

4040-849 OPTIMIZATION METHODS

OPTIMIZING CRYPTOGRAPHIC STRENGTH OF SUBSTITUTION LAYERS IN SYMMETRIC-KEY CRYPTOGRAPHIC ALGORITHMS

Christopher Wood

May 14, 2012

Abstract

The cryptographic security of symmetric-key encryption algorithms is commonly based Shannon's principles of confusion and diffusion [1]. Confusion can be defined as the statistical relationship between the ciphertext and private key of a cipher, while diffusion refers to the statistical redundancy of plaintext bits in the ciphertext bits. Consequently, it is increasingly important to optimize these characteristics in order to make them less susceptible to common attacks, such as linear and differential cryptanalysis. S(ubstitution)-boxes are the most traditional mathematical structures that are used to improve the levels of diffusion and confusion within symmetric-key cryptographic algorithms. Recent research efforts have revealed practical measurements of S-box constructions that indicate their susceptibility to linear and differential cryptanalysis [1]. In this work, we discuss the formulation of building cryptographically strong substitution layers in symmetric-key block algorithms with S-box designs into a mixed integer nonlinear programming (MINLP) problem that can be optimized to yield the high diffusion and confusion dividends in resulting algorithm definitions.

1 Problem Description

The cryptographic security of symmetric-key encryption algorithms is based upon Shannon's principles of confusion and diffusion [1]. Confusion can be defined as the complexity of the relationship between the secret-key and ciphertext, and diffusion is commonly referred to as the degree to which the influence of a single input plaintext bit is spread throughout the resulting ciphertext. When designing new cryptographic algorithms, it is increasingly important to optimize these characteristics in order to make them less susceptible to common attacks, such as linear and differential cryptanalysis.

Linear cryptanalysis is a popular attack technique that attempts to approximate the secret key used within a symmetric-key encryption algorithm using a set of known plaintexts. Mathematically, it is defined as the formulation of a linear expression that is composed of the input and output bits for a function within the algorithm (or the entire algorithm itself) that is satisfied with high probability (which is an indication of bit patterns generated by the target function). This expression is then cross-checked with a large set of known plaintexts in order to deduce the secret encryption key [2].

Differential cryptanalysis is another powerful attack technique that attempts to break symmetric key ciphers by exploiting high probability of certain occurrences of plaintext differences and ciphertext differences [2]. By increasing the level of nonlinearity of a cipher, the probability of the occurrences of these differences decreases significantly, thus thwarting attacks that use this information to uncover the secret encryption key.

Collectively, linear and differential cryptanalysis lose their effectiveness as the levels of confusion and diffusion within a symmetric-key encryption algorithm are increased. Therefore, it is a

fundamental design goal of such algorithms to incorporate mathematical elements that exhibit high measures of confusion and diffusion when composed together. Combinations of linear and nonlinear operations are the most common means by which these characteristics are realized in symmetric-key encryption algorithms, with the S(ubstitution)-box being the most popular nonlinear component available to cryptographic designers. An S-box is a well-defined function that maps elements in the domain \mathbb{F}_2^n to elements in the range \mathbb{F}_2^n . With this notion, we consider a configuration of an S-box to be a set of input and output pairs. Furthermore, we denote the canonical form of an S-box configuration as an integer sequence of length 2^n ordered by the input elements that contains the output elements in \mathbb{F}_2^n .

The mathematical design and cryptographic strength of S-boxes, which are commonly referred to as the substitution layers within symmetric-key encryption algorithms, has been the focus of extensive research in recent years. However, there is a lack of work in the literature that formulates the problem of S-box construction as a constrained optimization problem. Therefore, we address this lack of focus and attempt to state the S-box construction problem as a set of mixed-integer nonlinear programming problems with individual and multiple objective functions that can be solved to yield the optimum S-box configuration. The rest of this report is dedicated to the discussion of this effort.

1.1 Cryptographic Strength of Substitution Layers

Mathematically, an S-box can be represented as a function f that maps input values $a \in \mathbb{F}_2^n$ to output values $b \in \mathbb{F}_2^n$. In the context of symmetric-key cryptographic algorithms, such a function f should ideally be bijective in order to avoid bias towards any specific output element in the field.

We now present a series of definitions that are pertinent to the design of cryptographically strong S-Boxes [3].

