10.24425/acs.2022.143666

Archives of Control Sciences
Volume 32(LXVIII), 2022
No. 4, *pages 663–676*

Nonlinearity of incomplete Boolean functions: prioritizing spectra calculation

Piotr PORWIK

In this paper, a class of linear Boolean functions is analyzed. The Boolean function can be represented as disjoint cubes or in the form of a truth vector. The primary purpose of this analysis is to decide whether an incompletely defined function can be extended to a complete linear form. A simple algorithm for generating all states of this function has been proposed if the Boolean function can have a full representation. The algorithm is beneficial for large functions. The proposed approach can be applied to completely and incompletely defined Boolean functions.

Key words: Boolean function, numerical methods, Walsh transform

1. Introduction

Incompletely defined Boolean functions are widely used because such forms are flexible and convenient in many practical applications [2, 3]. A proper definition of the Boolean function requires the knowledge to which class the Boolean function belongs. The next question is how to find the extension of a given partially defined Boolean function and realize the function in affine or linear form. For an incompletely defined Boolean function, evaluating whether the function can be realized as linear can be difficult, especially for larger functions. These difficulties can be overcome. Spectral methods provide an efficient representation of Boolean functions, allowing one to discover properties that cannot be observed in their original domain. One of such method is the Fast Walsh-Hadamard Transform (FWHT) [15]. The Walsh transform is straightforward and uses a butterfly algorithm, similar to the well-known Fourier transform. Unfortunately, the main drawback of spectral techniques is the high computational cost for large Boolean functions. The FWHT, as well as the fast Möbius

P. Porwik (e-mail: piotr.porwik@us.edu.pl) is with Faculty of Science and Technology, University of Silesia, Będzińska 39, 41-200 Sosnowiec, Poland.

Received 20.04.2022. Revised 14.10.2022.

transform, have exponential complexity, increasing rapidly with the number of variables of the Boolean function [5, 17]. The aforementioned methods allow the analysis of functions of up to 40 variables due to rapidly increasing memory occupancy. Calculating the coefficients of the Walsh-Hadamard spectrum can be implemented directly from a reduced representation of the Boolean function – arrays of disjoint cubes. If the Boolean function is fully defined, then its linear forms can also be recognized by known transformations, such as the Reed-Muller or Möbius form. Unfortunately, the mentioned methods are useless when the function is only partially defined [11]. When the function is incomplete, it is essential to design an efficient method to construct the completely specified form if possible.

In addition to spectral techniques, approaches based on multivariate polynomials can be applied to measure nonlinearity of Boolean functions [4, 5], as well. Another approach offers quantum computations with oracle [17]. Quantum algorithms can also identify completely as well as incompletely defined linear Boolean functions using a query to the oracle. This approach uses the property that Boolean linear and affine functions are always balanced – the truth table of a function contains an equal number of 0's and 1's. In such cases, quantum algorithms based on the Bernstain-Vazirani or Deutsch-Jozsa idea are employed [16, 17]. Bernstein-Vazirani's algorithm (the one-query algorithm) can identify the linear Boolean function using a single query to the oracle. The complexity of the problem is measured by the number of oracle queries required to discover the exact form of the function. Classical computations would require $O(n)$ oracle calls; however, it can be solved using a single query to the oracle using Bernstein-Vazirani's quantum algorithm. Unfortunately, the direct use of this algorithm does not allow the detection of an affine function [17]. If a function is partially determined, the quantum technique allows the identification of a particular class of incompletely specified Boolean affine functions with a success probability of at least 2/3. The probability of success depends on the number of don't cares in a given function [17].

In the approach proposed in this paper, identifying a linear or affine Boolean function is always possible. Moreover, in the case of an incompletely specified function, it can be verified that the function has an expansion to a linear (affine) form. These possibilities are advantages of the presented method. The suggested in the paper spectral analysis overwhelms the limitations of other ways.

