J Glob Optim (2010) 47:597–613
DOI 10.1007/s10898-009-9481-4

A combined DCA: GA for constructing highly nonlinear balanced boolean functions in cryptography

Hoai Minh Le · Hoai An Le Thi · Tao Pham Dinh · Pascal Bouvry

Received: 6 April 2007 / Accepted: 22 October 2009 / Published online: 5 November 2009

Abstract Substitution boxes, aka S-boxes, are a key component of modern crypto-systems. Several studies and developments were carried out on the problem of building high-quality S-boxes in the last few years. Qualities of such boxes, such as nonlinearity and balance, steer the robustness of modern block ciphers. This work is concerned with the construction of highly nonlinear balanced Boolean functions. A deterministic optimization model which is the minimization of a polyhedral convex function on a convex polytope with 0–1 variables is introduced. A local deterministic optimization approach called DCA (Difference of Convex functions Algorithm) is investigated. For finding a good starting point of DCA we propose two versions of a combined DCA–GA (Genetic Algorithm) method. Numerical simulations prove that DCA is a promising approach for this problem. Moreover the combination of DCA–GA improves the efficiency of DCA and outperforms other standard approaches.

Keywords Cryptography · Boolean function · Nonlinear balanced Boolean function · Nonlinearity · Genetic Algorithm · Hybrid Genetic · DC programming · DCA · Mixed 0–1 polyhedral convex program · Exact penalty

H. M. Le · H. A. Le Thi (✉)
Laboratory of Theoretical and Applied Computer Science (LITA EA 3097) UFR MIM, University of Paul Verlaine - Metz, Ile du Saulcy, Metz 57045, France
e-mail: lethi@univ-metz.fr

T. Pham Dinh
Laboratory of Modelling, Optimization and Operations Research, National Institute for Applied Sciences - Rouen, BP 08, Place Emile Blondel, Mont Saint Aignan Cedex 76131, France

P. Bouvry
Computer Science Research Unit, University of Luxembourg, Campus Kirchberg, 6 Rue Richard Coudenhove-Kalergi, Luxembourg 1359, Luxembourg

1 Introduction

Since the development of computers there has been strong demand for means to protect information and to provide various security services. The main aspects of information security are privacy, data integrity, authentication, and non-repudiation. In this paper we deal with the problem of cryptography that transforms a message (plaintext) into an encoded message that only the sender and the receiver can understand (ciphertext). Cryptographic techniques are divided into two categories: symmetric-key ciphers, aka secret key ciphers, and public key ciphers. If both sender and receiver use the same key, or it is easy to obtain one form another then the system is referred to as symmetric key encryption. If the sender and receiver each uses a different key, and it is computationally infeasible to determine one form another without knowing some additional (kept secret) information then the system is referred to as a public key encryption. There are two classes of symmetric-key encryption schemes: block ciphers and stream ciphers. A block cipher breaks up the message into blocks of the fixed length and encrypts one block at a time. A stream cipher is one that encrypts data stream one bit or one byte at a time. A good overview of all major cryptography techniques can be found in [8]. A description of block ciphers including the Advanced Encryption Standard, aka AES (a successor of the Data Encryption Standard, aka DES) cipher is presented in [15].

The terms confusion and diffusion introduced by Shannon capture the two basic building blocks for any cryptographic system. In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext, what is usually achieved through a large set of permutations. Confusion makes the relationship between the statistics of the ciphertext and plaintext as complex as possible. This paper deals with the modeling of substitution boxes, aka S-boxes, that are the crypto-component bringing the confusion aspect of most modern symmetric-key block ciphers such as AES and 3DES.

Typical secret key ciphers are structured as a set of rounds. Each round is performing a set of permutations and substitutions that encode the plaintext. It is thus important to notice that S-box, though being an important crypto-primitive is only one of the sub-components of a cipher. Without knowing the complete cipher, it is difficult to predict the precise impact in terms of quality improvement a higher quality S-box. However, it is clear that any weakness of any component may later be exploited by an attacker. Indeed in cryptanalysis any hint may be used to reduce the complexity of the search space.

S-boxes are typically fixed lookup tables (e.g. AES, DES, 3DES), though some encryption algorithms may use dynamic ones (e.g. Blowfish, Twofish). We consider in the present article the construction of high-quality fixed S-boxes for secret key ciphers. Given that these tables are designed once for ever, we can afford to spend a high computational time for obtaining a very high quality table. We denote the substitution table of a n-input k-output Boolean function by $f = B^n \rightarrow B^k$, mapping each combination of n Boolean input values to some combination of k Boolean output values.

In the case of AES, we have $n = k = 8$ that gives a bijective S-box that allows to map any 8 bits input value to a 8 bits output (see Table 1). For instance the value $0x9a$ is mapped into value $0xb8$. Given that for two different input values, we have two different output values, this table is revertible.

For building quality S-boxes several criteria have to be taken into account such as nonlinearity, balance, immunity, etc. Designing suitable S-boxes is thus a difficult task where the objective is to optimize multiple criteria. We propose and compare in the coming sections different approaches for generating such Boolean functions. These functions feature high quality cryptography criteria in order to become good candidates for building high-quality S-boxes. In our work, nonlinearity and balance are the main criteria considered for building

Table 1 AES S-box

	00	01	02	03	04	05	06	07	08	09	**0a**	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	**b8**	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

high quality S-boxes based on Boolean functions. These properties have been widely studied in the literature (see e.g. [1,14] and references therein). S-boxes that do not feature these qualities would result in ciphers that are prone to linear cryptanalysis and therefore very weak.