Definition 1. The *Hamming weight* of an element $x \in \mathbb{F}_2^n$ is defined as $\text{wt}(x) = \sum x_i$, where x_i refers to the i th bit in x .

Definition 2. The *branch number* of an $n \times n$ -bit S-Box S is

$$B_N = \min_{a, b \neq a} [\text{wt}(a \oplus b) + \text{wt}(S(a) \oplus S(b))], \quad (1)$$

where $a, b \in \mathbb{F}_2^n$.

Definition 3. A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ exhibits the *avalanche effect* if and only if

$$\sum_{x \in \mathbb{F}_2^n} \text{wt}(f(x) \oplus f(x \oplus c_i^n)) = n2^{n-1}, \quad (2)$$

for all i ($1 \leq i \leq n$), where $c_i^n = \langle 0, 0, \dots, 1, \dots, 0 \rangle$ (where a 1 is in the n th position of the vector of cardinality n).

Definition 4. The *avalanche number* of a function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is defined as

$$A_N = \sum_{i=0}^{n-1} \sum_{x \in \mathbb{F}_2^n} \text{wt}(f(x) \oplus f(x \oplus 2^i)), \quad (3)$$

where $a, b \in \mathbb{F}_2^n$. This just sums of the weight for each bit i , where $1 \leq i \leq n$.

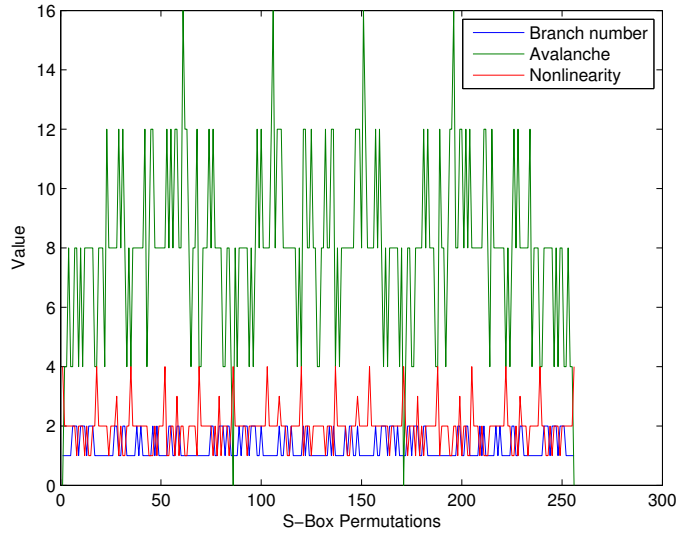


Figure 1: The results for the branch number, avalanche number, and degree of nonlinearity for a 2-bit S-box over all possible permutations in the design variables. Note that there are $4^4 = 256$ possible permutations over the S-box configuration design space because each element $x \in \mathbb{F}_2^2$ can map to any output value $y \in \mathbb{F}_2^2$.

Definition 5. The *degree of nonlinearity* of an $n \times n$ -bit S-Box from $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ can be measured by

$$P_S = \max_{0 \neq a, b} |\{x \in \mathbb{F}_2^n : S(x + a) - S(x) = b\}| \quad (4)$$

where $a, b \in \mathbb{F}_2^n$, and a low measure for P_S indicates a high degree of nonlinearity.

Cryptographers and mathematicians commonly use these measurements as a basis for the levels of confusion and diffusion within a symmetric-key encryption algorithm, which in turns correlates to an approximation of their susceptibility to linear and differential cryptanalysis (among other attacks). In particular, it has been shown that cryptographically secure symmetric-key algorithms utilize substitution layers that provide the following characteristics [1]:

1. High branch number
2. Avalanche number of exactly $n^2 2^{n-1}$
3. High degree of nonlinearity

2 Optimization Problem Formulation

The best way to find optimal S-box configurations is to perform an exhaustive search over the permutation design space (i.e. consider all configurations) and find one that yields the optimum results. The results from an exhaustive search experiment for a 2-bit S-box are shown in Figure 2. Exhaustive searches over the entire S-box permutation design space have a time complexity of $O(n^n)$ and thus are only feasible for very small order S-boxes. However, recent research efforts have

advanced the upper bound on these exhaustive searches to include 4-bit S-boxes, which were run in [4]. Unfortunately, measurements for the branch number, avalanche number, and nonlinearity degree were not provided.