2. Boolean function

Let $\mathbf{F}_2$ be a Galois Field having the addition (modulo 2) operator $\oplus$. Such a field is convenient for the processing of logic signals. The symbol + will be used to denote the addition over integers and the domain of an n-variable Boolean

function is denoted by the vector space $(\mathbf{F}_2^n, \oplus)$. Let $\mathbf{B}^n$ be the vector space of dimension n over the field $\mathbf{F}_2^n$. A Boolean function f, of n variables is the mapping $f\colon \mathbf{F}_2^n \to \mathbf{F}_2$ or, in other words, $f\colon \mathbf{B}^n \to \mathbf{B}^1$. A Boolean function $f(x_{n-1}, \ldots, x_1, x_0)$ can also be interpreted as the output column of the truth table of f, i.e. a vectors of binary elements $\mathbf{F}_2^n = \{\mathbf{p}_1, \ldots, \mathbf{p}_{2^n}\}$. This domain is a set of 2^n binary vectors having elements: $(0, 0, \ldots, 0)$, $(0, 0, \ldots, 1)$, $\ldots$, $(1, 1, \ldots, 1)$, say in lexicographic order. Thus, the Boolean function is a set of ordered pairs in which the first element is a binary input vector and the second element is a constant, either 0 or 1. If the order of $\mathbf{p}_i \in \mathbf{F}_2^n$ is fixed, then f is uniquely identified.

If the domain $\mathbf{D}$ of the Boolean function f is $\mathbf{B}^n$ then f is a completely specified. However if $\mathbf{D} \subsetneq \mathbf{B}^n$ then f is incompletely defined. If some values of the function f belong to the set $\mathbf{B}^n \backslash \mathbf{D}$ then f is also incompletely defined. The points where a value for f is not assigned will be denoted as *don't care*. The binary vectors for which the function is undefined will be marked as DC cubes and the power of the set DC will be marked as $card(DC) = d$. The undefined values of this Boolean function will be denoted by the symbol "–". Hence, the mapping $f\colon \mathbf{B}^n \to \mathbf{B}^1 \cup \{-\}$ is an n-variable incompletely specified function.

Definition 1 *Let the number of arguments of the Boolean function f_k be denoted by n. The Boolean function $f_k(x_n, x_{n-1}, \ldots, x_1)$ is called affine if f_k can be represented in the form $f_k(x) = a_n x_n \oplus a_{n-1} x_{n-1} \oplus \ldots \oplus a_1 x_1 \oplus c$, where a_j, $c \in \{0, 1\}$, and $k = c + \sum_{i=1}^{n} a_i 2^i$. In particular, if $c = 0$ then f_k is a linear Boolean function, so functions f_k are either linear Boolean functions or their compliments.*

Definition 2 *Boolean function f is not fully defined if $\#\{f(x) = \text{"}-\text{"}\} \geqslant 1$ and is weakly defined, when $\#\{f(x) = \text{"}-\text{"}\} \geqslant 2^{n-1}$, where symbol # denotes the cardinality of set, and symbol "–" denotes undefined place in the truth vector of f, so $f\colon f^n \to y \in \{0, 1, -\}$.*

There are various Boolean forms such as linear, bent, majority, balanced, threshold, and many others [6, 13]. The linearity (or nonlinearity) of binary functions is widely utilized in cryptography, data encryption, the development and analysis of ciphers, the generation of error-correcting codes, Reed-Muller forms, and many other domains [5, 6, 16, 17].

In some cases, a linear function is desirable. In others, it is not. For example, linear structures are highly desirable in optimizing digital circuits, while nonlinear functions are employed in encryption. So knowing the class of the function seems to be necessary.

3. The Walsh-Hadamard transform

In many cases, particular properties of the original problem can be more conveniently interpreted when transferred isomorphically from the original space to another one. The best example is the well-known Fourier Transform. Spectral or Fourier analysis is widely used in mathematics, and engineering [14, 15]. The theory of Fourier analysis is complex and sometimes complicated, although many algorithms are derived from it. In this paper, the more convenient discrete Walsh-Hadamard transform will be used. In practice, it is a discrete Fourier transform in the Walsh basis, and the Walsh functions are ordered according to the Hadamard rules [1, 15]. Given this approach, matrix notation offers a simple way to define such discrete functions.

Discrete Walsh functions of order n, denoted as $wal(w,t), w,\ t = 0,\ldots,2^n-1$ are defined as the rows of the $2^n \times 2^n$ Hadamard matrix [15]:

$$\mathbf{H}_n = \bigotimes_{i=1}^{n} \mathbf{H}_1 = \mathbf{H}_1 \otimes \mathbf{H}_{n-1} = \underbrace{\mathbf{H}_1 \otimes \ldots \otimes \mathbf{H}_1}_{n\text{-times}}, \tag{1}$$

where $\mathbf{H}_0 = [1]$, $\mathbf{H}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, and $\otimes$ denotes the Kronecker product.

In this notation, w identifies the index of the Walsh function, while t indicates a discrete point within the function's determination interval. The Hadamard matrix comprises discrete Walsh functions ordered in the Hadamard order.

A Boolean function f, represented by the binary truth-vector $\mathbf{\Delta}_f = [y_0,\ldots,y_{2^n-1}]^T$, can be transformed from the Boolean domain $\{0,1\}$ into the spectral domain. If the values of the truth vector of the Boolean function f are encoded using the mapping $\{0,1\} \rightarrow \{1,-1\}$, then instead of $\mathbf{\Delta}_f$, we denote such a vector as $\mathbf{Y}_f$. For a given vector $\mathbf{Y}_f$, the inner product $s_i = \langle \mathbf{Y}_f, wal(i,n) \rangle$ gives the correlation between the Boolean function f and the appropriate i-th Walsh function, where $i,\ n = 0,\ldots,2^n-1$. It means that a complete forward Walsh-Hadamard Transform is defined as:

$$\mathbf{s}_f = \mathbf{Y}_f \mathbf{H}_n \tag{2}$$

and

$$s_i = \sum_{n=0}^{2^n-1} y_i\ wal(i,n), \qquad i = 0,\ldots,2^n-1\,. \tag{3}$$

Variables n of a fully defined Boolean function f can be equivalently represented by a complete set of spectral coefficients. It follows from the fact that spectral

coefficients are specifically associated with input variables of the function f:

$$f(x_{n-1},\ldots,x_1,x_0) = \frac{1}{2^{n+1}}\left[2^n - s_0 - s_1\cdot(-1)^{x_0} - s_2\cdot(-1)^{x_1} - s_3\cdot(-1)^{x_0\oplus x_1}\ldots - s_{2^n-1}\cdot(-1)^{x_0\oplus x_1,\ldots,\oplus x_{n-1}}\right]. \quad (4)$$

Spectral coefficients can be applied in the process of function recognition. The s_0 coefficient is directly related to the number of minterms of Boolean function, then $s_0 = 2^n - 2a$ [8]. The number of minterms having the value of 1 will be denoted by a. The Walsh-Hadamard Transform can also be organized as a "fast" form (FWHT), similarly, as well-known, the FFT (Fast Fourier Transform) [10, 15]. Computational complexity and space complexity are the same as in FFT: $O(n2^n)$, and $O(2^n)$, respectively [1].

Instead of the classical representation of a Boolean function by truth vector, a function can be represented more compactly using disjoint cubes [7–9]. This representation is advantageous for large Boolean functions. Representing functions as cubes, however, requires a different method of Walsh spectrum calculating.

4. Disjoint cubes of Boolean function

Definition 3 *The true (false) set of the Boolean function f is denoted by $T_f(F_f)$. All the true vectors of f are denoted as $T_f = \{x \in \{0,1\}^n : f(x) = 1\}$ – it is the set of ON cubes. All the false vectors of f are denoted as $F_f = \{x \in \{0,1\}^n : f(x) = 0\}$ – it is the set of OFF cubes.*

For large Boolean functions, even a fast transform is not helpful due to the complexity of the problem. For such functions, representation employing disjoint cubes is more efficient. Based on previous considerations, the following subset is additionally introduced:

$$DC = \{x \in \{0,1\}^n : \ f(x) = \text{"–"}\}, \quad (5)$$

where the sign "–" is the don't care (DC) element. Elements of the subsets ON and DC can be combined into cubes. The remaining values form the OFF cubes. Cubes should be disjoint – it means that cubes intersection is prohibited [9]. An example of disjoint cubes of ON and DC type is shown in Karnaugh's map in Fig. 1. The OFF cube is not necessary. Therefore, it is not selected.

Based on a method described informally in [8], let k be the number of disjoint cubes of the type ON. Let w be the number of disjoint cubes of the DC type. Let symbol p denote the number of "–" type elements in the cube.

The cube's index for which the spectrum is determined is denoted by j. Spectral coefficients of the function f form the vector $\mathbf{s}_f = [s_0,\ldots,s_{2^n-1}]$.

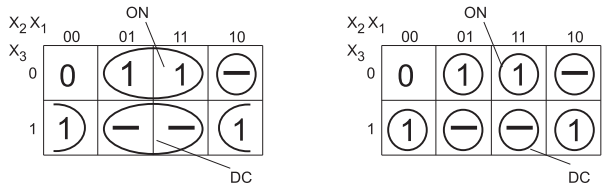

Figure 1: The Karnaugh map of the Boolean function $y = f(x_3, x_2, x_1)$, where disjoint *ON* and *DC* cubes are indicated. The *OFF* cube was not selected. Optimal cube selection (left) and non optimal (right)

Spectrum will be determined separately for the *DC* and *ON* cubes only:

$$s^{ON}_{0_{(j)}} = 2^n - 2^{p+1}, \qquad s^{DC}_{0_{(j)}} = 2^{n-1} - 2^p, \tag{6}$$

$$s^{ON}_{1(j)} = \begin{cases} (-1)^{1+x_1} \cdot (2^{p+1}) & \text{for } x_1 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{7}$$

$$s^{DC}_{1(j)} = \begin{cases} (-1)^{1+x_1} \cdot 2^p & \text{for } x_1 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{8}$$

$$s^{ON}_{2(j)} = \begin{cases} (-1)^{1+x_2} \cdot (2^{p+1}) & \text{for } x_2 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{9}$$

$$s^{DC}_{2(j)} = \begin{cases} (-1)^{1+x_2} \cdot 2^p & \text{for } x_2 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{10}$$

$$s^{ON}_{12(j)} = \begin{cases} (-1)^{(1+x_1+x_2)} \cdot (2^{p+1}) & \text{if } x_1 \neq \text{“–”} \wedge x_2 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{11}$$

$$s^{DC}_{12(j)} = \begin{cases} (-1)^{(1+x_1+x_2)} \cdot 2^p & \text{if } x_1 \neq \text{“–”} \wedge x_2 \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{12}$$

$$\ldots \tag{13}$$

$$s^{ON}_{123\ldots n(j)} = \begin{cases} (-1)^{1+\sum_{m=1}^{n} x_m} \cdot (2^{p+1}) & \text{for all } x_1, x_2, \ldots, x_n \neq \text{“–”}, \\ 0 & \text{otherwise;} \end{cases} \tag{14}$$

$$s^{DC}_{123\ldots n(j)} = \begin{cases} (-1)^{1+\sum_{m=1}^{n} x_m} \cdot 2^p & \text{for all } x_1, x_2, \ldots, x_n \neq \text{“–”}, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where $x_i \in \{0, 1, -\}$. The individual elements of the spectrum $\mathbf{s}_f = [s_0, \ldots, s_{2^n-1}]$ are determined as follows. Zero coefficient:

$$s_0 = \left(\sum_{j=1}^{w+k} s_{0_{(j)}} \right) - (k-1) \cdot 2^n - w \cdot 2^{n-1}, \tag{16}$$

and another coefficient:

$$s_i = \left(\sum_{j=1}^{w+k} s_{i_{(j)}} \right). \tag{17}$$

The above calculations can be performed separately or in parallel. Table 1 shows the steps for calculating the spectrum for a function represented by disjoint cubes.

Table 1: Spectrum of the Boolean function represented by disjoint cubes

x_3	x_2	x_1	cube type	s_{0_j}	s_{1_j}	s_{2_j}	s_{12_j}	s_{3_j}	s_{13_j}	s_{23_j}	s_{123_j}	index of cube
1	–	0	*ON*	4	–4	0	0	4	4	0	0	$j = 1$
0	–	1	*ON*	4	4	0	0	–4	4	0	0	$j = 2$
0	1	0	*DC*	3	–1	1	1	–1	–1	1	1	...
1	–	1	*DC*	2	2	0	0	2	–2	0	0	$j = w + k$
s				**–3**	**1**	**1**	**1**	**1**	**5**	**1**	**1**	

Representation of the disjoint cubes from Table 1 employing the Karnaugh map is depicted previously in Fig. 1 (left). A similar table can be constructed for non-optimal cubes selection, see Fig. 1 (right). In such a case, the table will have 7 rows with different local spectral coefficients, but a final spectrum **s** is the same as in Table 1. In most cases, the number of cubes of the Boolean function is much smaller than 2^n. The value of 2^n is critical in the algorithms based on the FWHT approach. The cube-based spectrum determination is devoid of these drawbacks, but in some cases, the computational complexity may be less favorable than in the FWHT method. This is shown below.

5. The spectral identification of a Boolean function

Corollary 1 *Any fully defined linear Boolean function f of n variables, encoded by means of the mapping $\{0,1\} \rightarrow \{1,-1\}$, is characterized by the following unique Walsh-Hadamard spectrum distribution [12]:*

$$s_i = \begin{cases} (-1)^c \times 2^n & \text{for } i = (k-c)/2, \\ 0 & \text{otherwise}, \end{cases} \tag{18}$$

where k and c have the same meanings as previously and $i = 0, 1, \ldots, 2^n - 1$.

Let Boolean function f is undefined in one point, so f can be extended to a linear (affine) form, if coordinates of some i-th row of $\mathbf{H}_n$, and the coordinates of $\mathbf{Y}_f$ are the same (or complementary), except for the undefined coordinate. This

means that the 2^n-1 coordinates of the i-th row of $\mathbf{H}_n$ and $\mathbf{Y}_f$ are the same ($c = 0$) or complementary ($c = 1$), hence either $s_i = (2^n - 1) \times (1) = 2^n - 1$, if $c = 0$, or $s_i = (2^n - 1) \times (-1) = -(2^n - 1)$, if $c = 1$. If the Boolean function f satisfies the assumptions of Corollary 1, then this function can be formed as an affine (linear) if undefined point in the vector $\mathbf{\Delta}_f$ is replaced by a value 0 (for $c = 1$) or 1 (for $c = 0$).

Observation. Let f be a partial Boolean function, represented by encoded truth vector $[y_0, y_1, \ldots, y_{2^n-1}]$, where $y_i \in \{+1, -1, 0\}$. Let f be undefined at d points, $y_i = 0$ for $i = 1, \ldots, d$. Function f can be extended up to the affine (linear) as follows:

a. If all undefined points are located within in the set T_f, then $s_0 = +d$.

b. If all undefined points are located within in the set F_f, then $s_0 = -d$.

c. If all undefined points are located in the set $T_f \cup F_f$, then $s_0 \neq d$ and for some i, $s_i = \pm(2^n - d) = (-1)^c \cdot (2^n - d)$.

6. Algorithm of extension of Boolean function to linear form

Input: The truth-vector $\mathbf{\Delta}_{f_k}$ of the n variable Boolean function f_k. The number of undefined points d in the vector $\mathbf{\Delta}_{f_k}$.

Output: All extensions (if any exist) of the partially defined Boolean function f.

1. For a given Boolean function, determine how the Walsh-Hadamard spectrum will be calculated:

   a. **If spectrum will be calculated by the Walsh-Hadamard Transform, then:** encode the truth-vector according to the mapping $\mathbf{\Delta}_{f_k}$: $\{0, 1, -\} \rightarrow \{1, -1, 0\}$, so $\mathbf{\Delta}_{f_k} \rightarrow \mathbf{Y}_k$. Calculate the Walsh-Hadamard spectral coefficients $s_i \in \mathbf{s}$, $i = 0, 1, \ldots, 2^n - 1$ of the vector $\mathbf{Y}_k$, by using (3) or the FWHT.

   b. **If the spectrum is calculated by disjoint cubes, then:** spectral coefficients are calculated from the formulas (6)–(17). Finally, the same spectral coefficients $s_i \in \mathbf{s}$, identically ordered, are computed.

2. Calculate the values:

   a. $\{s_i\} = \max\{|s_i|, i = 0, \ldots, 2^n - 1\}$,

   b. $z = \#\{s_i\}$, where symbol # denotes the cardinality of set.

3. $j = 1$.

4. $i = \underset{j=1,\ldots,z}{\arg} \{(s_i)_j\}$, where j denotes current number of the element s_i with the subscript i.
5. if $s_i \neq \pm(2^n - d)$ then the function f can not be extended up. Go to step 9.
6. if $s_i > 0$ then $c = 0$ else $c = 1$.
7. $k = 2i + c$.
8. $\mathbf{\Delta}_{f_k} = \frac{1}{2}[\mathbf{1} + (-1)^{1-c} \cdot wal(k, t)]$.
9. $j = j + 1$.
10. if $j < z$, then go to step 4.
11. End of the algorithm.

The advantage of the above algorithm is that it can be applied to fully defined and incompletely defined Boolean functions. It does not matter which method is used to calculate the spectral coefficients. For large functions, a method based on disjoint cubes should be preferred because, in most cases, the number of cubes is much smaller than 2^n. The spectrum of smaller functions can be calculated directly using the Fast Walsh-Hadamard (FWHT), as it is simpler in software implementation.

Example 1 Let the truth-vector of a given partial Boolean function f_k have the form $\mathbf{\Delta}_{f_k} = [--1\,0--0\,1]$. Number of the undefined points of f_k is equal $d = 4$. Number of input variables is $n = 3$. The Walsh-Hadamard transform forms the vector of spectral coefficients: $\mathbf{s}_{f_k} = [0\,0\,0\,0\,0 - 4\,0\,4]$.

Therefore $\{s_5, s_7\} = \underset{i=0,\ldots,2^n-1}{\max} \{|s_i|\}$, $s_5 = -2^n + d = -4$ and $s_7 = 2^n - d = 4$. Hence, function f can be extended up to its two forms:

$s_{i=5} < 0 : k = 2i + c = 2i + 1 = 2 \times 5 + 1 = 11$.

$s_{i=7} > 0 : k = 2i + c = 2i + 0 = 2 \times 7 + 0 = 14$.

Taking into account above discussion, we can see that two functions have been found: $f_{11} = 1 \oplus x_2 \oplus x_1$, $\mathbf{\Delta}_{f_{11}} = [10100101]$ and $f_{14} = x_3 \oplus x_2 \oplus x_1$, $\mathbf{\Delta}_{f_{14}} = [01101001]$.

It may also be observed that after mapping $\{0, 1\} \rightarrow \{1, -1]\}$ the function f_{11} has equivalent representation to the discrete Walsh function $(-1) \cdot wal(5, t)$ from the matrix $-\mathbf{H}_3$, and that the function f_{14} is equivalent to the function $wal(7, t)$ from the same matrix $\mathbf{H}_3$.

An algorithmic search of affine (linear) functions has been significantly improved. It is important because, instead of processing all possible Boolean functions (for a given n), only a few functions have to be checked. Brute force search is not practical.

Example 2 If the truth vector of a given Boolean function is performed as $\mathbf{\Delta}_f = [1 - - - - - - - - - - - - - - - - 1 - - - - - - - - 0 - - - - - -]$, $(n = 5)$ then on the basis of proposed algorithm, in a single algorithm pass, eight affine binary functions are recommended: $f_3, f_7, f_{11}, f_{15}, f_{17}, f_{21}, f_{25}, f_{29} = 1 \oplus x_4 \oplus x_3 \oplus x_2$. For the vector $\mathbf{\Delta}_f = [01 - 00110 - 001100110 - 1100101 - 00110]$ we obtain only a single linear function ($s_{x=27} = +32$), which can finally be represented as the binary vector [01100110100110011001100101100110].

7. The complexity of the algorithms

As it was explained, the complexity of the Fast Walsh-Hadamard transform requires $O(n2^n)$ additions and storing $O(2^n)$ elements at each of n calculation stages. The same computation and space complexity offer methods based on multivariate polynomials or Fast Möbius Transform [5]. Unfortunately, for larger functions, the Hadamard $2^n \times 2^n$ matrices $\mathbf{H}_n$ are inconvenient to use. For this reason, the cube representation of the Boolean function is preferred as the more compact description of large functions.

