One property to rule them all?
On the limits of trade-offs for S-boxes

Djurasevic, Marko ; Jakobovic, Domagoj; Picek, Stjepan

DOI
10.1145/3377930.3390247

Publication date
2020

Document Version
Accepted author manuscript

Published in
GECCO 2020

One property to rule them all? On the limits of trade-offs for S-boxes

Marko Djurasevic
Faculty of Electrical Engineering and Computing, University of Zagreb
Zagreb, Croatia
marko.durasevic@fer.hr

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

Stjepan Picek
Cyber Security Research Group, Delft University of Technology
Delft, The Netherlands
s.picek@tudelft.nl

ABSTRACT

Substitution boxes (S-boxes) are nonlinear mappings that represent one of the core parts of many cryptographic algorithms (ciphers). If S-box does not possess good properties, a cipher would be susceptible to attacks. To design suitable S-boxes, we can use heuristics as it allows significant freedom in the selection of required cryptographic properties. Unfortunately, with heuristics, one is seldom sure how good a trade-off between cryptographic properties is reached or if optimizing for one property optimizes implicitly for another property. In this paper, we consider what is to the best of our knowledge, the most detailed analysis of trade-offs among S-box cryptographic properties. More precisely, we ask questions if one property is optimized, what is the worst possible value for some other property, and what happens if all properties are optimized. Our results show that while it is possible to reach a large variety of possible solutions, optimizing for a certain property would commonly result in good values for other properties. In turn, this suggests that a single-objective approach should be a method of choice unless some precise values for multiple properties are needed.

CCS CONCEPTS

• **Security and privacy → Block and stream ciphers**; • **Computing methodologies → Discrete space search**;

KEYWORDS

Cryptography, S-boxes, Evolutionary Algorithms, Trade-off

ACM Reference Format:
Marko Djurasevic, Domagoj Jakobovic, and Stjepan Picek. 2020. One property to rule them all? On the limits of trade-offs for S-boxes. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Substitution boxes (S-boxes) are nonlinear mappings often used in block ciphers [8]. They are important as without them, a cipher would be linear and thus trivial to break with techniques like differential [2] or linear cryptanalysis [10]. Throughout the years (almost half a decade of research on block ciphers), there are numerous S-boxes designed that fit the cipher size (common sizes are from 3×3 up to 8×8. Most often with the same size of the input and output but there are exceptions, e.g., 6×4 S-box size.) and have strong cryptographic properties. Today, the most accepted approach is to employ certain algebraic constructions to build S-boxes [3].

At the same time, researchers explored whether random S-boxes or S-boxes created with heuristics can result in appropriate cryptographic properties. There, one of the main arguments for heuristics is the fact that such techniques can produce a variety of S-boxes where the designer can select the properties to emphasize on. Still, there are two problems when considering the heuristics for S-box design: 1) for larger sizes, heuristics cannot compete with algebraic constructions in the quality of obtained solutions (see, e.g., [9]), and 2) most of the heuristic research conducts experiments in an ad-hoc manner. More precisely, we identify several common approaches (and corresponding problems):

(1) Use a single-objective approach and concentrate on a single cryptographic property. This approach, as expected, commonly results in S-boxes with a very good property that was evolved. At the same time, to be useful in practice, S-box needs to fulfill several properties. Consequently, other properties are often either not "good enough" or they are "good by luck" (as we never evolved those properties).
(2) Use multi-phase approaches where first, concentrate on a single property, and once that is obtained, continue to other relevant properties. Alternatively, several properties are considered at the same time (additionally, some authors use weight factors). These approaches commonly result in various trade-offs as it is not trivial (or maybe even possible) to evolve S-boxes with all required properties in such ways.
(3) Use a multi-objective approach and evolve S-boxes with several properties. Again, this approach commonly results in trade-offs among properties.