Heuristic optimization approaches, have already been successfully applied to this problem (see e.g. [1,3,13,15] and references therein). However, up to now there is no deterministic models and methods for it. In this work we develop a *deterministic continuous optimization approach* based on DC (Difference of Convex functions) programming and DCA (DC optimization Algorithms). The contribution of the paper is 3-fold:

Firstly, we propose in this paper, an original deterministic mathematical model for the considered problem. By expressing the XOR operation (involved in the nonlinearity of Boolean functions) in a suitable way we first formulate the problem as a combinatorial optimization problem, namely the minimization of a polyhedral convex function on a convex polytope with 0–1 variables. Due to the very large dimension of this problem in practice, the standard methods in combinatorial optimization such as branch and bound, branch and cut, cutting plan can not be applied. Attempting to develop robust numerical solution approaches, we reformulate the problem as a polyhedral DC program by using exact penalty techniques in DC programming ([7]).

Secondly, we investigate DC programming and DCA for solving the related polyhedral DC program.

DC programming and DCA have been introduced by Pham Dinh Tao in 1985 and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994 to become now classic and increasingly popular ([4,7,10,11] and references therein).

DCA aims to solve a general DC program that takes the form

$$\alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^p\} \quad (P_{dc}) \tag{1}$$

where g, h are lower semicontinuous proper convex functions on $\mathbb{R}^p$. Such a function f is called DC function, and $g - h$, DC decomposition of f while g and h are DC components of f. The construction of DCA involves DC components g and h but not the function f itself: each iteration k of DCA consists of computing

$$y^k \in \partial h(x^k), x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\} \quad (P_k).$$

Hence, for a DC program, each DC decomposition corresponds to a different version of DCA. Since a DC function f has an infinite number of DC decompositions which have crucial impacts on the qualities (speed of convergence, robustness, efficiency, globality of computed solutions,…) of DCA, the search for a "good" DC decomposition is important from algorithmic point of views. Moreover, despite its local character, DCA with a good initial point can converge to global solutions. Finding a "good" initial point is then also an important stage of DCA. How to develop an efficient algorithm based on the generic DCA scheme for a practical problem is thus a judicious question to be studied, and the answer depends on the specific structure of the problem being considered. In the current paper, we propose a DCA that has interesting convergence properties: our algorithm converges to a local solution after a finitely many iterations and it consists of solving a linear program at each iteration. Moreover although our DCA is a continuous approach that works on a continuous domain, it provides an integer solution. This is unusual in continuous approaches and is original property of the proposed DCA. On the other hand, we investigate the computation of good initial points of DCA by a Genetic Algorithm (GA).

Thirdly, we propose a combination of the two approaches DCA and GA for solving the above mentioned polyhedral DC program. Two versions of the combined algorithm are developed on which an extensive computational study is investigated.

The paper is organized as follows. Section 2 introduces some notations and properties of Boolean functions. Section 3 deals with the optimization formulation and reformulation of the problem. DC programming and DCA for solving the resulting polyhedral DC program are investigated in Sect. 4. For the reader's convenience, at the beginning of this section we provide a brief introduction to these tools. Section 5 is devoted to the combination of DCA and GA to compute good starting points of DCA. Finally, computational results are reported in Sect. 6. They show that DCA is a promising approach for this problem. Moreover the combined DCA–GA improves the efficiency of DCA and outperforms other standard approaches.

2 Preliminaries

Definition 1 A Boolean function: $f : \{0, 1\}^n \to \{0, 1\}$ is a function which produces a Boolean result.

Definition 2 The binary truth table of a Boolean function of n variables, denoted $f(x)$, is the truth table that contains 2^n elements corresponding to all possible combinations of the n binary inputs.

For a given table $x = (x_1, x_2, \ldots, x_n)$, the Boolean function f can be determined by the last column of its binary truth table, namely a binary vector in dimension 2^n. Let $B := \{0, 1\}$. In this work we consider a Boolean function f as a vector in B^{2^n}. Hence the set of Boolean functions, denoted by F is exactly the set B^{2^n}.

Definition 3 The polarity truth table of a Boolean function denoted $\widehat{f}$ is defined by $\widehat{f}(x) = (-1)^{f(x)} = 1 - 2f(x)$, where $\widehat{f}(x) \in \{1, -1\}$.

Definition 4 (a) A linear Boolean function $L_w(x)$, selected by $w \in Z_2^n$, is a Boolean function given by ($\oplus$ denotes the Boolean operation 'XOR')

$$L_w(x) = wx = w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n. \tag{2}$$

(b) An affine Boolean function $A_w(x)$ is a Boolean function which can be represented in the form

$$A_w(x) = wx \oplus c \quad \text{where } c \in Z_2. \tag{3}$$

Two fundamental properties of Boolean functions are Hamming weight and Hamming distance.

Definition 5 (a) The Hamming weight of a Boolean function is the number of ones in the binary truth table.
(b) The Hamming distance between two Boolean functions is the number of positions for which their truth tables differ.

Property 1 (a) *The Hamming weight of a Boolean function is given by:*

$$hwt(f) := \sum_{x \in B^n} f(x) = \frac{1}{2}\left(2^n - \sum_{x \in B^n} \widehat{f}(x)\right). \tag{4}$$

(b) *The Hamming distance between two Boolean functions is computed as*

$$d(f, g) := \sum_{x \in B^n} f(x) \oplus g(x) := \left(2^n - \sum_{x \in B^n} \widehat{f}(x)\widehat{g}(x)\right). \tag{5}$$

Definition 6 (a) The balance of a Boolean function is defined as: $I_f := \frac{1}{2}\left|\sum_{x \in B^n} \widehat{f}(x)\right|$.
(b) The nonlinearity of a Boolean function is the Hamming distance to the closest affine function. That is, nonlinearity is the number of bits which must change in the truth table of a Boolean function to reach the closest affine function.

