Genetic Algorithm for Bent Functions Generating

Milan Stojanović and Suzana Stojković

Abstract— **The importance of unique bent functions (most significantly in cryptography) creates a demand for their generation. Bent function generation is an interesting problem and, in this paper, we explore the idea of using invariant spectral operations in a Genetic algorithm for generating bent functions. Invariant spectral operations, when executed on bent function, resulting function is also bent. If multiple operations are performed consecutively, then there is a possibility that the newly generated bent function is not unique. A genetic algorithm is used to search the solution space in order to produce the most unique bent functions, for the least number of invariant spectral operations.**

Index Terms— **Bent functions, invariant spectral operations, genetic algorithm.**

I. Introduction

Bent functions are Boolean functions most distant from affine functions. They were introduced by O.S. Rothaus in 1976. [1], and they have characteristics that are interesting for cryptographic applications. There are many algorithms for the generation of the bent function, see for example [2-9] and references therein.

A very important characteristic of the bent functions is flat Walsh spectra. All Walsh spectral coefficients of n-variable bent functions have the same absolute value equal to $2^{n/2}$. Invariant spectral operations are operations that do not change the absolute values of spectral coefficients, i.e., they only permute or change the sign of spectral coefficients. It follows that new bent functions can be generated from any known bent function by applying invariant spectral operations. References [7-9] elaborate methods for bent functions generation by using invariant spectral operations. The main disadvantage of those methods is that the same bent function can be generated by applying different sequences of operations.

Genetic algorithm is inspired by natural selection, that belongs to the evolutionary algorithm group. This algorithm is used to optimize a solution for a corresponding problem. It can be used most effectively when the search space is vast, but the solution does not need to be perfect, only optimal to some degree.

Milan Stojanović is master's-degree student in Computer Science at Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: milances.14@gmail.com)

Suzana Stojković is with Dept. of Computer Science, Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14 18000 Niš, Serbia (e-mail: suzana.stojkovic@elfak.ni.ac.rs).

This paper proposes the usage of a Genetic algorithm for the generation of bent functions. Bent functions belong to the vast space of Boolean functions. Therefore, the search for unique bent functions can be presented as executing a sequence of invariant spectral operations, and optimization is used in the sequence of operations, so that we will produce as many different bent functions as possible.

The paper is organized in the following way: Section II presents the ANF representation of bent function. Section III covers Invariant spectral operations. Oscar-Bent functions are presented in Section IV and the Genetic algorithm is defined in Section V. Section VI explains the problem definition and usage of the Genetic algorithm for the generation of bent functions. Section VII goes over the results, and Section VIII gives a conclusion.

II. ANF Representation of Bent Functions

A. Definition

An n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$ can be presented by the algebraic normal form (ANF), or the positive polarity Reed-Muller expansion as:

$$f(x_1, \ldots, x_n) = \sum_{i=0}^{2^n-1} S(i) \prod_{k=0}^{n-1} x_k^{i_k}$$

where $S(i)$ is the Reed-Muller spectral coefficient and $i_0 i_1 \ldots i_{n-1}$ is the binary representation of the index i.

Reed-Muller spectral coefficients of bent functions are equal to 0 for each input vector with the number of ones greater than $n/2$. The maximal number of variables in a product term is called the degree of f [8].

B. Disjoint quadratic function

The disjoint quadratic function contains $n/2$ disjoint quadratic terms, defined as:

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

III. Invariant Spectral Operations

A. Definition

Invariant spectral operations do not change the absolute values of Walsh spectral coefficients, they only permute or change the sign of spectral coefficients. These changes preserve the flat spectrum.

Due to the simplicity of invariant spectral operations in the Reed-Muller domain, all operations are introduced in this domain. For consistency, all examples will be provided starting from the Disjoint quadratic function for $n = 6$.

B. *Function complement*

Function complement is defined as:

$$f_2 = \overline{f_1} = f_1 \oplus 1$$

For example, if

$$f_1(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6$$

The resulting function will be:

$$f_2(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus 1$$

C. *Variable complement*

Variable complement replaces the input variable i by its complement $x_i' = x_i \oplus 1$.

If variable complement on variable x_4 is performed, the function f_1 is transformed to:

$$\begin{aligned} f_2(x_1, x_2, x_3, x_4, x_5, x_6) &= f_1(x_1, x_2, x_3, \overline{x_4}, x_5, x_6) \\ &= x_1x_2 \oplus x_3(x_4 \oplus 1) \oplus x_5x_6 \\ &= x_1x_2 \oplus x_3x_4 \oplus x_3 \oplus x_5x_6 \end{aligned}$$

D. *Disjoint spectral translation*

Disjoint spectral translation replaces the input variable i by $x_i' = x_i \oplus x_j$, where $i \neq j$.

