4040-849 Optimization Methods

Optimizing Cryptographic Strength of Substitution Layers in Symmetric-Key Cryptographic Algorithms

Christopher Wood

May 13, 2012

Abstract

The cryptographic security of symmetric-key block algorithms and other related primitives is based upon their adherence to 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 block ciphers and other related primitives is based upon their adherence to Shannon's principles of confusion and diffusion [1]. Confusion is typically referred to 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 bit is spread throughout the resulting ciphertext. Consequently, it is increasingly important to optimize these characteristics in order to make symmetric-key cryptographic algorithms less susceptible to common attacks, such as linear and differential cryptanalysis.

Linear cryptanalysis is an attack that attempts to take advantage of high probability occurrences of linear expressions involving plaintext bits, ciphertext bits, and subkey bits. Mathematically, the attack is based on the idea of approximating the operation of a portion of the block cipher in question with an expression that is linear in terms of the input (X) and output (Y) bits involved, as shown below [2]:

$$X_{i_1} \oplus X_{i_2} ... \oplus X_{i_r} \oplus Y_{j_1} \oplus Y_{j_2} ... \oplus Y_{j_s} \oplus = 0$$

Using this expression, attackers can guage the amount of randomness introduced by the algorithm. That is, if such an expression is satisfied frequently (with a relatively high probability), then we know that the probability of the expression holding for any two values $a, b \in \mathbb{F}_2^n$ is approximately $1/2$. However, when this probability shifts away from $1/2$, the amount of known plaintexts required to determine the key (or key block) that was used to reproduce the output goes down dramatically. Such a deviation from the expected probability of $1/2$ for any expression of the form above, which is referred to as the *linear bias*, determines the susceptibility of the block cipher to known plaintext attacks. Therefore, it is important to introduce non-linearity into the block cipher in order to defend against such attacks.

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]. To illustrate this attack technique, consider the input of a block cipher to be represented by the vector $\langle X_1, X_2, ..., X_n \rangle$ and the corresponding output to be $\langle Y_1, Y_2, ..., Y_n \rangle$, where each element X_i and Y_i corresponds to a single bit in the input and output, respectively. With this definition, we can represent the input difference of any two input vectors (ΔX) and any two output vectors (ΔY) as follows:

$$\Delta X = \langle \Delta X_1 \oplus \Delta X_2 \oplus ... \oplus \Delta X_n \rangle$$
$$\Delta Y = \langle \Delta Y_1 \oplus \Delta Y_2 \oplus ... \oplus \Delta Y_n \rangle$$

where $\Delta X_i = X_{i,1} \oplus X_{i,2}$ and $\Delta Y_i = Y_{i,1} \oplus Y_{i,2}$. It is an ideal block cipher the output different ΔY for a specific input different ΔX will occur with a probability of approximately $1/2^n$ (i.e. it produces random output based on every input block). Note that it is common to represent the input and output difference pairs as $(\Delta X, \Delta Y)$ (which are referred to as differentials).

With the idea of differential pairs in mind, we define differential cryptanalysis as the process of finding differentials that occur with a probability much higher than $1/2^n$, which subsequently gives rise to a statistical correlation between the relationship between the input and output of a block cipher and can allow one to deduce the private key used in the cipher. Therefore, it is ideal to maximize the amount of randomness introduced by the algorothm, which can be obtained from high diffusion and confusion levels. Thus, the efficiency of both of these attacks relate to the measures of confusion and diffusion within an algorithm and the extent to which they can be exploited.

Confusion and diffusion are usually realized through a unique combination of linear and nonlinear operations. It is understood that the nonlinear operations in the algorithms contribute more to the security of the corresponding algorithm than the other components. The most common source of nonlinearity in symmetric-key cryptographic algorithms is from a S(ubstitution)-box, which is a function with an equal sized domain and range that is configured to yield optimal confusion and diffusion between each input and output pair. As such, S-boxes, as the substitution layer in symmetric-key cryptographic algorithms, will be the focus of this project.

1.1 Cryptographic Strength of Substitution Layers

Mathematically, an S-box can be represented as a function f that maps input values a to output values b such that $a, b \in \mathbb{F}_2^n$. In the context of cryptographic applications, such a function f should 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 $\mathrm{wt}(x) = \sum x_i$, where x_i refers to the ith bit in x.