The "Balance" is a primary cryptographic criterion for S-boxes: an imbalanced function has sub-optimum unconditional entropy. For cryptographic Boolean functions, in order to have a *balanced* function, we need to have an equal number of 0's and 1's in the binary truth table.

The "nonlinearity" is considered as another primary criterion for S-boxes: indeed a linear function is prone to linear cryptanalysis and is thus leading to a weak cipher. The nonlinearity of a Boolean function is typically computed via the Walsh-Hadamard transform (WHT) of the function being measured.

Definition 7 The Walsh-Hadamard Transform (WHT) of a Boolean function is defined as: $\widehat{F}(w) := \sum_x \widehat{f}(x)\widehat{L}_w(x)$.

Property 2 The nonlinearity of a Boolean function f, denoted N_f, is related to the maximum magnitude of WHT values, and given by

$$N_f := 2^{n-1} - \frac{1}{2} \max_{w \in B^n} |\widehat{F}(w)|. \tag{6}$$

Extensions to S-boxes. As proposed in [3], for each k-output S-box, we can extract a single-output Boolean function by simply XOR-ing some subset of the output bits together. If $f(x) : B^n \to B^k$ is an n-input k-output S-box then each $\beta \in B^k$ defines a function that is a linear combination $f(x)$ of the k outputs of f. This is given by

$$f_\beta(x) = \beta_1 f_1(x) \oplus \cdots \oplus \beta_k f_k(x). \tag{7}$$

There are $2^k - 1$ non-trivial functions obtainable in this way. The notions of non-linearity is readily extended to the multiple-output case. For the k-output case the non-linearity is the worst (lowest) non-linearity of all the $2^k - 1$ non-trivial single output functions obtained as indicated above.

3 Mathematical formulation and reformulation

The objective of our work is to generate highly nonlinear balanced Boolean functions. In other words, we have to find a balanced Boolean function featuring a maximal nonlinearity criterion. According to the above notations and properties, the problem of maximizing the nonlinearity of a Boolean function can be written as:

$$\max_{f\in F} N_f = \max_{f\in F} \min_{w\in B^n} \frac{1}{2}\left(2^n - \left|\sum_{x\in B^n} \widehat{f}(x)\widehat{L}_w(x)\right|\right).$$

Let $\Phi(\widehat{f}) := \max\limits_{w\in B^n} \left|\sum_{x\in B^n} \widehat{f}(x)\widehat{L}_w(x)\right|$, we have

$$\max_{f\in F} N_f = 2^{n-1} - \frac{1}{2}\min_{f\in F} \Phi(\widehat{f}). \tag{8}$$

In this formulation the presence of the operation "XOR" in $\widehat{L}(w)$ (cf. Definition 4) is not suitable to apply continuous optimization techniques. We will express $\widehat{L}(w)$ in another way. By the recursive demonstration we get

$$\begin{aligned}\widehat{L}_w(x) &= 1 - 2wx = 1 - 2(w_1x_1 \oplus w_2x_2 \oplus \cdots \oplus w_nx_n)\\ &= 1 \text{ if } \langle w, x\rangle = \sum_{i=1}^{n} w_i x_i \text{ is a even number, } -1 \text{ otherwise.}\end{aligned}$$

Denoting $a_{wx} := \widehat{L}_w(x) \in \{-1, 1\}$ for $w, x \in B^n$, we can write the function Φ in the form

$$\Phi(\widehat{f}) = \max_{w\in B^n} \left|\sum_{x\in B^n} a_{wx}\widehat{f}(x)\right|. \tag{9}$$

Consequently, according to Definition 3 (we use f_x to denote $f(x)$)

$$\Phi(\widehat{f}) = \max_{w\in B^n} \left|\sum_{x\in B^n} a_{wx}(2f_x - 1)\right| = 2\max_{w\in B^n} \left|\sum_{x\in B^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in B^n} a_{wx}\right|. \tag{10}$$

Hence, from (8) it follows that $\max_{f\in F} N_f = 2^{n-1} - \min_{u\in B^n} \Psi(f)$, where $\Psi(f) := \max_{w\in B^n} \left|\sum_{x\in B^n} a_{wx}f_x - \frac{1}{2}\sum_{x\in B^n} a_{wx}\right|$. Thus, maximizing N_f amounts to minimizing $\Psi(f)$ on B^{2^n}. For finding a balanced Boolean function we add the next constraint

$$\left|\sum_{x\in B^n} f_x - 2^{n-1}\right| \le b, \tag{11}$$

with a nonnegative number b. Clearly that if $b = 0$, then the function is balanced. Finally we get the following optimization problem

$$\beta := \min \left\{ \Psi(f) : 2^{n-1} - b \leq \sum_{x \in B^n} f_x \leq 2^{n-1} + b,\ f \in B^{2^n} \right\}. \tag{12}$$

It is easy to see that the function Ψ is a polyhedral convex (by definition, a function is polyhedral if it is a pointwise supremum of a finite collection of affine functions). We are then facing the minimization of a convex polyhedral function with binary variables under linear constraints. It is known that the last problem is in fact equivalent to a *mixed zero-one linear program* (with exactly one continuous variable). For our convenience in the DC programming approach, for the moment, we consider the problem in the form (12). This problem is of a very large dimension: 2^n variables and 2^{n+1} constraints.

