}$	**114**	**116**	**115.92**	**116**	**0.396**	2.4×10^{-1}
	C_4	108	112	111.84	112	0.792	3.1×10^{-2}
10	C_1	464	472	469.4	470	1.863	7.9×10^{-1}
	C_2	482	484	482.72	482	0.97	1.8×10^{2}
	$\mathbf{C_3}$	**480**	**484**	**483.16**	**484**	**1.621**	1.6×10^{1}
	C_4	468	480	475.44	476	1.809	1.9×10^{0}

4.5. First Improvement Quasi-Tabu Search

FHC with the defined neighborhood is computationally too expensive to be included in a memetic algorithm. Thus, we decided to design an alternative trajectory-based strategy. Tabu Search is a very popular trajectory-based strategy. Two of its main keys are to move to the best neighbor at each iteration and to use a tabu list to avoid cycles. In our case, iterating to get the best neighbor is computationally expensive but applying a tabu list with the aim of reducing the neighborhood seems promising. As a result, a novel method that applies some of the principles of Tabu Search was devised. The strategy is called First Improvement Quasi-Tabu Search (FIQTS) and differently to Tabu Search it moves to the first neighbor that improves the current solution.

Algorithm 2 illustrates the working operation of FIQTS. First, the tabu bit-swapping positions $\mathcal{R}_0$ and $\mathcal{R}_1$ are initialized at random (line 2), $\mathcal{R}_0$ and $\mathcal{R}_1$ contain positions that are not allowed to bit-swap. Then the non-tabu bit-swapping positions $\mathcal{Q}_0$ and $\mathcal{Q}_1$ are initialized (line 3), such that

$$\mathcal{Q}_i \cup \mathcal{R}_i = \mathcal{S}.\mathcal{P}_i \qquad \mathcal{Q}_i \cap \mathcal{R}_i = \emptyset \quad i \in \{0,1\}.$$

Algorithm 2: First Improvement Quasi-Tabu Search (FIQTS).

```
   Input   :Initial solution $\mathcal{S}$, $T_{end}$
1  $T_{elapsed} = 0$ ;
   /* Tabu bit-swapping positions initialized randomly                    */
2  Set $\mathcal{R}_i = \{p_k : p_k \in \mathcal{S}.\mathcal{P}_i, k = 1, \cdots, 2^{n-4}\}, i \in \{0,1\}$ ;
   /* Non-tabu bit-swapping positions                                     */
3  $\mathcal{Q}_i = \mathcal{S}.\mathcal{P}_i \setminus \mathcal{R}_i, i \in \{0,1\}$ ;
   /* Construct the reduced neighborhood.  It is automatically updated when $\mathcal{S}$,
      $\mathcal{Q}_0$ or $\mathcal{Q}_1$ changes                                      */
4  $N_r(\mathcal{S}) = \{BS(\mathcal{S}, q_0, q_1) : q_0 \in \mathcal{Q}_0, q_1 \in \mathcal{Q}_1\}$ ;
5  Set $N_v(\mathcal{S}) = \emptyset$ ;                        /* Neighbors already visited  */
6  Set $Repeats = 0$ ;                          /* Repetitions of neighbors  */
7  LOF= False ;                                   /* Local Optimum Found  */
8  while LOF= False do
       /* Select a random neighbor $\mathcal{S}' \in N_r(\mathcal{S})$                          */
9      $\mathcal{S}' = BS(\mathcal{S}, q_0, q_1), \quad q_0 \in \mathcal{Q}_0, q_1 \in \mathcal{Q}_1$ ;
       /* Exchange the oldest element $r_{i_{old}} \in \mathcal{R}_i$ with $q_i \in \mathcal{Q}_i$               */
10     $swap(q_i \in \mathcal{Q}_i, r_{i_{old}} \in \mathcal{R}_i), \quad i \in \{0,1\}$ ;
       /* Check if the neighbor has not been visited                       */
11     if $N_v(\mathcal{S}) \cap \{\mathcal{S}'\} = \emptyset$ then
12         $N_v(\mathcal{S}) = N_v(\mathcal{S}) \cup \{\mathcal{S}'\}$;
13         if $\mathcal{S}' \prec \mathcal{S}$ then
14             $\mathcal{S} = \mathcal{S}'$ ;
15             $N_v(\mathcal{S}) = \emptyset$;
16             $Repeats = 0$;
17     else
18         $Repeats++$;
       /* Stopping criterion                                               */
19     if $N_v(\mathcal{S}) = N(\mathcal{S})$ or $Repeats \geq 2|N_r(\mathcal{S})|$ then
20         LOF = True ;
21     Update($T_{elapsed}$) ;
22     if $T_{elapsed} > T_{end}$ then
23         LOF =True ;
   Output:Best solution found $\mathcal{S}$
```

The sets $\mathcal{Q}_0$ and $\mathcal{Q}_1$ contain positions such that the decision variables $\mathcal{S}[q_0]$, $\mathcal{S}[q_1]$ are set equal to 0, 1 respectively, and that might be bit-swapped. In base of these sets the reduced neighborhood $N_r(\mathcal{S})$ is constructed (line 4). Each time that a neighbor is created by bit-swapping the decision variables in the positions q_0 and q_1, such elements are moved to $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively (lines 9, 10). Additionally, the elements that have been for a longer time in $\mathcal{R}_0$ and $\mathcal{R}_1$ are inserted in $\mathcal{Q}_0$ and $\mathcal{Q}_1$. The main principle is to promote the selection of different decision variables at each step when performing the bit-swapping. The stopping criterion is set in such a way that the iterative process ends when all the neighborhood has been visited and no better solution was found or when the random sampling process generates too many neighbors that have already been visited previously (line 19). Another stopping criterion that can be used is time (line 22).

As in the FHC method, FIQTS method was executed with the four cost functions. Table 2 shows the results obtained with FIQTS for n = 8, 10. As in the previous case, the cost function C3 attains a better performance. The FIQTS method requires less iterations than the FHC method to reach high-quality solutions, so it is faster to locate local optima.