In the given example, if x_3 is replaced by $x_3 \oplus x_6$, following function is generated:

$$\begin{aligned} f_2(x_1, x_2, x_3, x_4, x_5, x_6) &= f_1(x_1, x_2, x_3 \oplus x_6, x_4, x_5, x_6) \\ &= x_1x_2 \oplus (x_3 \oplus x_6)x_4 \oplus x_5x_6 \\ &= x_1x_2 \oplus x_3x_4 \oplus x_4x_6 \oplus x_5x_6 \end{aligned}$$

E. *Spectral translation*

In the general case, we can define spectral translation as adding linear member x_i to the function:

$$f_2 = f_1 \oplus x_i$$

If in our example x_2 is added, resulting function is:

$$f_2(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_2$$

F. *Permutation of variables*

Permutation of variables is defined as the interchange of two input variables $x_i \leftrightarrow x_j$, where $i \neq j$.

$$f_2(x_1, \dots, x_i, \dots, x_j, \dots, x_n) = f_1(x_1, \dots, x_j, \dots, x_i, \dots, x_n)$$

In the given example if we interchange input variables x_3 and x_6 the resulting function is:

$$\begin{aligned} f_2(x_1, x_2, x_3, x_4, x_5, x_6) &= f_1(x_1, x_2, x_6, x_4, x_5, x_3) \\ f_2(x_1, x_2, x_3, x_4, x_5, x_6) &= x_1x_2 \oplus x_6x_4 \oplus x_5x_3 \end{aligned}$$

G. *Generalized spectral translation*

The generalized spectral translation is defined for the function f which has n variables $(n = 2 * k, k \geq 3)$ and contains $n/2$ disjoint quadratic terms:

$$f(x_1, \dots, x_n) = \cdots x_{i_1}x_{j_1} \oplus x_{i_2}x_{j_2} \oplus \dots \oplus x_{i_{n/2}}x_{j_{n/2}}$$

Performing generalized spectral translation on function f adds a new term $x_{k_1}x_{k_2} \dots x_{k_{n/2}}$ where

$$k_1 \in \{i_1, j_1\},\ k_2 \in \{i_2, j_2\}, \dots, k_{n/2} \in \{i_{n/2}, j_{n/2}\}.$$

If the starting function is f_1 is and if $k_1 = 1$, $k_2 = 3$, and $k_3 = 6$, resulting function f_2 is:

$$f_2(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_1x_3x_6$$

IV. OSCAR-BENT FUNCTIONS

The bent function which does not have linear and constant members can be called Oscar-Bent function (the name derives from Oscar Rothaus, who first defined bent functions). For the bent function defined in (1), we can derive the Oscar-Bent function shown in (2) by using invariant spectral operations.

$$f_1(x_1, \dots, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \oplus x_1 \oplus 1 \qquad (1)$$

$$f_2(x_1, \dots, x_6) = x_1x_2 \oplus x_3x_4 \oplus x_5x_6 \qquad (2)$$

To transform a bent function to its Oscar-Bent function we need to remove linear and constant members, which is done by using two invariant spectral operations: function complement and spectral translation. By counting only Oscar-Bent functions, we can deduce that the number of unique bent functions found with this algorithm is calculated by multiplying the number of Oscar-Bent functions with 2^{n+1}. The multiplier is found by calculating all possible combinations using two invariant operations mentioned above.

V. GENETIC ALGORITHM

Genetic algorithm is a subclass of Evolutionary algorithm (EA), which is a subclass of Evolutionary computation and belongs to set of general stochastic search algorithm [10].

Population in both Genetic algorithms and in nature represents the set of individuals who are trying to survive and pass on their genes to the next generations. An individual can be interpreted as a set of genes and abilities, and how fit they are to survive in the current population and habitat. If we observe an individual as a solution to a problem, as well as in nature, optimization (survival of the fittest) will transpire, in the end, the fittest individual will represent an optimized solution to the problem. Given a population of individuals within some environment that has limited resources, competition for those resources causes natural selection (survival of the fittest). This in turn causes a rise in the fitness of the population. Given a quality function to be maximized, we can randomly create a set of candidate solutions, i.e., elements of the function's domain. We then apply the quality function to these as an abstract fitness measure – the higher the better. Based on these fitness values some of the better candidates are chosen to seed the next generation. This is done by applying recombination and/or mutation to them. Recombination is an operator that is applied to two or more selected candidates (the so-called parents), producing one or more new candidates (the children). The mutation is applied to one candidate and results in one new candidate. Therefore, executing the operations of recombination and mutation on the parents leads to the creation of a set of new candidates (the offspring). These have their fitness evaluated and then compete – based on their fitness (and possibly age) – with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached [11].