Continuous optimization formulation. We can reformulate (12) in the form of a continuous optimization problem. Let

$$p : \mathbb{R}^{2^n} \to \mathbb{R} \text{ be the function defined by } p(f) := \sum_{x \in B^n} \min\{f_x, 1 - f_x\}.$$

It is clear that p is a nonnegative concave function on $[0, 1]^n$. Moreover $p(f) = 0$ iff $f \in B^{2^n}$. Hence Problem (12) can be expressed as

$$\min \left\{ \Psi(f) : 2^{n-1} - b \leq \sum_{x \in B^n} f_x \leq 2^{n-1} + b,\ p(f) \leq 0 \right\}. \tag{13}$$

Using exact penalty techniques [7] leads us to the more tractable continuous optimization problem ($t > 0$ is the penalty parameter):

$$(Q)\quad \beta = \min \{\Psi(f) + tp(f) :\ f \in K\},$$

where $K := \left\{ f : 2^{n-1} - b \leq \sum_{x \in B^n} f_x \leq 2^{n-1} + b,\ 0 \leq f_x \leq 1,\ \forall x \in B^n \right\}$.
More precisely, it is proved that (13) and (Q) are equivalent in the sense that there exists $\tau_0 \geq 0$ such that for every $t > \tau_0$, the two problems have the same optimal value and the same set of optimal solutions.

We will prove in Sect. 4 that Problem (Q) can be reformulated as DC program and show how to use DCA for solving it.

4 Solution method based on DCA

Before discussing the algorithm based DCA for generating highly nonlinear balanced Boolean Function, let us introduce briefly DC programming and DCA. For a complete study of DC programming and DCA the reader is referred to [4,6,10,11] and references therein.

4.1 An introduction of DC programming and DCA

DC Programming and DCA constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. They address the problem of minimizing a function f which is a difference of convex functions on the whole space $\mathbb{R}^p$ or on a convex set $C \subset \mathbb{R}^p$. Generally speaking, a DC program takes the form

$$\alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^p\} \quad (P_{dc}) \tag{14}$$

where g, h are lower semicontinuous proper convex functions on $\mathbb{R}^p$. The convex constraint $x \in C$ can be incorporated in the objective function of (P_{dc}) by using the indicator function on C denoted χ_C which is defined by $\chi_C(x) = 0$ if $x \in C$, ∞ otherwise. Let

$$g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^p\}$$

be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$\alpha_D = \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^p\}. \quad (D_{dc}) \tag{15}$$

One can prove that $\alpha = \alpha_D$, (see e.g. [4,6]) and there is the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}).

DCA is based on the local optimality conditions of (P_{dc}), namely

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{16}$$

(such a point x^* is called *critical point* of $g - h$), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{17}$$

The condition (17) is necessary local optimality of (P_{dc}). It is also sufficient for many classes of DC programs. In particular it is sufficient for the next cases quite often encountered in practice:

i) In polyhedral DC programs with h being a polyhedral convex function (see [4,6,10,11] and references therein). In this case, if h is differentiable at a critical point x^*, then x^* is actually a local minimizer for (P_{dc}). Since a convex function is differentiable everywhere except for sets of measure zero, one can say that a critical point x^* is almost always a local minimizer for (P_{dc}).
ii) In case of the function f being locally convex at x^* ([6]).

The idea of DCA is simple: each iteration of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^k \in \partial h(x^k)$) and minimizes the resulting convex function (that is equivalent to determining $x^{k+1} \in \partial g^*(y^k)$).

Generic DCA scheme

Initialization: Let $x^0 \in \mathbb{R}^p$ be a best guess, $0 \leftarrow k$.

Repeat

Calculate $y^k \in \partial h(x^k)$

Calculate $x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\}$ (P_k)

$k + 1 \leftarrow k$

Until convergence of x^k.

It is important to mention the following main convergence properties of DCA:

- DCA is a descent method (the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing) *without linesearch*;
- If the optimal value α of the problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point x^* (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).
- DCA has a *linear convergence* for general DC programs.
- DCA has a finite convergence for polyhedral DC programs.

The solution of a practical nonconvex program by DCA must be composed of two stages: the search of an *appropriate* DC decomposition and the search of a *good* initial point. An

appropriate DC decomposition, in our sense, is one that corresponds to a DCA which is not expensive and has interesting convergence properties.

In the next subsection we will develop an instance of DCA to solve problem (Q) in its equivalent DC program, and study the convergence properties of the proposed algorithm.

4.2 A DCA scheme for solving Problem (Q)

We first reformulate (Q) in the form of DC program. Let χ_K be the indicator function of K, say $\chi_K(f) = 0$ if $f \in K$, $+\infty$ otherwise. Since K is a convex set, χ_K is a convex function on $\mathbb{R}^{2^n}$.

A natural DC decomposition of the objective function of (Q) is

$$\Psi(f) + tp(f) := G(f) - H(f),$$

with $G(f) := \chi_{K(f)} + \Psi(f)$ and $H(f) := -tp(f)$. It is clear that G and H are convex functions. Thus Problem (Q) is a DC program of the form

$$(Q_{dc}) \quad \beta := \min\{G(f) - H(f) : \quad f \in \mathbb{R}^{2^n}\}.$$

Let ψ_w be the function defined by : $\psi_w(f) := \left|\sum_{x\in B^n} a_{wx} f_x - \frac{1}{2}\sum_{x\in B^n} a_{wx}\right|$.

Then ψ_w is a convex function, and $\Psi(f) = \max_{w\in B^n} \psi_w(f)$. Therefore, as mentioned in Sect. 2, Ψ is a polyhedral convex function. Likewise the function H is also polyhedral convex . So (Q_{dc}) is a polyhedral DC program where all DC decomposition are polyhedral. This property enhances DCA as will be shown later in the convergence theorem of DCA.