As an alternative to performing these brute force searches, we consider the problem of optimizing the branch number, avalanche number, and degree of nonlinearity of symmetric-key cryptographic algorithm as MINLP problems, which can be formulated as shown in Figure 2.

3 Candidate Optimization Algorithms

Analyzing the function values for the branch number, avalanche number, and degree of nonlinearity over the S-box permutation design space indicates that each function is highly unstable and discontinuous. Therefore, traditional optimization methods that rely on the continuity of the functions are not likely to be very effective when solving these optimization problems. This section discusses some potential optimization algorithm candidates and describes why a genetic algorithm was chosen as the desired solution approach.

3.1 Traditional Optimization Algorithms

The *Branch and Bound* algorithm is a very common optimization algorithm that is used to solve MINLP problems. However, this algorithm is not suitable in the context of this problem for a variety of reasons, as shown below.

1. The branch and bound algorithm is greedy, in that it selects the appropriate decision branch to take to proceed towards the optimal value based on the current configuration. While this leads the algorithm towards an optimum value, it is usually a locally optimal solution. Furthermore, there is no penalty applied to the decision process that enables the design variables to change.
2. As shown in Figure 2, there are many locally optimum values for each objective function even with small order S-boxes. The usefulness of the BNB algorithm will surely degrade as larger order S-boxes are considered, because the number of intermediate local optimum values will increase significantly as well. This would cause the algorithm to converge to an optimal value too quickly and terminate with very few iterations.

3.2 Evolutionary Optimization Algorithms

Given the instability of each of these functions over the permutation design variable space, it is natural to solve these MINLP problem using a probabilistic optimization algorithm. For the purposes of this project, the main results were derived using a highly probabilistic genetic algorithm that attempts to explore the permutation design space as much as possible in search for the global optimum values. Genetic algorithms are commonly utilized in the context of cryptography to explore the design space for cryptographic functions. They serve as an intelligence optimization method that can ease the computational efforts of performing brute force searches among the design space, which is crucial for functions with larger domains [5].

Since each of the aforementioned objective functions exhibit the same instable behavior, it is possible to apply the same genetic algorithm structure to solve each one of these. In particular, the genetic algorithm configuration for each of these problems used the following parameters.

Branch Number - Minimize

$$B'_N(X) = -B_N(X) = -\min_{i,j \neq i} (\text{wt}(i \oplus j) + \text{wt}(X(i) \oplus X(j))),$$

subject to the constraints

$$0 \leq X(i) \leq 2^n - 1$$

where n is the number of bits needed to represent the design variables.

Avalanche Number - Minimize

$$A'_N(X) = -A_N(X) = -\sum_{i=0}^{n-1} \sum_{x \in \mathbb{F}_2^n} \text{wt}(f(x) \oplus f(x \oplus 2^i))$$

subject to the constraints

$$\begin{aligned} 0 \leq X(i) &\leq 2^n - 1 \\ \min_{i,j \neq i} (\text{wt}(i \oplus j) + \text{wt}(X(i) \oplus X(j))) - n2^{n-1} &\leq 0 \end{aligned}$$

where n is the number of bits needed to represent the design variables.

Degree of Nonlinearity - Minimize

$$P_S(X) = \max_{0 \neq a,b} |\{x \in \mathbb{F}_2^n : S(x+a) - S(x) = b\}|,$$

subject to the constraints

$$0 \leq X(i) \leq 2^n - 1,$$

where n is the number of bits needed to represent the design variables.

Figure 2: MINLP optimization problem definitions for the branch number, avalanche number, and nonlinearity degree measurements. It is important to note that the lack of additional constraints on these objective functions is intentional. Due to the nature of S-boxes and their application inside cryptographic algorithms, additional constraints might serve as additional information for attackers to exploit.

Table 1: Genetic algorithm configuration parameters.

Algorithm Option	Configuration Description
Population mutation function	Randomly scale each design variable based on the current generation
Generations	500
Tolerance Function limit	1×10^{-6} from last function value change
Stall generation limit	2^n identical function values
Initial population	S-box configuration $\langle 0, 1, \dots, n \rangle$

4 Genetic Algorithm Optimization Results

We now present the optimization results for the three objective functions listed in the Section 2 for 3- and 4-bit S-box configurations. Each objective function is first optimized individually and then they are joined together as a multi-objective MINLP problem.