To conclude, heuristic approaches commonly result in trade-offs, and thus suboptimal cryptographic properties. Additionally, even when certain properties reach optimal values, it is not clear whether those values are the result of luck or optimization process. Unfortunately, this is not a simple problem as we must work with large search spaces. Indeed, the search space size for an S-box of size $n \times m$ (n input bits and m output bits) equals $2^{n \cdot 2^m}$. For example, for the smallest practical S-box size (3×3), this gives search space size equal to 2^{24}, while for 8×8 size, the search space size equals

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$2^{2\,048}$. Finally, even if we restrict our attention to permutations only (where $n = m$), the search space size equals $2^n!$.

As already indicated, related works consider various settings when optimizing S-box properties, but they have in common a certain perspective where only the results are important. At the same time, a more thorough investigation of how difficult it was to obtain an S-box with certain properties or what are the ranges of attainable values are usually neglected.

In this paper, we consider evolutionary algorithms for the design of S-boxes with strong cryptographic properties. We consider various S-box sizes, solution encodings, and algorithms (single-objective and multi-objective). By doing so, we provide, to the best of our knowledge, the most detailed account of the difficulty of S-box evolution, and we provide answers to questions like 1) if we evolve only a single property, what is the worst value we can expect for some other property? 2) are there property combinations that are more aligned (i.e., so it is enough to evolve a single property only)? 3) is a multi-objective paradigm more appropriate when we optimize for several cryptographic properties? Our extensive experimental investigation can be summed into 1) permutation encoding should be the preferred choice for larger S-box sizes, 2) for smaller S-box sizes, there is almost no influence in the selection of cryptographic property that is optimized, encoding, or algorithms (as such, we suggest to use the cellular automata encoding and single-objective approach where only one property is optimized), and 3) multi-objective approach results in solutions covering smaller part of the search space, which can help avoid "surprises" that could happen with the single-objective approach that does not consider all relevant properties.

2 BACKGROUND

Let n, m be positive integers, i.e., $n, m \in \mathbb{N}^+$. We denote by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$ and by $\mathbb{F}_{2^n}$ the finite field with 2^n elements. Next, the set of all n-tuples of elements in the field $\mathbb{F}_2$ is denoted by $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements.

An S-box (substitution box) is a mapping F from n bits into m bits. An (n, m)-function F can be defined as a vector $F = (f_1, \cdots, f_m)$, where the Boolean functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ for $i \in \{1, \cdots, m\}$ are called the coordinate functions of F. The component functions of an (n, m)-function F are all the linear combinations of the coordinate functions with non all-zero coefficients. As for every n, there exists a field $\mathbb{F}_{2^n}$ of order 2^n, we can endow the vector space $\mathbb{F}_2^n$ with the structure of that field when convenient. The addition of elements of the finite field $\mathbb{F}_{2^n}$ is denoted with "+". The inner product of a and b equals $a \cdot b = a_1x_1 + \ldots + a_nx_n$ [3].

An (n, m)-function F is balanced if it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times. When F is balanced, it is a permutation (the function is bijective, meaning $n = m$).

The Walsh-Hadamard transform of an (n, m)-function F equals (see, e.g., [3]):

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^m} (-1)^{v \cdot F(x) + a \cdot x}, \ a, v \in \mathbb{F}_2^m. \tag{1}$$

The nonlinearity nl of an (n, m)-function F is the minimum nonlinearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^{m*}$ (for any set S, we denote $S \backslash \{0\}$ by S^*.) [3, 12]:

$$nl = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in \mathbb{F}_2^n \\ v \in \mathbb{F}_2^{m*}}} |W_F(a, v)|. \tag{2}$$

The nonlinearity of any (n, m) function F is bounded above by the covering radius bound:

$$nl \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{3}$$

Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ with $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$. We denote:

$$D_F(a, b) = \left\{ x \in \mathbb{F}_2^n : F(x) + F(x + a) = b \right\}. \tag{4}$$

The entry at the position (a, b) corresponds to the cardinality of the delta difference table $D_F(a, b)$ and is denoted as $\delta(a, b)$. The differential uniformity δ is then defined as [11]:

$$\delta = \max_{a \neq 0, b} \delta(a, b). \tag{5}$$

In Table 1, we give the best known/possible results for S-box properties and sizes we consider in this paper.