According to Sect. 4.1, applying DCA to (Q_{dc}) amounts to computing, at each iteration k: $v^k \in \partial H(f^k)$ and $f^{k+1} \in \partial G^*(v^k)$.

By the very definition of H, we can take v^k as follows:

$$v_x^k := -t \quad \text{if} \quad f_x^k \le 0.5, \ t \quad \text{otherwise}. \tag{18}$$

On the other hand, by the virtue of the above results concerning DC programming and DCA, the condition $f^{k+1} \in \partial G^*(v^k)$ is equivalent to:

$$f^{k+1} = \operatorname{argmin}\{\Psi(f) - \langle v^k, f\rangle : f \in K\}. \tag{19}$$

Since Ψ is a convex polyhedral function, Problem (19) is equivalent to a linear program. Indeed,

$$\begin{aligned}
&\min\{\Psi(f) - \langle v^k, f\rangle : f \in K\} = \min_{f\in K}\max_{w\in B^n}\left(\psi_w(f) - \langle v^k, f\rangle\right)\\
&\iff \min\left\{\xi - \langle v^k, f\rangle : \ \psi_w(f) \le \xi, \forall w \in B^n, f \in K\right\}\\
&\iff \min\left\{\begin{array}{c}
\xi - \langle v^k, f\rangle : \sum\limits_{x\in B^n} a_{wx} f_x - \frac{1}{2}\sum\limits_{x\in B^n} a_{wx} \le \xi, \forall w \in B^n\\
-\sum\limits_{x\in B^n} a_{wx} f_x + \frac{1}{2}\sum\limits_{x\in B^n} a_{wx} \le \xi, \forall w \in B^n,\\
2^{n-1} - b \le \sum\limits_{x\in B^n} f_x \le 2^{n-1} + b, 0 \le f_x \le 1, \ \forall x \in B^n
\end{array}\right\}.
\end{aligned} \tag{20}$$

The DCA applied to (Q_{dc}) can be now described as follows:

DCA Algorithm.
Let $f^0 \in \mathbb{R}^{2^n}$, and ϵ be a sufficiently small positive number.
Repeat
Set $v^k \in \partial H(f)$ via the formula (18);
Solving the linear program (20) to obtain f^{k+1};
Set $k := k + 1$.
Until $\| f^k - f^{k-1} \| < \epsilon$.

Denote by Ω the feasible set of the linear program (20), and $V(\Omega)$ the vertex set of Ω. Let f^* be a solution computed by **DCA**. The convergence of **DCA** can be summarized in the next theorem whose proof is essentially based on the convergence theorem of DCA applied to a polyhedral DC program.

Theorem 1 (Convergence properties of Algorithm **DCA**)

(i) **DCA** *generates a sequence* $\{f^k\}$ *contained in* $V(\Omega)$ *such that the sequence* $\{\Psi(f^k) + tp(f^k)\}$ *is decreasing.*
(ii) *For a number* t *sufficiently large, if at iteration* r *we have* $f^r \in \{0,1\}^{2^n}$, *then* $f^k \in \{0,1\}^{2^n}$ *for all* $k \geq r$.
(iii) *The sequence* $\{f^k\}$ *converges to* $\{f^*\} \in V(\Omega)$ *after a finite number of iterations. The point* f^* *is a critical point of Problem* (Q_{dc}). *Moreover if* $f^*_x \neq \frac{1}{2}$ *for all* $x \in B^n$, *then* f^* *is a local solution to* (Q_{dc}).

Proof (i) is consequence of DCA's convergence Theorem for a general DC program (see Sect. 4.1).

(ii) Let $t > t_1 := \max\left\{\frac{\Psi(f)-\eta}{\theta} : f \in V(\Omega), p(f) \leq 0\right\}$ where $\eta := \min\{\Psi(f) : f \in V(\Omega)\}$ and $\theta := \min\{p(f) : f \in V(\Omega)\}$. Let $\{f^k\} \subset V(\Omega)$ $(k \geq 1)$ be generated by **DCA**. If $V(\Omega) \subset \{0,1\}^{2^n}$, then the assertion is trivial. Otherwise, let $f^r \in \{0,1\}^{2^n}$ and $f^{r+1} \in V(\Omega)$ be an optimal solution of the linear program (20). Then from (i) of this theorem we have

$$\Psi(f^{r+1}) + tp(f^{r+1}) \leq \Psi(f^r) + tp(f^r).$$

Since $p(f^r) = 0$, it follows

$$tp(f^{r+1}) \leq \Psi(f^r) - \Psi(f^{r+1}) \leq \Psi(f^r) - \eta.$$

If $p(f^{r+1}) > 0$, then

$$t \leq \frac{\Psi(f^r) - \Psi(f^{r+1})}{p(f^{r+1})} \leq \frac{\Psi(f^r) - \eta}{\theta} \leq t_1$$

which contradicts the fact that $t > t_1$.

(iii) Since (Q_{dc}) is a polyhedral DC program, DCA has a finite convergence (property v), Sect. 4.1), say, the sequence $\{f^k\}$ converges to $\{f^*\} \in V(\Omega)$ after a finite number of iterations. Moreover $\{f^*\}$ is a critical point of $G - H$, i.e.

$$\partial G(f^*) \cap \partial H(f^*) \neq \emptyset. \tag{21}$$

If $f^*_x \neq 1/2, \forall x \in B^n$, then H is differentiable at f^* and then the condition (21) becomes $\partial H(f^*) \subset \partial G(f^*)$. This subdifferential inclusion is the necessary and sufficient local optimality condition for a polyhedral DC program whose H is polyhedral convex function. The proof is then complete. □

Finding a good starting point for DCA

How to find a good starting point is an important question in the use of DCA. For this, we relax the balanced constraint and generalize a Bent function. A Bent function is a boolean function that has the maximum possible nonlinearity. Bent functions have an even number of variables. The construction of Bent functions is available at http://www.ii.uib.no/~mohamedaa/odbf/index.html. When n is an odd number we solve the concave quadratic programming problem by applying again the DCA developed in [5]

$$0 = \min \left\{ \sum_{x \in B^n} f_x(1 - f_x) : f \in K \right\} \tag{22}$$

and use the computed solutions as starting points for DCA.

