Two New Recursive Approaches for Classification Based on Logic Gates

Rajni Goyal[1], Harshit Grewal[1]
[1]Amity University, Noida (U.P.) India.

Abstract- *There are so many methods of classification in literature but very few are based on properties of Boolean function. Autocorrelation is very important property of a given Boolean function. So, we have developed two methods to classify set of all Boolean functions based on Autocorrelation and Logic gates. We have used NSGA-II to get out results. Our method is novel and very efficient.*

Keywords: Boolean functions, Classifications, NSGA-II

I. INTRODUCTION

Cardinality of a set of Boolean functions for n-variables is too large. So, its very difficult to study all Boolean functions and check that given the Boolean function is cryptographically strong or not [1]. Classification of set of Boolean functions make this process very easy. So, Theoretical cryptographers always have great interest for any novel technique of classification. Classification methods are always welcomed by researcher because it makes representation of Boolean functions extremely easy, and Booleanfunction that belongs to same class will have same properties. In [2], Correia and Reis, have concentrated mainly three properties to classify the set of n-variable functions, first cardinality of functions in each class,second the cardinality of classes, and last the cardinality of NPN classes. In [3], authors classified Boolean functions into same cardinality equivalence classes based on permutation and combination. In [4], authors mentioned the procedure for selection of the representative from each class. Moreover in [5], authors divided set of Boolean functions into two groups namely linear and affine, and represented an algorithm to count the Boolean functions in each group. Some author tried to partition the whole set based on other properties of Boolean functions like Nonlinearity, resiliency etc. Methods listed in [6], and [3], are based on above properties. In [7], Rout n al. gave two new approaches, recursive and nonrecursive. They have partitioned the set of all Boolean functions such that exactly one affine function is representative of each class. This method was based on concatenation and recursive approach. In [8], two approaches are given, one is based on hamming distance and another on recursive. Above all, selection of representative is also animportant task. In [4], Golomb has also highlighted the procedure of selection of a representative function, with one member from each equivalence class. After analyzing all the work above, we formulated two approaches: recursive and evolutionary. In first approach, we started with 1-variable functions and went up to n-variables. For second approach, we applied NSGA-II and observed better results than previously listed results. Section-wise paper is arranged as follows: In Section 2, some denitions from literature are given. Section 3 gives a brief description evolutionary approach based on NSGA-II. In Section 4, we have developed two approaches, recursive and heuristic and classify the set of n-variables Boolean functions. Section 5 gives results and discussions of our work. In Section 6, we have concluded our work.

II. RELATED DEFINITIONS

(a) Boolean Function:

E_2 is a field of two elements and E^n is a n-dimensional vector space over E_2.

Boolean function $f(x_1, \ldots,)$, is a mapping from a set of all possible bit string of E^n to E_2. Collection of all possible Boolean function can be denoted by y and cardinality of this set is 2^{2^n}.

Number of 1's present in the string of $f(x_1, \ldots, x_n)$, is called weight of Boolean function. If number of 1's are equal to the number of 0's in the string of $f(x_1, \ldots, x_n)$, then Boolean function is called balanced Boolean function. A affine function is a Boolean function with algebraic degree one(at most). The n-variable affine function can be represent by

$f_{affine}(x) = a_1 . x_1 \oplus a_2 . x_2 \oplus a_3 . x_3 \oplus \ldots \oplus a_n . x_n \oplus c$, where a_i and $c \in E_2$. For $c = 0$ above function is called linear.

(b) Hamming Distance: The Hamming distance between two functions f_1 and f_2 can be defined as the number of truth table positions in which the functions f_1 and f_2 distinct, we denote hamming distance by denoted by $hd(f_1, f_2)$, So,

$$hd(f_1, f_2) = |\{x : f_1(x) \neq f_2(x)\}|,$$

where $|.|$ represents the cardinality of the set.

(c) Walsh Hadamard Transform: By Walsh Hadamard Transform W_{HT}, we represent a Boolean functions with different manner. For a given $t \in E_2$, the W_{HT} of a f (Boolean functions) can be represented by $W_f(t)$ and defined by

$$(t) = \sum_{x \in E^n} (-1)^{f(x)+t.x}.$$

(d) Autocorrelation: The autocorrelation of a Boolean function represents an indication of the imbalance of all first order derivatives of a Boolean function and provides a measure of self similarity for Boolean function. The derivative of Boolean function f(x), taken with respect to a vector t, where x and t belongs to, can be defined as:

$$A_c(f) = \sum (-1)^{\{f(x)+f(x+h)\}},$$

A Boolean function f is considered to be good if $A_c(f)$ is small .

III. An Evolutionary Approach To Classify Boolean Functions

In this approach, we have described the developed method for classification. Our developed method can be explained into two parts (i) Formulation of objective function and (ii) Applications (Nondominating Sorting Genetic Algorithms-II)[4].

