Minimal sign representation of Boolean functions: algorithms and exact results for low dimensions

Can Eren Sezener and Erhan Oztop

Dept. of Computer Science, Ozyegin University
Istanbul, Turkey

Abstract

Boolean Functions (BFs) are central in many fields of engineering and mathematics, such as cryptography, circuit design and combinatorics. Moreover, they provide a simple framework for studying neural computation mechanisms of the brain. Many representation schemes for BFs exist to satisfy the needs of the domain they are used in. In neural computation, it is of interest to know how many input lines a neuron would need to represent a given BF. A common BF representation to study this is the so called polynomial sign-representation where -1 and 1 are associated with True and False. The polynomial is treated as a real valued function and evaluated at its parameters, and the sign of the polynomial is then taken as the function value. The number of input lines for the modeled neuron is exactly the number of terms in the polynomial. This paper investigates the minimum number of terms, i.e. the minimum threshold density that is sufficient to represent a given BF, and more generally, aims to find the maximum over this quantity for all BFs in a given dimension. With this work, for the first time exact results for 4 and 5 variable BFs are obtained, and strong bounds for 6-variable BFs are derived. In addition, some connections between sign-representation framework and Bent functions are derived, which are generally studied for their desirable cryptographic properties.

Index terms — Boolean Function, Binary Classification, Polynomial Threshold Function, Polynomial Threshold Density, Higher-Order Neuron, Sign-Representation

1 Introduction

A simple neuron model that can potentially capture the nonlinear operations of the dendrites (Mel and Koch, 1990; Mel, 1994) in the cerebral cortex is the sigma-pi unit which computes a weighted sum of the product of the inputs and passes it through a nonlinear activation function (Ghosh & Shin, 1992; Giles & Maxwell, 1987; Rumelhart, Hinton, & Williams, 1986; Schmitt, 2005; Taylor & Coombes, 1993). When the output activation function is taken as the signum function, the model can be used to represent Boolean functions (BFs) by associating (-1,+1) with (True,False). This is often called the sign-representation of BFs. Realizing BFs is an active research area and there are a variety of BF representations that are instrumental in engineering and mathematics. For instance, Zhang, Yang, and Wu (2011) recently proposed two different binary higher order neural networks to represent BFs and investigated the upper bound on the number of hidden nodes required. Among the BF representations, the sign-representation framework allows a BF to be identified by the sign of a polynomial of the input variables with powers no more than 1. In the field of artificial neural networks, the sign-representing polynomial is usually referred as a Polynomial Threshold Function (PTF). There are an infinite number of sign-representing polynomials for a given BF and some of those come with different number of terms (monomials). For example, given a sign-representing polynomial, one can always add a new monomial with a sufficiently small coefficient such that the sign of the polynomial remains the same for all the inputs. It is of practical and theoretical interest to know how many monomials will suffice to represent a given BF. In particular, work towards this may indicate the relevance of 'multiplication' as a model of dendritic computation in the cerebral cortex, and lead to new ways of finding BFs with desirable cryptographic properties.

Although there are several algorithms for finding sign representations for a given BF (Guler, 2001), there is no known non-brute force algorithm for finding a minimal monomial sign representation. Oztop proved that any n dimensional BF can be represented with 0.75×2^n monomials (Oztop, 2006) and gave an exact algorithm to achieve this bound (Oztop, 2009). Later, Amano (2010) showed that almost all BFs can be sign represented with less than 0.617×2^n monomials. However, these bound are far from being tight. The theoretical lower bound for the maximum density is known to be 0.11×2^n (O'Donnell & Servedio, 2003). In this study, we aim to improve the upper bound for lower dimensions by computational investigations. In particular, we obtain exact results for 4 and 5 variable BFs, and tighter bounds for 6 variable BFs. Furthermore, we investigate the sign representation of Bent functions which are important for their cryptographic properties.

2 Algorithms for sign-representation

In this section, we first give the basic definitions and results on sign representation starting from the spectral representation of Boolean functions. Then, we introduce the heuristic algorithm we developed to obtain sign-representations for low dimensions. Most of the results in this section are given without proofs; interested readers are referred to Oztop (2006).

2.1 Polynomial (Spectral) Representation of Boolean Functions:

By associating -1 with True and +1 with False, a Boolean function (BF) can be considered as a real valued function $f : \{-1, 1\}^n \to \{-1, 1\}$. Consequently, BFs can be represented as vectors in $\{-1, 1\}^{2^n}$ by adopting a fixed order over the all assignment combinations for the arguments of f (assignment vectors). This allows us to switch back and forth between the functional and vector representations, f and $\mathbf{f}$ without ambiguity.

It can be shown that there is a unique multilinear polynomial p_f that exactly represents f:

$$p_f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{2^n} s_i \prod_{k \in S_i} x_k$$

where S_i runs over all the subsets over {1,2,..,n}. This representation is called the spectral representation of f, and the coefficients are called the spectral coefficients. With a fixed ordering over the monomials (i.e. the products appearing in the expression of p_f), the spectrum, which is the collection of spectral coefficients, can be considered as a vector $\mathbf{s} \in \mathbb{R}^{2^n}$ and used to represent f. The spectrum can be obtained by the Walsh-Hadamard transform (Bruck, 1990; Siu, Roychowdhury, & Kailath, 1995) or Lagrange Interpolation (Oztop, 2006). We refer to the spectrum obtained by Walsh-Hadamard transform as the Walsh Spectrum, which is simply the spectrum obtained by Lagrange Interpolation scaled by 2^n.

By choosing the orderings used in these two representations suitably[1], one can obtain a concise way of connecting the two representations: $\mathbf{f} = \mathbf{D}_n\mathbf{s}$ where $\mathbf{D}_n$ is the $2^n \times 2^n$ Sylvester-type Hadamard Matrix with columns representing the monomials evaluated at all possible input assignments taken in the adopted assignment order. It can be shown that $\mathbf{D}_n$ can be expressed recursively as (Siu et al., 1995):

$$\mathbf{D}_0 = \left[\begin{array}{c} 1 \end{array}\right] \text{ and } \mathbf{D}_{n+1} = \left[\begin{array}{cc} \mathbf{D}_n & \mathbf{D}_n \\ \mathbf{D}_n & -\mathbf{D}_n \end{array}\right] \text{ for } n > 0$$

Which immediately tells that $\mathbf{D}_n$ is symmetric and $(\mathbf{D}_n)^{-1} = 2^{-n}\mathbf{D}_n$. Using the latter, the spectrum $\mathbf{s}$ can be obtained from the vector representation $\mathbf{f}$ of a Boolean function as $\mathbf{s} = 2^{-n}\mathbf{D}_n\mathbf{f}$. It is often practical to note that the row-sum of $diag(\mathbf{f})\mathbf{D}_n$ gives the spectrum scaled by 2^n (i.e. the Walsh spectrum).

2.2 Sign Representation of Boolean Functions:

Polynomial representation of Boolean functions naturally extends to sign-representation: instead of requiring exact interpolation we ask only the sign of the polynomial to agree with the function at each input combination. We say a polynomial p sign-represents a Boolean function f, if and only if $f(x_1, x_2, \cdots, x_n) = \mathrm{sgn}(p(x_1, x_2, \cdots, x_n))$ for all $x_i \in \{-1, 1\}$ [2]. In vector notation this is equivalent to saying $\mathbf{f} = \mathrm{sgn}(\mathbf{D}_n\mathbf{a})$ where $\mathbf{a}$ is the coefficients of p ordered as in the spectrum definition. When this holds, we usually say $\mathbf{a}$ or the monomials of p sign-represents f, or $\mathbf{a}$ *solves* f. The equation involving the sgn() function can be easily converted to a strict linear inequality system: $\mathbf{Y}_f\mathbf{D}_n\mathbf{a} > \mathbf{0}$ where $\mathbf{Y}_f = diag(\mathbf{f})$. Therefore, the problem of finding a sign-representation for $\mathbf{f}$ is equivalent to solving the inequality system $\mathbf{Y}_f\mathbf{D}_n\mathbf{a} > \mathbf{0}$. This form of the problem is referred as the standard form, and can be used to derive this result:

Lemma 1.1. For a given Boolean function f, all solutions (the coefficients of sign-representing polynomials) are of the form $\mathbf{a} = 2^{-n}\mathbf{D}_n\mathbf{Y}_f\mathbf{k}$ with arbitrary $\mathbf{k} > \mathbf{0}$. In other words, any solution $\mathbf{a}$ must belong to the interior

[1]Monomials are ordered as $1, x_0, x_1, x_1x_0, x_2, x_2x_0, x_2x_1x_0, ..., x_{n-1}...x_1x_0$; assignments to $(x_0, x_1, x_2, ..., x_{n-1})$ are ordered as (0's represent 1's and 1's represent -1's): $000...0, 100...0, 010...0, 110...0, ..., 111...1$.