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

$$B_N = \min_{a,b \neq a} \left[\mathrm{wt}(a \oplus b) + \mathrm{wt}(S(a) \oplus S(b)) \right], \tag{1}$$

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

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

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

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

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

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

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

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

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

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

Designers of cryptographically secure cryptographic algorithms use these measurements as a basis for the levels of confusion and diffusion within a symmetric-key algorithm, and thus for 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

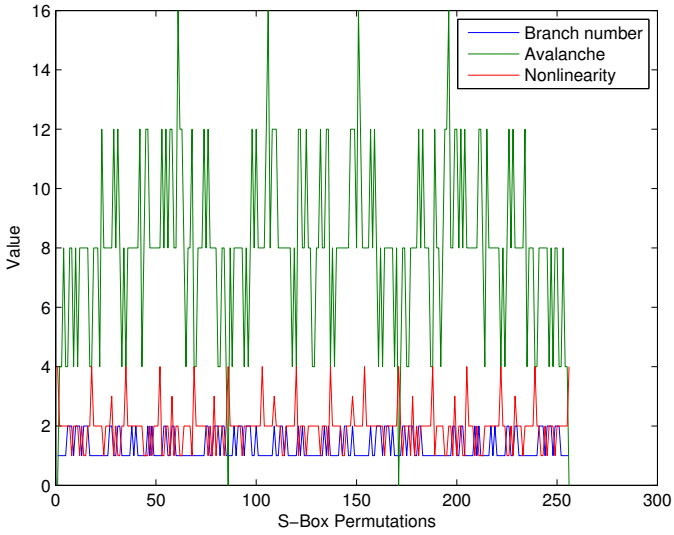

Figure 1: The results for the branch number, avalanche, and nonlinearity measurements for a 2-bit S-box over all possible permutations in the design variables.

2. Avalanche number of exactly $n^2 2^{n-1}$

3. High degree of nonlinearity

2 Optimization Candidate and Problem Formation

Due to the application of S-boxes inside cryptographic algorithms, it is ideal that such function configurations exhibit globally optimal values for each of these metrics. The best way to achieve this assurance is to perform an exhaustive search over the S-box design space (i.e. consider all possible input and output values) and find a configuration 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 each of the aforementioned metrics 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.

Branch Number - Minimize

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

subject to the constraints

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

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

Avalanche Number - Minimize

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

subject to the constraints

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

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.

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 likely to not be very effective when solving these optimization problems. In fact, the application of the famous Branch and Bound algorithm was applied to solve these problems and yielded very poor results. For that reason, we turn our focus to more evolutionary algorithms that have a probabilistic or random element to them as they explore the domain of the design variable space.

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 usally 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 aformentioned 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.

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^n} identical function values
Initial population	S-box configuration $\langle 0, 1, ..., n \rangle$

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

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 funtion is first optimized individually and then they are joined together as a multi-objective MINLP problem.

Also, since the genetic algorithm sovler 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

Based on Equation 2 from Section 1.1, we can easily tell that the optimal value for the avalanche number is exactly $n2^{n-1}$, where n is the number of bits needed to store the S-box values. Therefore, for 3- and 4-bit S-boxes, we conclude that the optimal values are $3 \cdot 3(2^2) = 3 \cdot 12 = 36$ and $4 \cdot 4(2^3) = 4 \cdot 32 = 128$, respectively, since we are concerned with the avalanche effect over every single bit. Now, 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 an illustration of the iteration pattern of the algorithm for a higher order 4-bit S-box configuration.

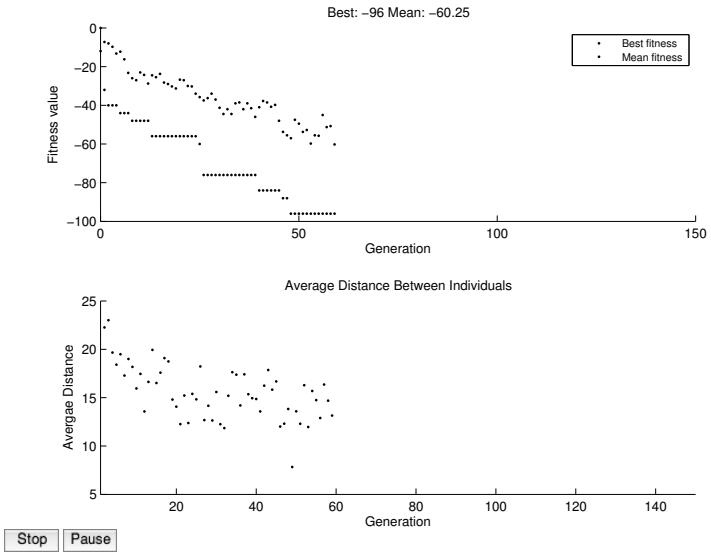

Figure 3: A single evolution of the optimal solution to the avalanche number of a 4-bit S-box configuration with a maximum of 500 generations.

Based on these results we can see that the genetic algorithm was able to reach an optimal value of 30, which is approximately 16% away from the global optimal value of 36. Given the time complexity difference between the exhaustive search and the genetic algorithm solver, it is clear that this is an acceptable result.

4.2 Branch Number

It has been proven that the maximum branch number for a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ 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 algorithm solver was able to obtain this optimal value in a small number of instances. The majority of the executions converged to the local optimal value of 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 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

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

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 Produced
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

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" [7]. However, almost-perfect nonlinear (APN) S-boxes have also been studied extensively and are shown to be almost equally resistent to the most sophisticated differential cryptanalysis attacks, where APN S-boxes have a P_S equal to exactly 2. Therefore, as shown in Table 4, the optimization algorithm 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 configuation value of $P_S = 1$.