(a) Formulation of Objective function : We have used criteria of autocorrelation to classify the Boolean functions. We know that for n-variables Boolean function the autocorrelation can be given by

$$A_C(f) = \sum (-1)^{\{f(x)+f(x+h)\}},$$

and highest value of 2^n. As there are total 2^{2^n} Boolean functions for n-variable, hence these functions attain autocorrelation between zero to 2^n. So, we classified set of all Boolean functions based on their autocorrelation and found interesting results. Firstly, we formulated an objective function, and then, applied NSGA-II on this with proper parameter. Obtained results are listed in Table 1 and Table 2.

As per def (2.5), value of autocorrelation can be given by the following formula:

$$A_C(f) = \sum (-1)^{\{f(x)+f(x+h)\}},$$

We assign f_1 equal to autocorrelation and take as our objective function, As for cryptographically secure function, we need minimum value of autocorrelation so we seek to find minimum value of $A_c(f)$. Hence ourobjective functions is :

$$f_1 = (f),$$

So,

$$\min f_1 = \min \text{ of } (\sum (-1)^{\{f(x)+f(x+h)\}}),$$

To optimize the above objective function, we have applied optimization Technique (NSGAII).

(b) Application of optimization technique: After forming the objective function ($_1$), we have applied optimization technique at f_1. With help of proper parameters we have found some desired results and these results are shown in section 5. The proper parameters that we have taken for our optimization technique are listed in Tables 3 and Table 4. For 1- variable there are total 4 Boolean functions. When we have applied NSGA-II on our objective function, we got -2 value of autocorrelation for two Boolean functions and +2 value of autocorrelation for other two Boolean functions, hence we have placed all functions in two classes. So, there is only two classes for one variable functions. We have observed few similarities among the functions of a class, such as all odd weight functions belonged to one class whereas even weight functions belonged to another class. Another observation is that both classes have same number of Boolean functions. When we applied NSGA-II on objective function for 2-variables Boolean functions, we got two values of autocorrelation: zero and four. So, we placed all Boolean functions in two classes. Boolean functions belonging to a single class, have same autocorrelation. We have observed few similarities among the functions of a class, such as all odd weight functions belonged to one class whereas even weight functions belonged to another class. Another observation is that both classes have same number of Boolean functions. Similarly, we can classify Boolean functions for higher variables as well. Main feature of this method is that, corresponding to a value of autocorrelation, we found all possible Boolean functions in a single run with help of proper parameters. In Table 8, 9 and Table 10, we have shown sub classification of classes for all two variables.

The inferences drawn from the above classification method are as follows:

(i) Functions belong to a class have same autocorrelation.
(ii) No. of classes are very less.
(iii) Cardinality of each class is not necessarily same.
(iv) Functions belong to same class have similar pattern of bits.

Table 1: Obtained Results

Autocorrelation	Boolean function (class) Results by NSGA-II
-2	01, 10
+2	00, 11

Table 2: Obtained Results

Autocorrelation	Boolean function (class) Results by NSGA-II
0	0001, 0111, 1110, 1000, 0100, 0010, 1011, 1101
4	0000, 0011, 1100, 1010, 0110, 0101, 1111,1001

Table 3: Parameters (for 1-variables)

Count of generations	100
Count of population	16
probability1(of C_R)	0.777
probability2(of M_T)	0.1181
SRN*	0.9999
Total number of bits (for each variable)	1
Total objective functions	1
Constraints	0

SRN* - Seed Random Number

Table 4: Parameters (for 2-variables)

Count of generations	500
Count of population	256
probability1(of C_R)	0.776
probability2(of M_T)	0.1182
SRN*	0.9998
Total number of bits (for each variable)	1
Total objective functions	1
Constraints	0

IV. A Recursive Approach To Classify Boolean Functions

(a) In this method, we classify n-variable Boolean Function based on Logic gates. By applying Logic gates, we have found different patterns and for n –variables these patterns have (n-1) digits.

We have started with one variable functions. There are total four functions for 1 – variable. We have applied **OR** Logic Gates between its bits. For two Boolean functions (out of all four Boolean functions), When we applied OR gate on its bits, we got value 1 and for remaining two Boolean functions we got 0.So, We have putted them in two different classes (Given in Table 5).

Table 5: Different Classes

INPUT		OUTPUT
A	B	A **OR** B(A+B)
0	1	1
0	0	0
1	1	1
1	0	1

Explanation: Lets take one Boolean function for one variable say **00.** After Appling **OR GATE** on its bits (0+0 =0(modulo2)) and we got **0**. So, we have putted this Boolean function in first class. Similarly, we have applied OR gate on others Boolean functions and analysed the result. We got only two values 1 and 0. So, we have classified all 1- variables Boolean functions into two classes and each class has two Boolean functions(cardinality is equal for both classes).