[2]It is also possible to work on general Boolean domains such as $\{1, 2\}$. Hansen and Podolskii (2013) investigates the density of BFs over such domains.

of the cone spanned by the columns of $\mathbf{D}_n\mathbf{Y}_f$. From this we can obtain a simple but useful result:

Lemma 1.2. Let $\mathbf{Q}_f = diag(\mathbf{f})\mathbf{D}_n$, and $\mathbf{A}$, $\mathbf{B}$ be matrices made up from an arbitrary partition of the columns of $\mathbf{Q}_f$. Then $\exists\mathbf{k} > \mathbf{0}$ such that $\mathbf{B}^T\mathbf{k} = \mathbf{0}$ if and only if $\mathbf{a} = [\mathbf{A0}]^T\mathbf{r}$ with some $\mathbf{r} > \mathbf{0}$ is a solution for $\mathbf{Q}_f\mathbf{a} > \mathbf{0}$

Proof. Without loss of generality assume $[\mathbf{AB}] = \mathbf{Q}_f = diag(\mathbf{f})\mathbf{D}_n$

$\Rightarrow:$ Assume $\mathbf{B}^T\mathbf{k} = \mathbf{0}$ for some $\mathbf{k} > \mathbf{0}$. According to Lemma 1.1 $\mathbf{a} = [\mathbf{AB}]^T\mathbf{k}$ satisfies $\mathbf{Q}_f\mathbf{a} > \mathbf{0}$ hence the result follows since $[\mathbf{AB}]^T\mathbf{k} = [\mathbf{A0}]^T\mathbf{k}$ (take $\mathbf{r} = \mathbf{k}$).

$\Leftarrow:$ Assume $\mathbf{a} = [\mathbf{A0}]^T\mathbf{r}$ with $\mathbf{r} > \mathbf{0}$ is a solution for $\mathbf{Q}_f\mathbf{a} > \mathbf{0}$ Then by Lemma 1.1 we must have $\mathbf{a} = [\mathbf{A0}]^T\mathbf{r} = [\mathbf{AB}]^T\mathbf{k}$ for some $\mathbf{k} > \mathbf{0}$. Thus, $\mathbf{B}^T\mathbf{k} = \mathbf{0}$ follows.

Corollary 1.1. A set of monomials sign-represents $\mathbf{f}$ if and only if the vector representation of the remaining monomials are orthogonal to $diag(\mathbf{f})\mathbf{k}$ for some positive vector $\mathbf{k}$.

Proof. Follows from the definition that $[\mathbf{AB}] = diag(\mathbf{f})\mathbf{D}_n$.

2.3 Minimum Sign Representation

Given a Boolean function f, the sign-representation of it is not unique, even excluding the trivial case of arbitrary positive scaling. Usually, one is interested in the minimum number of monomials that would be sufficient to represent the given Boolean function. This number is called the *threshold density* of the Boolean function and it is denoted with $\pi_f(n)$. With Lemma 1.2, we see that finding a minimal sign-representation for a Boolean function f is equivalent to finding a positive vector to nullify the maximum number of columns of $\mathbf{Q}_f = \mathbf{Y}_f\mathbf{D}_n$. More generally, one might ask: what is the minimum number of monomials that one can represent an arbitrary n-variable Boolean function? This number is called the *maximum threshold density* for a given dimension n and is denoted with $\Pi(n)$. Maximum threshold density was classically studied via Spectral Theory of Boolean functions, and initial bounds were obtained as $2^n - \sqrt{2^n} + 1$ see (see Saks, 1993; O'Donnell & Servedio, 2003). More recently an elementary proof was obtained establishing a better bound on the threshold density as 0.75×2^n by Oztop (2006). However even though this bound is significantly lower than the previous ones, it is far from being tight.

A brute force algorithm to find the maximum threshold density for a given dimension is easy to write:

1. For each problem f in dimension n, $(2^{(2^n)}$ many)
2. Let $\mathbf{Y}_f\mathbf{D}\mathbf{a} > \mathbf{0}$ be the standard form of the problem and let $\mathbf{Q} = \mathbf{Y}_f\mathbf{D}$.
3. Enumerate all the column submatrices of $\mathbf{Q}$ as $\mathbf{Q}_1, \mathbf{Q}_2, ...\mathbf{Q}_{2^{2^n}}$.
4. For each submatrix $\mathbf{Q}_i$ apply Fourier Motzkin (FM) elimination to check whether $\mathbf{Y}\mathbf{Q}_i\mathbf{a} > \mathbf{0}$ is satisfiable or not.

FM elimination is a Gaussian Elimination type procedure that is applied to inequalities. It projects the problem into lower dimensions repeatedly by quantifier elimination, and ends by finding either a feasible solution or by a proof that the solution does not exist (see e.g., Chandru, 1993, for details). Running time of FM elimination grows very rapidly with the increasing number of variables; however, this is not the main problem. Even if the satisfiability check of Step 4 could be done very efficiently by fast methods such as interior point methods (also see Section 2.7 for our solution), the outer loops of Steps 1 and 3 prohibit using this algorithm to investigate even relatively low dimensions to get insights on the minimum density. For example, take a single 7-dimensional BF. The number of subsets to be considered is 2^{128} for a single BF (there exists a total of 2^{128} such functions). If we can do the satisfiability check in 1 millionth of a second then it would take 10^{25} years (the age of the universe is 10^{11} years) for a *single BF.* For dimension six, this would take 60000 years for a single problem. So we cannot expect to get much help from numeric investigations of dimension 7 and over. However, there are a few dimensions, namely 4,5 and 6 that has not been explored fully and are amenable to analysis by using equivalence relations defined over BFs. Oztop (2006) conjectured that the minimum density might be given by $0.5 \times 2^n + 1 - (n \bmod 2)$. With our investigations in lower dimensions, we show that this is only true for dimension 2,3 and 4 and definitely not true for dimensions 5 and 6. In particular, we show somewhat surprisingly that the maximum density is much lower for these dimensions.

2.4 Developing a heuristic for sign-representation

Given the problem $\mathbf{Y}_f\mathbf{D}_n\mathbf{a} > \mathbf{0}$ or more compactly, $\mathbf{Q}_f\mathbf{a} > \mathbf{0}$, we would like to find an $\mathbf{a}$ with as many zeros as possible. Determining the monomials that can be eliminated is a hard combinatorial problem. However, once the candidate components to be zeroed are given, one can come up with efficient algorithms to check and find a solution.

Consider the constant Boolean function $f = 1$, which is represented in vector form as $\mathbf{f} = [1, 1, 1, 1, 1, 1, 1, 1]^T$ and has the spectrum $\mathbf{s} = [1, 0, 0, 0, 0, 0, 0, 0]^T$.

Any sign representation for f has the coefficients $\mathbf{a}$ given by:

$$\mathbf{a} = \mathbf{Q}^T \cdot \mathbf{k} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \\ k_7 \\ k_8 \end{bmatrix}$$

Here, we say that the i^{th} monomial is zeroed (eliminated), if and only if $a_i = \mathbf{Q}_i^T \mathbf{k} = \mathbf{0}$ with $\mathbf{k} > \mathbf{0}$ where subscript denotes the row index. For instance, we can easily see whether it is possible to zero the first and the second monomials independently:

$$\mathbf{a}_1 = \mathbf{Q}_1^T \mathbf{k} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 \end{bmatrix}^T$$
$$\mathbf{a}_2 = \mathbf{Q}_2^T \mathbf{k} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 \end{bmatrix}^T$$

It is easy to see that a_1 can never be eliminated since $a_1 > 0$ for all $\mathbf{k} > \mathbf{0}$. On the other hand, for all $\mathbf{k}$ that satisfy

$$k_1 + k_3 + k_5 + k_7 = k_2 + k_4 + k_6 + k_8 \qquad (1)$$

a_2 can be eliminated. These are two extreme cases of 'impossible' to eliminate and 'easy' to eliminate. Generalizing from this, we can define how easy it is to eliminate a monomial by looking at the number of solutions to zero the corresponding coefficient. Of course, with an unbounded $\mathbf{k}$, existence of a solution implies infinitely many solutions. So, let us take an arbitrary large finite scaling bound C such that $\sum_{u=1}^{2^n} k_u = C$ and $\mathbf{k} \in \mathbb{N}^{2^n}$. Now, we define the *hardness* of eliminating i^{th} monomial, $h_i = |K_i|^{-1}$, the inverse of the cardinality of the set of $\mathbf{k}$'s that nullify the i^{th} column of $\mathbf{Q}$.

Lemma 2.1. Given $\mathbf{Q} = \mathbf{Y}_f \mathbf{D}$ and two column indices p and r for any BF f, we can assert:

$$|\mathbf{1}^T \mathbf{Q}_p| > |\mathbf{1}^T \mathbf{Q}_r| \Leftrightarrow |K_p| < |K_r| \qquad (2)$$

where the subscripts denote the column indices.

Remark. This can also be conveniently stated as $|s_p| > |s_r| \Leftrightarrow |K_p| < |K_r|$ since the spectral coefficient corresponding to column i is given by $s_i = 2^{-n}\mathbf{1}^T\mathbf{Q}_i$

Proof. To eliminate a monomial (i.e. make the coefficient of it zero) we have to satisfy an equation in the form of (1) with some k_i's on the left and others on the right. Let L and R be the set of indices of k_i's that are on the left-hand-side and right-hand-side of the equation, respectively. Since we assumed that the sum of all k_i's are equal to C we must have $\sum_{i\in L} k_i = \sum_{j\in R} k_j = C/2$. The number of positive integer vectors $(k_1, k_2, ..., k_n)^T$. that satisfy $\sum_{i\in\{1,2,..,n\}} k_i = M$ is equal to the ways we can write M as the sum of n positive integers. This problem is known as *compositions of an integer*, and the count is given by the binomial coefficient $comb(M-1, n-1)$. Thus, the number of k_i's that can satisfy our elimination condition is given by

$$\begin{pmatrix} C/2-1 \\ |L|-1 \end{pmatrix}\begin{pmatrix} C/2-1 \\ |R|-1 \end{pmatrix} \tag{3}$$

where $|L|$ and $|R|$ denotes the cardinality of L and R, respectively. Noting that $|R| = 2^n - |L|$ we immediately see that the value of (3) is maximum when $|L| = |R| = 2^{n-1}$, and it strictly decreases as $|R| - |L|$ increases. Since $2^{-n}(|R| - |L|)$ is equal to the spectral coefficient $|s_i|$ of the monomial considered, the proof is complete.

Lemma 2.1 implies that when the monomials are considered in isolation, the monomials with lower spectral coefficients ($|s_i|$'s) are easier to eliminate (i.e. will generate equations that have large solution sets). Even though we do not know whether two isolated 'easy' monomials will be still easy when considered simultaneously, we may hope that the large solution sets induced by the elimination conditions will have a high chance of being a non-empty intersection. In fact, it can be shown that even in the worst case $2^{n/2} - 1$ monomials with the smallest non-zero spectral coefficients can always be thrown away from the spectrum to obtain a sign representation (Saks, 1993). However, this is largely short of what we can achieve with the heuristic algorithm we develop in the next section.

2.5 An algorithm for sign-representation with a small number of monomials

Following the motivation given in the previous section, the main strategy adopted is to eliminate the monomials starting from the easiest ones (i.e. with smaller $|s_i|$ values) and to gradually move on to the harder ones (i.e. those with larger $|s_i|$ values). So we pick the first h monomials with the lowest $|s_i|$, and check if such an elimination is possible. If it is, we try to eliminate one more monomial (i.e. increment h). If it is not, we pick the next monomial with the highest or equal $|s_i|$ value. We continue this process until h can no longer be increased. Until this point, the algorithm is greedy; it will pick the monomial with the smallest spectral coefficient in magnitude, and continue with the next smallest as long as they all can be eliminated. When the elimination is not possible the algorithm looks for a replacement for the most recent monomial tried to be eliminated by attempting to eliminate the next monomial with the highest $|s_i|$. The important point is that, the total number of elimination attempts is in the order of 2^n as no monomial is attempted to be eliminated more than once. This linear search (in the number of monomials) works surprisingly well considering the computational cost that would be needed for a brute force search. The pseudo code for this procedure is given in Table-1.

The algorithm given in Table-1 uses a subroutine at Step 11, which determines whether there exists a positive vector $\mathbf{k}$ in the null space of $\mathbf{H}$. If exists, such $\mathbf{k}$ will ensure that (i) $\mathbf{a} = \mathbf{D}_n \mathbf{Y}_f \mathbf{k} = \mathbf{Q}_f^T \mathbf{k}$ is a solution to the inequality $\mathbf{Q}_f \mathbf{a} > \mathbf{0}$ (due to Lemma 1.2), and (ii) the coefficients of $\mathbf{a}$ corresponding to the monomials in $\mathbf{H}$ are zeroed. The problem of finding $\mathbf{k} > \mathbf{0}$ to satisfy $\mathbf{k}^T \mathbf{H} = \mathbf{0}$ can be formulated as a constrained quadratic minimization or a linear programming (LP) problem and thus can be solved efficiently.

This algorithm solves 2^n LP problems as opposed to 2^{2^n} for brute force. Each LP problem has 2^n variables and 2^n or less constraints. LP problems can be solved by methods such as the Interior Point Method in polynomial number of steps (in the number of variables). Therefore, the running time of this algorithm is bounded above by $2^n \times (2^n)^{O(1)}$ and the time complexity can be given as $2^{O(n)}$.

```
input  : $\mathbf{y_f}$ (the BF $f : \{-1,1\}^n \to \{-1,1\}$ in vector form)
output: $\mathbf{w}, h$ (coefficients of a sign representation with $2^n - h$ zeros)
1  Let $\mathbf{D}$ be a Sylvester- type Hadamard matrix of order $2^n$
2  Let $\mathbf{s} = \mathbf{y}_f^T\mathbf{D}$ (spectrum of $\mathbf{y}_f$ scaled by $2^n$)
3  Let $\mathbf{Q} = diag(\mathbf{y}_f)\mathbf{D}$
4  Let $\mathbf{A} = [\mathbf{Q}; |\mathbf{s}|]$ (augment $\mathbf{Q}$ with the absolute value of the
   spectrum)
5  Sort the columns of $\mathbf{A}$ in ascending order using components of $|\mathbf{s}|$ as
   the sort key
6  $h = 1$ (denotes the number of monomials to attempt to eliminate)
7  $\mathbf{w} = \mathbf{0}$ (coefficients of the sign-representing polynomial)
8  while $h < \#A$ (number of columns in A) do
9      $\mathbf{H}$ = first h columns of $\mathbf{A}$ excluding the last row
10     (i.e. $\mathbf{H} = \mathbf{A}[1 : h, 1 : 2^n]$)
11     if There exists $\mathbf{k} > 0$ such that $\mathbf{k}^T\mathbf{H} = \mathbf{0}$ then
12         $h = h + 1$
13         $\mathbf{w} = \mathbf{Q}^T\mathbf{k}$ ($\mathbf{w}$ is a solution because $\mathbf{Qw} = \mathbf{QQ}^T\mathbf{k} > 0$)
14     else
15         delete $h^{th}$ column of $\mathbf{A}$ (undo the effect of the last elimination
           attempt on A)
16     end
17 end
18 return $\mathbf{w}, h$
```

Table 1: Heuristic algorithm for sign-representing y

2.6 Results of the heuristic algorithm

Oztop (2009) introduced the restricted prime function $f_n(\mathbf{x})$ that tells whether the n-bit binary number $\mathbf{x}$ is prime or not, and used it to obtain sign-representation through the 3-Quarters Algorithm. In Table 1, we compare the results of our heuristic algorithm with those reported in (Oztop, 2009). Surprisingly, our sign-representations are much more parsimonious. As the dimension increases the heuristic algorithm can obtain representations by using only one fifth of the number of monomials needed by 3-Quarters. In particular, for non-negative integers up 1023, a polynomial with 100 monomials is found to be sufficient to represent primality with the heuristic algorithm whereas this would require 633 monomials with 3-Quarters.

Restricted Prime Functions	# of monomials by the 3-Quarters (% of full set)	Computation duration (s) for 3-Quarters	# of monomials by our algorithm (% of full set)	Computation duration (s) for our algorithm
p_4	7 (64.06%)	0.07	5 (31.25%)	0.09
p_5	17 (57.42%)	0.08	6 (18.75%)	0.14
p_6	39 (61.52%)	0.10	9 (14.06%)	0.49
p_7	82 (64.06%)	0.15	12 (9.38%)	3.16
p_8	147 (57.42%)	0.30	(8.59%)	32.0
p_9	315 (61.52%)	1.10	(10.35%)	449
p_{10}	633 (61.82%)	10.0	(9.77 %)	5,602
p_{11}	1259(61.47%)	84.0	(11.04%)	93,853

Table 2: Result comparison

Our heuristic algorithm was implemented in MATLAB (R2013) and runs on a standard desktop PC with a 3.07 GHz CPU and 16 GB RAM. The last column of Table-2 depicts the computing times for each function. Although the algorithm is simple, it serves as a very good tool to study the sign-representation of BFs for low dimensions. It must be noted that the algorithm given in Table-1 is the simplest version of the outlined heuristic logic. In the actual implementation, we used additional heuristics to reduce the computation time: In the given algorithm, the monomials are first sorted according to $|\mathbf{s}|$, then the eliminations are done starting from the first monomial (i.e. the monomial with the smallest coefficient), and subsequent monomials are tried one-by-one. In our implementation, we start the elimination from the first block of 2^{n-1} monomials, if such an elimination is not possible, then the elimination falls back to the first 2^{n-2} monomials, and then to the first 2^{n-3} monomials and so on (somewhat similar to binary search) until a solution is found. When a valid solution is found, the elimination proceeds linearly as indicated in the standard algorithm given in Table-1.

In what follows we describe the obtained results with the help of this algorithm and other mathematical insights for dimensions 4, 5 and 6.

2.7 Obtaining exact results for the maximum density

The proposed heuristic algorithm can only provide us a lower bound for the minimum threshold density. If the algorithm fails to find a solution, say with m number of monomials, this does not constitute a proof that there is no solution with m number of monomials. Of course, the repeated execution of the heuristic algorithm (with randomized elimination ordering for monomials with equal spectral coefficients) gives us an indication that perhaps with m monomials there can be no solution. To confirm that no solutions with m monomials exist, we need another algorithm to give us a proof or rather a *certificate* that can be used to prove that no solution exists with m monomials. Next lemma shows that a positive vector is the required certificate.

Lemma 2.2. If a set of monomials (corresponding to a submatrix $\mathbf{A}$, of $\mathbf{Q}_f$) sign-represents $\mathbf{f}$ (i.e. $\mathbf{Aa} > \mathbf{0}$ for some $\mathbf{a}$) then for any positive vector $\mathbf{k}$, $\mathbf{A}^T\mathbf{k} \neq \mathbf{0}$.

Proof. Assume that there exists $\mathbf{k}$ with $\mathbf{A}^T\mathbf{k} = \mathbf{0}$. Then this would contradict with the fact that $\mathbf{Aa} > \mathbf{0}$ as $\mathbf{0} = \mathbf{k}^T\mathbf{Aa} > \mathbf{k}^T\mathbf{0} = \mathbf{0}$.

So if we find a positive $\mathbf{k}$ for which $\mathbf{A}^T\mathbf{k} = \mathbf{0}$ we prove that $\mathbf{A}$ cannot constitute a solution, i.e. there is no $\mathbf{a}$ such that $\mathbf{Aa} > \mathbf{0}$. What about the other direction; if there is no solution can we always find a certificate to show this? The answer is yes as stated in the next Lemma.

Lemma 2.3. If a set of monomials (corresponding to a submatrix $\mathbf{A}$, of $\mathbf{Q}_\mathbf{f}$) cannot sign-represent $\mathbf{f}$ (i.e. there is no $\mathbf{a}$ such that $\mathbf{Aa} > \mathbf{0}$) then there exists a positive vector $\mathbf{k}$, $\mathbf{A}^T\mathbf{k} = \mathbf{0}$ or equivalently $\mathbf{k}^T\mathbf{A} = \mathbf{0}^T$.

Proof. Assume we apply FM elimination to check for the satisfiability of $\mathbf{Aa} > \mathbf{0}$. The process ends with a new inequality system $\hat{\mathbf{A}}\mathbf{a} > \mathbf{0}$ where each column is all zero or all positive or all negative vector (Oztop, 2006). As such if $\hat{\mathbf{A}} \neq \mathbf{0}$, $\hat{\mathbf{A}}\mathbf{a} > \mathbf{0}$ can always be satisfied since we have no constraints on $\mathbf{a}$, which implies the satisfiability of $\mathbf{Aa} > \mathbf{0}$ (Chandru, 1993). Therefore, the only way FM elimination can generate an unfeasibility is by ending up with $\hat{\mathbf{A}} = \mathbf{0}$. But each row of $\hat{\mathbf{A}}$ has been generated by a non-negative combination of the rows of $\mathbf{A}$ (according to the FM elimination rules). So summing all the rows will give us the (transpose of the) desired certificate $\mathbf{k}$, as it will be a positive combination of all the rows of $\mathbf{A}$.

Let f be an n-variable Boolean function that can be solved with $m+1$ monomials. If we suspect that there is no solution with m monomials we can use Lemma 2.3 to prove it: scan all submatrices $\mathbf{H}$ with m columns from $\mathbf{Q}$ and check whether there exists $\mathbf{k} > \mathbf{0}$ such that $\mathbf{k}^T\mathbf{H} = \mathbf{0}$. The latter step can be done efficiently as discussed in Section 2.5 (Algorithm step 11). However, we need to check for each $\binom{2^n}{m}$ submatrices to prove that no m-monomial solution exists. If we do that, with the knowledge that there exists a solution with $m+1$ monomials we establish the threshold density of f exactly as $m+1$. To find the maximum threshold density of all n-variable Boolean functions we need to repeat this for all 2^{2^n} functions! Obviously, this is a very time-consuming process. In Section 4, by using appropriate equivalence classes defined over the BFs, we show how novel results on maximum threshold density can be obtained for low dimensions.

3 Results for lower dimensions ($n \leq 4$)

Oztop (2006) previously established the maximum threshold densities for 1, 2 and 3 variable BFs as $\Pi(1) = 1$, $\Pi(2) = 3$, $\Pi(3) = 4$ and asserted that $\Pi(4) \leq 9$. We verified these results and showed that in fact the inequality for dimension 4 is an equality. For this we first applied the heuristic algorithm to BFs of dimensions 1, 2 and 3 and obtained the same results with Oztop (2006). Later, we showed $\Pi(4) \leq 9$ by applying the heuristic algorithm to all $65,536$ 4-variable BFs[3]. It is worth noting that only (896) 1.37% of the all 4-variable BFs required 9 monomials, and the rest could be solved with less number of monomials by using the heuristic algorithm. This ratio was 50% with the 3-Quarters Algorithm. In order to prove $\Pi(4) = 9$, it suffices to show that there exists a dichotomy that cannot be solved with 8 monomials or less. This can be done by exhaustively showing that all 8-monomial combinations cannot be a solution for a given dichotomy. We took a hard looking problem[4], and constructed certificates for each 8-monomial subset to prove that it cannot constitute a solution for the selected function, thereby

[3]Results can be found in the Appendix

[4]To be concrete, we used one of the functions from the equivalence class C_8^3, see the Appendix

establishing the maximum threshold density for dimension 4 as $\Pi(4) = 9$.

4 Equivalence of Boolean Functions and Exact results for 5 dimensions

The number of Boolean functions grow very rapidly as the number of variables increase. For $n = 5$, there are $2^{2^5} = 4,294,967,296$ Boolean functions. Considering that it takes approximately 0.2 seconds to solve a dichotomy of 5 variables with a standard PC, solving all the dichotomies is not impossible. However, by defining an equivalence relation over the BFs that keeps the threshold density as an invariant, it is possible to obtain the maximum threshold density for dimension 5 or more a lot faster by working on a single representative from each equivalence class. There are several group transformations defined over BFs in the literature that preserve threshold density and induce natural equivalence classes. In this section, we present two important transformations and show that the threshold density stays invariant under these transformations.

4.1 Algebraic Normal Form

The Algebraic Normal Form (ANF) is widely used to represent Boolean functions both in engineering and mathematics. In ANF, a Boolean function $f(x_1, x_2, \ldots x_n)$ is represented as a polynomial over F_2^n, the field of integers modulo 2 with normal addition and multiplication. Alternatively, this can be seen as an XOR of conjunctions of Boolean variables $x_1, x_2, \ldots x_n$ with 0 and 1 representing False and True respectively.

$$f(x_1, x_2, \ldots x_n) = \bigoplus_{i=1,2\ldots 2^n} a_i \prod_{k \in S_i} x_k$$

where $a_i, x_k \in \{0, 1\}$, S_i runs over all subsets of $\{1, 2, \ldots n\}$, and $\bigoplus$ denotes the XOR ($\oplus$) operator over a sequence (e.g., $\bigoplus_{i=1,2\ldots n} x_i = x_1 \oplus x_2 \oplus \ldots x_n$).

It is easy to transform a BF given in ANF, to the exact representing polynomial (equivalently to find its spectrum) using this isomorphism:

1. replace binary constants 0 and 1 with bipolar constant 1 and -1 respectively

2. replace F_2^n addition with multiplication
3. replace F_2^n multiplication $x_1 \cdot x_2$ with $(-x_1x_2 + x_2 + x_1 + 1)$

Note that this generates the full spectral representation (i.e. the exact interpolating polynomial where T and F are represented with -1 and $+1$). It is also easy to obtain a sign-representation which may have more zero coefficients. For the last replacement we can use $(x_1x_2 - x_1 - x_2)$ and still satisfy the sign match requirement. This will be handy for generating close-to-minimal sign-representations for some special BFs (see Section 5.4).

4.2 NPN and Spectral Equivalence

One of the most commonly used equivalence relation to analyze Boolean functions is so called the NPN (Negation-Permutation-Negation) equivalence. The equivalence class of a BF f can be obtained by applying the composition of three types of transformations.

1. Negation of input variables (e.g. $f(x_1, x_2, x_3) \to f(x_1, x_2, \neg x_3)$)
2. Permutation of input variables (e.g. $f(x_1, x_2, x_3) \to f(x_2, x_1, x_3)$)
3. Negation of the output (e.g. $f(x_1, x_2, x_3) \to \neg f(x_1, x_2, x_3)$)

Edwards (1975) introduced another classification method called Spectral Classification that uses five transformations to classify Boolean functions. Three out of five transformations in Spectral Classification are the same with the NPN Classification. Additional two transformations are:

4. XORing an input variable with other variables (e.g. $f(x_1, x_2, x_3) \to f(x_1, x_2 \oplus x_3 \oplus x_1, x_3)$)
5. XORing the function with input variables (e.g. $f(x_1, x_2, x_3) \to x_1 \oplus x_2 \oplus f(x_1, x_2, x_3)$)

As the number of transformations increase, the number of equivalence classes decrease. If we know that the composition of the five transformation preserves the threshold density, we can work on a single representative from each equivalence class. Therefore the smaller the number of the equivalence classes, the better it is for our investigation of the threshold density. It is not difficult to show that the threshold density is an invariant under all the 5 transformation above. We show this in the polynomial representation framework of Boolean functions in the next section. These five transformations defined over the Boolean functions have been studied extensively in F_2^n and coincides with the Affine Transformations of functions defined over F_2^n which we describe next.

4.3 Affine Transformation and Equivalence

Definition. An affine function $L_{b,c} : F_2^n \to F_2$ is defined as
$L_{b,c}(\mathbf{x}) = \mathbf{b}^T\mathbf{x} \oplus c$ where $\mathbf{b} = (b_1, b_2, \cdots, b_n) \in F_2^n$, and $c \in F_2$.
An affine transform $T : 2^{F_2^n} \to 2^{F_2^n}$ maps a Boolean function f to another Boolean function g by (i) multiplying the input vector of f with a non-singular binary matrix; (ii) negating a subset of the input variables; and (iii) applying another affine function to the output. The formal definition is as follows:

Definition. An affine transform maps a function f to a function g such that,

$$g\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = f\left(\mathbf{M} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \oplus \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix}\right) \oplus (b_1 \cdot x_1 \oplus b_2 \cdot x_2 \oplus \cdots \oplus b_n \cdot x_n) \oplus c$$

where the operations are defined in the F_2 field. Thus, affine transform in this form maps as $0, 1$ vectors to $0, 1$ vectors that define BFs. As BFs can also be represented by polynomial functions (i.e. spectral representation), the Affine transform can also be defined to map polynomial functions to polynomial functions that are defined from $\{1, -1\}^n$ to $\{1, -1\}$ (or vector of rational numbers to vector of rational numbers by taking the coefficient of those polynomials as vectors). This can be achieved by a simple lexical translation: Replace each constant as $0 \to 1$ and $1 \to -1$; replace addition in F_2^n ($\oplus$) with normal multiplication ($\cdot$); replace the inner product $\mathbf{b}^T\mathbf{x}$ with $B = \prod_{i=1..n} {x_i}^{b_i}$ product of variables with corresponding ones in $\mathbf{b}$. Finally, replace $\mathbf{m}_i \cdot \mathbf{x}$, where $\mathbf{m}_i$ is the i^{th} row of $\mathbf{M}$ with $X_i = \prod_{j \in S_i} x_j$ where S_i is the set of the indices of non-zero elements of $\mathbf{m}_i$. Overall, when we make the transition from ANF representation to the spectral representation the corresponding Affine transformation that maps f_{spect} to g_{spect} is given as:

$$g_{spect}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (-1)^c(\prod {x_i}^{b_i}) f_{spect}\begin{pmatrix} (-1)^{k_1}X_1 \\ (-1)^{k_2}X_2 \\ \vdots \\ (-1)^{k_n}X_n \end{pmatrix}$$

It is not difficult to show that the transformation T defines an equivalence relation. We will call Boolean functions f and g are equivalent if they belong to the same equivalence class, i.e. if and only if there exists an affine transformation T, such that $T(f) = g$.

We use ${C_i}^n$ to denote the i^{th} equivalence class of n-variable BFs. (the list of equivalence classes for $n \leq 5$ can be found in the Appendix.) We also extend the use of the threshold density $\pi(f)$, and maximum threshold density $\Pi(n)$ notations over the equivalence classes ${C_i}^n$ as threshold density is an invariant of the affine transformation as shown next.

4.4 Effect of Affine Transformations on Sign-Representation

Given a sign representation problem[5] in the standard form 'solve **a** to satisfy $\mathbf{Y}_f\mathbf{D}_n\mathbf{a} > \mathbf{0}$', components of a solution **a** gives the coefficients of the sign representing polynomial. For instance with the ordering adopted in this report, a_2 corresponds to the coefficient of x_1; and a_8 corresponds to the coefficient of $x_1x_2x_3$ and so on. If we multiply the monomial x_1 and $x_1x_2x_3$ with x_2x_3, we arrive at $x_1x_2x_3$ and x_1 respectively considering we are only interested in evaluating these polynomials at $\{1, -1\}^n$. Therefore, the coefficients of the representing polynomial, and consequently the components of **a** will swap under the multiplication with x_2x_3 transformation. In fact, the effect of an affine transformation is exactly to permute and sign change of some components of **a**, and therefore does not effect the number of zero coefficients. To be concrete, let us look at the affine transformations in four parts to identify their specific effects on the coefficients of sign-representing polynomials.

1. $(-1)^c f(\mathbf{x})$ negates all the coefficients if c is 1. Otherwise, coefficients remain the same.

[5]As exact interpolation or spectral representation is a special sign representation, all the arguments in this section applies to spectral representation of BFs as well.

2. $(\prod {x_i}^{b_i})f(\mathbf{x})$ rearranges the coefficients by swapping every coefficient exactly once (assuming $\exists b_i \neq 0$). The coefficient of a given monomial u is swapped with the coefficient of the monomial $u(\prod {x_i}^{b_i})$. For instance, if $u = x_1x_2x_3$ and $b = (1, 1, 0, \ldots 0)$ then we would have $(x_1x_2)(x_1x_2x_3)$ resulting in x_3, and symmetrically $(x_1x_2)(x_3)$ will result in $u = x_1x_2x_3$. This means the coefficients of the monomials $x_1x_2x_3$ and x_3 will swap.

3. $f(-1^{\mathbf{k}} \circ \mathbf{x})$, where multiplication ($\circ$) and exponentiation is component-wise, negates the coefficients of the monomials that include x_i if $k_i = 1$.

4. Replacing x_i with $X_i = \prod\limits_{j \in S_i} x_j$, rearranges the order of coefficients. If a monomial u contains x_i, coefficient of u is swapped with the coefficient of the monomial $u(\prod\limits_{j \in S_i} x_j)$.

Lemma 4.1. Let $\mathbf{a}$ be the coefficient vector of the polynomial that sign-represents f. Let T be an affine transform that maps f to g, $T(f) = g$. A coefficient vector $\hat{\mathbf{a}}$ that sign-represents g can be generated by reordering and negating some of the indices of $\mathbf{a}$.

Proof. Follows directly from the effect analyses above.

Corollary 4.1. The number of components that are zero in $\mathbf{a}$, i.e. the number of zero coefficients in a sign-representation is an invariant of the affine transformation.

Corollary 4.2. All equivalent Boolean functions under the affine transformation have equal threshold densities.

Equivalence classes of 5-variable Boolean functions has been known for a long time, and the number of non-equivalent functions are quite low (48) (Berlekamp & Welch, 1972; Maiorana, 1991). Although it is known that the number of equivalence classes grows exponentially (see Table-3 for some samples), the equivalence relation presents an opportunity to study the minimal sign-representation for lower dimensions which we exploit in this report. The affine transformation and the induced equivalence classes have been exploited in other domains such as cryptography and circuit complexity via invariants of the transform such as auto correlation, nonlinearity, and absolute indicators (Preneel, 1994; Fuller, 2003). However, to our knowledge, our study is the first to utilize the affine equivalence classes for the study of minimum threshold density.

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
# Boolean functions	4	16	256	65,536	4×10^9	18×10^{18}
# Equivalence classes	1	2	3	8	48	150,357

Table 3: Numbers of Boolean Functions and Equivalence classes

The equivalence classes of the n-dimensional BF for $n \leq 5$ are given in Anderson (2007) and Fuller (2003), and reproduced in the Appendix. From each equivalence class, we (arbitrarily) select a representative and study its threshold density to obtain the maximum threshold densities for the selected dimensions.

4.5 Results for 5-variable Boolean Functions

We first applied the heuristic algorithm to one member from each of the 48 equivalence classes of 5-variable Boolean functions enumerated by Fuller (2003), and obtained an upper bound on the threshold density for each class. Then, we searched for solutions with smaller monomial subsets to obtain a tighter bound for $\Pi(5)$. Subsequently, we were able to show that (see Appendix for the details) any 5-variable Boolean function can be sign-represented with at most 11 out of 32 monomials (i.e. $\Pi(5) \leq 11$). To show that this bound is the smallest possible, we showed that there exists functions (that belong to C^5_{33}, C^5_{46} and C^5_{48}, see the Appendix) that cannot be sign-represented with 10 monomials by finding proof certificates as described in Section 2.7. By doing so, we established $\Pi(5) = 11$ for the first time. This is a big improvement over the previous bound of 24 given by the 3-Quarters Theorem (Oztop, 2006). An interesting observation is that the heuristic algorithm proposed in this report found the threshold density of almost all studied Boolean functions (i.e. the representatives of their equivalence classes) correctly. In a few cases, the results of the algorithm was off by one monomial from the threshold densities.

Examining the functions with the highest threshold densities, we see that all belong to three equivalence classes (out of 48), namely, C^5_{33}, C^5_{46} and C^5_{48}. The equivalence class C^5_{46} is composed of *Semi-Bent functions* (i.e. the Boolean functions with Walsh coefficients $\pm 2^{(n+1)/2}$, see Section 5.3) which are known to have nice cryptographic properties. To the best of our knowledge, functions of the classes C^5_{33} and C^5_{48} which are far from being linear (as are the Semi-Bent functions) do not belong to a specific BF class that was addressed before. It might be interesting to investigate the cryptographic properties of these functions, and assess how they compare with those of Semi-Bent functions'.

Furthermore, for the first time we have shown that two equivalence classes with the same spectrum summary (the list of unique absolute values of the spectrum and their number of occurrences) can have different threshold densities. E.g. C^5_{32} and C^5_{33} have the same spectrum summary $(4 \times 12,\ 28 \times 4)$[6]; however, $\pi(C^5_{32}) = 9$ and $\pi(C^5_{33}) = 11$. This discovery is significant because spectrum summaries are often taken indicators for the non-linearity and the hardness of the BFs for sign-representations. Our finding show that this two disassociate: the number of monomials is not an invariant property of the BFs that share the same spectrum summaries.

5 Six Dimensions

We have obtained the upper bound on $\Pi(5)$ by solving a representative function from each of the 48 affine equivalence classes for 5-variable Boolean Functions as discussed in the previous section. Maiorana (1991) showed that in dimension 6, there exists 150,357 affine equivalence classes, which were also verified by Fuller (2003). In fact, Fuller explicitly computed representatives for each equivalence class in her Ph.D. thesis (Fuller, pers.comm., 20 Feb 2014), which we used to obtain the best known upper bound for $\Pi(6)$ as $\Pi(6) < 27$. However, before presenting this result, we first obtain an upper bound which does not use any information on the affine equivalence classes of the 6-variable Boolean functions.

5.1 Using $\Pi(5)$ for bounding $\Pi(6)$

The surprising low value of 11 for $\Pi(5)$ gives us an easy way to prove a bound for $\Pi(6)$ which supersedes any known bound so far. This result is obtained by the application of a general decomposition of n-variable Boolean functions into two $n-1$ variable Boolean functions. Consequently, we arrive at the non-trivial bound of $\Pi(6) \leq 44$.

[6]Spectrum summary of 4×12, 28×4 means that the Walsh spectrum is composed of 4 ± 12's and 28 ± 4's.

Theorem 5.1. Any n-variable Boolean function can be represented with $4 \times \Pi(n-1)$ monomials.

Proof. Any n-variable Boolean function can be expressed in terms of two $n-1$ variable Boolean functions based on a chosen input variable. To see this, let f be an n-variable Boolean function, and without loss of generality take x_0 as the chosen input variable (and assume instead of True and False we are using -1 and 1 respectively).

$$f(x_0, x_1, \ldots, x_n) = \begin{cases} g(1, x_1, \ldots, x_{n-1}) & if\ x_0 = 1 \\ h(-1, x_1, \ldots, x_{n-1}) & if\ x_0 = -1 \end{cases} \tag{4}$$

where g and h are $n-1$ variable Boolean functions. Analogously, any polynomial function p_f that sign-represents f can be partitioned using the algebraic equation:

$$p_f(x_0, x_1 \cdots, x_{n-1}) = (x_0+1)p_g(x_1, x_2 \cdots, x_{n-1}) + (1-x_0)p_h(x_1, x_2 \cdots, x_{n-1}) \tag{5}$$

where p_g and p_h are sign-representing polynomials of g and h induced by p_f. As we are only interested in the sign of p_f we can replace p_g and p_h with *any* sign-representation for g and h. In other words, p_g and p_h are sign representations for g and h if and only if $(x_0+1)p_g + (1-x_0)p_h$ is a sign representation for f. But then, this says that f has a sign representation with at most twice the total number of monomials in p_g and p_h.

Corollary 5.1. Any 6-variable Boolean function can be solved with $4 \times \Pi(5) = 44$ monomials or less.

Proof. Denoting the number of monomials in p with $\tau(p)$, we have

$$\tau(p_f) \leq 2 \times (\tau(p_g) + \tau(p_g))$$

as a direct consequence of (5). Furthermore, we know every 5-variable BF can be sign-represented with 11 or less monomial. It follows that for any 6-variable Boolean function there is always a sign representation with $44 = 4 \times 11$ or less number of monomials.

5.2 Bent functions and upper bound

When we inspect the maximum threshold densities we have obtained so far, we see that the increase in the number of monomials required is greater when moving from an odd dimension to an even dimension (Table 4).

Maximum Threshold Densities					
$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
1	3	4	9	11	≤ 26

Table 4: Maximum threshold densities for $n \leq 6$

When the 'hardest' dichotomies (in the sense of requiring high number of monomials) in even dimensions are inspected (see the Appendix), one can see that those functions have Walsh (i.e. spectral coefficients scaled by 2^n) coefficients with equal absolute value. In fact, these are exactly the type of Boolean function that are dubbed *Bent* by Rothaus (1976).

Definition (Bent function). An n-variable Boolean function is called Bent if and only if the Walsh coefficients of the function are all $(\pm 2)^{(n/2)}$.

Remark. Bent functions exist only in even dimensions.

Algorithm described in Table-1 attempts to eliminate the monomials starting from the spectral coefficients with the smallest absolute values and then moves on to the ones with the higher values. In the case of Bent functions, the spectral coefficients are constant. So there are no low-hanging fruits, and the heuristic algorithm simply picks the monomials to test for elimination randomly. Everything else equal, this on its own would cause the heuristic to obtain the poorest results on Bent functions. Indeed, when we inspect the exact results on Bent functions further, we see that they are the hardest (in terms of number of monomials required) functions to represent in dimensions 2 and 4, and they appear to be the hardest in dimension 6. This suggests that the reason Bent functions are found to require more monomials with the heuristic algorithm is not really a performance issue but rather a fundamental difficulty with the Bent functions. Based on our empirical observations, we conjecture that the sign-representations of Bent functions in a given dimension require more monomials than that of any other BFs.

Conjecture 5.1. For all even n, maximum threshold density, $\Pi(n)$, is equal to the threshold density of a Bent function.

If all Bent functions belonged to a single equivalence class, we could say that all n-variable Bent functions have the same threshold density. Therefore, Conjecture 5.1 would imply that the density of any Bent function would be equal to $\Pi(n)$. All 2 and 4 variable Bent functions belong to a single affine equivalence class, namely to C_2^2 and C_8^4. However, this does not hold for six dimensions as Rothaus (1976) showed that all 6-variable Bent functions belong to four different equivalence classes. Although existence of multiple equivalence classes of Bent functions for $n \geq 6$ is a fact, it is an open question whether all must have the same threshold density. Therefore even if Conjecture 5.1 is correct for a given dimension n, one needs to compute the threshold density for each affine equivalence class of Bent functions to determine $\Pi(n)$.

In spite of the fact that there are only 4 inequivalent 6-variable Bent functions, it is not easy to resort to exhaustive search to check whether all four Bent equivalence classes share the same threshold density, since it would require checking all (or a substantial fraction of) 2^{64} monomial subsets for a solution.

5.3 Constructing Sign-Representations of Bent Functions

In section 5.1, we proposed a method to construct n-variable Boolean functions using two $(n-1)$-variable Boolean functions. Here, we show that Bent functions can be constructed using Semi-Bent functions.

Definition (Semi-Bent function)**.** For an odd n, if Walsh Spectrum of an n-variable Boolean function f, can only take the values 0 and $\pm 2^{(n+1)/2}$, then f is a *Semi-Bent* function.

Theorem 5.2. In the decomposition of Theorem 5.1, if f is a Bent function, then g and h must be Semi-Bent functions.

Proof. Let f be a Bent function, $\mathbf{y}_f$ be its vector representation, $\mathbf{w}_f = \mathbf{y}_f \cdot \mathbf{D}$ be it's spectrum, and $\mathbf{D}$ be a Sylvester-type Hadamard Matrix. As in (3), we partition f into g and h, but this time in the vector form:

$$[\mathbf{y}_g^T \mathbf{y}_h^T] \begin{bmatrix} \mathbf{D} & \mathbf{D} \\ \mathbf{D} & -\mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 \mathbf{w}_2 \end{bmatrix} \tag{6}$$

$$\mathbf{y}_g^T \mathbf{D} + \mathbf{y}_h^T \mathbf{D} = \mathbf{w}_1 \tag{7}$$

$$\mathbf{y}_g^T \mathbf{D} - \mathbf{y}_h^T \mathbf{D} = \mathbf{w}_2 \tag{8}$$

Here, $\mathbf{y}_g$ and $\mathbf{y}_h$ correspond to the vector representations of g and h. Spectrums g and h are $\mathbf{w}_g = \mathbf{y}_g^T \mathbf{D}$ and $\mathbf{w}_h = \mathbf{y}_h^T \mathbf{D}$. By organizing (7) and (8), we get

$$\mathbf{w}_g = \mathbf{y}_g^T \mathbf{D} = (\mathbf{w}_1 + \mathbf{w}_2)/2 \tag{9}$$

$$\mathbf{w}_h = \mathbf{y}_h^T \mathbf{D} = (\mathbf{w}_1 - \mathbf{w}_2)/2 \tag{10}$$

The Walsh spectrum of an n-variable Bent function is always composed $+2^{n/2}$'s and $-2^{n/2}$'s; hence, $\mathbf{w}_1$ and $\mathbf{w}_2$ are vectors of $\pm 2^{n/2}$. It is clear from the identities (9) and (10) that $\mathbf{w}_g$ and $\mathbf{w}_h$ can only be composed of the three different values: $-2^{n/2}$, 0, and $2^{n/2}$. Therefore, they are Semi-Bent functions.

5.4 Upper Bound for 'Simple' Bent Functions

It is known that all function of the form (in ANF),

$$f(x_1, x_2, \ldots x_n) = x_1x_2 \oplus x_3x_4 \oplus \ldots \oplus x_{n-1}x_n \tag{11}$$

are Bent functions (Meier & Staffelbach, 1990). We call them Simple Bent for brevity (and because of their apparent simple ANF form).

Theorem 5.3 For n even, a Simple Bent function in n variables can always be sign-represented with $3^{n/2} \approx 1.732^n$ monomials.

Proof. If we transform the function f in ANF to a sign-representing polynomial f', as suggested in the section 2, we get $f' = (x_1x_2 - x_1 - x_2) \cdot (x_3x_4 - x_3 - x_4) \ldots \cdot (x_{n-1}x_n - x_{n-1} - x_n)$. Expanding this will generate exactly $3^{n/2}$ terms; because there are $n/2$ parenthesis blocks with three terms with no common variables, hence no two term can cancel or combine into one term in the expansion.

Other Simple Bent functions and their Sign-representations can be generated by applying affine transformations to the Simple Bent function defined above, each having the same threshold density. Since the only Bent functions in dimension 2 and 4 are the Simple Bents, using Theorem 5.3 we can obtain an upper bound on the threshold density for these dimensions as $\pi(2) \leq 3$ and $\pi(4) \leq 9$. These inequalities can actually be replaced with equalities (as shown computationally) and determine the maximum density, i.e. Simple Bents are the 'hardest' problems in these dimensions. It is tempting to speculate that Simple Bents are the hardest problem for (even) $n > 4$ as well. This is also corroborated by our computational investigations with 6-variable Bent functions. We were able to find sign representations for all the Bent functions with 24 monomials, except for the Simple Bents for which we could not find a sign representation with less than 26 monomials. However, the intuition that the Simple Bents are hardest must be wrong. This surprising fact becomes apparent when Theorem 5.3 is combined with the lower bound given for the maximum density in the literature.

Corollary 5.2 Simple Bent functions cannot be the hardest problems among all n-variable Boolean functions for all $n > n_0$ for some constant n_0.

Proof. Let's assume that the Simple Bent functions are the hardest problems (i.e. they require the most number of monomials for their sign-representation compared to any other BF). This implies $\Pi(n) \leq 3^{n/2}$. The tightest known lower bound for $\Pi(n)$ is reported as 0.11×2^n (O'Donnell & Servedio, 2003). Combining the two, we can write $\Pi(n)$ as $0.11 \times 2^n \leq \Pi(n) \leq 3^{n/2}$. However, for sufficiently large n, 0.11×2^n will be greater than $3^{n/2}$. In fact it can be calculated that for $n \geq 16$ the inequality will fail. Thus, we can say that with 16 or more variables there must appear harder problems than the Simple Bent functions.

5.5 Results from 6 dimensional Bent functions

Rothaus (1976) showed that all 6-variable Bent functions belong to one of the four affine equivalence classes. Below we reproduce one representative from each Bent affine equivalence class from Rothaus (1976), which we study to obtain the sign-representations and the threshold densities for 6-variable Bent functions.

$$\begin{aligned}
g_1(x_6,x_5,x_4,x_3,x_2,x_1) =& x_1x_2 \oplus x_3x_4 \oplus x_5x_6\\
g_2(x_6,x_5,x_4,x_3,x_2,x_1) =& x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_1x_2 \oplus x_1x_4 \oplus x_2x_6 \oplus x_3x_5 \oplus x_4x_5\\
g_3(x_6,x_5,x_4,x_3,x_2,x_1) =& x_1x_2x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_3x_6\\
g_4(x_6,x_5,x_4,x_3,x_2,x_1) =& x_1x_2x_3 \oplus x_2x_4x_5 \oplus x_3x_4x_6 \oplus\\
& x_1x_4 \oplus x_2x_6 \oplus x_3x_4 \oplus x_3x_5 \oplus x_3x_6 \oplus x_4x_5 \oplus x_4x_6
\end{aligned}$$

If Conjecture 5.1 is correct, we can obtain bounds on the maximum threshold density of 6-variable Boolean functions by only studying these Bent functions. However, as mentioned before, our heuristic algorithm is ineffective against Bent functions since the spectrum is flat. In the heuristic algorithm given in Table 1, the order of the monomial elimination is not specified in the case of ties. When working with Bent functions we modified the algorithm to have the monomials picked via random permutations to search for solutions with less number of monomials. The minimum number of monomials in the solutions we could obtain for the 6-dimensional Bents are so far

26, 24, 24 and 24 for g_1, g_2, g_3 and g_4 respectively[7]. Later, we showed that all 6-dimensional BFs can be solved with less than 27 monomials by studying a representative form of each affine equivalence class[8] (of a total of 150,357) establishing that $\Pi(6) \leq 26$. So as expected Bent functions appear to be the hardest functions in dimension 6. Furthermore, g_1 which appears to be the hardest 6-variable Bent function, is a Simple Bent. According to Theorem 5.3, any Simple Bent can be solved with $\sqrt{3}^n$ monomials. Therefore, the bound implied by Theorem 5.3 is 27 which is only off by 1 monomial from the best result we have obtained for $\Pi(6)$.

5.6 About Higher Dimensions

As the number of variables (n) increase, the number of Boolean functions increase super exponentially (2^{2^n}). Equivalence classes give us a way to handle this complexity for lower dimensions. However, the affine equivalence classes for Boolean functions have been computed only up to 6-variable BFs. There are no studies that completely enumerate the affine equivalence classes (or representatives of) over 7-variable Boolean functions. Given that the number of 7-variable Boolean functions is approximately $10^{38.5}$, new methods of classification other than affine equivalence may be necessary to characterize sign representation of 7-variable Boolean functions. As noted before, the correctness of Conjecture 5.1 would allow us to focus on only Bent functions to determine the maximum threshold density of even dimensions. However, we are still in dark, as not all 8-variable Bent functions have been classified so far (Carlet, 2010).

6 Conclusion

This article presented both theoretical and computational results regarding the maximum threshold density $\Pi(n)$, that is, the minimum number of monomials required to sign-represent an n-variable Boolean function. We proposed a heuristic algorithm that provides a fast way to create sign-representations

[7]Possibly the hardest problems of dimension 6, are the ones that belong to the equivalence class of the Simple Bent function ($g_1(x_6, x_5, x_4, x_3, x_2, x_1) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6$); so, we give its sign-representation explicitly as a 26-monomial polynomial in the Appendix

[8]Results of the 150,357 problems can be obtained from http://cargo.erensezener.com/sign-rep-files.zip

with a few number of monomials. Using this algorithm we established $\Pi(4) = 9$ for the first time, and obtained minimum-sign representation solutions for all 4-variable Boolean functions. Then we studied the affine equivalence classes of Boolean functions and showed that the threshold density is invariant under affine transformations. This enabled us to study sign-representation problem for dimensions 5 and 6 and obtain interesting results for the first time. By solving representative functions from each of the equivalence classes, we established $\Pi(5) = 11$, a surprising result since only extra two monomials were sufficient to account for the doubling of the number of monomials passing from dimension 4 to 5. The result $\Pi(5) = 11$, also allowed us to obtain an easy theoretical upper bound for $\Pi(6)$ as $\Pi(6) \leq 44$, superseding previous known bound of 47 (Oztop, 2006). Again using heuristic search on the all equivalence classes of 6-variable Boolean functions we significantly improved this bound to $\Pi(6) \leq 26$. From the biological point of view, these results say that a neuron would need at most 11 presynaptic computation nodes to represent a five variable classification problem. This is 9 for a four variable problem, and it is less than 27 for a 6-variable problem. Although these classification problems seem small for a biological neuron; in higher dimension requiring a neuron to learn a fully specified Boolean function becomes out of question as it has a limited time and slow sampling rate to 'experience' each input-output pair defining the function[9].

In addition to general Boolean functions, we specifically looked at the sign representation of Bent functions and observed that they appear to be the hardest Boolean functions to sign-represent. We obtained an upper bound for the threshold density of 'Simple Bent' functions that is not limited to lower dimensions: the Simple Bent functions, i.e. those of the form $x_1x_2 \oplus x_3x_4 \oplus \ldots \oplus x_{n-1}x_n$ can always be sign-represented with $\sqrt{3}^n = 1.732^n$ monomials. Although this type of functions are the hardest to sign represent in dimensions 2, 4, and seemingly so in dimension 6; we showed that they cannot be the hardest functions for $n \geq 16$, i.e. harder functions await us in 16 dimensions. This is a curious case, and leaves two possibilities: Either there are types of Boolean functions that are harder than the Bent functions, or some new types of Bent functions have densities higher than $\sqrt{3}^n$ and they determine the maximum threshold density $\Pi(n)$. If the former is true, then

[9]Taking the sampling rate of a neuron as 10 Hertz, the neuron has to spend 13.5 years to experience all the assignment possibilities of a 32-variable problem. For a 35-variable problem this becomes 100 years

identifying those non-Bent functions might reveal interesting properties for domains such as cryptography. If the latter is true, it might be possible to obtain a tighter bound on the maximum density, $\Pi(n)$ by studying those new Bent functions.

7 Appendix

7.1 Summary of Equivalence Classes

In this section we list the summaries of results for equivalence classes of $n \leq 5$. Classes are sorted and numbered arbitrarily. Density denotes the threshold density of a class, $\pi(C_i^n)$. Function numbers give the binary representation of a BF that belongs to the equivalence class when translated from the hexadecimal number. The definition of spectrum summary is given in section 4.5. To make all these clear we give an example:

From Table-7 that lists the results of 3-variable BFs, let's pick a class, C_2^3, whose class no (i) is 2. The function number of C_2^3 is 0xab in hexadecimal, and 10001001 in binary. Based on this binary value, we can obtain the assignment vector $\mathbf{f} = [-1, 1, 1, 1, -1, 1, 1, -1]^T$ of $f \in C_2^3$ by replacing 1's with -1's and 0's with 1's. Then we can obtain the spectrum $\mathbf{s}_f = 2^{-3} \cdot \mathbf{D}_3 \cdot \mathbf{f}$, and the Walsh Spectrum $\mathbf{w}_f = [2, -2, -2, -6, 2, -2, -2, 2]^T$ by scaling $\mathbf{s}_f$ by 2^3. Lastly, we match the unique values of $|\mathbf{w}_f|$ with the number of occurrences of the each value and obtain the function's spectrum summary: $1 \times 6, 7 \times 2$.

Class No (i)	Density	Function Number	Summary of Spectrum
1	1	1	2×1

Table 5: Densities of C_i^1

Class No (i)	Density	Function Number	Summary of Spectrum
1	1	0x1	1×4, 3×0
2	3	0x2	4×2

Table 6: Densities of C_i^2

Class No (i)	Density	Function Number	Summary of Spectrum
1	1	0xaa	1×8, 7×0
2	4	0xab	1×6, 7×2
3	3	0xac	4×4, 4×0

Table 7: Densities of C_i^3

Class No (i)	Density	Function Number	Summary of Spectrum
1	1	0xaa55	1×16, 15×0
2	5	0xab55	1×14, 15×2
3	4	0xbb55	1×12, 7×4, 8×0
4	5	0xaba5	1×10, 3×6, 12×2
5	3	0xaaff	4×8, 12×0
6	5	0xaba4	2×8, 8×4, 6×0
7	5	0xab12	6×6, 10×2
8	9	0xac90	16×4

Table 8: Densities of C_i^4

Class No (i)	Density	Function Number	Summary of Spectrum
1	1	0xaa55aa55	1×32, 31×0
2	6	0xaa55ab55	1×30, 31×2
3	5	0xaa55bb55	1×28, 15×4, 16×0
4	6	0xaa5dbb55	1×26, 7×6, 24×2
5	4	0xaaddbb55	1×24, 7×8, 24×0
6	6	0xaa5dbb51	1×24, 3×8, 16×4, 12×0
7	6	0x2a5dbb51	1×22, 1×10, 10×6, 20×2
8	6	0xaaddbb51	1×22, 3×10, 4×6, 24×2
9	10	0x2a5dbf51	1×20 1×12, 30×4
10	6	0x6a5dbb51	1×20 6×8, 15×4, 10×0
11	6	0x2addbb51	1×20, 1×12, 4×8, 14×4, 12×0
12	5	0xa8ddbb51	1×20, 3×12, 12×4, 16×0
13	10	0xaeddda51	1×18, 1×10, 15×6, 15×2
14	7	0x0a5dbf51	1×18, 1×14, 12×6, 18×2
15	6	0x8addda51	1×18, 3×10, 9×6, 19×2