Table 2. Summary of the results attained by FIQTS applying various cost functions.

n	CF	Min	Max	Mean	Median	σ_{N_n}	$\bar{t}(s)$
8	C_1	108	112	110.16	110	1.448	3.8×10^{-2}
	C_2	112	116	114	114	0.7	1.1×10^{0}
	$\mathbf{C_3}$	**114**	**116**	**115.96**	**116**	**0.283**	1.4×10^{-1}
	C_4	108	116	112.6	112	1.629	3.7×10^{-2}
10	C_1	466	472	470.16	470	1.888	6.8×10^{-1}
	C_2	482	484	482.68	482	0.957	9.4×10^{1}
	$\mathbf{C_3}$	**480**	**484**	**483.64**	**484**	**1.12**	1.0×10^{1}
	C_4	472	480	478.48	480	2.27	3.4×10^{-1}

According to the results in Tables 1 and 2, the cost function C_3 is the most adequate cost function for our optimizers. Thus, the cost function C_3 is chosen to be used from now on. Moreover, it is noticeable that not only the FIQTS method is slightly better in terms of quality, but it also needs less time. In order to select the method to be integrated in our population-based strategy, it is important to select an inexpensive one with the aim of evolving more generations. In order to decide which method is the most suitable to use in the rest of experiments, we compare the performance of the FHC and FIQTS methods with the cost function C_3, employing a fixed time of 0.5 and 5.0 s for n equal to 8 and 10, respectively. Table 3 shows the results obtained for 50 independent runs for each method. Additionally, Figure 1 shows the evolution of the nonlinearity for the FHC and FIQTS methods with the cost function C_3. The FIQTS method quickly achieves high nonlinearity, so it is preferable for short-term executions, meaning that this is the method selected to be integrated in our population-based approach.

We would like to note that we also designed trajectory-based strategies based on simulated annealing and iterated local search. However, they were not superior to FIQTS.

Table 3. Comparison between FIQTS and FHC with the cost function C_3 in executions at fixed time.

n	Method	Min	Max	Mean	Median	σ_{N_n}
8	**FIQTS**	**114**	**116**	**115.96**	**116**	**0.283**
	FHC	114	116	115.92	116	0.396
10	**FIQTS**	**480**	**484**	**481.52**	**482**	**1.182**
	FHC	480	482	480.08	480	0.396

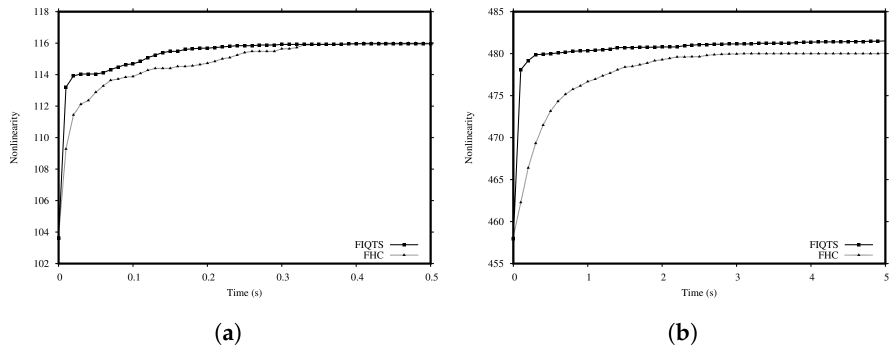


Figure 1. Evolution of the Mean Nonlinearity for FIQTS and FHC. (**a**) 8-variables, (**b**) 10-variables.

5. Population-Based Proposals

A very successful way to improve the performance of Evolutionary Algorithms (EAs) is to hybridize with local search or other trajectory-based techniques. In fact, Memetic Algorithms (MAs) [29] that combine genetic algorithms with an individual improvement technique, have been applied successfully to many combinatorial optimization problems. Taking into account the integration between the population-based strategy and the refinement scheme we can classify the MAs in two groups [30]:

- **Lamarckian Memetic Algorithm** (LMA): modifications performed in the individual improvement procedure are written back in every individual representation.
- **Baldwin Memetic Algorithm** (BMA): modifications change the fitness of the individuals without altering its representation.

In our case, we apply a LMA (see Algorithm 3). In order to fully define the LMA the following components have to be defined: *initialization*, *fitness or cost function* for the evaluation step, *mating selection*, *crossover* and *mutation* for the reproduction step, *survivor selection* and *individual improvement*. In the following, we describe each of the different components for our first basic variant. Then, some components that extend them to generate a more effective strategy are presented.

Algorithm 3: Lamarckian Memetic Algorithm (LMA).

```
1  Initialize($P_0$)
2  Evaluate($P_0$)
3  Improvement($P_0$)
4  $t = 0$
5  while not stopping criterion do
6      $P'_t$ = Mating Selection($P_t$)
7      $P'_t$ = Reproduction($P'_t$)
8      Evaluate($P'_t$)
9      $P'_t$ = Improvement($P'_t$)
10     $P_{t+1}$ = Replacement ($P_t, P'_t$)
11     $t = t + 1$
```

5.1. A Lamarckian Memetic Algorithm with a Generational Replacement with Elitism (LMA-GRE)

Our first variant (LMA-GRE) does not consider an explicit control of diversity. In the following the details of its different components are given.

In this first variant, the initial population P_0 is generated at random. Particularly, N balanced BFs are generated following a uniform distribution, that is, each balanced BF is equiprobable. Regarding the evaluation, for each individual the WHT is calculated and then the cost function C_3 is used to compare individuals. The mating selection is performed with the well-known binary tournament [28]. After the mating selection, the offspring population is built. Two different variation operators that maintain the balance of BFs are applied: *crossover* and *mutation*.

Algorithm 4 describes the working operation of the crossover operator. It is a novel—although quite straightforward—operator. It takes as input two individuals ($\mathcal{I}_j, \mathcal{I}_k$) and generates two new individuals ($\mathcal{C}_j$, $\mathcal{C}_k$). First, it calculates the positions where both individuals contain a one in the truth table (line 3–4). These positions are preserved in the offspring, that is, they are set to one (lines 17–18). Additionally, we calculate the positions that are set to one in only one of the individuals (line 5–6). They are saved in the $\mathcal{R}_{1_j}$ and $\mathcal{R}_{1_k}$ sets. Then, until these sets are empty (line 7–16), we select one value randomly for each of the sets. Then, these positions are inherited by the corresponding children (lines 10–11) or exchanged with probability p_c (lines 13–14). Note that at each step, the selected positions are removed from the corresponding sets (lines 15–16), and that the remaining positions are set to zero (lines 1–2), meaning that balancedness is maintained.