5 The combined DCA–GA algorithm

The use of evolutionary computation (EC) techniques to evolve solutions for both theoretical and practical problems has seen a dramatic increase in popularity and success over last decade. The most popular and widely applied EC technique is called Genetic Algorithm (GA) ([9]) whose computational scheme is based on a single population of individuals representing a single species. Genetic algorithms are simple general purpose procedures, which can be applied to many different search and optimization problems. They are able to explore huge search spaces quickly and to find good regions of individuals with high fitness.

Hybridization of GAs for optimization purposes has been widely studied in the past years, combining the capacity of GAs to explore huge search spaces and good regions of solutions exploiting the power of local search algorithms.

In order to use a genetic algorithm for solving this problem, we need to apply a set of transformation to our problem. It is, for instance, necessary to convert the notion of Boolean functions (our phenotype) to a genetic representation of the individuals(genotype), then we need to map our objective function to a fitness function. Eventually we also need to define the genetic operators (selection, crossover and mutation).

Representation How to represent a problem in the genetic algorithms (GAs) is a key issue. It has a direct influence on the performance and on the choice of genetic operators. We already saw that a Boolean function can be presented in the form of a binary vector. Thus naturally, in the proposed genetic algorithm, we use a binary coding to represent a Boolean function.

Initial population For binary strings, we can consider that the size of initial population N can be determined by ([12]) : $N = [1 + \log(-l(/\ln P_2^*)/\log 2]$ (P_2^* denotes the probability that at least one allele is present at each locus can be found and l is the length of binary string). The initial population of functions is randomly generated under the constraint of being balanced.

Fitness function Naturally, we use nonlinearity of function (6) as fitness function in our algorithm.

Selection This operator represents the fact that fitter individuals have a greater chance to survive and to produce offspring. We use an elitist procedure to preserve a number of the best individuals of the tth generation in the $(t + 1)$th generation. For selecting the individuals to

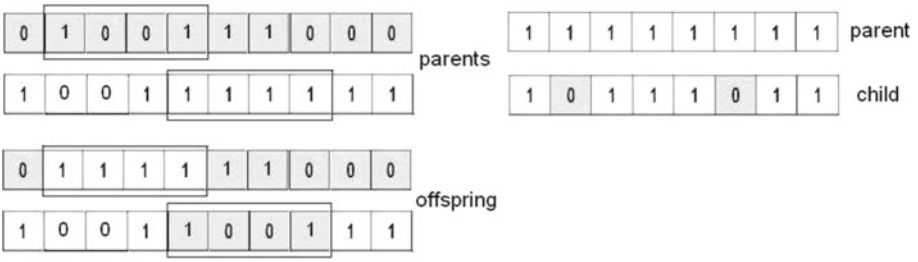

Fig. 1 Crossover operator ($left$) mutation operator ($right$)

be operated (crossover) we use a tournament selection: n individuals are selected randomly from the population ($n = 2$ in our algorithm). The best fit individual will be the winner of the tournament.

Crossover We saw that in our problem, we are interested in the balance criteria, which means that the number of 1 and 0 shall be equal. For that, we must adopt a crossover operator that preserves the balance of the Boolean function. More precisely, starting from two parents that are balanced, we shall obtain two children that also are balanced functions. The procedure is simple: for each relative, we randomly select a balanced sub-sequence and the crossover operation consists of exchanging these two sub-sequences. We used a crossover probability $p_c = 0.6$.

Mutation The mutation allows to avoid early convergence during the evolutionary process. We experimentally tuned the mutation rate to $p_m = 0.005$ in our algorithm by exploring the range $p_m \in [0.001, 0.01]$. For preserving the balance of Boolean functions, we ensure that the total number of mutations is even (Fig. 1).

Termination condition We terminate the GA after a fixed number of iterations denoted N_g.

The combined DCA–GA algorithms

For exploiting the efficiency of both DCA and GA we propose the combined DCA–GA method that is composed of two phases: the first phase consists of applying GA while in the second phase DCA is used. The idea is motivated by the fact that GA can find a "good" feasible solution to Problem (Q) and then applying DCA from this solution provides a better solution, due to Theorem 1. We develop here below two versions of the combined DCA–GA method that differ by the phase 1: in the first one (**DCA–GA-1**) we use GA only while in the second one (**DCA–GA-2**) we apply the DCA at each iteration of GA when a new individual which is better than the best current individual (of which the nonlinearity is denoted N_f^*) is detected.

The algorithms are given below:

DCA–GA-1 scheme

Phase 1—GA

Generate initial population P_t of N balanced functions.

$nbIteration \leftarrow 0$

Evaluate population P_t.

While $nbIteration \leq N_g$

Insert N_e best individuals of P_t in P_{t+1}
For i $\leftarrow$ 1 to $(N - N_e)$
Select individuals i_1 and i_2 from P_t
i$'$ $\leftarrow$ Crossover(i_1, i_2)
i$''$ $\leftarrow$ Mutate(i$'$)
insert individual i$''$ in P_{t+1}
$P_t \leftarrow P_{t+1}$
Evaluate population P_t
Choose the best individual of the final population T^*
Phase 2—DCA: Apply DCA with starting point T^* until its convergence.