Also, since the genetic algorithm solver is a probabilistic algorithm that randomizes the population at each generation, it is important to note that each algorithm was run many times to obtain the optimal results. Due to the highly instable objective functions, it is not enough to run the algorithm once and conclude that the results are optimum.

4.1 Avalanche Number

From Section 1.1 we know that the optimal value for the avalanche number of an n -bit S-box is $n^2 2^{n-1}$. Therefore, for 3- and 4-bit S-boxes, we conclude that the optimal values are $3^2(2^2) = 9(4) = 36$ and $4^2(2^3) = 16(8) = 128$, respectively, since we are concerned with the avalanche effect over every single bit. Now, we compare these optimal values against those obtained from the optimization algorithm. Table 4.1 contains the optimal results for a 3-bit S-box configuration that were collected over a variety of optimization runs, and Figure 4.1 contains a diagram of the best iteration pattern of the algorithm for a higher order 4-bit S-box configuration.

Based on these results we can see that the genetic algorithm was able to reach an optimal value of 30 for the 3-bit S-box, which is approximately 16% away from the global optimal value of 36. Similarly, the genetic algorithm was able to reach an optimal value of 96, which is approximately 25% away from the global optimal.

Clearly, we see that the accuracy of the genetic algorithm decreases as the order of the S-box gets higher. This is most likely due to the fact that the function for the avalanche number is very unstable and contains many local optimal values, and even with a probabilistic population generation procedure it is difficult to explore beyond one of these local optimal values. However, given the time complexity difference between the exhaustive search and the genetic algorithm solver, this may be a reasonable compromise in terms of security if a better configuration cannot be found manually.

4.2 Branch Number

It has been proven that the maximum branch number for an n -bit S-box is equal to n [6]. Therefore, in the case of 3- and 4-bit S-boxes, we know that the maximum possible branch numbers are 3 and 4, respectively. Given the small permutation design space for the 3-bit S-boxes, the genetic

Table 2: Some results from the genetic algorithm solver for the avalanche number of 3-bit S-boxes.

S-Box Configuration	Optimal Function Value	Generations
6 1 1 2 1 6 6 1	30	63
5 3 2 5 2 5 5 2	30	59
6 1 1 6 1 4 6 1	30	51
1 6 6 1 6 5 1 6	30	51
6 1 5 6 1 6 2 1	28	58
6 1 3 6 3 6 4 1	26	57
6 3 3 4 2 5 5 2	22	63
1 2 6 1 7 0 1 6	22	58
0 7 7 4 6 1 1 2	20	51
3 4 7 1 6 3 1 6	20	51

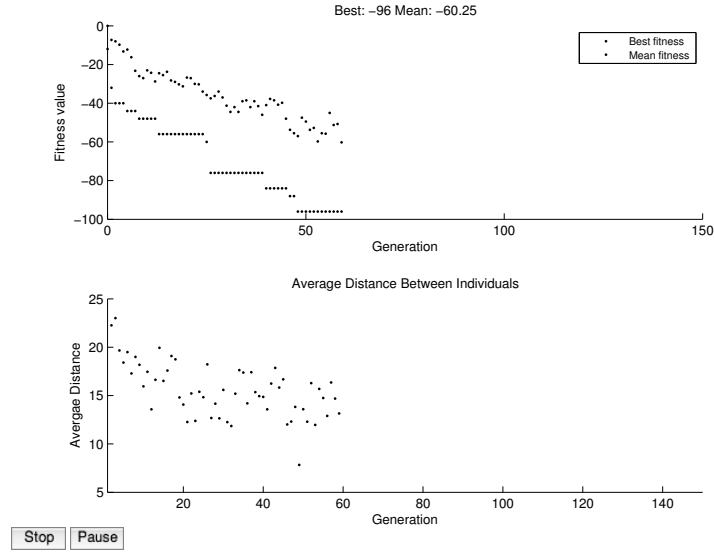


Figure 3: A single evolution of the optimal solution to the avalanche number of a 4-bit S-box configuration with a maximum of 500 generations. Note that the solver converged to a sub-optimal result after computing 60 generations, which is an indication that the solution reached a locally optimal solution.