Algorithm 4: Crossover strategy applied in all our optimizers.

```
   Input  :Parents $\mathcal{I}_j, \mathcal{I}_k$, cross probability $p_c$
1  Set $\mathcal{C}_j[i] = 0 \quad \forall i \in \mathcal{N}_n$;
2  Set $\mathcal{C}_k[i] = 0 \quad \forall i \in \mathcal{N}_n$;
   /* With $\mathcal{I}_j, \mathcal{I}_k$ generate the offspring $\mathcal{C}_j, \mathcal{C}_k$                              */
   /* First take the intersection of their $\mathcal{P}_1$ sets                             */
3  $\mathcal{Q}_{1_j} = \mathcal{I}_j.\mathcal{P}_1 \cap \mathcal{I}_k.\mathcal{P}_1$ ;
4  $\mathcal{Q}_{1_k} = \mathcal{I}_j.\mathcal{P}_1 \cap \mathcal{I}_k.\mathcal{P}_1$;
   /* Take the differences                                                  */
5  $\mathcal{R}_{1_j} = \mathcal{I}_j.\mathcal{P}_1 \setminus \mathcal{Q}_{1_j}$ ;
6  $\mathcal{R}_{1_k} = \mathcal{I}_k.\mathcal{P}_1 \setminus \mathcal{Q}_{1_k}$ ;
   /* Cross the elements from $\mathcal{R}_{1_j}$ with $\mathcal{R}_{1_k}$                                */
7  while $\mathcal{R}_{1_k} \neq \emptyset$ do
8      Select randomly $r_j \in \mathcal{R}_{1_j}$ and $r_k \in \mathcal{R}_{1_k}$;
9      if $\mathcal{U}(0,1) < p_c$ then
10         $\mathcal{Q}_{1_j} = \mathcal{Q}_{1_j} \cup \{r_j\}$;
11         $\mathcal{Q}_{1_k} = \mathcal{Q}_{1_k} \cup \{r_k\}$;
12     else
13         $\mathcal{Q}_{1_j} = \mathcal{Q}_{1_j} \cup \{r_k\}$;
14         $\mathcal{Q}_{1_k} = \mathcal{Q}_{1_k} \cup \{r_j\}$;
15     $\mathcal{R}_{1_j} = \mathcal{R}_{1_j} \setminus \{r_j\}$;
16     $\mathcal{R}_{1_k} = \mathcal{R}_{1_k} \setminus \{r_k\}$ ;
   /* With $\mathcal{Q}_{1_j}$ and $\mathcal{Q}_{1_k}$ build the offspring $\mathcal{C}_j, \mathcal{C}_k$                     */
17 Set $\mathcal{C}_j[i] = 1 \quad \forall i \in \mathcal{Q}_{1_j}$;
18 Set $\mathcal{C}_k[i] = 1 \quad \forall i \in \mathcal{Q}_{1_k}$;
   Output:Offsprings $\mathcal{C}_j, \mathcal{C}_k$
```

It is important to note that when parents are close to each other, that is, their differences are not large, then the crossover is not very destructive. In particular, if both parents are the same individual no changes are performed. This automatic adaptive behavior is usually appropriate in the design of crossover operators [28].

Regarding the mutation, the applied operator is described in Algorithm 5. The process iterates over all positions set to one and with a given probability (p_m) it applies the BS operator to exchange this position with a position that contains a zero value. This second position is selected at random.

Algorithm 5: Mutation strategy applied in all our optimizers.

```
   Input  :Individual $\mathcal{I}$, mutation probability $p_m$
1  foreach $p_1 \in \mathcal{I}.\mathcal{P}_1$ do
2      if $\mathcal{U}(0,1) < p_m$ then
           /* Select $p_0 \in \mathcal{I}.\mathcal{P}_0$ at random to perform a bit-swapping          */
3          $BS(\mathcal{I}, p_0, p_1)$ ;
   Output:Individual mutated $\mathcal{I}$
```

The way in which the mutation and crossover operators are applied is quite standard. Algorithm 6 illustrates the creation of the offspring population by employing the crossover and mutation operators. Note that at each generation N offspring are created.

Algorithm 6: Reproduction strategy applied in all our optimizers.

```
    Input  : Parents $P$, cross probability $p_c$, mutation probability $p_m$
    /* Generate two individuals at each iteration                    */
  1 for $i = 0; i < N; i+ = 2$ do
  2     Select orderly $\mathcal{I}_i, \mathcal{I}_{i+1}$ ;
        /* Perform crossover                                         */
  3     $\mathcal{I}'_i, \mathcal{I}'_{i+1} = Crossover(\mathcal{I}_i, \mathcal{I}_{i+1}, p_c)$;
        /* Perform mutation to each offspring                        */
  4     $\mathcal{I}'_i = Mutation(\mathcal{I}'_i, p_m)$ ;
  5     $\mathcal{I}'_{i+1} = Mutation(\mathcal{I}'_{i+1}, p_m)$ ;
    Output: Offspring population $P'$
```

In pure generational schemes, the offspring are the survivors [31]. Since elitism usually contributes to the attainment of high-quality solution some Generational Replacement with Elitism (GRE) variants exist [31]. In our proposal (see Algorithm 7) when the best solution of the previous generation is better than the best offspring, the best solution of the previous generation replaces a randomly selected offspring.

Algorithm 7: Generational Replacement with Elitism (GRE) Technique.

```
    Input  : Population $P$, offspring $P'$
  1 Set $\mathcal{I}$ = individual with best cost in $P$;
  2 Set $\mathcal{I}'$ = individual with best cost in $P'$;
  3 if $\mathcal{I} \prec \mathcal{I}'$ then
        /* Replace at random any individual $\mathcal{I}'_r$ from $P'$ with $\mathcal{I}$    */
  4     Select $\mathcal{I}'_r \in P'$ ;
  5     Set $\mathcal{I}'_r = \mathcal{I}$ ;
  6 Set $P = P'$;
    Output: New population $P$
```

Finally, the intensification procedure must be specified. In LMA-GRE, FIQTS (Algorithm 2) is applied to each offspring. There is just a subtle difference which is the stopping criterion. In this case, FIQTS stops after a certain period of time. The reason for this is that with the aim of evolving a high number of generations, a strict control on the time invested by the intensification procedure is required. Note that LMA-GRE does not incorporate explicit control of diversity, so it is used precisely to validate the advantages of other proposals that incorporate strategies to control the diversity.

5.2. LMA with a Replacement Considering Elitism and Dynamic Diversity Control (LMA-REDDC)

The second variant of our optimizer includes a replacement phase that takes diversity into account. Particularly, the replacement phase is similar to the one devised for the Graph Partitioning Problem [10]. This replacement operator (Algorithm 8) operates as follows. It takes as input the population, offspring and a threshold D to perform penalties and it selects the members of the next population. First, it selects the individual with the lowest cost (lines 1–6). Then, iteratively it selects the remaining survivors (lines 7–17). Particularly, at each iteration among the non-selected individuals that are not too close to already selected survivors (line 10–11) it selects the one with the lowest cost (lines 12–13). Note that it might happen that all individuals are too close to each other (line 14). In this case, the farthest individual is selected (lines 14–15).