(b) Now applying same OR GATE, we have classified all the Boolean functions for 2- variables. There aretotal 16 Boolean functions for 2-variables. We took all Boolean functions and applied OR Logic gate on all Boolean functions and got 4 different patterns. Based on thes e patterns we have putted them in 4 different classes. Boolean functions with same pattern (after applying OR Gate) belong to same class. So, based on this method we got total 4 classes and each class have equal number of Boolean functions.

Explanation: We have taken any one Boolean function of two variable say **1001** and applied **OR GATE** on its bits (two consecutive bits at a time). After applying OR GATES, we found pattern 11(1+0(first two bits), 0+1(last two bits)). After applying this process on all Boolean functions of 2- variables, we got total4 different patterns (11, 10, 01,00). So, we have constructed total 4 classes and each class have four Booleanfunction (equal number of Boolean functions). All classes are listed in Table 6.

(c) Based on same process, wehave classified all Boolean functions for 3 – variables. There are total 256 Boolean functions and after applying the above process, we got total 16 classes and each class has 16 Boolean functions. All the classesfor three variables are listed in Table7.

Explaination:

Consider a 3-variable Boolean Function be **01010011** on this function we apply four **OR** logic gates. First **OR** gate has been applied on first two digit position of Function. Second OR gate has been applied on 3rd and 4th position digits. Third OR gate has been applied on 5th and 6th position digits. Fourth OR gateis applied on 7th and 8th position digits. After applying these gates on Boolean function. we got pattern **1100**(0+1, 0+1, 0+0, 1+1). Hence we have putted this Boolean function under this pattern class.

(d) Similarly, we applied this procedure on all 256 Boolean Functions.

We have observed few similarities among the functions of a class, such as all odd weight functions belonged to one class whereas even weight functions belonged to another class. Another observation is that all classes have same number of Boolean functions. Similarly, we can classify Boolean functions for highervariables as well.

The inferences drawn from the above classification method are as follows:

(i) No. of classes are very less.
(ii) Cardinality of each class is same.
(iii) Functions belong to same class have similar pattern of bits.
(iv) As Boolean function and its complement have same properties So, both functions belong to same class.

Table 6: Obtained Results

Pattern	Boolean Functions	Number of Boolean Functions
0	00,11	2
1	01,10	2

Table 7: Obtained Results

Pattern	Boolean Functions	Number of Boolean Functions
00	0000, 1111, 0011, 1100	4
01	0010, 1101, 0001, 1110	4
10	1011, 0100, 1000, 0111	4
11	1010, 0101, 1001, 0110	4

Table 8. Obtained Results

Pattern	Boolean Functions	Number of Boolean Functions
0000	00110011, 00111100…..,…..	16
0010	00110111, 00111000,…..	16
0001	00110001, 00111110,…..	16
0011	00111010, 00110101,….	16
1010	01110111, 01111000,….	16
1011	01111010, 01110101,…..	16
1001	01110001, 01111110,….	16
1000	01110000, 01111111,…..	16
0101	00010001, 00011110,…	16
0110	00011011, 00010100,…	16
0100	00010000, 00011111,….	16
0111	00011010, 00010101,….	16
1100	10100000, 10101111,….	16
1111	10101010, 10100101,…..	16
1101	10101101, 10100010,….	16
1110	10101011, 10100100,….	16

Table 9. Sub classification of Class-I of 1-variable functions

No. of Boolean function	Hamming Distance (from the base Boolean function 00000000)	weight
1	0	0
1	2	2

Table 10: Sub classification of Class-I of 2-variable functions

No. of Boolean function	hamming Distance (from the base Boolean function 00000000)	weight
4	1	1
4	3	3

Table 11. Sub classification of Class-II of 2-variable functions

No. of Boolean function	hamming Distance (from the baseBoolean function 00000000)	weight
1	0	0
6	2	2
1	4	4

V. Results And Discussion

In Table 12 and Table 13 all Boolean Functions are shown.

After applying the method developed in Section 3 and in Section 4, we classified the Boolean functions. These functions belong to different classes, and same class functions have symmetry amongst them. The parameters taken to classify functions (by NSGA-II) for 1 and 2-variables are listed in Table 3, 4 respectively. Our methods are new and with less complexity as compared to [8,4,7,3].

VI. Conclusion

In this piece of work, we have partitioned the set of n-variables Boolean functions into equivalence classes of same size. Main characteristic of our first method is that, we need only one member of previous class to define whole class. Classification have been done by two different approaches. Our methods are efficient and less complicated. In evolutionary approach method, we have found all possible results in single run (for each n-variables) only. This is the main characteristic of this method. Acknowledgement The authors are thankful to the Amity University, Noida.