Table 3: Some results from the genetic algorithm solver for the branch number of 3-bit S-boxes.

S-Box Configuration	Optimal Function Value	Generations
3 4 0 7 5 2 6 1	3	500
3 6 4 1 5 0 2 7	3	500
4 7 6 1 2 1 5 6	2	500
5 3 1 5 6 5 0 7	2	500
1 3 0 2 7 2 6 3	2	500
6 3 4 1 5 6 0 4	2	500
3 1 5 4 2 6 4 1	2	500
6 5 5 4 4 3 2 6	2	500
2 6 6 2 4 3 2 1	2	500
4 3 1 2 6 7 5 6	2	500

algorithm solver was able to obtain this optimal value in only a small number of instances. The majority of the executions converged to the local optimal value of 2, as shown in Table 4.2.

Extending the order of the S-box to 4 bits made the solver even less effective at finding the optimal value of 4. As shown in Figure 4.2, although each generation covered a large area of the state space (as indicated by the average bit-wise distance between parents and children), the optimal value still converged to the local optimum of 2 even after 500 generations. In fact, there was no execution of the solver that yielded a higher value for the branch number.

This is an obvious result due to the instability of the branch number function over the configuration permutation state space. Since there are so many local optimal values for the function that increases with the order of the S-box, the solver simply converges to the nearest optimal and then orbits around that value for each of the following generations. Based on these results we conclude that the usage of a genetic algorithm is not particularly helpful when optimizing the branch number of S-boxes.

4.3 Degree of Nonlinearity

It has been shown that the most optimal value for P_S is 1, meaning that the S-box is "perfectly nonlinear" in that it is resistant to any forms of differential cryptanalysis [7]. However, almost-perfect nonlinear (APN) S-boxes have also been studied extensively by Nydberg and are shown to be almost equally resistant to the most sophisticated differential cryptanalysis attacks, where $P_S \leq 2$ for aushc S-box configurations. Therefore, as shown in Table 4, the genetic optimization solver was able to find a variety of S-box configurations qualify them as APN, which is an indication that the S-box is secure against common attacks. Unfortunately, the solver was not able to find a configuration that yielded the perfectly nonlinear configuration value of $P_S = 1$.

Extending the order of the S-box to 16 elements and running the same test yielded similar APN results, as shown in Figure 4.3. The solver was able to converge to the optimal value of 2 for in a relatively small number of generations, which is a good indication that the instability of the function does not deter the algorithm as significantly as it did with the branch number function. Based on these results, we can assume that this pattern would likely continue into higher order S-boxes, and thus we conclude that the genetic algorithm is very well suited to finding the S-box configuration that achieves an acceptable degree of nonlinearity.

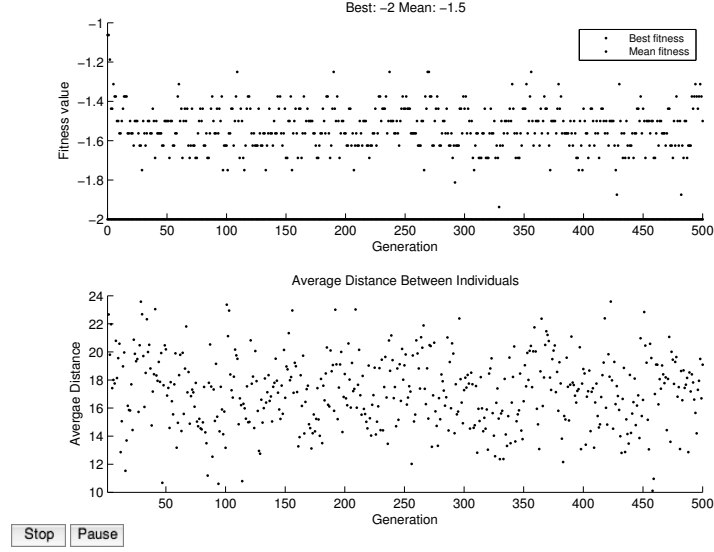


Figure 4: A single evolution of the optimal solution to the branch number of a 4-bit S-box configuration with a maximum of 500 generations.

Table 4: Some results from the genetic algorithm solver for the degree of nonlinearity of 3-bit S-boxes.