Algorithm 8: REDDC Survivor Selection Technique.

```
Input  : Population $P$, offspring $P'$, distance $D$
1  Set $P_c = P \cup P'$ ;                                  /* Current Population */
2  foreach $\mathcal{I} \in P_c$ do
3      $\mathcal{I}.cost = C_3(\mathcal{I})$ ;             /* Cost of individual $\mathcal{I}$ */
4  sort($P_c$) ;
   /* $\mathcal{I}_0$ is the individual with lowest cost in $P_c$ */
5  $P = \{\mathcal{I}_0\}$ ;                    /* New population starts with the best */
6  $P_c = P_c \setminus \{\mathcal{I}_0\}$ ;              /* Update current population */
7  for $i = 1; i < N; i++$ do
       /* Calculate the distance to the closest neighbor (DCN) */
8      foreach $\mathcal{I} \in P_c$ do
9          $\mathcal{I}.DCN = \min\{d_H(\mathcal{I}, \mathcal{I}') : \mathcal{I}' \in P\}$;    /* Hamming distance 7 */
10         if $\mathcal{I}.DCN < D$ then
11             $\mathcal{I}.cost = \infty$ ;                  /* Penalize the closest */
12     sort($P_c$) ;
13     $\mathcal{S} = \mathcal{I}_0$ ;                        /* Select the best ($\mathcal{I}_0$) */
       /* If all are penalized choose the farthest */
14     if $\mathcal{S}.cost = \infty$ then
15         $\mathcal{S} = \mathcal{I}_k$ s.t. $\mathcal{I}.DCN \leq \mathcal{I}_k.DCN, \forall \mathcal{I} \in P_c$ ;
16     $P = P \cup \{\mathcal{S}\}$;
17     $P_c = P_c \setminus \{\mathcal{S}\}$;
Output: New population $P$
```

The threshold D is quite important. When using a large D value, exploration is promoted, whereas small values promote intensification. In order to balance from exploration to exploitation, this threshold is set dynamically following the next equation: $D = D_0 - (D_0) \times T_{elapsed}/T_{end}$, where D_0 is a parameter of the optimizer, $T_{elapsed}$ is the elapsed time of the optimization (in seconds), and T_{end} is the stopping criterion (in seconds). This means that at the first generation D is set to D_0 and it decreases linearly so that at the end of the optimization it is set to zero.

5.3. A Memetic Algorithm Based on Clusters

Our last variant (MAC-REDDCC) includes two modifications to improve the performance of LMA-REDDC. One of the drawbacks of LMA-REDDC is that in most generations no individual with a distance lower than D is accepted (line 10). Usually, when crossover is applied to distant individuals, an exploration step is performed. While maintaining distant individuals to explore different regions is important, using close individuals to better intensify is also important. For this reason, a novel algorithm that tries to explore and intensify during the whole search is proposed.

In order to incorporate this design principle two modifications are incorporated. The most important one is the incorporation of a newly design replacement strategy. The REDDC with Clustering (REDDCC) strategy is explained in Algorithm 9.

REDDCC alters the replacement strategy by applying a clustering technique that takes into account the stopping criterion. The main difference between the REDDC and the REDDCC strategies is that for each individual the REDDCC strategy allows a certain number of individuals (cluster size C_{size}) to be closer than D, but farther than $D_C < D$ (see lines 12–15), while in the REDDC strategy this is not allowed.

Algorithm 9: REDDCC Survivor Selection Technique.

```
Input  :Population $P$, offspring $P'$, distances $D$ and $D_C$, cluster size $C_{size}$
1  Set $P_c = P \cup P'$ ;                                      /* Current Population */
2  foreach $\mathcal{I} \in P_c$ do
3      $\mathcal{I}.{cost} = C_3(\mathcal{I})$ ;                /* Cost of individual $\mathcal{I}$ */
4  sort$(P_c)$ ;
   /* $\mathcal{I}_0$ is the individual with lowest cost in $P_c$ */
5  $P = \{\mathcal{I}_0\}$ ;                    /* New population starts with the best */
6  $P_c = P_c \setminus \{\mathcal{I}_0\}$ ;    /* Update current population */
   /* The first cluster with one element is formed */
7  $C_0 = \{\mathcal{I}_0\}$ ;                  /* Cluster centered in $\mathcal{I}_0$ */
8  $\mathcal{C} = \{C_0\}$ ;                    /* Clusters set */
9  for $i = 1; i < N; i++$ do
       /* Calculate the distance to the closest neighbor (DCN) */
10     foreach $\mathcal{I} \in P_c$ do
           /* Distance given by the Hamming distance */
11         $\mathcal{I}.{DCN} = \min\{d_H(\mathcal{I},\mathcal{I}'_k) : \mathcal{I}'_k \in P\}$ ;
12         if $\mathcal{I}.{DCN} < D \wedge |C_k| \geq C_{size}$ then
13             $\mathcal{I}.{cost} = \infty$ ;   /* Penalize but allowing some close individuals */
14         if $\mathcal{I}.{DCN} < D_C$ then
15             $\mathcal{I}.{cost} = \infty$ ;   /* Penalize too close individuals */
16     sort$(P_c)$ ;
17     $\mathcal{S} = \mathcal{I}_0$ ;          /* Select the best $(\mathcal{I}_0)$ */
       /* If all are penalized choose the farthest */
18     if $\mathcal{S}.{cost} = \infty$ then
19         $\mathcal{S} = \mathcal{I}_k$ s.t. $\mathcal{I}.{DCN} \leq \mathcal{I}_k.{DCN}, \forall \mathcal{I} \in P_c$ ;
20     $C_i = \{\mathcal{S}\}$ ;                /* Cluster centered in $\mathcal{S}$ */
21     $\mathcal{C} = \mathcal{C} \cup C_i$ ;
22     for $j = 0; j < i; j++$ do
23         $\mathcal{S}.{DCN} = d_H(\mathcal{S},\mathcal{I}_j)$  $\mathcal{I}_j \in P$;
24         if $\mathcal{S}.{DCN} < D$ then
25             $C_j = C_j \cup \{\mathcal{S}\}$ ;    /* Update the clusters previously added */
26             $C_i = C_i \cup \{\mathcal{I}_j\}$ ;  /* Update the current cluster */
27     $P = P \cup \{\mathcal{S}\}$ ;            /* Update the new population */
28     $P_c = P_c \setminus \{\mathcal{S}\}$ ;   /* Update the current population */
Output:New population $P$, clusters $\mathcal{C}$
```

Note that in comparison to previous variants, the general procedure (see Algorithm 10) was also changed. First, since the REDDCC strategy requires D_c and D as inputs, both values are calculated at each generation (lines 12–15). Some preliminary experiments showed that a larger degree of diversification was only required at initial stages, meaning that D and D_C could be set to small values after a proportion of the granted execution time (T_{end}). Taking this into account, a new parameter, K_E is added. This parameter refers to the proportion of time where exploration is promoted (line 6). The updating strategies were designed so that D is decreased from D_0 to D_{C_0} and D_C is decreased from D_{C_0} to 0 after this period of time (lines 13-14). Then, in subsequent generations the values of D and D_C are not changed (line 15).