The genetic algorithm can be described by the pseudo-code in Fig. 1.

```
InitializePopulation();
EvaluatePopulation();
while i < MaxIteration and
BestFitness < MaxFitness do
        Fitness = FitnessCalculation();
        Selection();
        ParentSelection();
        Reproduction();
        i++;
        BestFitness = Max(Fitness);
end while
return BestFitness
```

Fig. 1. Pseudo code detailing the genetic algorithm

A. Parent selection

Parent selection represents a strategy of selecting good parents to get a better next generation. The strategy should consist of some random chance in selection, so diverse parents will be used, and we can diverge from the local maximum (which can be reached by using the same group of parents).

- There are different strategies, for this paper, we have used:
- Roulette selection – odds of selection are determined by individual fitness and a corresponding piece of the roulette wheel is given; a random number is generated to represent a ball spin.
- Rang selection – is like Roulette selection, but fitness is scaled to give more chances to weaker individuals.
- Tournament selection – from a randomly selected group of individuals the best individual is chosen based on fitness.

B. Recombination and mutation

Recombination and mutation are used to produce a new solution to find the best one which solves the problem. Both methods may and may not be performed (based on chance which is determined on startup).

There are several recombination methods, but the most common is a crossover with one crossover point which is randomly selected. Genes from the first parent are copied to the crossover point, after which genes from the second parent are copied.

Mutation, if performed, results in randomly changing individual genes. For each gene, independently, it is determined whether the mutation will be performed or not.

C. Adult selection

Adult selection defines how will the new generation join the existing group of adults. Since the "habitat" can only sustain an already defined number of individuals, adult selection is needed to determine who will survive. Several methods are implemented:

- Full generational replacement – as the name applies, the parental generation is replaced with the new generation.
- Generational mixing – both generations are mixed, and the best of mixed generations survives.
- Overproduction – this method is a mixture between full generational replacement and generational mixing, in which the new generation has twice as many individuals as the parental generation, and only a half of the best child individuals survive to form the new parental generation.

Elitism can also be used, elitism enables keeping the best solution for the next solution, regardless of chosen adult selection.

VI. Genetic Algorithm For Bent Functions Generating

The problem can be defined by finding the most bent function by using the least number of invariant spectral operations. Since invariant spectral operations are performed on a bent function, a starting bent function needs to be defined. In our case, we start from the disjoint quadratic function.

For each implementation of a genetic algorithm, it is crucial to define an individual (which represents a solution to a problem) and a fitness function (which represents how good the solution is).

A. Individual representation

An individual is represented as a sequence of invariant operations which are performed on the most recent bent function.

B. Fitness function

It is recommended that the fitness function should be defined so it would have a minimum (the worst solution) and the maximum (the best solution), even though the boundaries can be arbitrary, the custom is to choose boundaries as 0 and 1.

To determine how good is the solution, we need to go back to the problem definition which states that we should find the most unique bent functions for the least number of invariant spectral operations. From this, we can derive that the fitness function can be calculated as the number of unique Oscar-Bent functions divided by the number of used invariant spectral operations.

By searching for the unique Oscar-Bent functions, we can generate the most bent functions, given that from the one Oscar-Bent function we can derive 2^{n+1} bent functions.

VII. Experimental Results

The application was developed in C#, and tests were performed on the laptop with the following configuration:

- CPU: Intel® Core™ i5-8250U CPU @ 1.6GHz
- RAM: 16 GB
- OS: 64bit Windows 10

Multiple parameters can be changed, and which can influence results (both performance and result wise). Testing all permutations of the possible combination of parameters is not a trivial task, and it is time-consuming. Therefore, some parameters were hardcoded with values that we perceived as best with our experience and using educated guesses.

Parameters that were hardcoded for all tests:

- Adult selection – Generational mixing

- Parent selection – Tournament selection
    - Tournament size – 20% of the population
- Possibility of gene mutation – 10%
- Possibility of recombination – 90%

A. Test 1 – Different number of genes

In this test we have chosen the number of variables to equal 6, population size is set to 10, and the number of generations is limited to 100. In this test, we will change the number of genes and compare the number of unique Oscar-Bent functions in an average of 5 runs. Results are shown in Table I.

TABLE I.
RESULTS OF TEST 1

Number of genes	**Number of generated OBF**	**Time (s)**
100	97.8	0.066
1 000	948.4	0.454
10 000	9 364.4	8.31
100 000	92 058	82.724

Through a different number of genes, we have seen that with linear growth of the number of genes, the number of unique Oscar-Bent functions grows in a linear fashion, with the growth factor between 9.5 and 10. When we analyze the time needed, it grows exponentially which is expected since the solution space grows exponentially as well.

B. Test 2 – Different number of variables

As in the previous test, the population size is set to 100, the number of generations is limited to 100 and the number of genes to 10 000. Here we will fluctuate number of variables. Results are shown in Table II.

TABLE II.
RESULTS OF TEST 2

n	**Number of generated OBF**	***Time (s)***
8	9 652.6	16
10	9 786.8	35.316
12	9 834.4	111.346
14	9 881.2	413.6

While an increasing number of variables we can observe that the number of unique Oscar-Bent functions increase with low percentages. Factor of growth for the time needed increases with each step, but it does not increase exponentially.

C. Test 3 – Application limits

In this test, we emphasized the performance limits of the application, not to the numbers we have, therefore we have run this test only once. Here, we have kept the number of genes to 10 000 and the number of generations to 100, as in the last test. But we have changed population size to 100. The results are shown in Table III.

TABLE III.
RESULTS OF TEST 3

n	**Number of generated OBF**	**Time (s)**
8	9 673	169.18
10	9 768	405.08s
12	9 845	1318.69s
14	9 877	1801.56s
16	N/A	N/A

VIII. CONCLUSION

We have seen that the usage of Genetic algorithm can be used in the generation of new bent functions. The performance of this approach indicates that future work can give promising results.

Memory is the biggest obstacle when working with bent functions. This problem can be approached by tracking only Oscar-Bent functions, which is performed in this paper. Further, all functions are kept in memory, which is a problem when expecting many unique Oscar-Bent functions, which we have seen in test 3. Future work will address this problem.

REFERENCES

[1] O. S. Rothaus, “On “Bent” Functions”, Journal of Combinatorial Theory, Series A, Vol. 20. No. 3, pp. 300-305, 1976.

[2] H. Dobbertin, “Construction of bent functions and balanced Boolean functions with high nonlinearity”, LCNS, vol. 1008, pp. 61-74, Springer, Berlin, Germany, 1995.

[3] J. Climent, F. Garcia, V. Requena, “On the iterative construction of bent functions”, in Proc. of the 5th WSEAS Int. Conf. on Inf. Security and Privacy (ISP06), pp. 15–18 World Scientific and Engineering Academy and Society (WSEAS), Wisconsin, USA, 2006

[4] J. Climent, F. Garca, V. Requena, “On the construction of bent functions of n+2 variables from bent functions of n variables”, Advances in Mathematics of Communications, vol. 2, pp. 421–431, 2008.

[5] C. Carlet, “A larger class of cryptographic Boolean functions via a study of the Maiorana McFarland construction”, LNCS, vol. 2442. pp. 549-564, Springer, 2002.

[6] H. Dobbertin, G. Leander, A. Canteaut, C. Carlet, P. Felke, P. Gaborit, “Construction of bent functions via niho power functions”, Journal of Combinatorial Theory, vol. 113, no. 5, pp. 779–798, 2006.

[7] M. Stanković, C. Moraga, R. Stanković, “Some Invariant spectral Operations for Functions with Disjoint Products in the Polynomial Form”, in Proc. EUROCAST 2017, LNCS, Vol. 10672, pp. 262-269, Springer, 2017.

[8] S. Stojković, M. Stanković, C. Moraga, R. Stanković, “Generation of Binary Bent Functions by Walsh Invariant spectral Operations Performed in Reed-Muller Domain”, Proceedings of 13th International Workshop on Boolean Problems, Bremen, Germany, pp. 255-266, September 19-21. 2018.

[9] R. S. Stanković, M. Stanković, C. Moraga and J. T. Astola, "Construction of Ternary Bent Functions by FFT-like Permutation Algorithms", 2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL), pp. 88-93, 2020.

[10] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, India, pp. 261-265, 22-24 Dec. 2016.

[11] A.E. Eiben, J.E. Smith, “Introduction to Evolutionary Computing”, 2nd ed., Berlin, Germany: Springer, 2015.