S-Box Configuration	Optimal Function Value	Generations
4 2 1 4 1 1 5 4	2	51
6 3 6 5 1 5 5 1	2	51
0 3 3 6 7 6 2 2	2	51
3 4 1 5 4 0 6 6	2	51
4 4 7 2 4 1 5 6	2	51

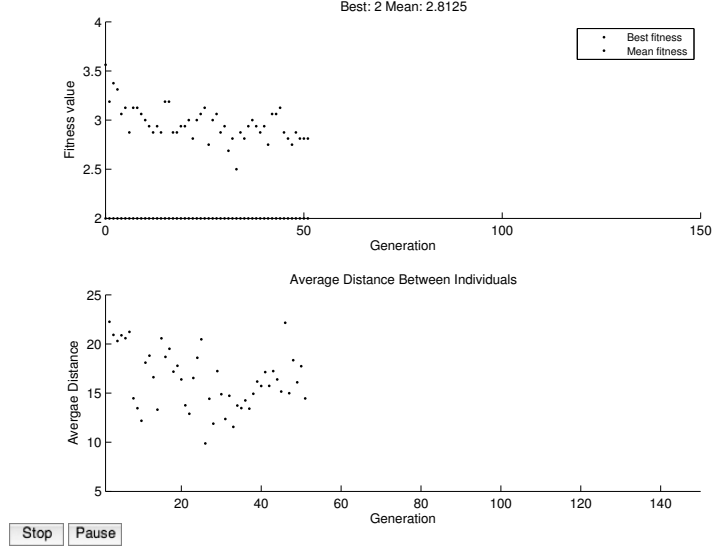


Figure 5: A single evolution of the optimal solution to the nonlinearity degree of a 4-bit S-box configuration with a maximum of 500 generations.

4.4 Multi-Objective Optimization

Separately, the branch number, avalanche number, and degree of nonlinearity all contribute to the overall levels of confusion and diffusion of a symmetric-key encryption algorithm. However, it is not enough to consider these problems separately because they are very likely to have functions that create a great deal of destructive interference when superimposed on top of one another. Therefore, it is ideal to find a shared solution S-box configuration that satisfies all three objective functions simultaneously. The most common way to solve this problem is to formulate the optimization problem as a linear combination of all three objective functions. The benefit of this approach is that we can prioritize the influence that each objective function has on the overall solution by assigning weights to each value in this linear combination. Such a linear combination is shown in Figure 4.4.

Since the weighted priorities of each objective function in this linear combination are values that should be tweaked by the cryptographer or mathematician in response to their own security priorities, they are not treated as design variables in this problem. Based on the research that was conducted into each of these metrics, various weights were assigned to this expression and then run through the genetic algorithm solver to see if a shared optimal solution could be generated. A subset of the results for 3-bit S-boxes is shown in Table 5, and a single execution of the solver is shown in Figure 4.4.

Based on these results, it is clear that each of the objective functions in this linear combination do interact destructively when combined to form the overall objective. This claim is supported by Figure 2. The immediate implication of this is that any change in the population is likely to improve one objective function at the expense of degrading another. As the simulation in Figure 4.4 indicates, even with large significant changes in the children after every generation of the genetic algorithm, the joint objective function tends to stay at the same optimal value because of the degrading penalty suffered by such changes.

Therefore, we conclude that the genetic algorithm is not an effective technique for exploring the

Joint MINLP Problem - Minimize

$$f(X) = w_1 A'_N(X) + w_2 B'_N(X) + w_3 P_S(X),$$

subject to the constraints

$$\begin{aligned} 0 \leq X(i) &\leq 2^n - 1 \\ \min_{i,j \neq i} (\text{wt}(i \oplus j) + \text{wt}(X(i) \oplus X(j))) - n2^{n-1} &\leq 0, \end{aligned}$$

where n is the number of bits needed to represent the design variables and w_i , where $1 \leq i \leq 3$, are *not* design variables.

Figure 6: The joint MINLP problem formalization as a linear combination of all three objective functions under the same constraints.

Table 5: Sample linear weights and the corresponding function value tuples for the avalanche number, branch number, and degree of nonlinearity for 3-bit S-boxes. The high weights on the branch number and nonlinearity degree selection comes from the need for higher measures of confusion (from nonlinear operations) than diffusion (from linear operations) in the S-box.

w_1	w_2	w_3	S-Box Configuration	(A_N, B_N, P_S)
1	2^8	2^8	5 6 2 5 5 6 2 5	(40, 1, 8)
1	2^4	2^8	4 3 3 6 2 5 5 4	(50, 1, 6)
1	2^8	2^4	2 5 2 2 2 5 2 2	(24, 1, 8)
1	2^4	2^4	3 5 4 2 3 5 4 2	(40, 1, 8)

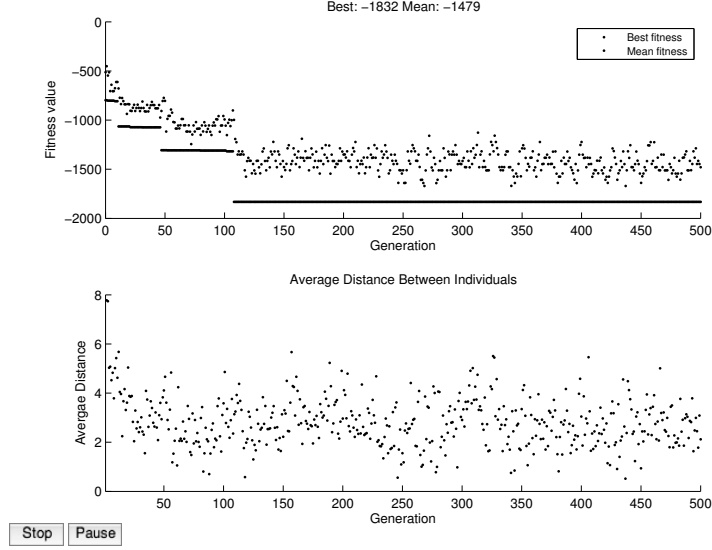


Figure 7: Weights: $1, 2^8, 2^8$

configuration permutation space and trying to optimize all three objective functions together at once.

5 Conclusions

Based on the results from the individual optimization problems and combination problem, we can conclude that the use of genetic algorithms (or any evolutionary algorithm) have very limited or specialized use in the context of cryptography. Most cryptographic functions (and even measurements of those functions) are truly random as across the function domain, and this seemingly random behavior leads to discontinuous and unstable functions with many local optimal solutions. While this rules our traditional optimization algorithms that rely on the smoothness of functions almost immediately, it still leaves some room for probabilistic evolutionary algorithms.

However, as was shown, the application of such evolutionary algorithms is only moderately useful when optimizing a single dimension (or measurement) of cryptographic algorithms; it is not effective when optimizing multiple dimensions at once. The reason for this is that when superimposing each of these functions on top of one another, we are left with a great deal of destructive interference. That is, the global optimal values rarely align for multiple measurement functions for cryptographic algorithms.

Since cryptographers seek to optimize multiple dimensions at once, the application of such optimization algorithms is likely yield little or no significant benefits to the designers. This conclusion supports the lack of research done in this problem in the field of cryptography. However, that is not to say that the application of all evolutionary algorithms should be ruled out altogether. Perhaps as researchers further refine the security measurements for symmetric-key cryptographic algorithms we may be able to define a multi-objective MINLP that can be optimized to find an optimal configuration for S-boxes. However, until that time, we are left with intuitive and mathematical derivations for S-box configurations.

References

- [1] K. Kim, “A study on the construction and analysis of substitution boxes for symmetric cryptosystems,” 1990.
- [2] H. M. Heys, “A tutorial on linear and differential cryptanalysis,” Tech. Rep., 2001.
- [3] P. P. Mar and K. M. Latt, “New analysis methods on strict avalanche criterion of s- boxes.”
- [4] M.-J. O. Saarinen, “Cryptographic analysis of all 4 x 4 - bit s-boxes,” Cryptology ePrint Archive, Report 2011/218, 2011, <http://eprint.iacr.org/>.
- [5] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker, “The skein hash function family,” 2009.
- [6] “On feistel structures using a diffusion switching mechanism.” in *Fast Software Encryption, 13th International Workshop, FSE 2006*, 2006, pp. 41–56. [Online]. Available: <http://www.iacr.org/cryptodb/archive/2006/FSE/3254/3254.pdf>
- [7] “”Provable” security against differential and linear cryptanalysis,” in *Fast Software Encryption, 19th International Workshop, FSE 2012*, 2012.