Extending the order of the S-box to 16 elements and running the same test yielded the same APN results, as shown in Figure 4.3. Therefore, we can conclude that the genetic algorithm is very well suited to finding the S-box configuration that achieves an acceptable degree of nonlinearity.

4.4 Multi-Objective Optimization

Based on the fact that cryptographers and mathematicians attempt to optimize all of these measurements together, it is ideal to find an optimal value among all of the objective functions simultaneously. The most common way to solve this is to re-write the optimization problem as

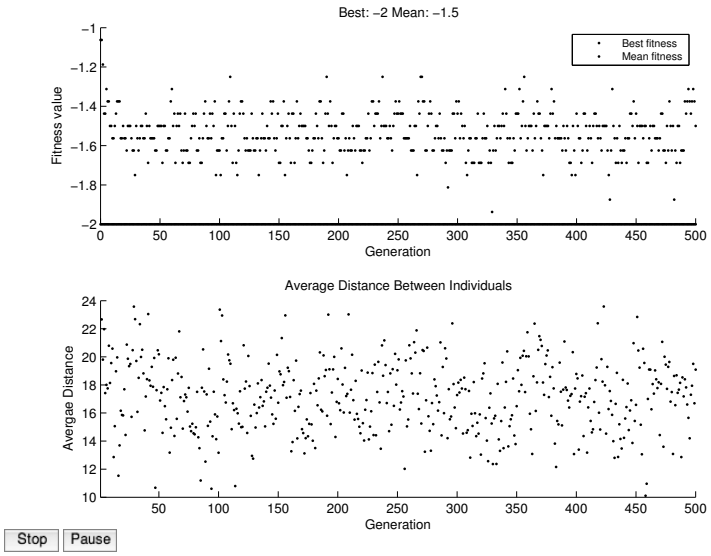

Figure 4: A single evolution of the optimal solution to the branch number of a 4-bit S-box configuration with a maximum of 500 generations.

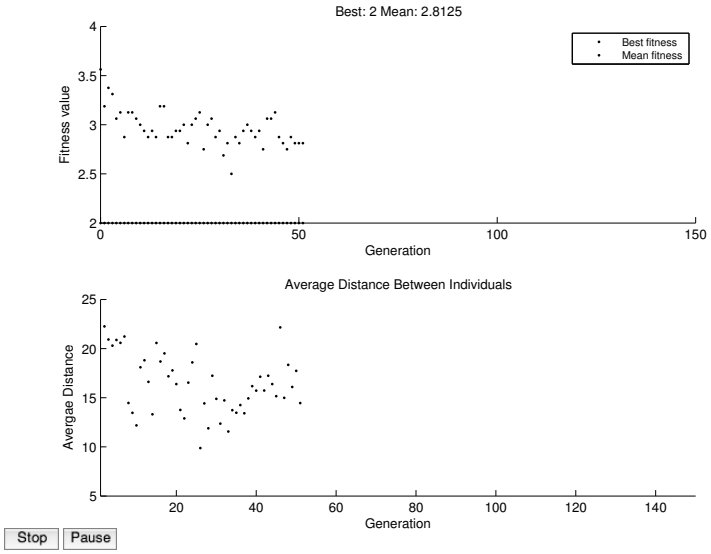

Figure 5: A single evolution of the optimal solution to the nonlinearity degree of a 4-bit S-box configuration with a maximum of 500 generations.

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

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

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.

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)

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. For the purposes of this project, the linear combination shown in Figure 4.4 was used to formalize the multi-objective MINLP problem.

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 crypgoraphic algorithms; it is not effective when optimizing multiple dimensions at once. The reason for this is that when superimposing each

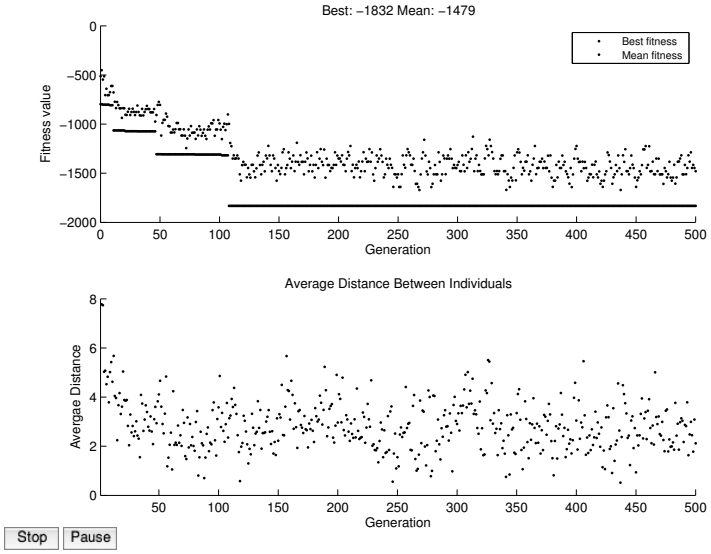

Figure 7: Weights: $1, 2^8, 2^8$

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 hask 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.