Algorithm 10: MAC-REDDCC Method.

```
Input  : $D_{C_0}, D_0, C_{size}, N, T_{ls}, K_{ls}, p_{cc}, p_m, p_c, K_E, T_{end}$
1  Initialize($P_0$) ;
2  Evaluate($P_0$);
   /* Each individual makes up a different cluster */
3  $\mathcal{C} = \{C_k : C_k = \{\mathcal{I}_k\}, \mathcal{I}_k \in P_0\}$ ;
4  Improvement($P_0, T_{ls}, K_{ls}$) ;
5  $t = 0, T_{elapsed} = 0$ ;
6  $T_{exploration} = K_E \times T_{end}$ ;
7  while $T_{elapsed} < T_{end}$ do
8      $P'_t$ = Mating Selection with Clusters($P_t, \mathcal{C}, p_{cc}$) ;
9      $P'_t$ = Reproduction($P'_t, p_c, p_m$);
10     Evaluate($P'_t$);
11     $P'_t$ = Improvement($P'_t, T_{ls}, K_{ls}$) ;
12     if $T_{elapsed} \leq T_{exploration}$ then
13         $D = D_0 - (D_0 - D_{C_0}) \times T_{elapsed}/T_{exploration}$ ;
14         $D_C = D_{C_0} - D_{C_0} \times T_{elapsed}/T_{exploration}$ ;
15     else $D = D_{C_0}, D_C = 0$ ;
16     $P_{t+1}, \mathcal{C}$ = REDDCC ($P_t, P'_t, D, D_C, C_{size}$) ;
17     $t = t + 1$ ;
Output: Best solution(s) found
```

Additionally, a new parameter (p_{cc}) is added to facilitate attaining a proper balance between exploration and exploitation. This parameter refers to the probability that two individuals from different clusters are crossed in order to create a pair of offspring. In other cases, individuals that belong to the same cluster are crossed. The new selection process is given in Algorithm 11.

Algorithm 11: Mating Selection for MAC-REDDCC.

```
Input  : Population $P$, clusters $\mathcal{C}$, $p_{cc}$
   /* Perform two binary tournament at each iteration */
1  for $i = 0; i < N; i+ = 2$ do
       /* Choose two parents from different clusters */
2      if $\mathcal{U}(0,1) < p_{cc}$ then
3          Select randomly $\mathcal{I}_{j_1}, \mathcal{I}_{j_2} \in \mathcal{C}_j$ and $\mathcal{I}_{k_1}, \mathcal{I}_{k_2} \in \mathcal{C}_k, k \neq j$ ;
4          $\mathcal{I}'_i = best(\mathcal{I}_{j_1}, \mathcal{I}_{j_2})$ ;
5          $\mathcal{I}'_{i+1} = best(\mathcal{I}_{k_1}, \mathcal{I}_{k_2})$ ;
       /* Choose two parents in the same cluster */
6      else
7          Select randomly $\mathcal{I}_{k_1}, \mathcal{I}_{k_2}, \mathcal{I}_{k_3}, \mathcal{I}_{k_4} \in \mathcal{C}_k$ ;
8          $\mathcal{I}'_i = best(\mathcal{I}_{k_1}, \mathcal{I}_{k_2})$ ;
9          $\mathcal{I}'_{i+1} = best(\mathcal{I}_{k_3}, \mathcal{I}_{k_4})$ ;
Output: Parents $P'$
```

In the case of the improvement phase, the only difference is that we grant more time for local search when the offspring is generated by crossing parents from different clusters. The reason is that in such a case the crossover operator is more disruptive, so it is expected that more time is required to attain a new high-quality solution. Algorithm 12 illustrates the new improvement phase. A new parameter, K_{ls}, is added. This parameter is used for setting the time granted to those individuals generated with parents belonging to different clusters (line 3). If the parents of the offspring belong to the same cluster the local search time granted is T_{ls} (line 5).

Algorithm 12: Improvement Phase for MAC-REDDCC.

```
   Input  : Population $P$, $T_{ls}, K_{ls}$
   /* Perform the improvement to the population                    */
1  for $i = 0; i < N; i++$ do
2      if $\mathcal{I}_i$ was generated with parents belonging to different clusters then
3          $\mathcal{I}_i = FIQTS(\mathcal{I}_i, K_{ls} \times T_{ls})$ ;
4      else
5          $\mathcal{I}_i = FIQTS(\mathcal{I}_i, T_{ls})$ ;
   Output: Population improved $P$
```

5.4. Parameterization Study

One of the inconveniences of the last proposal (MAC-REDDCC) is that several parameters must be set. Particularly these are the following parameters:

1. D_{C_0}: initial distance threshold used to maintain a proper diversity in each cluster (see Algorithm 10 line 14).
2. D_0: it is responsible for controlling the degree of diversity maintained in the whole population (see Algorithm 10 line 13).
3. C_{size}: indicates the maximum size allowed for each cluster (see Algorithm 9 line 12). Note that since the acceptance of individuals only depends on the information of the closest already selected survivor, some clusters might eventually contain more that C_{size} individuals.
4. N: number of individuals in the population and number of offspring generated at each generation (see Algorithm 6 line 1).
5. T_{ls}: stopping criterion of the local search procedure for offspring with parents belonging to the same cluster (see Algorithm 12 line 5).
6. K_{ls}: if the offspring is generated with parents belonging to different clusters, the local search time applied to this kind of individuals is $T_{ls} \times K_{ls}$ (see Algorithm 12 line 3).
7. p_{cc}: indicates the probability to cross individuals belonging to different cluster (see Algorithm 11 line 2).
8. p_m: probability of performing swaps to mutate the individual (see Algorithm 5 line 2).
9. p_c: probability of interchanging each gene (see Algorithm 4 line 9).
10. K_E: indicates the proportion of time with additional promoted exploration (see Algorithm 10 line 6).

In order to properly adjust the parameters, for each parameter 5 different values were tested and since stochastic algorithms are considered, each tested configuration was run 50 times. For each independent run, the stopping criterion was set to 72 h of execution. Due to the limitations of time it is not possible to do an exhaustive search in the parameters space by considering the dependencies between all of them. Particularly, the optimization of each parameter was performed independently from each other. This was done as follows. First, since the problem for 8 variables can be solved easily up to the best-known results for nonlinearity, we focus on the parameters setting for the 10 variable problem. Taking into account some preliminary results, the zero or initial parameter setting was set according to Table 4. With the initial parameter setting we carried out 50 independent runs with the stopping criterion set equal to 72 h. Table 5 shows the results obtained for this experiment. We can see from the results showed in Table 5, that using the initial parameters setting, the algorithm achieves the best results attained by metaheuristics in the 10-variable problem. The mean nonlinearity is high (its mean is 486.96), which indicates that in several of the independent executions it was able to find solutions with nonlinearity equal to 488. The mean nonlinearity will be our main indicator to consider the quality of each parameterization. The seventh column shows the proportion of independent executions that reach this nonlinearity value, which is called the success ratio (S_r). The initial parameter setting has a success ratio $S_r = 50\%$, which means that 25 of the 50 independent runs achieve solutions with nonlinearity equal to 488. Since 488 is the maximum nonlinearity that had been obtained ever in a single execution of a metaheuristic, our aim was to find a parameterization with a success ratio close to 100%.

Table 4. Initial Parameterization used in the Adjustment Phase.

D_{C_0}	D_0	C_{size}	N	T_{ls}	K_{ls}	p_{cc}	p_m	p_c	K_E
20	100	10	200	0.01	1.0	0.2	0.002	0.5	1.0

Table 5. Results attained with the initial parameterization.

n	Min	Max	Mean	Median	σ_{N_n}	S_r
10	484	488	486.96	487	1.087	50%

In order to improve the parameterization, each parameter was modified independently maintaining the remaining parameter values as in the best configuration found so far. Parameters were optimized following the order in which they were previously presented. After this parameterization stage we obtained the parameterization shown in Table 6. In this last case the success ratio increased to 100%, meaning that our proposal could generate in every execution the best solution attained up to now by metaheuristics.

Table 6. Parameter setting attained after the adjustment procedure.

D_{C_0}	D_0	C_{size}	N	T_{ls}	K_{ls}	p_{cc}	p_m	p_c	K_E	Mean	S_r
14	**60**	**20**	**350**	**0.08**	**2.0**	**0.2**	**0**	**0.2**	**0.2**	**488**	**100%**

5.5. Comparison Among Population-Based Metaheuristics

In order to show the benefits of the components included in MAC-REDDCC, it is interesting to compare it against LMA-GRE and LMA-REDDC. All these algorithms were executed using the set of parameters obtained after the parameterization steps. Specifically, each of the three methods were executed 50 independent times with the parameters shown in Table 7.

Table 7. Parameters involved in MAC-REDDCC, LMA-REDDC and LMA-GRE.

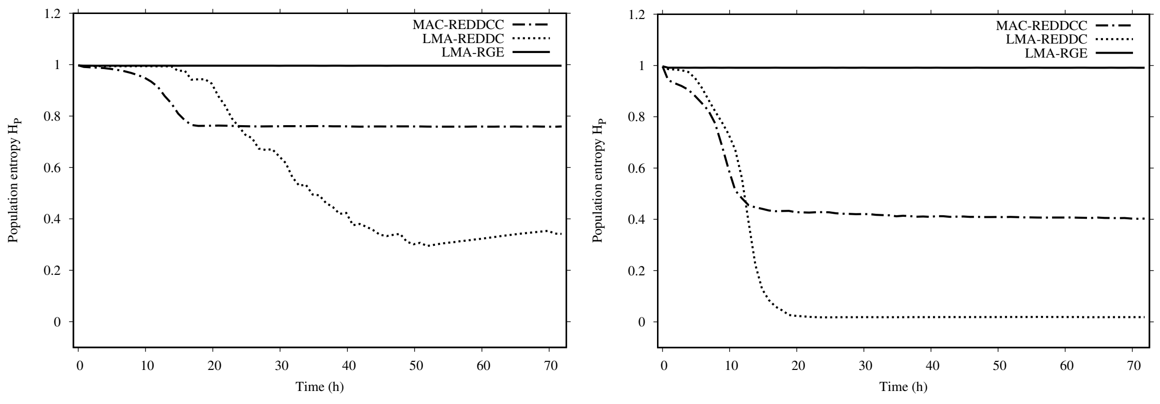
Method	D_{C_0}	D_0	C_{size}	N	T_{ls}	K_{ls}	p_{cc}	p_m	p_c	K_E
MAC-REDDCC	**14**	**60**	**20**	**350**	**0.08**	**2.0**	**0.2**	**0**	**0.2**	**0.2**
LMA-REDDC	×	**60**	×	**350**	**0.08**	×	×	**0**	**0.2**	**0.2**
LMA-GRE	×	×	×	**350**	**0.08**	×	×	**0**	**0.2**	×

Table 8 summarizes the attained results. Results obtained by LMA-GRE in the 10-variable case are poor in comparison with the ones obtained with LMA-REDDC and MAC-REDDCC. Thus, the importance of using a scheme based on diversity in order to find proper solutions for the problem of generating BFs with high nonlinearity is clear. When compared to LMA-REDDC, MAC-REDDCC attains slightly better results, so the additional changes performed were useful to increase the robustness and attain a 100% success ratio.

Table 8. Comparison between the results attained by the population-based optimizers.

n	Method	Min	Max	Mean	Median	σ_{N_n}
8	MAC-REDDCC	116	116	116	116	0
	LMA-REDDC	116	116	116	116	0
	LMA-GRE	116	116	116	116	0
10	**MAC-REDDCC**	**488**	**488**	**488**	**488**	**0**
	LMA-REDDC	486	488	487.44	488	0.907
	LMA-GRE	482	484	483.92	484	0.396

Figure 2 shows the population entropy for the different evolutionary methods. We can see that the LMA-GRE never converges. LMA-REDDC is the method with the fastest decrease in diversity. The clustering technique attains a larger global entropy with a smaller entropy inside clusters, which is key to the success of MAC-REDDCC.



(**a**) Population entropy $n = 8$ (**b**) Population entropy $n = 10$

Figure 2. Population entropy for MAC-REDDCC, LMA-REDDC and LMA-GRE.

5.6. Hybridization with an Algebraic Technique

Motivated by the related work from Burnett [18] and Izbenko [19], we considered the application of an algebraic procedure to improve our results. Contrary to their work, we decided not to use bent functions, instead we employ the work from Tang [32] to build an initial population of individuals with high nonlinearity. Thus, the only change appears in the *initialization* procedure of the population.

Tang [32] proposes a method capable of constructing balanced BFs of even number of variables with relatively high nonlinearity. Their construction is based on a modification of the Maiorana-McFarland [20] method to construct bent BFs. As a result Tang et al. were able to obtain a large amount of balanced BFs with nonlinearity

$$N_n \geq 2^{n-1} - 2^{n/2-1} - 2^{\lceil n/4 \rceil}. \tag{35}$$

Since in the case of the 8-variable BF problem, we had already obtained the best-known solution, we decided to make experiments with the hybrid method only for the 10-variable BF problem. The initialization strategy allows to generate up to 1310400 BFs with nonlinearity equal to 488. In order to make 50 independent runs and each one with a population size equal to 350, we built 17,500 ($50 \times 350 = 17,500$) individuals with nonlinearity equal to 488 with the initialization algorithm [32]. Each population was employed with the three evolutionary methods: MAC-REDDCC, LMA-REDDC and LMA-GRE. The parameter values are the indicated in Table 7. The attained results are shown in Table 9. We can see that the information in Table 9 does not help us to discern which hybrid method is better in comparison with the others because all the methods achieve in all the executions BFs with nonlinearity equal to 492. Note that this is currently the best-known solution and that no meteheuristic had ever reached this value. In order to discern which method is preferable we count the amount of different individuals in the final population that achieve a nonlinearity equal to 492. This comparative is shown in Table 10. From the information in the Table 10, we can see that in MAC-REDDCC and LMA-REDDC every individual in the final population is different and has nonlinearity equal to 492. However, the LMA-GRE is able to maintain only one solution with such nonlinearity. We consider that using the MAC-REDDCC and LMA-REDDC is preferable, because maintaining a diverse population with high quality means that several regions are explored and there is a larger probability to finding better solutions (if they exist). However, when coupling algebraic strategies we are not able to discern between MAC-REDDCC and LMA-REDDC.

Table 9. Comparison among the results attained by the hybrid methods.

n	Method	Min	Max	Mean	Median	σ_{N_n}
	MAC-REDDCC	492	492	492	492	0
10	LMA-REDDC	492	492	492	492	0
	LMA-GRE	492	492	492	492	0

Table 10. Comparison between hybrid methods for the amount of individuals found with nonlinearity equal to 492.

Method	Min	Max	Mean	Median
MAC-REDDCC	350	350	350	350
LMA-REDDC	350	350	350	350
LMA-GRE	1	1	1	1

Figure 3 shows the entropy of the population along the execution time for the three methods tested. As we can see in Figure 3, MAC-REDDCC never decrease its entropy but is able to reach very high quality solutions, so in this sense its behavior seems more adequate to try to reach a solution with a higher nonlinearity.

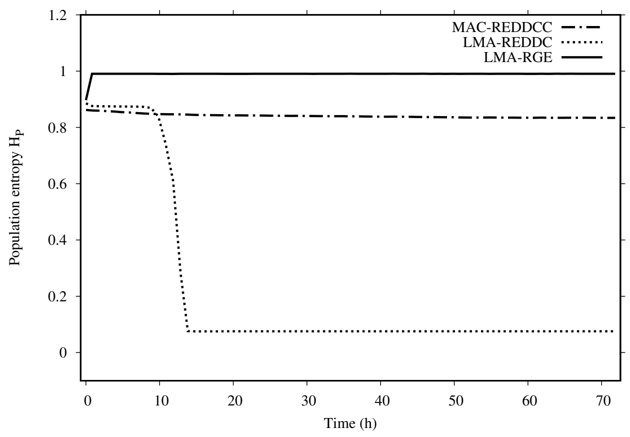


Figure 3. Population entropy for MAC-REDDCC, LMA-REDDC and LMA-GRE when coupled with an algebraic-based initialization

6. Conclusions and Future Work

The problem of generating Cryptographic Boolean Functions (CBFs) with high nonlinearity is an extremely complicated problem. This problem has been addressed with many strategies in the last 30 years. Many heuristic methods have been proposed to generate CBFs with high nonlinearity and the results obtained have been improving over the years. However, the results obtained so far were not as extraordinary as those obtained with algebraic techniques. According to our knowledge, heuristic methods had never been able to generate 10-variable CBFs with nonlinearity equal to 492, but the algebraic constructions do.

In this paper, we work with the hypothesis that a diversity-based metaheuristic can provide a better way of generating CBFs with high nonlinearity, improving further the best current results obtained by other metaheuristic methods and reducing the gap between algebraic and metaheuristic approaches. Based on this, a set of trajectory-based and population-based metaheuristic methods are proposed. Among the proposals, the most novel is a population-based metaheuristic that incorporates explicit diversity management with a clustering technique that allows intensifying and exploring throughout the optimization process, since it forces some members of the population to be distant but some are allowed to be close. This method is called MAC-REDDCC and according to our knowledge, this is the first diversity-aware scheme applied to the generation of CBFs with high nonlinearity.

MAC-REDDCC operates with a set of novel components proposed in this paper. The most important component is the REDDCC strategy, which is responsible for the explicit diversity management. The cost function employed to guide the search is another important component. It takes into account several information from the Walsh Hadamard Transform (WHT), what makes it more suitable to guide the search. The intensification procedure employed to improve the members of the population is another important component. This novel trajectory-based search method is inspired by the Tabu Search algorithm and is called First Improvement Quasi-Tabu Search (FIQTS).

The results obtained with the MAC-REDDCC method improve the best current results reported in the literature by other pure metaheuristic methods, and match the results reported by hybrid metaheuristics with algebraic techniques. As other researchers, we propose a hybridization between the MAC-REDDCC method and a simple algebraic technique. The results obtained with the hybridization match the best results known for 10-variable CBFs, since CBFs with nonlinearity equal to 492 are generated. Thus, MAC-REDDCC is the first metaheuristic method that is able to generate CBFs with that nonlinearity. Our proposal is a single objective mechanism that efficiently solves the problem of creating CBFs for general purposes. This means that it is not difficult to apply it with different fitness or cost functions for different purposes, which opens new lines for future research.

As a future work, the scalability of our proposals should be improved. After performing some initial experiments with 12 variables, we concluded that we need to do some changes—probably in the cost functions and in the local improvement techniques—to be able to deal with 12 or more variables. Particularly, in order to develop a better intensification procedure, it would be interesting to use a neighborhood coupled with the Walsh Hadamard transform. In this way, the improvement process could be faster. In other words, extending the smart hill climbing algorithm to operate with other cost functions seems promising. Additionally, it seems interesting to employ other properties such as the autocorrelation or the algebraic degree inside the cost function. Some authors have obtained improvements with these kinds of transformations, so integrating them with the advances developed in this paper seems promising. Particularly, applying multi-objective optimizers with alternative functions considering more information from the WHT as well as designing ad-hoc operators seems encouraging. Additionally, the applicability of the CBFs generated with our proposals should be further explored. Finally, note that some authors have used specific hardware to speeding up the attainment of high-quality CBFs [33,34]. In order to scale our algorithms, these kinds of hardware as well as supercomputing might be applied.

Author Contributions: Author Contributions: Conceptualization, I.L.-L., G.S.-G. and C.S.; methodology, I.L.-L. and C.S.; software, I.L.-L. and C.S.; validation, I.L.-L., G.S.-G. and C.S.; formal analysis, I.L.-L., G.S.-G., C.S., D.O. and O.R.; investigation, I.L.-L., G.S.-G. and C.S.; writing—original draft preparation, I.L.-L., G.S.-G., C.S., D.O. and O.R.; writing—review and editing, I.L.-L., G.S.-G., C.S., D.O. and O.R.; supervision, I.L.-L., G.S.-G., C.S., D.O. and O.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The third author acknowledges the financial support from CONACyT through the "Ciencia Básica" project no. 285599. Additionally, authors acknowledge the support from "Laboratorio de Supercómputo del Bajio" through the project 300832 from CONACyT.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiao, G.Z.; Massey, J.L. A spectral characterization of correlation-immune combining functions. *IEEE Trans. Inf. Theory* **1988**, *34*, 569–571. [CrossRef]
2. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography (Chapman 8 Hall/Crc Cryptography and Network Security Series)*; Chapman & Hall/CRC: Boca Raton, FL, USA, 2007.
3. Millan, W.; Clark, A.; Dawson, E. Smart hill climbing finds better boolean functions. In *Workshop on Selected Areas in Cryptology 1997, Workshop Record*; Citeseer: University Park, PA, USA, 1997; pp. 50–63.
4. Picek, S.; Jakobovic, D.; Miller, J.F.; Marchiori, E.; Batina, L. Evolutionary methods for the construction of cryptographic boolean functions. In *European Conference on Genetic Programming*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 192–204. [CrossRef]
5. Picek, S.; Ege, B.; Batina, L.; Jakobovic, D.; Chmielewski, Ł.; Golub, M. On using genetic algorithms for intrinsic side-channel resistance: The case of aes s-box. In *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 13–18. [CrossRef]
6. Picek, S.; Carlet, C.; Guilley, S.; Miller, J.F.; Jakobovic, D. Evolutionary algorithms for boolean functions in diverse domains of cryptography. *Evol. Comput.* **2016**, *24*, 667–694. [CrossRef] [PubMed]
7. Sarkar, P.; Maitra, S. Nonlinearity bounds and constructions of resilient Boolean functions. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 515–532. [CrossRef]
8. Doganaksoy, A.; Dündar, B.G.; Göloglu, F.; Saygı, Z.; Sulak, F.; Uguz, M. Constructions of highly nonlinear balanced boolean functions. In Proceedings of the Ulusal Kriptoloji Sempozyumu, Ankara, Turkey, 18–20 November 2005.
9. Segura, C.; Hernandez, A.; Luna, F.; Alba, E. Improving diversity in evolutionary algorithms: New best solutions for frequency assignment. *IEEE Trans. Evol. Comput.* **2017**, *21*, 539–553. [CrossRef]
10. Romero Ruiz, E.; Segura, C. Memetic algorithm with hungarian matching based crossover and diversity preservation. *Comput. Sist.* **2018**, *22*. [CrossRef]
11. Aldana-Bobadilla, E.; Kuri-Morales, A. A clustering method based on the maximum entropy principle. *Entropy* **2015**, *17*, 151–180. [CrossRef]
12. Elhosary, A.M.; Hamdy, N.; Farag, I.A.; Rohiem, A. State of the art in Boolean functions cryptographic assessment. *Int. J. Comput. Netw. Commun. Secur.* **2013**, *1*, 88–94.
13. MacWilliams, F.J.; Sloane, N.J.A. *The Theory of Error-Correcting Codes*; Elsevier: Amsterdam, The Netherlands, 1977.
14. Millan, W.; Clark, A.; Dawson, E. An effective genetic algorithm for finding highly nonlinear Boolean functions. In *International Conference on Information and Communications Security*; Springer: Berlin/Heidelberg, Germany,1997; pp. 149–158. [CrossRef]
15. Millan, W.; Clark, A.; Dawson, E. Heuristic design of cryptographically strong balanced Boolean functions. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 489–499.
16. Clark, J.A.; Jacob, J.L. Two-stage optimisation in the design of Boolean functions. In *Australasian Conference on Information Security and Privacy*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 242–254. [CrossRef]
17. Clark, J.A.; Jacob, J.L.; Maitra, S.; Stanica, P. Almost Boolean functions: The design of Boolean functions by spectral inversion. In Proceedings of the 2003 Congress on Evolutionary Computation, 2003. CEC '03, Canberra, ACT, Australia, 8–12 December 2003; IEEE: Piscataway, NJ, USA, 2003; Volume 3, pp. 2173–2180. [CrossRef]
18. Burnett, L.; Millan, W.; Dawson, E.; Clark, A. Simpler methods for generating better Boolean functions with good cryptographic properties. *Australas. J. Comb.* **2004**, *29*, 231–248.
19. Izbenko, Y.; Kovtun, V.; Kuznetsov, A. The design of boolean functions by modified hill climbing method. In Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 27–29 April 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 356–361. [CrossRef]
20. McFarland, R.L. A family of difference sets in non-cyclic groups. *J. Comb. Theory Ser. A* **1973**, *15*, 1–10. [CrossRef]

21. Picek, S.; McKay, R.I.; Santana, R.; Gedeon, T.D. Fighting the symmetries: The structure of cryptographic boolean function spaces. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Coimbra, Portugal, 8–11 September 2015; ACM: New York, NY, USA, 2015; pp. 457–464. [CrossRef]
22. Picek, S.; Jakobovic, D.; Miller, J.F.; Batina, L.; Cupic, M. Cryptographic Boolean functions: One output, many design criteria. *Appl. Soft Comput.* **2016**, *40*, 635–653. [CrossRef]
23. Picek, S.; Santana, R.; Jakobovic, D. Maximal nonlinearity in balanced boolean functions with even number of inputs, revisited. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 3222–3229. [CrossRef]
24. Goyal, R.; Yadav, S.P. An evolutionary approach to construct cryptographically strong Boolean functions. *Int. J. Syst. Assur. Eng. Manag.* **2012**, *3*, 1–5. [CrossRef]
25. Črepinšek, M.; Liu, S.H.; Mernik, M. Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM Comput. Surv.* **2013**, *45*, 35:1–35:33. [CrossRef]
26. Segura, C.; Coello, C.A.C.; Segredo, E.; Aguirre, A.H. A novel diversity-based replacement strategy for evolutionary algorithms. *IEEE Trans. Cybern.* **2016**, *46*, 3233–3246. [CrossRef] [PubMed]
27. Clark, J.A.; Jacob, J.L.; Stepney, S. Searching for cost functions. In Proceedings of the IEEE Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; pp. 1517–1524. [CrossRef]
28. Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 74.
29. Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurr. Comput. Program, C3p Rep.* **1989**, *826*, 1989.
30. Chen, X.; Ong, Y.S.; Lim, M.H.; Tan, K.C. A multi-facet survey on memetic computation. *IEEE Trans. Evol. Comput.* **2011**, *15*, 591–607. [CrossRef]
31. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing (Natural Computing Series)*; Springer: Berlin/Heidelberg, Germany, 2008.
32. Tang, D.; Zhang, W.; Tang, X. Construction of balanced Boolean functions with high nonlinearity and good autocorrelation properties. *Des. Codes Cryptogr.* **2013**, *67*, 77–91. [CrossRef]
33. Asghar, A.; Iqbal, M.M.; Ahmed, W.; Ali, M.; Parvez, H.; Rashid, M. Logic algebra for exploiting shared SRAM-table based FPGAs for large LUT inputs. In Proceedings of the 2017 First International Conference on Latest Trends in Electrical Engineering and Computing Technologies (INTELLECT), Karachi, Pakistan, 15–16 November 2017; pp. 1–4. [CrossRef]
34. Asghar, A.; Iqbal, M.M.; Ahmed, W.; Ali, M.; Parvez, H.; Rashid, M. Exploring Shared SRAM Tables in FPGAs for Larger LUTs and Higher Degree of Sharing. *Int. J. Reconfig. Comput.* **2017**, *2017*. [CrossRef]