Finally, a fixed point is an input value that maps to the same output value:

$$x = F(x), \forall x. \tag{6}$$

The minimal number of fixed points is zero, while the maximal number equals 2^n.

The rationale for the consideration of nonlinearity and differential uniformity properties is evident as those are two properties directly related to the resilience against many cryptanalyses. At the same time, the number of fixed points is more subtle as too many fixed points are not good (for instance, all fixed points give a linear S-box), but the maximal allowed number depends on the whole cipher design. Having more fixed points could benefit the cipher form the implementation perspective, as fixed points are wires in the hardware design, and they are energy-efficient. Thus, having

Table 1: Best known values for bijective S-boxes. For 8×8, we present the best-known results while there is no guarantee better results are not possible. For bijective S-boxes (and in $\mathbb{F}_2$), both nonlinearity and differential uniformity can be even values only. The worst possible values are 0 for nonlinearity (i.e., the S-box in linear) and 2^n for the differential uniformity. The best possible differential uniformity equals two, and such functions are called the Almost Perfect Nonlinear (APN) functions. APN functions exist for both odd and even number of variables. For n odd (and when $n = m$), the best possible nonlinearity equals the Sidelnikov-Chabaud-Vaudenay bound [5]. This bound is possible for n odd only, and such functions are called the Almost Bent (AB) functions.

Size	nl	δ
3×3	2	2
4×4	4	4
5×5	12	2
6×6	24	2
7×7	56	2
8×8	112	4

more fixed points can result in S-box that is more power-efficient (and the whole cipher that is more energy-efficient).

Results show that the maximal nonlinearity is possible even with half of the values being fixed points [4]. For differential uniformity, there are no such results, so to the best of our knowledge, it is not known how many fixed points are possible while maintaining the best possible differential uniformity.

3 RELATED WORKS

We divide the related works into those considering a single cryptographic property vs. those that consider multiple properties.

Clark et al. used the principles from the evolutionary design of Boolean functions to evolve S-boxes with the desired cryptographic properties [6]. They used simulated annealing and hill-climbing to evolve bijective S-boxes with high nonlinearity for sizes up to 8×8. This work is interesting as they use two-phase optimization where in the first phase, the authors do not optimize for any of the considered cryptographic criteria (nonlinearity, autocorrelation spectrum). In contrast, in the second phase, they optimize for one of the cryptographic properties only. Finally, they provide the results for the third property, algebraic degree, but they never optimize for it. P. Tesar used a combination of genetic algorithm and a total tree searching to generate 8×8 S-boxes with nonlinearity equal up to 104 [23]. The author used a single-objective approach and concentrated on a single property only. Kazymyrov et al. used a gradient descent technique where they start with an S-box that has good differential uniformity, and they conduct a number of steps until they find an S-box with good nonlinearity, among other cryptographic properties. Picek, Knezevic, and Jakobovic used evolutionary computation in order to evolve bent (n, m)-functions [17]. There, the authors experimented with several different fitness functions considering one or two terms in the single-objective optimization. The authors considered only nonlinearity property. Picek and Jakobovic used genetic programming to evolve constructions resulting in S-boxes with good cryptographic properties [16]. They used a single-objective approach and worked with differential uniformity property only.

Picek et al. used CGP and GP to evolve 3×3 and 4×4 S-boxes and discussed how to obtain permutation-based encoding with those algorithms [21]. The authors used the single-objective optimization where both nonlinearity and differential uniformity are used. Picek et al. explored the evolution of S-boxes of size 8×8 with better resistance against side-channel attacks as measured with the transparency order and modified transparency order properties [15]. The authors use single-objective optimization with two terms, but they also report results for several more cryptographic properties. Picek et al. investigated the genetic programming approach to evolve cellular automata rules that are then used to generate S-boxes [18]. The authors used a single-objective optimization with multiple terms, and they obtained results outperforming other metaheuristic techniques for sizes 5×5 up to 7×7.

Finally, