Indian Statistical Institute
Kolkata

On Search of Highly Nonlinear Boolean Functions

Manmatha Roy
Supervisor: Subhamoy Maitra

A thesis submitted in partial fulfilment of
the degree of M.Tech in Computer Science

11th July 2020

Contents

PART I

Preliminary

CHAPTER 1

Boolean Functions

At the very beginning, we will define some basic definitions related to boolean functions and their analysis keeping cryptographic applications in mind.

1.1 Definition of Boolean Functions

Let F_2^n be the vector space over prime field F_2. And x be a vector of length n over F_2. Then boolean function f is defined as a mapping from F_2^n to F_2.

$$f : F_2^n \to F_2$$

Like any other mapping from finite domain, boolean functions can be also be uniquely described by enumeration of function values at all the elements from its domain set.

$$f = \{(\sigma, f(\sigma)) \quad | \quad \forall \sigma \in F_2^n\}$$

For particularly boolean functions, this representation became so popular, it even got a name for this. It is called truth table of boolean function. It is visually depicted by a table of 2^n rows and 2 columns, where in the first column there are all possible vectors of F_n^2 and in the second column there are values of the boolean function at those vectors taken in the same order as they were in the first column.

We suppose that the arguments of a function (i.e., vectors of length n) follow in lexicographical order. This assumption does not any way affect the definition of boolean function apart from uniqueness of the representation.

For example, if $n = 3$, the order is $(000), (001), (010), (011), (100), (101), (110), (111)$
Let us come back to the truth table representation For instance, the following are Boolean functions: $g\colon F_2^2 \to F_2$ such that $g(00) = g(11) = 1, g(01) = g(10) = 0$; Then the truth table of g can be depicted as follows

x_1x_2	$g(x_1x_2)$
00	1
01	0
10	0
11	1

It is to deduce that there exists 2^{2^n} boolean functions of the form $f : F_2^n \to F_2$. We will call this set of boolean functions of n variable B_n.

1.2 Algebraic Normal Form

Apart from truth table representation, another way of representing boolean functions is through algebraic formulation of the boolean system.

Algebra of finite fields asserts that there exist unique multivariate polynomial over $GF(2^n)$ for each of boolean functions of n variable.

Let $\oplus$ and $\cdot$ denote the usual addition and multiplication operation under F_2. Note that sometimes we will be doing abuse of notation here by skipping the $\cdot$ sign and end up writing xy instead of $x \cdot y$. It is known that any boolean function can be uniquely represented by its algebraic normal form.

$$f(x_1x_2...x_n) = \sum_{k=1}^{n} \sum_{i_1 i_2 ... i_k} a_{i_1 i_2 ... i_k} x_{i_1} x_{i_2} ... x_{i_n} \oplus a_0$$

where for each k indices $i_1, ..., i_k$ are pairwise distinct and sets $\{i_1, ..., i_k\}$ are exactly all different nonempty subsets of the set $\{1, ..., n\}$; coefficients $a_{i_1 ... i_k}$, a_0 take values from F_2.

If someone has any experience of building equivalent algebraic expression for a boolean function from its truth table through Karnough Map techniques, she may remember the problem of non uniqueness of the final algebraic expression. We want to stress on the fact that algebraic normal form uniquely represents a boolean function. Thanks to earlier research in this area, given a truth table one can derive it's algebraic normal form. The algorithm is quite involved.

Some more definitions are now ready to be given at this point.

- For a boolean function f , the number of variables in the longest item of it's algebraic normal form, is called the algebraic degree of a function (or briefly degree) and is denoted by $\deg(f)$. A boolean function is called affine, quadratic, or cubic when it's degree is 1, 2, or 3 respectively. A special case of affine function, when $a_0 = 0$, is called linear function.
- A boolean function is called homogeneous, if all the terms of it's algebraic normal form contain equal number of variables only.
- A boolean function is called non-degenerate, if it's algebraic normal form contains all the variables.

1.3 Walsh Hadamard Transformation

Here comes the central tool of analysis of boolean functions.

The Walsh-Hadamard transform of a boolean function f in n variables is the integer-valued function on F_n^2 defined as

$$W_f(\gamma) = \sum_{x \in F_2^n} (-1)^{\langle x.\gamma \rangle + f(x)} \quad \forall \gamma \in F_2^n$$

Numbers $W_f(\gamma)$ are called Walsh-Hadamard coefficients of a boolean function f . The ordered multiset $W_f = \{W_f(x) \colon x \in F_2^n\}$,where vector $x \in F_2^n$ in lexicographical order, is the Walsh-Hadamard spectrum of a function f.

There are many important lemmas involving Walsh-Hadamard transformation, many of them central to the analysis of boolean functions. We state here some of them, we would be using in our work [CS17a].

Theorem 1.3.1. *Walsh-Hadamard spectrum determines a boolean function in a unique manner.*

Theorem 1.3.2. *For a boolean function f in n variables and an arbitrary vector $x \in F_2^n$,*

$$(-1)^f(x) = \frac{1}{2^n} \sum_{\gamma \in F_2^n} W_f(\gamma)(-1)^{\langle x.\gamma \rangle}$$

Theorem 1.3.3. *A function f is balanced if and only if $W_f(0) = 0$.*

Theorem 1.3.4. *For a Boolean function f in n variables, Parseval's equality holds*

$$\sum_{\gamma \in F_2^n} (W_f(\gamma))^2 = 2^{2n}$$

Theorem 1.3.5. *For every Boolean function f in n variables,*

$$\max_{\gamma \in F_2^n} |W_f(\gamma)| \geq 2^{\frac{n}{2}}$$

1.4 Important Cryptographic Properties of Boolean Functions

Property	Definition
Hamming Weight	The Hamming weight of a binary string T, denoted by $wt(T)$, is the number of 1's in T.
Balanced	An n-variable function f is said to be balanced if it's truth table contains an equal number of 0's and 1's, i.e., $wt(f) = 2^{n-1}$
Nonlinearity	Let $A(n)$ be the set of all n-variable affine functions. Then nonlinearity of a function f of n variable is defined as $$nl(f) = \min_{g \in A(n)} d(f, g)$$
Correlation immunity	A function f is said to be correlation immune (CI) of order m (m-CI) if for every m indices $1 \leq i_1 < i_2 < ... < i_m \leq n$ and for every tuple $(a_1 a_2 ... a_m) \in F_2^m$, $$Prob(f(x) = 1 \mid (x_{i_1}, x_{i_2}, x_{i_3} ... x_{i_n} = (a_1, a_2, ... a_m)) = Prob(f(x) = 1)$$
Auto-correlation	Another important results in the area of boolean function analysis is that given any n variable boolean function f, it is always possible to get a boolean function g with degree at most $\lceil \frac{n}{2} \rceil$ such that fg is of degree at most $\lceil \frac{n}{2} \rceil$ Here the functions are considered to be multivariate polynomials over F_2 and fg is the polynomial multiplication over F_2. • Given $f \in B_n$, a nonzero function $g \in B_n$ is called an annihilator of f if $fg = 0$. By $AN(f)$ we mean the set of annihilators of f. • Given $f \in B_n$, the algebraic immunity of f, denoted by $\mu(f) = deg(g)$, where $g \in B_n$ is the minimum degree nonzero function such that either $fg = 0 or (1 + f)g = 0$ [CS17b] [CS17d] [CS17e].

1.5 Document Structure

The rest of the text is organised as follows

Chapter 2 introduces bent functions, their properties in a detailed manner.

Chapter 3 provides sketches on Patterson-Weidemann construction construction technique of boolean functions of odd no of variable beating bent concatenation bound.

Chapter 4 details our efforts on computer aided search techniques for finding Patterson-Weidemann functions

Chapter 5 details our efforts for the same using quantum powered systems

Chapter 6 conclusion and sketches direction of further research

CHAPTER 2

On Bent Functions

2.1 Non-linearity Revisited

In earlier portion of the text, Walsh Hadamard transformation of the boolean function was defined as

$$\begin{aligned} W_f(\gamma) &= \sum_{x \in F_2^n} (-1)^{\langle x.\gamma \rangle \oplus f(x)} \quad \forall \gamma \in F_2^n \\ &= \sum_{x \in F_2^n} (-1)^{l_\gamma \oplus f(x)} \quad \forall \gamma \in F_2^n \end{aligned} \tag{2.1}$$

Where $l_\gamma = \langle x.\gamma \rangle \in L_n$ the set of all linear functions of n variable.

Careful inspection allow us to interpret the expression in the following way

- Fix any $\gamma \in F_2^n$
- Fixing γ also fixes $l_\gamma = \langle x.\gamma \rangle$
- $\forall x \in F_2^n$, compare function values of our function of interest f and l_γ.
- Start a counter with zero. count $+1$ when they are matching, -1 otherwise.
- After all x have been processed, counter value is Walsh-Hadamard value at γ

Therefore it can reformulated in the following way

$$\begin{aligned} W_f(\gamma) &= \#(f = l_\gamma) - \#(f \neq l_\gamma) \\ &= 2^n - 2.\#(f \neq l_\gamma) \\ &= 2^n - 2.d(f, l_\gamma) \\ &= 2^n - 2.wt(f \oplus l_\gamma) \end{aligned} \tag{2.2}$$

At this point, we recall the definition of nonlinearity of a boolean function f in n variables. It is the hamming distance with respect to the set of all affine functions. Mathematically it can be expressed as follows

$$nl(f) = \min_{g \in A_n} dist(f, g)$$

If, among all the linear functions of n variable, the minimum distance of f occurs for the linear function l_γ, then above equation implies for that $W_f(\gamma)$ has the maximum value in the Walsh spectrum of f.

On the other hand, if the minimum distance occurs for the affine function h_γ, then $-W_f(\gamma)$ will have the maximum value.

Therefore, in terms of the Walsh spectrum, the nonlinearity of f is given by

$$nl(f) = 2^{n-1} - \frac{1}{2} \max_{\gamma \in F_{2^n}} |W_f(\gamma)|$$

As we already stated in earlier section, for any boolean function f, maximum of absolute value of Walsh-Hadamard spectrum is lower bounded by $2^{\frac{n}{2}}$, nonlinearity of any boolean function is upper bounded at $2^{n-1} - 2^{\frac{n}{2}-1}$ [CS17a]

2.2 Definition of Bent Functions

If n is even, then the maximal possible value of nonlinearity can be found using the expression

$$2^{n-1} - 2^{\frac{n}{2}-1}$$

. When n is odd, then maximum nonlinearity bound can't be found using this expression, in fact it still remains as one of the unsolved p. Leaving this complications for the later part of this text, we move to define bent functions.

1. ***A bent function is a maximal nonlinear Boolean function with an even number of variables. Its nonlinearity is*** $2^{n-1} - 2^{fraqn2-1}$

2. ***A bent function is a boolean function in*** n ***variables (*** n ***is even) such that***

$$|W_f(\gamma)| = 2^{\frac{n}{2}} \quad \forall \gamma \in F_2^n$$

3. ***A bent function is a boolean function in*** n ***variables (*** n ***is even) such that for any nonzero vector*** y ***its derivative*** $D_y f(x) = f(x) \oplus f(x \oplus y)$ ***is balanced, that is, it takes values*** 0 ***and*** 1 ***equally often***

First two definitions directly follows from our discussion. Intuition for the third one can be described as follows.
It is called 'bent' because it is maximum possible distance from all of the linear and affine functions lying in the same space.

2.3 Properties Of Bent Functions

Though our work involves boolean functions of odd no of variables only with high nonlinearity, we will provide some interesting results on bent functions. The reason is that even if the type of functions we will be working with are not bent functions, they have uncanny similarity with bent functions. For example many of the attempts of the construction methods were inspired by some construction methods of the bent function. In some sense, bent functions are closest neighbours of the functions we would be dealing with. We will study bent functions to have a rough understanding on different aspects of functions with high nonlinearity.

The following property of bent functions is very important and plays a key role in proving many results in bent functions

Degree of Bent Functions

The degree $\deg(f)$ of a bent function f in $n \geq 4$ variables is not more than $\frac{n}{2}$. If $n = 2$, a bent function is quadratic.

Invariance Under Affine Transformations

It is desirable that a nonlinearity criterion remains invariant under a large group of transformations. For many applications, this symmetry group should contain the group of affine transformations. The perfect nonlinear or bent functions are indeed invariant under this group, as we shall see now.
First we will define affine equivalence in a more formal way.

f and g are affinely equivalent if there is a nonsingular $n \times n$ matrix A and a vector b of length n, such that

$$g(x) = f(Ax \oplus b) \quad \forall x \in F_2^n$$

Additionally f and g are extended affinely equivalent if there is a nonsingular $n \times n$ matrix A, vectors b and c of length n, and a constant $\lambda \in F_2$, such that

$$g(x) = f(Ax \oplus b) \oplus \langle c, x \rangle \oplus \lambda \quad \forall x \in F_2^n$$

Now we are ready to state our main theorems.

1. a Boolean function $f(Ax \oplus b)$ is bent, where A is an $n \times n$ invertible matrix over F_2 and b is an arbitrary vector of length n.
2. a function $f \oplus l$ is bent for any affine function l .

Thus, class B_n is closed up to any non-degenerate affine transformation of variables and addition of any affine boolean function.[CS17c]

Duality of Bent Functions

For a bent function f , the dual function $f\prime$ in n variables is defined by the equality

$$W_f(\gamma) = 2^{\frac{n}{2}}(-1)^{f\prime(\gamma)}$$

This definition is correct since $|W_f(\gamma)| = 2^{\frac{n}{2}} \quad \forall \gamma \in F_2^n$ It is not hard to prove that the function $f\prime$ is a bent function too. It holds that $f = f\prime$ [CS17c].

Hamming Weight of Bent Functions

Every bent function in n variables is of Hamming weight of $2^{n-1} \pm 2^{\frac{n}{2}-1}$ [Tok15a]

Nondegeneracy of Bent Functions

A bent function in n variables is nondegenerate; that is, all variables are presented in its algebraic normal form. [Tok15a]

2.4 Equivalent Represenation of Bent Functions

Hadamard Matrices

A Hadamard matrix is a square $k \times k$ matrix A with elements ± 1 such that $AA\prime = kI$. Let us enumerate the rows and columns of a $2n \times 2n$ matrix with binary vectors x and y of length n.

Theorem 2.4.1. *The following statements are equivalent [Tok15a]:*

2.4.1.1. *A Boolean function f in n variables is bent.*

2.4.1.2. *$A = (a_{xy}), where a_{xy} = 2^{-fraqn2} W_f(x \oplus y)$, is a Hadamard matrix.*

2.4.1.3. *$D = (d_{xy}), where d_{xy} = (-1)^{f(x \oplus y)}$, is a Hadamard matrix.*

Difference Sets

From the beginning, bent functions were studied in connection with difference sets. Let a finite Abelian group G have order v and be presented in the additive form. A subset $D \subseteq G$ of size k is called a difference set with parameters (v, k, λ) if every nonzero element $g \in G$ can be represented in the form $g = b - d$ exactly in λ ways, where b and d are elements of the set D.

Theorem 2.4.2. *A Boolean function f in n variables is a bent function if and only if the set $D = \{(x, f(x))|x \in F_2^n\}$ is a difference set with parameters $(2^{n+1}, 2^n, 2^{n-1})$ in the additive group Z_2^{n+1}*

Strongly Regular Graph

Consider the Cayley graph $G_f = G(F_2^n, supp(f))$ of a Boolean function f. All vectors of length n are vertices of the graph. There is an edge between two vertices x and y if vector $x \oplus y$ belongs to $supp(f)$.A regular graph G is called strongly regular if there are nonnegative integers λ and μ such that for any vertices x and y the number of vertices incident to x and y is both equal λ or μ and depends on the presence or absence of the edge between x and y. The following is proven:

Theorem 2.4.3. *A Boolean function f is bent if and only if the Cayley graph $G_f$is strongly regular and $\lambda = \mu$.*

Bent Rectangles

Let f be a Boolean function in n variables, $n = r + k$. Let us represent the vector of values $textit\mathbf{f}$ of a function f in the form $textit\mathbf{f} = (f_1, ..., f_{2^r})$, where every vector f_i has length 2^k. Let f_i be a boolean function in k variables for which f_i is the vector of values. Let us consider the matrix M_f of size $2^r \times 2^k$ with spectral vectors $W_f(1), ..., W_f(2^r)$ as rows. A matrix of size $2^r \times 2^k$ is called a bent rectangle if every line (row or column) multiplied by $2^{(r-\frac{n}{2})}$ is a spectral vector of the appropriate boolean function. The following theorem holds

Theorem 2.4.4. *A Boolean function f is a bent function if and only if the matrix M_f is a bent rectangle*

2.5 Construction of Bent Functions

Maiorana-Mc Farland's class	the best known construction of bent functions defined in bivariate form (explicit construction). $f_{\pi,g}(x,y) = x.\pi(y) + g(y)$ with $\pi : F_2^m \to F_2^m$ and $g : F_2^m \to F_2$
Dillon's Partial Spreads class PS^-	well known construction of bent functions whose bentness is achieved under a condition based on a decomposition of its supports (not explicit construction) $support(f) = \cup_{i\in[2^{m-1}]} E_i^*$ where E_i are m dimensional vector spaces with $E_i \cap E_j = \phi$[Tok15b] [Tok15c]
Dillon's Partial Spreads class PS_{ap}	Functions are defined explicitly in bivariate form :$f(x,y) = g(xy^{2^m-2})$ with g as a balanced Boolean function on F_2^m which vanishes at 0 [Tok15b] [Tok15c]
.	

2.6 Application of Bent Functions

Bent Functions in Cryptography

- In the arena of symmetric key cryptography, where the outputs of several linear feedback shift registers (LFSRs) are combined by a nonlinear Boolean function to generate the keystream, the correlation between the keystream and one of the LFSR outputs (or a linear combination of the LFSR outputs) is used to obtain the key. In other words, a correlation attack can be mounted if there is a high correlation between the combining function and a linear function, which implies low nonlinearity. Hence, as it is well known, high nonlinearity provides resistance against correlation and fast correlation attacks. Though bent functions are not directly used in building cryptographic primitives, study on them sheds helps the design as well as analysis of the ciphers a lot.
- Their derivatives $D_f : x \to f(x) + f(x+a)$ are balanced, this has an important relationship with the differential attack on block ciphers.

Bent Functions in Coding Theory

- The covering radius plays an important role in error correcting codes. It essentially measures the maximum errors to be corrected in the context

of maximum-likelihood decoding. The Covering radius $\rho(1, n)$ of the Reed-Muller code $RM(1, n)$ coincides with the maximum nonlinearity $nl(f)$.

- It is well-known that Kerdock codes are constructed from bent functions. Moreover, bent functions can also be used to construct linear codes with few weights. Such codes have applications in secret sharing, authentication codes, regular graphs.

2.7 Open Problems

Equivalent Representation

Bent functions can be represented in terms of Hadamard matrices, difference sets, block schemes, linear spreads, sets of subspaces in the Boolean cube, strongly regular graphs, and bent rectangles;The problem is to obtain a new equivalent representation of bent functions in order to get a classification of them [Tok15a]

Exact No of Bent Functions

It is known that there are exactly $8,896,5425430528 \approxeq 2^{32.3}, and 2^9 \cdot 193887869660028067003488010240 \approxeq 2^{106.29}$ bent functions in two, four, six, and eight variables, respectively. But what is the exact number of bent functions if $n \geq 10$ [Tok15e]?

Construction of Bent Functions

Propose new direct constructions of bent functions. For now the simplest constructive class of bent functions is the Maiorana-McFarland class. Are there other (larger) classes of bent functions with examples that are so easily constructed [**Tok17b**] [**Tok17c**] ?

Algebraic Criteria of Bent Functions

Propose an algebraic characterization of bent functions—that is, find necessary and sufficient conditions on a trace form (or polynomial form) of a boolean function to be bent. Think about other algebraic representations of bent functions [Tok15g].

Cryptographic Properties of Bent Functions

Study the connections between bentness and other cryptographic properties. For example, prove that there are bent functions in n variables of the maximal

possible algebraic immunity $fraqn2$. For this you need to prove the special combinatorial conjecture[**Cs17c**] [Tok15d]?

Duality of Bent Functions

Let A be a subset of F_n^2. Let B be the set of all binary vectors from F_n^2 that are at the maximal possible distance from the set A. Now let $A\prime$ be the set of all vectors that are at the maximal possible distance from B. We call a set A metrically regular if $A = A\prime$. In the case of regular sets, it is possible to say that there is duality between the definitions of A and B; the set A defines B and vice versa. It is proven that the set of vectors of all bent functions (affine functions) is metrically regular. Are there other such sets in the Boolean cube? Give a classification of them [CS17c] [Tok15d]?

Bounds on no of Bent Functions

Obtain the new (better) upper and lower bounds for the number of bent functions in n variables. For now there is a large gap between the simple lower $2^{2^{\frac{n}{2}}+log(n-2)-1}$ and upper $2^{2^{n-1}+\frac{1}{2}\binom{n}{\frac{n}{2}}}$ bounds for this number. There are several improvements of these bounds, but they are not too big [Tok15e] ?

Bent Decomposition Problem

Is it true that an arbitrary Boolean function in n variables $(niseven, n \geq 2)$ of degree not more than $\frac{n}{2}$ can be represented as the sum of two bent functions in n variables [Tok15f]?

Optimal Codes via Bent Functions

A code C of length 2^n is a constant-amplitude code if every nonzero code word is a vector of values of some bent function in n variables. Such codes are very important for transmission in code division multiple access. The following natural problem arises: How does one get constructions of linear optimal constant amplitude codes? What is the maximal possible dimension of such a code for any even n [Tok15c]?

PART II

Problem Definition

CHAPTER 3

The Curious Case of Odd n

The design of Boolean functions on an odd number of variables n achieving very high nonlinearity, constitutes one of the most challenging problems encountered in the area of cryptography, coding theory, and combinatorics. Until 1983, it was believed one can not get functions of odd no of variables having nonlinearity exceeding bent concatenation bound.

Bent Concatenation Bound: It was proved that for odd n, one can get a function f of high nonlinearity using two bent functions of $(n-1)$ variable in following way

$$f = (1 \oplus x_n) f_0 \oplus x_n f_1$$

One can prove that, nonlinearity of f is equal to $2^{n-1} - 2^{\frac{n-1}{2}}$. For long time it was believed to be highest nonlinearity one can expect from boolean functions of odd no variables. This constructions essentially concatenates two smaller truth table to build a big one only thus giving rationale behind the name.

In their seminal paper [PW83], Patterson and Wiedemann in 1983 could come up with a construction, involving combinatorial arguments over algebraic elements, for functions with non linearity exceeding bent concatenation bound. Since then numerous attempts have been made in finding high nonlinearity functions of odd variables. We summarize all the best results achieved so far in this direction of research in bent functions.

1. In 1970, it has been shown that for $n = 5$, the maximum nonlinearity of n-variable boolean functions is exactly equal to the bent concatenation bound, which is 12.
2. In 1980, the question for $n = 7$ could be solved and it has been noted that here also the maximum nonlinearity is the bent concatenation bound which is 56.
3. In 1983, it was shown that one can construct a 15-variable boolean function f with nonlinearity $(2^{15-1} - 2^{\frac{15-1}{2}} + 20) = 16276$. This construction was named after the two authors of the original e.g. Patterson Weidemann construction. It is well known that using this function,

one can construct any n-variable Boolean function F with nonlinearity $(2^{n-1} - 2^{\frac{n-1}{2}} + 20.2^{\frac{n-15}{2}})$ for $n > 15$. In fact, F can be written as $f \oplus g$, where g is an $(n-15)$-variable bent function. Note that this construction does not preserve the type of function e.g. structure of F and f are not same.

4. In 2009, 9-variable Boolean functions with nonlinearity $241 = (2^{9-1} - 2^{\frac{9-1}{2}} + 1)$ were identified in the rotation-symmetric class and subsequently this result was improved to 242 by defining the generalized rotation symmetric class [**KY10**].

5. In 2016, boolean functions with 21 variables with non linearity exceeding bent concatenation bound could be found by modifying Patterson Weidemann construction, e.g. incorporation of notion of generalized rotation symmetric boolean function. One thing has to be noted here about this result. The functions we got in this result has non linearity $2^{21-1} - 2^{\frac{21-1}{2}} + 20$. While concatenation of Patterson Weidemann type functions of 15 variable, with bent functions of 6 variable one can get nonlinearity $(2^{n-1} - 2^{\frac{n-1}{2}} + 20.2^{\frac{n-15}{2}})$, which remains as upper bound of non linearity for 21 variable till date. But such functions are not Patterson Weidemann type functions. Hence this new functions can be considered as Patterson Weidemann type functions of 21 variable with highest non linearity [KM16].

In light of above mentioned progress of research in this area, we would be now focusing on Patterson Weidemann constructions and its other expositions as well as later modifications.

3.1 The Patterson Weidemann Construction

Before diving into actual work of the duo , let us have fast recap on some related definitions from the topic of boolean functions.

- **Support:** Support of a boolean function is defined as

$$sup(f) = \{x \in F_2^n | \quad f(x) = 1\}$$

  It is to be noted that support of a boolean function uniquely determines the function.

- **Trace representation of linear function:** Trace representation of a linear function $l_\alpha = \alpha \cdot x$ can be described in the following way

$$l_\alpha = tr_1^n(\alpha x)$$

  where $tr_1^n(x) = \sum_{i=0}^{n-1} x^{2^i}$
  It is to be noted that, support of a linear function l_α defined as

$$Sup(l_\alpha) = \{x \in F_2^n | \quad tr_1^n(x) = 1\}$$

  In the same way, support of the corresponding affine function h_α defined as

$$Sup(h_\alpha)\{x \in F_2^n | \quad tr_1^n(x) = 0\}$$

- For two functions f and g on same no of variables n, their hamming distance can be described as symmetric difference between supports of those functions

$$d(f, g) = wt(sup(f) \oplus sup(f))$$

Now we are ready to look into Patterson-Weidemann construction.

Let a and b be two ppositive integers such that $n = a.b$ and a and b are coprime to each other. Then we define the following galois fields for our purpose

- $M = GF(2^{ab})$
- $L = GF(2^a)$
- $J = GF(2^b)$
- $K = GF(2)$

Let M^*, L^*, J^*, K^* the corresponding multiplicative group. Now the index of L^* in M^* is

$$m = \frac{2^{ab-a}}{2^a - 1}$$

Then M^* can be written as

$$M^* = \cup_{i \in [m]} L^* \cdot x_i$$

where $x_1, x_2, ..., x_m$ are coset representatives from respective equivalence classes of cosets.

Patterson and Weidemann considered functions whose supports are of the form

$$sup(f) = \cup_{i \in [p]} L^* \cdot x_i, \quad p \leq m$$

Let $I_{a,b}$ be the set of all such functions. What they showed that for any $f \in I_{a,b}$ and for arbitrary linear(affine) functions $l_\alpha(h_\alpha)$, following inequalities hold

- $d(f, \mathbf{0}) = l(2^a - 1)$
- $d(f, \mathbf{1}) = 2ab - l(2^a - 1)$
- $d(f, h_\alpha) = 2^{ab-1} - 2^a \cdot t(\alpha) + l$
- $d(f, l_\alpha) = 2^{ab-1} - 2^a \cdot t(\alpha) + l$

In earlier expression $\mathbf{0}$ and $\mathbf{1}$ are constant functions with all 0 values and all 1 values respectively, $t(\alpha)$ is the number of cosets of the form $L \cdot x_i$ which are present the support hyperplane of the corresponding linear(or affine) functions. Now by definition of nonlinearity of boolean functions,

$$nl(f) = min(l(2^a - 1), 2^{ab} - l(2^a - 1), 2^{ab-1} - (2^a \cdot t(\alpha) + l, 2^{ab-1} - (2^a \cdot t(\alpha) + l) \quad \forall \alpha \in F_2^n$$

For any of these functions with nonlinearity $nl(f) \geq 2^{ab-1} - 2^{\frac{ab-1}{2}}$ each of the four terms in the *min* expression should be greater than $2^{ab-1} - 2^{\frac{ab-1}{2}}$ for all values of $\alpha \in F_2^n$.Rearranging them we get,

- $\frac{2^{ab-1} - 2^{\frac{ab-1}{2}}}{2^a - 1} < l < \frac{2^{ab-1} + 2^{\frac{ab-1}{2}}}{2^a - 1}$
- $\frac{1}{2^a}(\frac{2^{ab-1} - 2^{\frac{ab-1}{2}}}{2^a - 1} - 2^{\frac{ab-1}{2}}) < l(\alpha) < \frac{1}{2^a}(\frac{2^{ab-1} + 2^{\frac{ab-1}{2}}}{2^a - 1} + 2^{\frac{ab-1}{2}})$

Patterson-Weidemann's next observation was when $b = 3$ then the cosets of $L^* \in M^*$ form the Desarguesian projective plane $PG(2, 2^a)$. Now, suppose $n = 15, a = 5, b = 3$. Let L^* and J^* be the multiplicative group of $GF(2^3)$ and $GF(2^5)$ respectively. One can identify the group M^* to the group $\psi(M^*)$ of left multiplications by the elements of M^* in $GL_K(M)$ and this correspondence is an isomorphism. Let $\phi_2 \in GL_K(M)$ be the Frobenius automorphism of M defined by

$$\phi_2(x) = x^2 \quad \forall x \in M$$

The group $\langle\phi_2\rangle$ is a cyclic group of order ab and lies within $GL_K(M)$. The group $\langle\phi_2\rangle$ acts trivially on the projective plane $PG(2, 2^a)$. Then they considered the action of the group

$$G = [\psi(L^*) \cdot \psi(J^*)]\langle\phi_2\rangle/\psi(L^*)$$

,where $[\psi(L^*) \cdot \psi(J^*)]\langle\phi_2\rangle$ is the semidirect product of $\psi(L^*)$ and $\psi(J^*)$ by $\langle\psi\rangle$, on the supports of the functions they picked earlier. This is in view of constructing supports of functions in $I_{5,3}$ which are invariant under the action of G and also satisfy the required inequalities.

Here comes the counting part, which clears the rationale behind this setup

- All of the $(2^3-1)(2^5-1) = 217$ elements from the cyclic subgroup, derived as direct product of L^* and J^*, are of same value and that holds true for all the elements in each of cosets too.
- Then comes the invariance under frobenius transformation constraints. This divides the $\frac{2^{15}-1}{(2^3-1)(2^5-1)} = 151$ elements into 10 groups of 15 size each and 1 with only 1 element. Class of single element can be initially assumed to be of zero value.
- Weight restriction imposes that one must choose 5 orbits of the 10 orbits of size 15. . Therefore, the total number of possible choices is $\binom{10}{5}$. Exhausting all the possibilities they have obtained two solutions up to complementation. The functions corresponding to these two solutions have nonlinearity 16276.

In summary, Patterson-Weidemann construction can be summarized in following way.

1. Picking up the class of functions of whose supports are of the form

   $$sup(f) = \cup_{i\in[p]} L^* \cdot x_i, p \leq m$$

   It is to be noted that existence of such functions will known prior their work. In fact such functions are being heavily used in Dilon's construction of bent functions.
2. To come up with simple expressions in form of inequalities for feasibility of such functions
3. Search space for above mentioned inequalities still remains large. Here comes the trickiest part. They considered only those functions whose supports are invariant under certain group action. This trick substantially reduced the search space.

3.2 Notion of Interleaved Sequences

Let $a = \{a_0, a_1, a_2, ..., a_{m-1}\}$ be an arbitrary binary string of length $m = 2^n - 1$. Now fundamental rule of finite field tells us that given a primitive element $\varsigma \in GF(2^n)$ we can always come with a function f such that

$$f(0) = 0$$
$$f(\varsigma^i) = a_i$$

where $i \in [m-1]$. It is to be clearly noted that function f is specific to the given sequence a as well as the the primitive element ς. It should be written as $f_{a,\varsigma}$. For the sake of convenience we would do an abuse of notation and continue with f. If we change the primitive element then we obtain a different function.

Now comes converse part. if f is a function from $GF(2^n)$ to $GF(2)$ with $f(0) = 0$ and $\varsigma \in GF(2^n)$ is a primitive element of the respective field then the sequence $\{f(1), f(\varsigma), f(\varsigma^2), ..., f(\varsigma^{n-2})\}$ is referred to as the sequence associated to f with respect to ς.

More formally speaking, there is a bijection here between these two sets.

$$\begin{aligned} if \quad & \sigma : \Sigma \times A \to F \qquad \delta : F \times \varsigma \to A \\ and \quad & \varsigma \in \Sigma, \quad a \in A \\ then \quad & \delta(\sigma(\varsigma, a), \varsigma) = a \end{aligned}$$

where Σ be the set of all primitive roots of $GF(2^n)$, A be the set of 2^{n-1} length binary strings, F being the set of all boolean functions of n variable.

Now we will define interleaved sequence.
Suppose m is a composite number such that $m = d \cdot k$ where d and k are both positive integers greater than 1, a is a binary sequence $\{a_0, a_1, a_2, ..., a_{m-1}\}$ where $a_i \in F_2 \forall i \in [m]$, then the (d, k)-interleaved sequence $a_{d,k}$ corresponding to the binary sequence a is defined as

$$\begin{matrix} a & a_1 & a_2 & \cdot & \cdot & a_d \\ a_d & a_{1+d} & a_{2+d} & \cdot & \cdot & a_{d+d} \\ a_{2d} & a_{1+2d} & a_{2+2d} & \cdot & \cdot & a_{d+2d} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{(k-1)d} & a_{1+(k-1)d} & a_{2+(k-1)d} & \cdot & \cdot & a_{d+(k-1)d} \end{matrix}$$

Let $d \cdot k = m = 2^n - 1$. And we define $a_{i+jd} = f(\varsigma^{i+jd})$. This will be called as interleaved sequence of f with respect to ς.

the connection beween Patterson-Weidemann construction and interleaved sequences of functions is here that

- First note that, if we fix $\varsigma \in \Sigma$, any interleaved sequence along with the assumption that $f(0) = 0$ acts like support of a function. It unquely identifies the function.
- Assumption of invariances under certain group actions on the support set of the function, puts extra restriction on the choice of our interleaved sequence from set of all interleaved sequences of same length e.g. reduced search space. The sequence is to be arranged in such a way that this reduction of search space can be handled with ease.
- It is to be kept in mind the way of deriving nonlinearities of Patterson Weidemann functions in earlier setup is not valid here. They have to carried out again for this setup. Arranging the interleaved sequence should not make this task too complicated.

Now we are ready to state our first observation related to interleaved sequence.

If the support of a function $f \in F_n$ is invariant under the action of a cyclic subgroup K of order k of $GF(2^n)^*$ then the $(2^{n-1}, k)$ interleaved sequence of f with respect to any primitive element of $GF(2^n)$ has a fixed binary sequence of length $\frac{2^n-1}{k}$ as rows. Conversely, if the $\frac{2^n-1}{k}$ interleaved sequence of f with respect to a primitive element $\varsigma \in GF(2^n)$ has a fixed binary sequence of length $\frac{2^n-1}{k}$ as rows, then the support of f is invariant under the action of K.

Proof: If $d = \frac{2^n-1}{k}$, then ς^d is clearly a generator of the cyclic subgroup K.

$$k = \langle \varsigma^d \rangle$$

If the support of f is invariant under the action of K then $f(\varsigma^i) = f(\varsigma^{i+jd})$ for $j \in [k-1]$ and $i \in [d-1]$. As a consequence the i-th column of $a_{d,k}$ is constant for each i.
On the other hand if $\mathbf{a}$ has a fixed binary sequence of length d as rows, then $a_i = a_{i+jd}$. Then the support of the function corresponding to this interleaved sequence will be invariant under action of any group of order k.

Now we will see that how does the interleaved sequences behave with addition restriction of invariance under the action of the group of Frobenius automorphisms ϕ_2.

We will first define a suitable equivalence relation on the index set of our interleaved sequence.

> Let ρ be an equivalence relation on $\{0, 1, 2, ..., (d-1)\}$ by $i_1 \rho_d i_2$ if and only if $i_1 \cong 2^j i_2 \mod d$ for some non-negative integer j where $i_1, i_2 \in \{0, 1, 2, ..., (d-1)\}$.

The set $\{0, 1, 2, ..., (d-1)\}$ is the set of column numbers of the (d, k) interleaved sequence of a boolean function. Thus ρ_d partitions this set into equivalence classes.

Now we will state our second observation related to this interleaved sequence.

> A function f invariant under the action of K is also invariant under the action of ϕ_2 if and only if the (d, k)-interleaved sequence of the function has a fixed binary sequence of length d as rows and the columns in the same equivalence class with respect to ρ_d are either 'all zero' columns(denoted as **0**) or 'all one' columns(denoted as **1**).

Proof : Any function f invariant under the action of K and ϕ_2 is invariant under the action of K and hence by earlier observation the (d, k)-interleaved sequence of this function has a fixed binary sequence of length d as rows.

Consider the i-th column where Under the Frobenius automorphism ψ_j, the element ς^i is mapped to $\varsigma^{2^j.i}$. If the support of f is invariant under the Frobenius automorphisms then

$$f(\varsigma^i) = f(\varsigma^{2^j.i})$$

Using euclid's division algorithm, we can always come up with such $q_{i,j}$ and $r_{i,j}$ so that

$$2^j \cdot i = q_{i,j} d + r_{i,j}$$

where $0 \leq r_{i,j} < d$.

Now if we peek into our interleaved sequence, element $f(\varsigma^{2^j.i})$ will be sitting on the $q_{i,j}$-th row and $r_{i,j}$ -th column in the (d, k)-interleaved sequence of f.

Since, due to the restriction of the invariance under the action of K, the columns of this interleaved sequence corresponding to f are either **0** or **1** columns, the $r_{i,j}$ -th column of the interleaved sequence has the same value as $f(\varsigma^i)$. In particular

$$f(\varsigma^{r_{i,j}}) = f(\varsigma^{ri,j})$$

.
Thus all the columns which are in the same equivalence class of ρ_d has the same value.

Conversely if the function f has the (d,k)-interleaved sequence with the above mentioned property then for any $j \geq 0$ and $i \in \{0,1,2,...,(d-1)\}$, $f(\varsigma^{2^j.i})$ appears in the $q_{i,j}$ -th row and $r_{i,j}$ -th column. But $r_{i,j} \cong 2^j.i \bmod d$, therefore $r_{i,j}$ and i are in the same equivalence class of ρ_d. Hence

$$f(\varsigma^i) = f(\varsigma^{2^j.i})$$

Thus the function is invariant under the action of ϕ. Again since the interleaved sequence under consideration has a fixed binary sequence of length d as rows the corresponding function f is invariant under the action of the group K.

We will now move to analysing the structure the interleaved sequence of the linear(affine) functions.

A function of the form $f(x) = tr_1^n(\alpha x^c)$ where $\alpha \in GF(2^n)$ and $gcd(c, 2^{n-1}) = 1$ is called a bijective monomial. When $c = 1$, f is the linear function l_α.

Suppose $t|n$ and $d = \frac{2^{n-1}}{2^{t-1}}$. Then the following theorem holds

Let $f(x) = tr_1^n(x^c)$. Then for all $(d, 2^{t-1})$ interleaved sequence of f with respect to a primitive element $\varsigma \in GF(2^n)$ is such that

1. The columns are either **0** or cyclic shifts of the binary sequence corresponding to $tr_i^t(x)$ when evaluated at $\varsigma^{c.i.d} \forall i \in [2^-1]$ which contains 2^{t-1} 1's.
2. The number of **0** columns is $d - 2^{n-t}$

Proof: Let ς be a primitive 2^{n-1}-th root of unity. Then the entry in the i-th column and j-th row of the $(d, 2^{t-1})$- interleaved sequence of f is

$$\begin{aligned} u_{i,j} &= f(\varsigma^{i+jd}) \\ &= tr_1^n(\varsigma^{c(i+jd)}) \\ &= tr_1^t(tr_t^n(\varsigma^{c(i+jd)})) \\ &= tr_1^t(\sum_{p=0}^{\frac{n}{t}-1} \varsigma^{2^p(ci+cjd)}) \\ &= tr_1^t(\sum_{p=0}^{\frac{n}{t}-1} \varsigma^{2^p.ci} \varsigma^{cjd}) \end{aligned} \tag{3.1}$$

The sequence $\{u_{i,j} | j = 0,1,...,2^t - 2\}$ corresponds to the linear function of the form $tr_1^t(z_i x)$, where $z_i = \sum_{p=0}^{\frac{n}{t}-1} \varsigma^{2^p.ci}$ when evaluated at the points ς^{cpd} where $p \in [2^t - 1]$.

Now observe if for some i, z_i becomes 0, then the i-th column consists of only zeros. Otherwise it is cyclic shift of the binary sequence generated by $tr_1^t(x)$

when evaluated at the points ς^{cpd} where $p \in [2^t - 1]$. Now we are done with the first part of the proof.

For the second part, observe that since each non-zero column is a sequence corresponding to a linear function with respect to the element ς^{cd}, the number of 'one's in each them is 2^{t-1}.
On the other hand, the number of 'one's in the binary sequences corresponding to $f(x)$ and $tr_1^n(x)$ are equal, therefore the total number of 'one's in the binary sequence corresponding to $f(x)$ is 2^{n-1}.

Let the number of non-zero columns be r. Then

$$r.2^{t-1} = 2^{n-1}$$

$$or, r = 2^{n-t}$$

So, the number of zero columns is $d - r = d - 2^{n-t}$

Now we will do the trickest part of this section, calculation on non linearity through interleaved sequence.

First note that for any affine function $h_\alpha = tr_1^n(\alpha x) + 1$ as a we can always prepare a $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence. By earlier calculation, the number of **1** columns in the interleaved sequence of h_α is $d - 2^{ab-a}$.

Now note that for Patterson-Weidemann functions type, where the support of f is union of cosets of the type $L * x_i$ earlier deductions dictates that corresponding $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence of f will have a fixed binary sequence of length $\frac{2^{ab}-1}{2^a-1}$ as rows.

Remember the original work of Patterson-Weidemann in earlier section of this chapter. They introducced $t(\alpha)$ as the number of cosets in the support of f that are contained in h_α. This is equivalent to the number of **1** columns of the $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence of f that correspond to the **1** columns of the $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence of h_α.

As our final goal is to search for a function $f \in I_{ab}$ having nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for any $\alpha \in GF(2^n)$. Therefore we need,

$$2^{n-1} - 2^{\frac{n-1}{2}} < d(f, h_\alpha) < 2^{n-1} + 2^{\frac{n-1}{2}}$$

In particular for $\alpha = 0$, we get,

$$2^{n-1} - 2^{\frac{n-1}{2}} < l(2^a - 1) < 2^{n-1} + 2^{\frac{n-1}{2}}$$

where l is the number of **1** columns in the $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence of f and consequently

$$\frac{2^{n-1} - 2^{\frac{n-1}{2}}}{2^a - 1} < l < \frac{2^{n-1} + 2^{\frac{n-1}{2}}}{2^a - 1}$$

Now, for other values of α, out of $\frac{2^{ab}-1}{2^a-1}$ columns of h_α, number of **1** columns is $\frac{2^{ab}-1}{2^a-1} - 2^{ab-a}$ and among them $t(\alpha)$ number of **1** columns match with the **1** columns of f.

This is same as saying that $t(\alpha)$ number of **0** columns of l_α match with the **1** columns of f. From this we have

$$d(f, l_\alpha) = t(\alpha)(2^a - 1) + (l - t(\alpha))(2^a - 1) + (2^{ab} - l + t(\alpha)2^{a-1}$$

$$= 2^{ab} + t(\alpha)2^a - l$$

Similarly we get

$$d(f, h_\alpha) = 2^{ab} - t(\alpha)2^a + l$$

Again for the function f to have nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ for any $\alpha \in GF(2^n)$, both of $d(f, h_\alpha)$ and $d(f, l_\alpha)$ have to greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. Therefore we need,

$$\frac{1}{2^a}(\frac{2^{ab-1} + 2^{\frac{ab-1}{2}}}{2^a - 1} - 2^{\frac{ab-1}{2}}) < t(\alpha) < \frac{1}{2^a}(\frac{2^{ab-1} - 2^{\frac{ab-1}{2}}}{2^a - 1} + 2^{\frac{ab-1}{2}})$$

Note that we got exactly those inequalities what we got in earler section.

It is to be noted that any $(\frac{2^{ab}-1}{2^a-1}, 2^a - 1)$-interleaved sequence with a fixed binary sequence of length $\frac{2^{ab}-1}{2^a-1}$ as rows correspond to a function in I_{ab} and conversely. If we construct such an interleaved sequence with l non-zero columns satisfying above equations then by the above discussion the function corresponding to this sequence will have nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$. However it is usually impossible to search all the possibilities. Because of this reason Patterson and Wiedemann have put extra restriction in the form of invariance under aforesaid operations and exhaustively searched the more restricted search space for $n = 15$. However the analogous search space even for $n = 21$ becomes too large to search exhaustively

3.3 Modified Patterson Wiedemann Construction

Patterson Weidemann construction when viewed through the lenses of interleaved sequences, appears very neat. But there is one caveat. Search space remains same as it was in the original construction. It essentially makes it difficult to adopt for higher values of n.

A fantastic enhancement to the method was proposed later [GKM06].

It reduces the search space further by applying notion of generalized rotation symmetric boolean functions e.g. they restricted the search space further by considering only the subset of k-RSBF.

Suppose K be a proper subgroup of $GF(2^n)^*$ of order k and index d containing $GF(2^t)^*$ where $t|n$.

Consider any f which is invariant under the action of K and ψ. Also let us suppose that all the interleaved sequences considered are with respect to a particular primitive element $\varsigma \in GF(2^n)$. The index of $\mathrm{GF}(2^t)^*$ in $GF(2^n)^*$ is

$$d_1 = \frac{2^n - 1}{2^t - 1}$$

Since $kd = 2^n - 1 = d_1(2^t - 1)$ and $(2^t - 1)|k$ it can easily deducted that $d|d_1$. By the invariance property of the support set of f under the action of K and ψ, the (d, k)-interleaved sequence of f has a

- fixed binary sequence of length d as rows
- all the columns in the same equivalence class of ρ_d are of same type e.g. either **1** columns or **0** columns.

If originally l columns of the (d, k)-interleaved sequence of f were **1** columns, then $(d_1, 2^t - 1)$-interleaved sequence of f then each column of the (d, k)-interleaved sequence splits up into $\frac{d_1}{d}$ number of columns. Thus the total number of **1** columns in the $(d_1, 2^t - 1)$-interleaved sequence is $l.\frac{d_1}{d}$.

Since invariance under action of set implies invariance under any subset of that set, support of the function f will remain invariant under the action of $GF(2^t)^*$, as $GF(2^t)^* \in K$. Therefore the $(d_1, 2^t 1)$-interleaved sequence of f is also repetition of a fixed binary sequence of length d_1 as rows.

Using earlier knowledge, function f has nonlinearity greater than $2^{n-1} - 2^{\frac{n-1}{2}}$ if and only if

$$\frac{2^{n-1} - 2^{\frac{n-1}{2}}}{2^t - 1} < l.\frac{d_1}{d} < \frac{2^{n-1} + 2^{\frac{n-1}{2}}}{2^t - 1}$$

$$\frac{1}{2^t}(\frac{2^{n-1} + 2^{\frac{n-1}{2}}}{2^t - 1} - 2^{\frac{n-1}{2}}) < t(\alpha) < \frac{1}{2^t}(\frac{2^{n-1} - 2^{\frac{n-1}{2}}}{2^t - 1} + 2^{\frac{n-1}{2}})$$

Here we analyse the invariance under teh action of the group ψ in a tighter way.

Because ρ_d is an equivalence relation defined on the column numbers of (d,k)-interleaved sequence of f. Suppose that there are r equivalence classes. Define r distinct binary variables $l_0, l_1, ..., l_{r-1}$ such that $l_j = 1$ if the j-th equivalence class consists of $\mathbf{1}$ columns else $l_j = 0$. Let s_j be the size of the j-th equivalence class where $j \in [r]$.

In the (d,k)-interleaved sequence of the function f if the j-th equivalence class has columns with entries l_j then corresponding to these columns there are $s_j \frac{d_1}{d}$ columns in the $(d_1, 2^{t-1})$-interleaved sequence of f with entries l_j. Let S_j be the set of column numbers of these columns.

Consider the $(d_1, 2^{t-1})$-interleaved sequence of $tr_1^n(\varsigma^i x)$. Let T_i be the set of column numbers of $\mathbf{0}$ columns of this interleaved sequence. Let $c_{ij} = |T_i \cap Sj||$. From this we obtain

$$t(\varsigma^i) = \sum_{j=0}^{r-1} c_{ij} l_j$$

The number c_{ij} is the number of $\mathbf{0}$ columns of the $(d_1, 2^{t-1})$-interleaved sequence of $tr_1^n(\varsigma^i x)$ having the same column numbers as the columns corresponding to the j-th equivalence class in the $(d_1, 2^{t-1})$-interleaved sequence of f. Thus earlier constraints can be written as

$$\frac{1}{2^t}\left(\frac{2^{n-1} + 2^{\frac{n-1}{2}}}{2^t - 1} - 2^{\frac{n-1}{2}}\right) < \sum_{j=0}^{r-1} c_{ij} l_j < \frac{1}{2^t}\left(\frac{2^{n-1} - 2^{\frac{n-1}{2}}}{2^t - 1} + 2^{\frac{n-1}{2}}\right)$$

The number of inequalities in the above system is $2^n - 1$. Below we prove that it is enough to solve r inequalities among them, where r is the number of equivalence classes with respect to ρ_d.

Let f invariant under the action of K and ψ. If i and j are in the same equivalence class of ρ_d i.e., $j \cong i2^k \bmod d$ for some $k \geq 0$, then

1. $W_f(\varsigma^i) = W_f(\varsigma^j)$
2. $t(\varsigma^i) = t(\varsigma^j)$

Proof:
Let $j = i2^k - qd$

$$\begin{aligned}
W_f(\varsigma^j) &= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^j x)+f(x)} \\
&= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^j \varsigma^{qd} x)+f(\varsigma^{qd} x)} \\
&= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^{i2^k} x)+f(x)} \\
&= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^{i2^k} x)+f(x)} \\
&= W_f(\varsigma^{i2^k}) \\
&= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^{i2^k} x^{i2^k})+f(x)} \\
&= \sum_{x \in GF(2^n)} (-1)^{tr(\varsigma^i x)+f(x)} \\
&= W_f(\varsigma^i)
\end{aligned} \tag{3.2}$$

The $(d_1, 2^{t-1})$-interleaved sequence of $tr(\varsigma^i x)$ has $t(\varsigma^i)$ zero columns corresponding to **1**columns of the $(d_1, 2^{t-1})$-interleaved sequence of $f(x)$. The total number of zero columns of the $(d_1, 2^{t-1})$-interleaved sequence of $tr(\varsigma^i x)$ is $d_1 - 2^{n-t}$ and the total number of **1** columns and **0** columns of the $(d_1, 2^{t-1})$-interleaved sequence of $f(x)$ are $l\prime$ and $d - l\prime$ respectively.

The number of zero columns of $tr(\varsigma^i x)$ that correspond to **1** columns of $f(x)$ is $t(\varsigma^i)$. The number of **1** columns of $f(x)$ that correspond to nonzero columns of $tr(\varsigma^i x)$ is $l\prime - t(\varsigma^i)$ and the number of zero columns that correspond to the nonzero columns of $tr(\varsigma^i x)$ is $2^{n-t} - (l\prime - t(\varsigma^i))$. Thus the Walsh transform

$$W_f(\varsigma^i) = (2^{t-1})t(\varsigma^i) + (1)(l\prime - t(\varsigma^i)) + (-1)(2^{n-t} - (l\prime - t(\varsigma^i)))$$

. Thus for any i, j if $W_f(\varsigma^i) = W_f(\varsigma^j)$ then $t(\varsigma^i) = t(\varsigma^j)$

3.4 The Case of $n = 15$

We now discuss about preparing interleaved sequences corresponding to Patterson Weidemann type bent functions of 15 variable.

1. We consider $15 = 5 \times 3$.
2. $|L^*| = 2^3 - 1, |J^*| = 2^5 - 1, |K| = |J^* \times L^*| = (2^5 - 1)(2^3 - 1) = 31.7$
3. since $k = 31.7$, $d = \frac{2^15-1}{31.7} = 151$
4. ρ_{151} has 11 equivalence classes where $\rho_{151} : i_1 \rho_{151} i_2 \implies i_1 \cong 2^s i_2$ mod 151 for some $s \in Z^+$.
5. By definition, $d_1 = 151.7$
6. Using apropriate inequality, we get $74 < l < 76$

3.5 The Case of $n = 21$

We now discuss about preparing interleaved sequences corresponding to Patterson Weidemann type bent functions of 15 variable.

1. We consider $21 = 7 \times 3$.
2. $|L^*| = 2^3 - 1, |J^*| = 2^7 - 1, |K| = |J^* \times L^*| = (2^7 - 1)(2^3 - 1) = 127.7$
3. since $k = 127.7$, $d = \frac{2^15-1}{31.7} = 2359$
4. ρ_{2359} has 115 equivalence classes where $\rho_{2359} : i_1 \rho_{2359} i_2 \implies i_1 \cong 2^s i_2$ mod 2359 for some $s \in Z^+$.
5. By definition, $d_1 = 2359.7$
6. Using apropriate inequality, we get $58 < l < 72$

3.6 Algorithm: Prepare-Inequalities(n, ς)

Here is the concrete algorithms which can be used to generate inequalities for any odd n as long as $n = a.b$. $gcd(a, , b) = 1$ holds true.

1. Factorize $n = a.b \quad a < b$
2. $d = \frac{2^n-1}{2^a-1} \quad k = (2^b - 1)$
3. Evaluate $tr_1^n(\varsigma^i) \quad \forall i \in [2^n - 1]$ and fill up the interleaved sequence $a = (d.k)$ accordingly.
4. Find no of **0** columns as $d - 2^{n-a}$. Let Z be the array containing the index of **0** columns.
5. Find the equvalence classes on the index no of columns as induced by ρ_d. Let E contains all the representatives of each of the equivalece classes.
6. Set $i = 0$
7. Let L of length equals to d such that $E[L[j]]$ is the representative of the equivalence class of ρ_d containing j.
8. Initilize C and K to 0
9. for $j = 0, 1, ..., 2^b$, do the following
   a) $K[j] = (Z[j] - E[i]) \bmod d$
   b) $m = K[j] \bmod d$
   c) $C[L[m]] = C[L[m]] + 1$
10. Out put $C = (c_{i,0}, c_{i,0}, ...c_{i,|E|})$
11. $i = i + 1$. Go to Step 7 till $i < |E|$.

PART III

Our Works

CHAPTER 4

Solving PWF21 Inequalities

4.1 Definition of Integer Programming

Finding solutions for the derived set of inequalities along with with appropriate weight condition, falls into a well known mathematical model called Integer Linear Programming.

Standard form of such problems are defined as follows

$$\begin{aligned} \max \quad & c\prime x \\ where, \quad & Ax \leq B \\ & x \geq 0 \\ & x \in Z_n^+ \end{aligned} \tag{4.1}$$

Two things are to noted at this point.
Firstly feasibility problem e.g. to find a feasible solution to given instance, is no less hard than optimization e.g. finding optimum value of the optimization problem, as long as integer linear problem is concerned.

Secondly any other problem with form other than above, can be converted into standard form. For example, in our case, problem instance has following form

$$\begin{aligned} \max \quad & c\prime x \\ subject \quad & to, \\ & Ax \leq b \\ & Gx \geq h \\ & x \geq 0 \\ & x \in Z_n \end{aligned} \tag{4.2}$$

We negate G and h to convert it into standard form

$$
\begin{aligned}
\max \quad & c\prime x \\
subject \quad & to, \\
& Ax \leq b \\
& Gx \geq h \\
& x \geq 0 \\
& x \in Z_n
\end{aligned}
\tag{4.3}
$$

On Hardness

Now we move to hardness part of our problem instance. Integer programming is NP-complete. In particular, the special case of $0-1$ integer linear programming, in which unknowns are binary, and only the restrictions must be satisfied, is one of Karp's 21 NP-complete problems. There is one caveat.

> NP-Completeness does not mean all instances of the given problem are equally difficult. In fact there might exist many easy instances of the problem. NP Completeness asserts that there exists some instances which are difficult to solve.

ILP does not have much such 'easy type instance' known to us. There are few. Probably most famous of them is 'Totally Unimodular Matrices'

> If the problem instance is of the following form
>
> $$
> \begin{aligned}
> \max \quad & c\prime x \\
> subject \quad & to, \\
> & Ax = B \\
> & x \geq 0 \\
> & x \in Z_n
> \end{aligned}
> \tag{4.4}
> $$
>
> and A is unimodular e.g. matrix for which every square non-singular submatrix is unimodular; equivalently, every square submatrix has determinant $0, +1$ or -1.
> Then usual polynomial time LP algorithm returns exact solution.

Our careful inspection along with suggests that problem instance we are working with , is indeed a 'hard instance'

4.2 Using Exact Solvers For Integer Programming

The naive way to solve an ILP is to simply remove the constraint that x is integer, solve the corresponding LP. This particular step got a very intuitive name e.g. LP relaxation. Then round the entries of the solution to the LP relaxation. But, not only may this solution not be optimal, it may not even be feasible; that is, it may violate some constraint.

When the matrix A is not totally unimodular, there are a variety of algorithms that can be used to solve integer linear programs exactly. One class of algorithms is *cutting plane methods* which work by solving the LP relaxation and then adding linear constraints that drive the solution towards being integer without excluding any integer feasible points.

Another class of algorithms are variants of the *branch and bound method*. For example, the branch and cut method that combines both branch and bound and cutting plane methods. Branch and bound algorithms have a number of advantages over algorithms that only use cutting planes. One advantage is that the algorithms can be terminated early and as long as at least one integral solution has been found, a feasible, although not necessarily optimal, solution can be returned. Further, the solutions of the LP relaxations can be used to provide a worst-case estimate of how far from optimal the returned solution is. Finally, branch and bound methods can be used to return multiple optimal solutions.

Experimental Setup

All the experiments we performed, were carried out in the following environment.

System	E5-2690 v3 CPU with 24 cores and 96GB of RAM
Tools	cplex(branch-and-cut, cutting plane), lpsolve(branch-and-cut), GLPK(branch-and-cut, cutting plane), Gurobi(branch-and-cut, cutting plane, trees of tree search)

Our rigorous experimentation with these solvers shows that gurobi always outperforms other solvers in all aspects. So the all the observations that would be given follow, should be considered as observations from the experiments with gurobi.

Experiments Performed With Exact Solvers

1. **SEARCHING WHOLE SPACE:**
   We encoded the problem *lp* format asked ILP solver to search for solutions. But running for 24 hours on above mentioned setup, none of them were able to find any solution.

2. **SEARCHING PARTIAL SPACE:**
   A smaller solution space is easier to search for the solvers. Therefore, we cut the solution space by fixing a few bits from the known solutions of the equations. We created different problem instances for the solver, fixing different number of bits, varying from 46 - 65 (bits selected randomly).

Observations From Experiments

Observation 1: Sensitivity of (existing) solutions This observation is related to the question of how much change we can inflict to a solution, before it potentially converts to a new solution. In other words, how much portion of the solution determines a solution 'uniquely'?

We started with partial solutions e.g. setting some variables as per one of the existing solutions, and then tried to solve the set of inequalities as a feasibility problem using a LP solver. Number of variables, being set in this way, was varied from 100 to 40 with a pretty fine grained manner during the whole course of experiment. Not to mention that same procedure was repeated with all of the four existing solutions. We observed that when we were setting less than 55 variables then, our solver does not return any solution even within 24 hours. On the contrary, setting more than 55 variable returns the original solution(from the partial solutions were taken) in a couple of minutes(sometimes hours). This clearly demonstrates that even randomly chosen 55 variable, when set in 'right way' will eventually gravitate to 'the solution'.

Observation 2: Distance of (existing) solutions To further dig into the the above mentioned issue, we considered known solutions in pairs only to find out at what positions both of them agree and then accordingly set the bits while searching in partial solution space. We again observed convergence into existing solution in a couple of minutes. We repeated the experiment with all pairs of known solutions available to us. There was no exception in it e.g. always converged into a existing solution.This observation is probably a stronger than the previous one.

Observation 3: (Almost) Dimensional Reduction Motivation was to find mutually (something like) 'orthogonal pairs' of equations from the set of equations. We observed that making some equation to satisfy lower bounds pushes some other equations to violate upper bounds. In existing solutions they were spared by narrow margin at respective bounds. We searched for such suitable pair of collections of equations and found 29 of them. The interesting part of the whole thing is, if we now redefine (setting low margins) upper and lower bounds of those equation, following an existing solution, and leave rest of the equations with usual bounds, and this new set of inequalities when given to a solver, returns the original solution in couple of hours. Note that this observation possibly tells something stronger than the first one. Instead of setting individual variables, here we are narrowing margins (as advised by an existing solution) for 'mutually orthogonal' pairs and getting same solution back. And that too only 29 in number. This observation can be used directly as a goodness heuristics in search procedure

4.3 Using Pseudo Boolean Solvers

Before going into details of this experimentation, we will have quick look into various related definitions.

LITERALS

A literal is a propositional variable v or its negation $\neg v$.

PSEUDO BOOLEAN CONSTRAINTS

A pseudo-boolean (PB) constraint (sometimes called linear zero-one constraint or simply linear inequality) in normal form is a constraint of the form

$$c_1 l_1 + \cdots + c_n l_n \geq k,$$

where $c_1, \ldots, c_n$ (the weights) and k (the threshold) are positive integers, and $l_1, \ldots, l_n$ are boolean literals.

PSEUDO BOOLEAN FORMULA

A PB formula P is a set of PB constraints over a set of variables V.

ASSIGNMENT

An assignment σ is called satisfying assignment or solution of P if the assignment satisfies all the constraints in P.

CONJUNCTIVE NORMAL FORM

A Boolean formula F over the set of variables V is in Conjunctive Normal Form (CNF) if F is expressed as conjunction of clauses wherein each clause is a disjunction over a subset of literals. The problem of Boolean Satisfiability (SAT) seeks to ask whether there exists a satisfying assignment of F.

Experimental Setup

All the experiments we performed, were carried out in the following environment.

System	E5-2690 v3 CPU with 24 cores and 96GB of RAM
Tools	RoundingSAT(Psuedo-boolean solver), CryptoMiniSAT(SAT solver)
Encoding	OPB(Psuedo-boolean solver), DIMACS(SAT solver)

EXPERIMENTS PERFORMED WITH PB SOLVERS

Just like exact solvers, we performed experiments in both of whole solution space as well as partial solution space in the same way we did in case of exact solvers.

OBSERVATION FROM THE EXPERIMENTS

Experimentation with PB and SAT solvers confirms our obseravtions in our experiments with exact solvers. In 24 hours, the solver could solve instances only if more than 56 bits are set.

> In each of the cases, there were no other solutions apart from the known solution. We blocked the solution which has been used to set the bits, and the solver returned that the problems are unsatisfiable, therefore we are sure that no other solution exists other than the used one. In fact this observation leads to a stronger intution about structure of the solutions. It says that solutions are fairly apart from each other. This might be related to another unsolved problem in this arena e.g. hamming distance of bent funnction in some way.

In table **??**, we summerize the time taken by RoundingSAT to solve different instances. CryptoMiniSat, despite it was given almost 24x resources to solve an instance, was taking much more time to solve.

4.4 Using Metaheuristics Methods

Apart from those methods for exact solving, we tried with several metaheuristics based techniques to come up with a feasible solution. List of the techniques is given below

1. Tabu Search
2. Simulated Annealing
3. l_p Alternating Direction Method of Multiplers

We mostly used opensource implemenation of these techniques to adapt to our requirements and did experiments. Unfortunately our attempts with these methods could be done with best of our efforts. and nothing fruitful came up with our limited experimentation.

known bits	seed	solution	time (s)
46	49	3	TO
47	34	2	TO
48	18	1	TO
49	35	2	TO
49	22	1	TO
49	37	2	TO
49	17	1	TO
50	33	2	TO
51	50	3	TO
52	21	1	TO
53	41	2	TO
54	59	3	TO
54	43	2	TO
55	6	0	TO
55	5	0	TO
55	52	3	TO
56	54	3	TO
56	24	1	TO
56	23	1	TO
56	19	1	TO
56	56	3	TO
57	57	3	3418.46
57	36	2	TO
57	53	3	TO
58	26	1	TO
59	58	3	36740.99
60	42	2	9591.37
61	7	0	10028.86
61	45	2	12669.94
61	38	2	15543.45
61	27	1	25297.87
61	60	3	57606.54
61	55	3	6520.71
61	40	2	TO
62	30	1	18222.45
62	25	1	33855.4
62	44	2	5110.37
62	39	2	67118.54
63	4	0	2423.26
63	15	0	2853.66
63	20	1	3835.54
64	29	1	1808.03
64	64	3	2086.71
64	8	0	649.92
65	32	1	18226.49
65	9	0	1989.78
65	13	0	646.51
65	31	1	656.8

Table 4.1: Time taken to solve different PB instances while some of the bits are fixed. Each row represents a PB instance. Column 1 represents number of bits fixed from the solution mentioned in column 3. Bits were selected randomly with random seed mentioned in column 2. Column 4 represents time taken to solve the instance (in seconds). TO represents that the solver timed out after running for 24 hours.

CHAPTER 5

PWF21 in Quantum Paradigm

5.1 Quantum Annealing Technique

Quantum annealing controls quantum fluctuations to search for the minimum of a cost function, possibly a multivariate function. The method is a generic approximate method , a lots of real-life problems, which are possible to be formulated as a combinatorial optimization problem, can also be formulated as quantum annealing. This set of optimization problems include portfolio and route optimization problems. These set of problems has a huge social importance and researchers from various fields have tried to solve these problems in numerous different methods. Quantum-mechanical methods has been attracted significant attraction from researchers in recent days.

The procedure for performing quantum annealing include writing the cost function in terms of Ising model of magnetism. The relation between the Ising model and the annealing is the following : the Hamiltonian of the Ising model should be chosen in such a way that solution to the combinatorial optimization problem should be represented by the ground state or lowest energy state. Once such a model is chosen, the Hamiltonian is modified by adding a term that represents the quantum-mechanical fluctuations. This induces quantum transitions between states.

Choosing the quantum term is an interesting factor. Generally, a value too large in comparison to the cost function is chosen as the term in the beginning, leading to a uniform probability to the states, such that all the states from quantum-mechanical superposition exist with equal probability. In practice this states are easy to prepare. In the second step, the the quantum term is gradually reduced. Therefore, the state of the system starts to follow the time-dependent Schrödinger equation, which is the natural evolution of a physical system over time. When this coefficient reaches, reaches zero, only one of the term of the Hamiltonian remains, which is the Ising model. If we identify the ground state of the model, that remains in this process, we have found the solution. For a successful process, at the end of the annealing procedure, the probability of the solution is the largest.