Disjoint cubes can be generated based on the method detailed presented in [8, 9]. Spectral coefficients are directly computed based on *ON* and *DC* cube representation only. From Eqs. (6)–(17) follows, the computational time will increase as the function variables (n) increase, as well as the total number of cubes $(e = w + k)$. For any cube, the 2^n separate spectral coefficients s_i are computed (see Table 1). For e cubes, $e \cdot 2^n$ calculations are performed. If we omit simple addition operations, then for every cube we are dealing with calculations of the type $e \cdot (-1)^a \cdot 2^b$ with approximate $O(1)$ computational complexity for each s_i spectral coefficient, $i = 0, \ldots, 2^n - 1$. The computational complexity of complete spectral coefficients calculations is $O(e2^n)$. It means that for $e \leqslant n$, the computational complexity of the cube-based method is smaller than for the FWHT method. For $n > e$, the computational complexity of the cube-based procedure increases, and it is worse than for FWHT. The cube method requires storing $O(2^n)$ elements for each cube because calculations can be performed "in place". However, the cube-based method overcomes the limitations of large square matrices $\mathbf{H}_n$ of size $2^n \times 2^n$.

In some cases where a given cube contains *DC* points, the spectral coefficient is zero, and calculations are not needed (see Eqs. (6)–(17)). The computational complexity is then more favorable than the Walsh-Hadamard transform.

It is known that Boolean functions can be represented in different ways [3, 4, 9, 10, 13], for example as:

- ANF – Algebraic Normal Form,
- NNF – Numeric Normal Form,

- TT – Truth Table,
- Walsh spectral coefficients,
- Disjoin cubes.

The NNF and ANF representations will not be analyzed here. The interested reader may find extensive literature on this topic. The most important positions are given in the References. Figure 2 shows that Boolean function linearity evaluation is conducted using the truth table (TT) of f. If f is represented as NNF or ANF form, then TT must first be built with a cost of O. Similarly, if linearity of the function will be evaluated based on the polynomial form. In mentioned cases, only complete Boolean functions can be evaluated, which is denoted by the C letter. Linearity of functions, evaluated by Walsh spectral coefficients (FWHT or cube-based representation), allows analysis of both complete (C) and incomplete (I) Boolean functions. This is an advantage of spectral methods because, as shown, it enables the extension of incomplete functions to complete version with the possibility of obtaining a linear form, if possible. As a reminder, in Fig. 2 the brute force method is also mentioned, but it has no practical signifi-

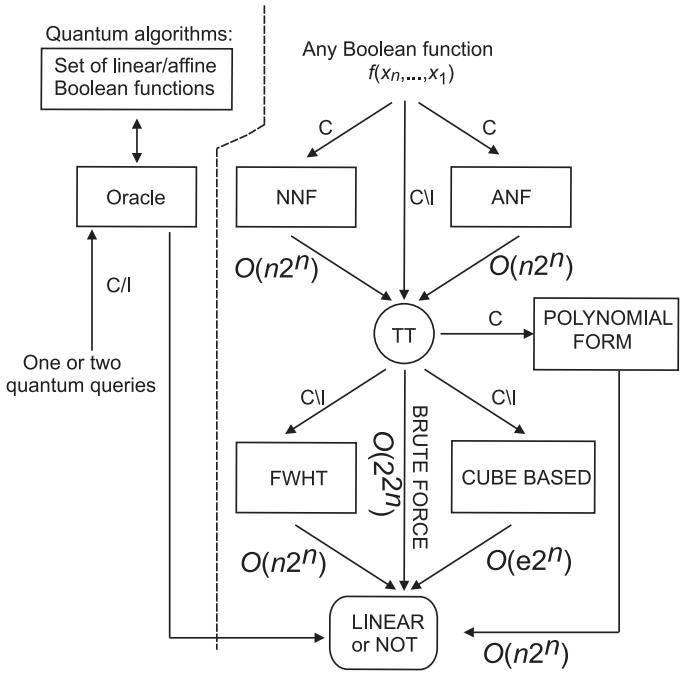

Figure 2: Different representation of a Boolean function $f(x_n, \ldots, x_1)$, along with complexity of transformation to True Table (TT) form and various final representations of function f

cance. As we know, for functions of n variables, 2^{2^n} different Boolean functions can be constructed, of which only 2^{n+1} are linear functions and their complements. The full review method is costly and ineffective, especially for the large n. Quantum algorithms form a separate group of Boolean function classifications. However, quantum algorithms use different mechanisms than Walsh-Hadamard spectrum-based or disjoint cube-based approaches. These algorithms' taxonomy is also shown in Fig. 2.