Table 12: Obtained Results

0000	**0010**	**0001**	**0011**	**1010**	**1011**	**1001**	**1000**
00110011	00110111	00110001	00111010	01110111	01111010	01110001	01110000
00111100	00111000	00111110	00110101	01111000	01110101	01111110	01111111
00110000	00111011	00111101	00110110	01111011	01110110	01111101	01110011
00111111	00110100	00110010	00111001	01110100	01111001	01110010	01111100
11001100	11000111	11000001	11001010	10001000	10001010	10000001	10000000
11000000	11001000	11001110	11000101	10001011	10000101	10001110	10001111
11001111	11001011	11001101	11000110	10000100	10000110	10001101	10000011
11000011	11000100	11000010	11001001	10000111	10001001	10000010	10001100
00000000	00000111	00000001	00001010	10111011	10110110	10111101	10110000
00000011	00001011	00001110	00000101	10110100	10111001	10110010	10111111
00001100	00000100	00001101	00000110	10110111	10111010	10110001	10110011
11111111	11110111	00000010	00001001	10111000	10110101	10111110	10111100
11110011	11111000	11110001	11111010	01000100	01000110	01001101	01000000
11111100	11111011	1111110	11110101	01000111	01001001	01000010	01001111
11110000	11110100	11111101	11110110	01001000	01001010	01000001	01000011
00001111	00001000	11110010	11111001	01001011	01000101	01001110	01001100
16	16	16	16	16	16	16	16

Table 13: Obtained Results

0101	**0110**	**0100**	**0111**	**1100**	**1111**	**1101**	**1110**
00010001	00011011	00010000	00011010	10100000	10101010	10101101	10101011
00011110	00010100	00011111	00010101	10101111	10100101	10100010	10100100
00011101	00010111	00010011	00010110	10100011	10100110	10100001	10100111
00010010	00011000	00011100	00011001	10101100	10101001	10101110	10101000
11101110	11101011	11100000	11101010	01010000	01010101	01011101	01011011
11101101	11100100	11101111	11100101	01011111	01010110	01010010	01010100
11100010	11100111	11100011	11100110	01010011	01011001	01010001	01010111
11100001	11101000	11101100	11101001	01011100	01011010	01011110	01011000
11011101	11010111	11010000	11011010	01100000	01100110	01101101	01100111
11010010	11011000	11011111	11010101	01101111	01101001	01100010	01101000
11010001	11011011	11010011	11010101	01100011	01101010	01100001	01101011
11011110	11010100	11011100	11011001	01101100	01100101	01101110	01100100
00100010	00100111	00100000	00101010	10010000	10011001	10011101	10010111
00100001	00101000	00101111	00100101	10011111	10011010	10010010	10011000
00101110	00101011	00100011	00100110	10010011	10010101	10010001	10011011
00101101	00100100	00101100	00101001	10011100	10010110	10011110	10010100
16	16	16	16	16	16	16	16

REFERENCES

[1] Rathod R. Diagnosis and Remedial Teaching in Mathematics-A Review. Anusandhan, Vol 9, Issue 17, (2019).

[2] Braeken A, Borissov Y, Nikova S, and Preneel B. Classification of Boolean Functions of 6 Variablesor Less with Respect to Cryptographic Properties. Sixth International Scientific Conference FMNS2015, At Blagoevgrad, Volume: 1 (2015).

[3] Das JK, Choudhury PP. Classification of n-Variable Boolean Functions through Hamming Distance and their Application in System Biology. International Journal of Computer Applications (IJCA), (0975 8887), NCWBCB (2014).

[4] Deb K, Pratap A, Agarwal S, Meyarivan T. A Fast and Elitist Multiobjective Genetic Algorithm, IEEE Trans. on Evolutionary Computation, 6(2), 182-197, (2002).

[5] Golomb SW. On the classification of Boolean functions. IRE Transactions on Circuit Theory, vol. 6, no. 5, pp. 176186 (1959).

[6] Harrison MA. On the classification of Boolean functions by the general linear and affine groups. Journal of the Society for Industrial and Applied Mathematics, Vol. 12, N0. 2, pp. 285-299.

[7] Correia VP, Reis AI. Classifying n-input Boolean functions, in Proceedings of the 7th Workshop IBERCHIP (IWS '01), pp. 5866, Montevideo, Uruguay, (2001).

[8] Rout RK, Chaudhary PP, Sahoo S. Classification of Boolean Functions where affine functions are Uniformly Distributed. Hindawi Publishing Corporation, Journal of Discrete Mathematics, Vol. 2013, Artical ID 270424.

[9] Slepian D. On the number of symmentric types of Boolean functions of n variables. Society for industrial and Mathematics, Vol. 05, Issue 02, pp. 185-193.

[10] Shtrakov S, Damyanov I. On the Classification of Boolean Functions. Proceedings of the Sixth International Scientific Conference FMNS2015, Vol. 1, pp. 124-130, (2015).