5.2 D'WAVE Architecture

We use a D'Wave quantum computer to execute the above mentioned quantum annealing procedure. The quantum processing unit (QPU) for D'Wave works in the following manner:

- Similar to all modern quantum devices build till date, the QPU here also runs in a framework that resembles the von Neumann architecture and runs within that framework which contains a arithmatic and logical unit - ALU, memory I/O and control unit.
- We may think that the QPU is an ALU that has qubits instead of bits, couplers and the control unit has been made specialized to handle all these.
- A quantum machine instruction - QMI is defined, which specifies how one should specify the desired output. This QMI programs the network of qubits and couplers, and mentions how one can get the output, which is a vector consisting of spin values, $s_i \in \{\pm 1\}$.

Specifying the desired output in accordance to the input is the most significant part in quantum processing unit's description. The input to the Ising Model optimization is, in fact this desired output. The problem of optimization is :

Given a graph $G = (V, E)$ with fields h_i on vertices V and interactions J_{ij} on edges E, find a spin vector S that minimizes the objective function

$$E(S) = \sum_{(i,j)\in E} J_{ij} s_i s_j + \sum_{i \in V} h_i S_i$$

If the graph is not planar, then the optimization problem turns out to be NP-hard. We provide the following data to the quantum machine instruction :

- $h = \{h_i\}$
- $J = \{J_{ij}\}$,
- T : anneal time interval

The calculation by the quantum device can be affected by external noise or other limitations, and that in turns mean that the result returned is not always guaranteed to be the ground state solution. Therefore, it is a wise decision to run the annealing procedure multiple times before for each input. And this requires another input parameter: how many times should we repeat the procedure, or, how many solutions do we want.

If we provide more parameters to the quantum machine instruction, we'll get different annealing procedures, generally termed as anneal path features. A simple example of that can be specifying the anneal as a piecewise linear waveform. This can be used for modifying the evolution near phase transition.

Another example can be modifying a single qubit annealing schedule that limits the degree by specifying anneal offsets. Furthermore, if we specify the initial qubit values to quickly check the neighbouring values, that procedure is called reverse anneal protocol.

As it was needed in GM, here in annealing based quantum processing unit, we do not need to specify the instructions needed in every step. Instead, we are needed to specify the desired output and an optimal spin assignment, objective function defined in terms of input. The quantum algorithm does the remaining tasks. Therefore it is clear that a declarative programming paradigm works better than imperative one in case of quantum annealing.

5.3 QUBO Formulation of Our Problem

For a variety of combinatorial optimization problems, it has been established in recent research that QUBO, or, Quadratic Unconstrained Binary Optimization problem can perform quite nicely.
QUBO solvers can be used efficiently to solve different problems by reformulation, that are easy to apply. After the reformulation, the problems are given to the QUBO framework. Along with quantum annealing, QUBO model is used in Fujitsu's digital annealing. Neuromorphic computing also uses QUBO as a subject for study. Due to all these reasons, QUBO models are widey in experimentation by y D-Wave Systems and neuromorphic computers developed by IBM.

The QUBO model is expressed by the optimization problem:

$$minmize \quad y = x\prime Qx$$

where x is a vector of binary decision variables and Q is a square matrix of constants.

We can assume that the Q matrix is in upper triangular form or symmetric. We can achieve this form without loss of generality in the following manner:

Upper triangular form: $\forall\ i$ and j with $i > j$, replace q_{ij} by $(q_{ij} + q_{ji})$. Once this is done, replace all q_{ij} for $i < j$ by 0. (If the matrix is already symmetric, this procedure doubles the q_{ij} values above the main diagonal, and then fixes all values below the main diagonal to 0).
Symmetric form: $\forall\ i$ and j except $i = j$, replace q_{ij} by $\frac{(q_{ij}+q_{ji})}{2}$.

Handling Inequality Constraints

We assume that A and b have integer components. Now. if the problem has inequality constraints, we can put them in this form by including slack variables and then representing the slack variables by a binary expansion.

(For example, this would introduce a slack variable s to convert the inequality $4x_1 + 5x_2 - x_3 \leq 6$ into $4x_1 + 5x_2 - x_3 + s = 6$, and since clearly $s \leq 7$

s can be represented by the binary expansion $s_1 + 2s_2 + 4s_3$ where s_1, s_2, s_3 are extra binary variables. We convert the constrained quadratic optimization models to an equivalent unconstrained QUBO models by the following manner :

- represent slack variables as x variables
- convert the constraints $Ax = b$ into quadratic penalties
- add these quadratic penalties to the objective function

To be pecific, for a positive scalar P, we add a quadratic penalty $P(Ax - b)\prime(Ax - b)$ to the objective function and we get

$$\begin{aligned} y &= x\prime Cx + P(Ax - b)\prime(Ax - b) \\ &= x\prime Cx + x\prime Dx + c \\ &= x\prime Qx \end{aligned} \tag{5.1}$$

where the matrix D and the additive constant c result from the matrix multiplication shown. We drop the additive constant, and the equivalent unconstrained version of the constrained problem turns to be

$$minmize \quad y = x\prime Qx$$

5.4 Experimenting with DWAVE

When we converted our problem instance into QUBO format, it became of size 231×1035. This polynomial increment of problem size happened beacuse of conversion of ineqiality constarints into equality constarints. We didnot have full access to DWAVE system but a trial access. Amount to time it got access to DWAVE sampler was insufficient to come up with any valid solution. We will try this again with a full access account in near future.

CHAPTER 6

Conclusion and Future Works

Despite knowing that integer linear programming is NP-Hard, main rationale behind our desperate attempts lies in the fact that security of cryptographic(especially symmetric key cryptography) constructions lies in non linearity of underlying non linearity components and weakness of a cipher can be demonstrated considering even a single instance of it. For example, four feasible solutions were already achieved for the set of inequalities and it would be considered as access to great resource in cryptanalysis of symmetric key ciphers.

In our experimentation, we could conclude about certain structures about the problem especially hamming distance of the solutions and orthogonality of certain set of inequalities. These observation could be used to come up with new heuristics to solve the problem. This is to be noted that we lacked computational resources to run the search procedure which we hope to circumvent in later phase of this research. May be we will be able to come with significant results then.

Another attempt worth taking is to employ quantum power to solve them. Here again comes comes the issue of hardness of particular instance. As we all know that behaviour of quantum annealing techniques is not homogeneous on all instances of the problem.

Another thing to noted here that solutions to this problem will not lead us to achieve highest nonlinearity for 21 variable boolean functions. Usual Patterson Weidemann functions of 15 variable concatenated with 6 variable bent functions still holds the record for 21 variable. It still remains as open challenge to come up with a direct construction for 21 variable.

Bibliography

[CS17a] Cusick, T. W. and Stanica, P. 'Chapter 02 - Fourier Analysis of Boolean Functions'. In: *Cryptographic Boolean Functions and Applications (Second Edition)*. Ed. by Cusick, T. W. and Stanica, P. Second Edition. Academic Press, 2017, pp. 7–29.

[CS17b] Cusick, T. W. and Stanica, P. 'Chapter 04 - Correlation Immune and Resilient Boolean Functions'. In: *Cryptographic Boolean Functions and Applications (Second Edition)*. Ed. by Cusick, T. W. and Stanica, P. Second Edition. Academic Press, 2017, pp. 55–82.

[CS17c] Cusick, T. W. and Stanica, P. 'Chapter 05 - Bent Boolean Functions'. In: *Cryptographic Boolean Functions and Applications (Second Edition)*. Ed. by Cusick, T. W. and Stanica, P. Second Edition. Academic Press, 2017, pp. 83–108.

[CS17d] Cusick, T. W. and Stanica, P. 'Chapter 07 - Stream Cipher Design'. In: *Cryptographic Boolean Functions and Applications (Second Edition)*. Ed. by Cusick, T. W. and Stanica, P. Second Edition. Academic Press, 2017, pp. 143–185.

[CS17e] Cusick, T. W. and Stanica, P. 'Chapter 08 - Block Ciphers'. In: *Cryptographic Boolean Functions and Applications (Second Edition)*. Ed. by Cusick, T. W. and Stanica, P. Second Edition. Academic Press, 2017, pp. 187–221.

[GKM06] Gangopadhyay, S., Keskar, P. H. and Maitra, S. 'Patterson–Wiedemann construction revisited'. In: *Discrete Mathematics* vol. 306, no. 14 (2006). R.C. Bose Centennial Symposium on discrete mathmematics and Applications, pp. 1540–1556.

[KM16] Kavut, S. and Maitra, S. 'Patterson–Wiedemann Type Functions on 21 Variables With Nonlinearity Greater Than Bent Concatenation Bound'. In: *IEEE Transactions on Information Theory* vol. 62, no. 4 (Apr. 2016), pp. 2277–2282.

[PW83] Patterson, N. and Wiedemann, D. 'The covering radius of the$(2^{15}, 16)Reed - Mullercodeisatleast16276$'. In: *IEEE Transactions on Information Theory* vol. 29, no. 3 (May 1983), pp. 354–356.

[Tok15a] Tokareva, N. ‘Chapter 06 - Equivalent Representations of Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 49–53.

[Tok15b] Tokareva, N. ‘Chapter 08 - Combinatorial Constructions of Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 63–72.

[Tok15c] Tokareva, N. ‘Chapter 09 - Algebraic Constructions of Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 73–80.

[Tok15d] Tokareva, N. ‘Chapter 11 - Distances Between Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 89–96.

[Tok15e] Tokareva, N. ‘Chapter 13 - Bounds on the Number of Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 107–122.

[Tok15f] Tokareva, N. ‘Chapter 14 - Bent Decomposition Problem’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 123–132.

[Tok15g] Tokareva, N. ‘Chapter 15 - Algebraic Generalizations of Bent Functions’. In: *Bent Functions*. Ed. by Tokareva, N. Boston: Academic Press, 2015, pp. 133–149.