8. Conclusion

Checking whether a given fully specified Boolean function can be represented according to Definition 1 stated in the Section 2, is very inconvenient. These problems can be overcome by spectral analysis. As shown, the spectral method can be applied to both fully and incompletely defined functions. A fast algorithm will generate its full representation if a partially specified Boolean function can be extended. The calculation of the partial spectra for each cube can be conducted independently if cube based technique is employed. Thus, the computations can be performed in parallel, which speeds up the calculations, especially for large Boolean functions. The algorithm proposed in this work can find all linear and affine extensions obtained from the original, not fully defined Boolean function in a single pass. It should be considered a unique advantage of the method. Spectrum-based algorithms have advantages over methods based on oracle or polynomial computing.

References

[1] N.U. Ahmed and K.R. Rao: *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag, Berlin, Heidelberg, 1975. DOI: 10.1007/978-3-642-45450-9.

[2] M. Andrecut: On the inherent competition between valid and spurious inductive inferences in boolean data. *International Journal of Modern Physics C*, **28**(12), (2017), 1750146. DOI: 10.1142/S0129183117501467.

[3] T. Ayav: Prioritizing MCDC test cases by spectral analysis of boolean functions. *Software Testing, Verification and Reliability*, 27:e1641, 08 (2017). DOI: 10.1002/stvr.1641.

[4] E. Bellini: Yet another algorithm to compute the nonlinearity of a boolean function. *arXiv.org*, 2014. DOI: 10.48550/arXiv.1404.2471.

[5] E. Bellini, M. Sala, and I. Simonetti: Nonlinearity of boolean functions: An algorithmic approach based on multivariate polynomials. *Symmetry*, 14(2(213)) (2022). DOI: 10.3390/sym14020213.

[6] L. Budaghyan, C. Carlet, T. Helleseth, and A. Kholosha: Generalized bent functions and their relation to Maiorana-McFarland class. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 1212–1215. IEEE, 2012. DOI: 10.1109/ISIT.2012.6283049.

[7] R. de Wolf: *A Brief Introduction to Fourier Analysis on the Boolean Cube*. Number 1 in Graduate Surveys. Theory of Computing Library, 2008. DOI: 10.4086/toc.gs.2008.001.

[8] B. Falkowski, I. Shefer, and M. Perkowski: Fast computer algorithm for the generation of disjoint cubes for completely and incompletely specified boolean functions. *Proc. 33rd Midwest Symp. on Circuits and Systems*, pages 1119–1122, 1990. DOI: 10.1109/MWSCAS.1990.140922.

[9] B.J. Falkowski, I. Schafer, and C.-H. Chang: An effective computer algorithm for the calculation of disjoint cube representation of boolean functions. *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 1308–1311 vol. 2, 1993. DOI: 10.1109/MWSCAS.1993.343341.

[10] M. Karpovski: *Finite Orthogonal Series in the Design of Digital Devices: Analysis, Synthesis, and Optimization*. A Halsted Press book. Wiley, 1976.

[11] D.M. Miller and M. Soeken: An algorithm for linear, affine and spectral classification of boolean functions. In *Advanced Boolean Techniques*, pages 195–215. Springer, 2020. DOI: 10.1007/978-3-030-20323-8_9.

[12] P. Porwik: Efficient spectral method of identification of linear boolean function. *Int. Journal Control and Cybernetics*, **33**(4), (2004), 663–678.

[13] B. Steinbach and C. Posthoff: *Logic Functions and Equations*. Springer Netherlands, 2009. DOI: 10.1007/978-1-4020-9595-5.

[14] E. Uyan, Çağdaş Çalik, and A. Doğanaksoy: Counting boolean functions with specified values in their walsh spectrum. *Journal of Computational and Applied Mathematics*, **259** (2014), 522–528. DOI: 10.1016/j.cam.2013.06.035.

[15] R. Wang: *Introduction to Orthogonal Transforms: With Applications in Data Processing and Analysis*. Cambridge University Press, 2012. DOI: 10.1017/CBO9781139015158.

[16] A. Younes: A fast quantum algorithm for finding the coefficients of the affine linear boolean function. *arXiv.org*, July 2014. DOI: 10.48550/arXiv.1407.6402.

[17] A. Younes: A fast quantum algorithm for the affine boolean function identification. *Eur. Phys. J. Plus*, **130**(2), (2015), 34. DOI: 10.1140/epjp/i2015-15034-4.