Class No (i)	Density	Function Number	Summary of Spectrum
16	6	0xa8dd9b51	1×18, 1×14, 2×10, 6×6, 22×2
17	6	0x88ddbb51	1×18, 3×14, 28×2
18	3	0x88ddbb11	4×16, 28×0
19	8	0x8c5dda51	1×16, 12×8, 19×0
20	5	0xa89d9b51	2×16, 8×8, 22×0
21	10	0x8eddda51	1×16, 8×8, 16×4, 7×0
22	7	0xaefdda51	1×16, 1×12, 6×8, 15×4, 9×0
23	7	0x025dbf51	2×16, 4×8, 16×4, 10×0
24	6	0x88ddda51	1×16, 2×12, 4×8, 14×4, 11×0
25	6	0x88dd9b51	2×16, 2×12, 14×4, 14×0
26	9	0xceddda51	1×14, 3×10, 13×6 , 15×2
27	10	0x0eddda51	1×14, 3×10, 13×6, 15×2
28	7	0x425dbf51	2×14, 2×10, 10×6, 18×2
29	7	0x8cddda51	1×14, 5×10, 7×6, 19×2
30	6	0x88dddb51	3×14, 1×10, 7×6, 21×2
31	6	0x289d9b51	2×14, 4×10, 4×6, 22×2
32	9	0x86fdda51	4×12, 28×4
33	11	0x88dddb71	4×12, 28×4
34	10	0xcefdda51	1×12, 10×8, 15×4, 6×0
35	9	0x0efdda51	2×12, 8×8, 14×4, 8×0
36	10	0x288d9b51	2×12, 8×8, 14×4, 8×0
37	8	0x8cfdda51	3×12, 6×8, 13×4, 10×0,
38	7	0x8cdddb51	4×12, 4×8, 12×4, 12×0
39	7	0x8ccdda51	4×12, 4×8, 12×4, 12×0
40	5	0x289d9b41	6×12, 10×4, 16×0
41	10	0x488ddb51	4×10, 16×6, 12×2
42	9	0xccfdda51	6×10, 10×6, 16×2
43	9	0x688d9b51	6×10, 10×6, 16×2
44	10	0x288d9b41	6×10, 10×6, 16×2
45	9	0x288d1b41	16×8, 16×0
46	11	0xdcfdda51	16×8, 16×0
47	9	0x68ad9b51	16×8, 16×0
48	11	0x688ddb51	12×8, 16×4, 4×0

Table 9: Densities of C_i^5

7.2 Sign-Representation of the hardest 6-variable equivalence class

In dimension 6, BFs of $C^6_{150,354}$ seem to require the most number of monomials. A function $f \in C^6_{150,354}$, and its sign-representing polynomial, $p(f)$ is given below:

$$\begin{aligned} f =&[1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, 1, 1, -1, \\ &-1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, 1, -1, \\ &-1, 1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1] \end{aligned}$$

$$\begin{aligned} p(f) =& 28x_3 - 35x_1 + 37x_5 - 35x_6 + 13x_1x_6 + 11x_2x_5 - 24x_2x_6 + 27x_3x_5 - 44x_3x_6 \\ &+ 17x_5x_6 + 28x_1x_2x_4 + 35x_1x_2x_5 - 24x_1x_2x_6 - 33x_1x_4x_6 + 29x_2x_4x_5 \\ &+ 25x_2x_5x_6 + 21x_3x_4x_6 + 38x_1x_2x_3x_4 + 37x_1x_2x_3x_5 + 31x_1x_2x_4x_5 \\ &- 23x_1x_2x_4x_6 + 14x_1x_2x_5x_6 + 17x_2x_3x_4x_6 + 19x_1x_4x_5x_6 \\ &- 31x_3x_4x_5x_6 - 19x_1x_3x_4x_5x_6 \end{aligned}$$

References

Amano, K. (2010). New upper bounds on the average ptf density of boolean functions. In O. Cheong, K.-Y. Chwa, & K. Park (Eds.), *Algorithms and computation* (Vol. 6506, p. 304-315). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-17517-6_28

Anderson, N. A. (2007). *The classification of boolean functions using the rademacher-walsh transform.* Retrieved from http://eprints.qut.edu.au/15828/

Berlekamp, E., & Welch, L. (1972, Jan). Weight distributions of the cosets of the (32,6) reed-muller code. *Information Theory, IEEE Transactions on*, *18*(1), 203-207. doi: 10.1109/TIT.1972.1054732

Bruck, J. (1990). Harmonic analysis of polynomial threshold functions. *SIAM Jorunal of Discrete Mathematics*, *3*(2), 168-177.

Carlet, C. (2010). Boolean functions for cryptography and error-correcting codes. In *Boolean models and methods in mathematics, computer science, and engineering* (p. 257-397). Cambridge University Press.

Chandru, V. (1993). Variable elimination in linear constraints. *The Computer journal*, *36*(5), 463-470.

Edwards, C. R. (1975, January). The application of the rademacher-walsh transform to boolean function classification and threshold logic synthesis. *IEEE Trans. Comput.*, *24*(1), 48–62. Retrieved from http://dx.doi.org/10.1109/T-C.1975.224082 doi: 10.1109/T-C.1975.224082

Fuller, J. E. (2003). *Analysis of affine equivalent boolean functions for cryptography* (Doctoral dissertation, Queensland University of Technology). Retrieved from http://eprints.qut.edu.au/15828/

Ghosh, J., & Shin, Y. (1992). Efficient higher order neural networks for classification and function approximation. *International journal of Neural Systems*, *3*(4), 323-350.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, *26*(23), 4972-4978.

Guler, M. (2001). A model with an intrinsic property of learning higher order correlations. *Neural Networks*, *14*(4-5), 495-504.

Hansen, K., & Podolskii, V. (2013). Polynomial threshold functions and boolean threshold circuits. In K. Chatterjee & J. Sgall (Eds.), *Mathematical foundations of computer science 2013* (Vol. 8087, p. 516-527). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-40313-2_46

Maiorana, J. A. (1991). A classification of the cosets of the reed-muller code r (1, 6). *Mathematics of Computation*, *57*(195), 403-414.

Meier, W., & Staffelbach, O. (1990). Nonlinearity criteria for cryptographic functions. In J.-J. Quisquater & J. Vandewalle (Eds.), *Advances in cryptology ? eurocrypt ?89* (Vol. 434, p. 549-562). Springer Berlin Heidelberg. doi: 10.1007/3-540-46885-4_53

Mel, B. W. (1994). Information processing in dendritic trees. *Neural Computation*, *6*, 1031-1085.

Mel, B. W., & Koch, C. (1990). Sigma-pi learning: on radial basis functions and cortical associative learning. In *Advances in neural information processing systems 2* (p. 474-481). Morgan Kaufmann publishers Inc.

O'Donnell, R., & Servedio, R. (2003). Extremal properties of polynomial threshold functions. In *Eighteenth annual conference on computational complexity* (p. 3-12).

Oztop, E. (2006). An upper bound on the minimum number of monomials required to separate dichotomies of {-1, 1}$^{\mathrm{n}}$. *Neural Computation*, *18*(12), 3119-3138.

Oztop, E. (2009, September). Sign-representation of boolean functions using a small number of monomials. *Neural Netw.*, *22*(7), 938–948. Retrieved from `http://dx.doi.org/10.1016/j.neunet.2009.03.016` doi: `10.1016/j.neunet.2009.03.016`

Preneel, B. (1994). *Analysis and design of cryptographic hash functions* (Unpublished doctoral dissertation). Cathoic University of Leuven.

Rothaus, O. (1976). On ?bent? functions. *Journal of Combinatorial Theory, Series A*, *20*(3), 300 - 305. Retrieved from `http://www.sciencedirect.com/science/article/pii/0097316576900248` doi: http://dx.doi.org/10.1016/0097-3165(76)90024-8

Rumelhart, D. E., Hinton, G. E., & Williams, R. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. McClelland, group, & PDP (Eds.), *Parallel distributed processing* (Vol. 1: Foundations, p. 151-193).

Saks, M. (1993). Slicing the hypercube. In K. Walker (Ed.), *London mathematical society lecture note series 187: Surveys in combinatorics* (p. 211-255). Cambridge University Press.

Schmitt, M. (2005). On the capabilities of higher-order neurons: a radial basis function approach. *Neural Computation*, *17*(3), 715-29.

Siu, K. Y., Roychowdhury, V., & Kailath, T. (1995). *Discrete neural computation.* Englewood Cliffs, NJ: Prentice Hall.

Taylor, J. G., & Coombes, S. (1993, March). Learning higher order correlations. *Neural Netw.*, *6*(3), 423–427. Retrieved from `http://dx.doi.org/10.1016/0893-6080(93)90009-L` doi: `10.1016/0893-6080(93)90009-L`

Zhang, C., Yang, J., & Wu, W. (2011, May). Binary higher order neural networks for realizing boolean functions. *Neural Networks, IEEE Transactions on*, *22*(5), 701-713. doi: `10.1109/TNN.2011.2114367`