# Cartesian Genetic Programming Approach for Generating Substitution Boxes of Different Sizes

Stjepan Picek
Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia
stjepan@computer.org

Julian F. Miller
Department of Electronics
University of York, UK
julian.miller@york.ac.uk

Domagoj Jakobovic
Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia
domagoj.jakobovic@fer.hr

Lejla Batina
Radboud University
Nijmegen, The Netherlands
lejla@cs.ru.nl

## ABSTRACT

Substitution Boxes (S-boxes) play an important role in many modern-day cryptography algorithms. Indeed, without carefully chosen S-boxes many ciphers would be easy to break. The design of suitable S-boxes attracts a lot of attention in cryptography community. The evolutionary algorithms (EAs) community also had several attempts to evolve S-boxes with good cryptographic properties. When using EAs one usually uses permutation representation in order to preserve the bijectivity of the resulting S-boxes. In this paper we experiment with Cartesian Genetic Programming (CGP) and Genetic Programming (GP) in order to evolve bijective S-boxes of various sizes that have good cryptographic properties. Besides the standard CGP representation, we use an approach that allows CGP and GP to be mapped to the permutation encoding.

## Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

## Keywords

S-box, Cryptograpy, Cartesian Genetic Programming, Genetic Programming, Permutation Encoding, Bitstring Encoding

## 1. INTRODUCTION

For secure data transmission, most often a symmetric-key cryptography is used, and more precisely block ciphers [9]. All block ciphers need some nonlinear element to be secure. A well known element of this kind is the Substitution Box (S-box). S-boxes may be considered as an array of Boolean functions (vectorial Boolean functions).

Two most popular sizes for S-boxes are $4\times4$ and $8\times8$. The first S-box size is usually used in lightweight cryptography that is primarily intended for the constrained environments; an example of such lightweight cipher is PRESENT [2]. On the other side, $8\times8$ size S-boxes are used when speed and security of a cipher are of primary importance; as an example consider the AES algorithm [5]. For a detailed description of relevant S-box properties, we refer the reader to [3,4,6–8]

In this paper we investigate the efficiency of two algorithms previously unexplored when discussing the evolution of S-boxes. Those algorithms are Cartesian Genetic Programming (CGP) and Genetic Programming (GP). Furthermore, as a baseline case we experiment with a Genetic Algorithm (GA).

## 2. EXPERIMENTAL SETUP AND RESULTS

We experiment with two S-box encodings: the first one is **permutation encoding**, where $n \times n$ S-box is defined as an array of $2^n$ integer numbers with values between 0 and $2^n - 1$. The main advantage of the permutation encoding is that the **bijectivity property** is automatically preserved and this encoding is used with simple GA as a baseline.

The second encoding uses CGP and GP; To be able to represent $n \times n$ S-box with CGP, we need a CGP genotype that has $n$ inputs and $n$ outputs (8 in our case). Each output can be regarded as a single Boolean function that has its truth table representation.

However, these outputs cannot be used to directly construct an S-box truth table, because there is no guarantee that they will represent a bijective S-box. Instead, we can use them to construct a valid permutation, which in turn describes the behavior of the S-box. This approach allows us to have balanced (bijective) solutions even when the truth table based on the original outputs is not balanced. To translate from these multiple outputs to the permutation encoding we use Algorithm 1. In this algorithm, the initial permutation is constructed in an array *balanced*[] and is then permuted according to the decoded values of the CGP outputs. With GP we follow the same approach, only here we need to evolve $n$ independent trees.

The function set for CGP is OR, XOR, AND, XNOR and



*GECCO '15 July 11-15, 2015, Madrid, Spain*



ACM ISBN 978-1-4503-3488-4/15/07.
DOI: http://dx.doi.org/10.1145/2739482.2764698

**Algorithm 1** Translate to permutation encoding.

**Require:** i = 0, m = input_size, n = output_size
  **for all** values $i < 2^n$ **do**
    balanced[i] = i
    **for** j = n-1; j != 0; j– **do**
      evolved[i] = evolved[i] + truth_table[i][j]*($2^j$)
    **end for**
  **end for**
  SORT balanced[] array using evolved[] array as key
  **for all** values $i < 2^n$ **do**
    **for** j = n-1; j != 0; j– **do**
      truth_table[i][j] = (balanced[i] * $2^j$) & 0x01
    **end for**
  **end for**