DCA–GA-2 scheme
Phase 1—GA using DCA
Generate initial population P_t of N balanced functions
Evaluate population P_t.
$nbIteration \leftarrow 0$
While $nbIteration \leq N_g$
For i $\leftarrow$ 1 to $(N - N_e - 2)$
Select individuals i_1 and i_2 from P_t
i$'$ $\leftarrow$ Crossover(i_1, i_2)
i$''$ $\leftarrow$ Mutate(i$'$)
insert individual i$''$ in P_{t+1}
Evaluate population P_{t+1}
Choose the two best individuals T_1 and T_2 in P_{t+1}
If $N_f(T_1) \geq N_f^*$ (resp. $N_f(T_2) \geq N_f^*$) then perform q iterations of DCA
from starting point T_1 (resp. T_2) to obtain solution T_1^* (resp. T_1^*2) and
insert T_1^* (resp. T_2^*) in P_{t+1}
Insert N_e best individuals of P_t in P_{t+1}.
$P_t \leftarrow P_{t+1}$.
Choose the best individual $T^*$of the final population
Phase 2—DCA: apply DCA from starting point T^* until its convergence.

Since the genetic operations keep T_1 and T_2 in balance, they are feasible solutions for the Problem (13). Furthermore, as DCA is a descent method, performing some iterations of DCA from two children T_1 and T_2 always improves their quality in terms of nonlinearity. Nevertheless applying DCA until its convergence in Phase 1 (for finding a good starting point of DCA in Phase 2) did not appear to be interesting from a numerical point of view.

6 Computational results

The algorithms have been implemented in C and tested on a PC Duo CPU(3.2 MHz, 4MB cache) equipped with 16 GB RAM. The CPLEX code (version 10.1) has been used for solving the linear program (20). We take $b = 0$ in the balanced constraint (11), $\epsilon = 10^{-6}$ in **DCA**. In **GA** we set $N_e = 4, N_g = 20$ when $n \leq 13$ and $N_g = 10$ when $n > 13$. In **DCA–GA-2** the value of q (the number of iterations of **DCA** in Phase 1) is 5 when $n \leq 11$, 2 when $11 < n \leq 13$ and 1 when $n > 13$.

In the first experiment, we compare in terms of nonlinearity the performance of our three algorithms based on DCA: **DCA**, **DCA–GA-1** and **DCA–GA-2** with three probabilistic algorithms: our GA proposed and two standard algorithms [2], namely a combination of random search for Boolean functions with high nonlinearity and Hill Climbing algorithm

(**R HC**), and **GA HC** - the best genetic algorithm studied in [2]. In [2] several versions of Genetic Algorithms with and without Hill Climbing are proposed among them the genetic algorithm with Hill Climbing (**GA HC**) is the best.

In the Table 2 we present the best nonlinearity achieved after testing 10, 000 functions of **R HC**, **GA HC** and the nonlinearity given by our algorithms with n varying from 8 to 15. When $n \geq 16$ the CPLEX code fails to solve the linear program at the very initial step of reading the data (remember that the dimension of our optimization problem is 2^n variables and 2^{n+1} constraints). We also indicate there the number of iterations (N^0it) and the computational times (Time) of our algorithms (in the combined **DCA–GA-2** N^0it is the number of iterations of DCA in Phase 2 while Time is the total computational times of both phases). The performance of the first phase of the combined algorithm **DCA–GA-2** is reported in the column "DCA–GA-2 1st phase" (note that the results of **GA** is also the one of the first phase of **DCA–GA-1**). The computational times of **R HC**, **GA HC** have not been indicated in [2].

In the second experiment we study the effect of the number of generations of GA (N_g) in our combined DCA–GA algorithms. In the Table 3, we present the numerical results of DCA–GA-1 and DCA–GA-2 with the different number of N_g in GA for the case $n = 9$.

Comments on the computational results

(i) In all case the nonlinearity Boolean functions obtained by our three algorithms based on DCA are better than that of the two standard algorithms **R HC**, **GA HC** (except for the case $n = 14$) and our **GA**. In particular even DCA without phase 1 is better than **GAs**. Moreover they are all balanced functions. It is worth mentioning that a *slight* improvement of the nonlinearity may involve a *significant* improvement of block ciphers. Indeed each weakness of a cipher may be exploited by cyber-criminals. And any information that can be gathered can make a deep cut in the cryptanalysis search space. It is however difficult to predict the size of this kind of impact without specifying the rest of the cipher (remember that S-box is only one of the crypto-components). The authors of AES believe that AES is secure even if we replace its S-box by one showing some linear features, but for other ciphers, the major part of the cipher quality relies on its S-box.

(ii) The combined DCA–GA algorithms are always better than DCA in terms of nonlinearity. Moreover **DCA–GA-2** is the best. The results show that GA using DCA (Phase 1 of **DCA–GA-2**) is the best way for finding a good starting point to DCA. Nevertheless, this interesting combination (GA using DCA) is not sufficient for obtaining good results: the second phase improves significantly the results of the first one (the same observation for **DCA–GA-1**). This means that DCA is crucial in our approaches.

(iii) Not surprisingly, the combined algorithms are more expensive than DCA. The ratio of computational times between DCA and **DCA–GA-1** (resp. **DCA–GA-2**) is in the interval [0.45, 1.0] (resp. [0.4, 0.74]). On the other hand, in **DCA–GA-1** (resp. **DCA–GA-2**) the computational times of the first phase varies between 13% ($n = 14$) and 54% ($n = 9$) (resp. 36% ($n = 13$) to 67% ($n = 8$)) of the total times. We note that the computational time is not relevant in this work where we restrict ourself to ciphers with fixed table, since the fixed S-boxes are built once for ever (while for the dynamic ones, it is indeed more important).