**Table 1: Average fitness, $fitness_1$, CGP.**

| Gen./mut. | 1 | 4 | 7 | 10 | 13 |
|---|---|---|---|---|---|
| 100 | 296.6 | 308.56 | 309.92 | 298 | 294.36 |
| 500 | 341.56 | 341.64 | 341.44 | 341.76 | 341.76 |
| 900 | 340.44 | 342.8 | 342.36 | 342.72 | 342.48 |
| 1 300 | 342.76 | 342.8 | 342.84 | 343.12 | 343.08 |
| 1 700 | 342.96 | 343.6 | 343.36 | 343.24 | 343.28 |

AND with one input inverted. The number of input connections $n_n$ for each node is two and the number of program output connections $n_o$ is one. For number of rows and levels-back parameter we use the most common choice [1]. In this setting levels-back parameter is set to be equal to the number of columns and number of rows is 1. The function set for the GP is the same as for the CGP in order to give as fair as possible experimental treatment. The terminals correspond to $n$ Boolean variables. The number of independent trials is 50 and stopping criterion is 500 000 evaluations for each experiment.

We aim to evolve S-boxes that have as high as possible nonlinearity property and as low as possible $\delta$-uniformity; the fitness function to maximize is:

$$fitness_1 = nonlinearity + (2^n - \delta). \quad (1)$$

For further information about S-boxes and their cryptographic properties, we refer readers to [5, 7, 8].

For $4 \times 4$ S-box size all three algorithms reach the optimal value so we do not present the statistics here. When experimenting with the $8 \times 8$ size, average fitness values for different mutation rates and genotype sizes for CGP are given in Table 1. The best value of 344 corresponds to the nonlinearity of 98 and $\delta$-uniformity of 10. Average fitness values for GP with population size 500 are presented in Table 2.

GP with tree depths 5 and 7 reaches the maximal value of 346 which corresponds to the nonlinearity of 100 and $\delta$-uniformity of 10. This set of values is also the best we found for the $8 \times 8$ size. The average fitness when experimenting with GA equals 342.5 with a maximal value of 344 (corresponding to the nonlinearity 98 and $\delta$-uniformity 10).

When discussing the results, the first conclusion we can reach is that CGP and GP can be used to evolve S-boxes.

**Table 2: Average fitness, $fitness_1$, GP.**

| Pop./depth | 3 | 5 | 7 |
|---|---|---|---|
| 500 | 342.02 | 343.97 | 344.05 |

When comparing the EAs, we see that all three algorithms perform similarly, with occasional better solution for one or other algorithm.It is somewhat surprising that GP performs so well in comparison with CGP since it needs to evolve independent trees which is not easy task for larger S-box sizes.

### *Summary and Future Work.*

In this paper we investigate how to use CGP and GP algorithms to evolve S-boxes of various sizes. We present an approach which transforms any CGP/GP output into a valid permutation, which is used to encode an S-box. As far as the authors are aware, this is the first usage of permutation encoding with GP. Further work will concentrate on evolution of S-boxes with regards to the area/speed perspective where we also plan to actually implement and test such evolved solutions.

## 3. REFERENCES


[1] In J. F. Miller, editor, *Cartesian Genetic Programming*, Natural Computing Series. Springer Berlin Heidelberg, 2011.

[2] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proc 9th Int Workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, pages 450–466. Springer-Verlag, 2007.

[3] L. D. Burnett. *Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography.* PhD thesis, Queensland University of Technology, 2005.

[4] J. A. Clark, J. L. Jacob, and S. Stepney. The design of S-boxes by simulated annealing. *New Generation Computing*, 23(3):219–231, Sept. 2005.

[5] J. Daemen and V. Rijmen. *The Design of Rijndael.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[6] S. Picek, B. Ege, L. Batina, D. Jakobovic, L. Chmielewski, and M. Golub. On Using Genetic Algorithms for Intrinsic Side-channel Resistance: The Case of AES S-box. In *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, CS2 '14, pages 13–18, New York, NY, USA, 2014. ACM.

[7] S. Picek, B. Ege, K. Papagiannopoulos, L. Batina, and D. Jakobovic. Optimality and beyond: The case of 4x4 s-boxes. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014*, pages 80–83, 2014.

[8] S. Picek, K. Papagiannopoulos, B. Ege, L. Batina, and D. Jakobovic. Confused by confusion: Systematic evaluation of DPA resistance of various s-boxes. In *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, pages 374–390, 2014.

[9] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C.* John Wiley and Sons, Inc., New York, NY, USA, 1995.