(iv) In phase 1 of the combined DCA–GA algorithms, it is not needed to wait for the GA to converge in order to reach the best solution: DCA–GA-1 and DCA–GA-2 are able to obtain the same results even if we stop GA before 40 generations.

Table 2 Comparative results between the six algorithms

n	DCA			GA			DCA–GA-1			DCA–GA-2 1st Phase			DCA–GA-2			GA HC	R HC
	N_f	Time	N^0it	N_f	Time	N^0it	N_f	Time	N^0it	N_f	Time	N^0it	N_f	Time	N^0it	N_f	N_f
8	**116**	1 min	8	112	1 min	20	**116**	2.1 min	7	**116**	1.2 min	20	**116**	1.8 min	5	114	114
9	234	2.2 min	5	230	2.4 min	20	**238**	4.4 min	6	232	2.8 min	20	**238**	4.2 min	4	236	232
10	476	17.5 min	11	478	8.3 min	20	486	22.3 min	9	482	9.7 min	20	**488**	24 min	8	482	476
11	974	1 h	9	974	20 min	20	976	1.25 h	4	978	1.5 h	20	**980**	2.5 h	5	**980**	968
12	1984	3.0 h	10	1,976	30 min	20	**1,998**	3.0 h	6	1,996	2.7 h	20	**1,998**	5.5 h	9	1,980	1,961
13	3,998	6.4 h	5	3,990	1.2 h	20	4,002	6.9 h	5	3,998	2.9 h	10	**4,028**	8.0 h	6	3,994	3,968
14	8,024	7.8 h	5	8,014	1.2 h	20	8,028	9 h	3	8,030	5.2 h	10	8,030	10.5 h	2	**8,036**	7,996
15	16,186	10.2 h	3	16,120	2.5 h	20	**16,248**	13.2 h	2	16,198	8.2 h	10	**16,248**	16 h	2	16,144	16,085

Table 3 Result with different number of the generations of GA

N_g	GA			DCA–GA-1			DCA–GA-2		
	N_f	N^0it	Time	N_f	N^0it	Time	N_f	N^0it	Time
10	228	10	57 s	236	7	2 min 47 s	**238**	7	3 min 02 s
20	230	20	1 min 28 s	**238**	6	3 min 17 s	**238**	6	3 min 46 s
30	232	30	1 min 45 s	**238**	6	3 min 52 s	**238**	8	4 min 02 s
40	232	40	2 min 12 s	**238**	6	4 min 23 s	**238**	6	4 min 31 s

7 Conclusion

We have proposed, for constructing highly nonlinear balanced Boolean Functions in Cryptography, a new and efficient approach based on a hybrid approach mixing GA and DCA. The considered combinatorial optimization problem has been beforehand reformulated as a DC program with a natural choice of DC decomposition, and the resulting **DCA** then solves a finite sequence of linear programs. **DCA** is original because it gives an integer solution while it works in a continuous domain.
The results show that DCA is an efficient approach for this problem and both **DCA** and the combined **DCA–GA** algorithms are significantly superior to standard heuristic approaches.

References

1. Canteaut, A., Carlet, C., Charpin, P., Fontaine, C.: Propagation characteristic and correlation-immunity of hight nonlinenar Boolean function. Advances in Cryptographie—EUROCRYPT 2000, LNCS, vol. 1807 Springer (2000)
2. Clark, J.A.: Optimisation heuristics for cryptology. PhD thesis, Faculty of Information Technology, Queensland University of Technology (1998)
3. Clark, J.A., Jeremy, J.L., Stepney, S.: The design of S-boxes by simulated annealing, CEC 2004: International Conference on Evolutionary Computation, Portland OR, June (2004), pp. 1533–1537. IEEE (2004)
4. Le Thi, H.A., Pham, D.T.: Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. J. Glob. Optim. **11**(3), 253–285 (1997)
5. Le Thi, H.A, Pham, D.T.: A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. Optimization **50**, 93–120 (2001)
6. Le Thi, H.A., Pham, D.T.: The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
7. Le Thi, H.A., Pham, D.T., Huynh, V.N.: Exact penalty techniques in DC programming. Technical Report LMI-INSA Rouen, pp. 1–28 (2005)
8. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1994)
10. Pham, D.T., Le Thi, H.A.: Convex analysis approach to DC programming: theory, algorithms and applications. Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, 22, pp. 289–355 (1997)
11. Pham, D.T., Le Thi, H.A.: DC optimization algorithms for solving the trust region subproblem. SIAM J. Optim. **8**, 476–505 (1998)
12. Reeves, C.R., Rowe, J.E.: Genetic Algorithms-Principles and Perspectives. ISBN: 0306480506, Kluwer (2002)

13. Sarkar, P., Maitra, S.: Construction of nonlinear Boolean functions with important cryptographic properties. In: Advances in Cryptographie–EUROCRYPT 2000, Lecture Note in Computer Science, vol. 1807 pp. 485–506. Springer (2000)
14. Seberry, J., Zhang, X.M., Zheng, Y.: Nonlinearly balanced Boolean functions and their propagation characteristics. Advances in Cryptographie—EUROCRYPT 2000 In: Advances in Cryptology—CRYPT0'93, Springer (1994)
15. Stallings, W.: Cryptography and Network Security. 3rd edn. Prentice Hall, Englewood Cliffs (2003)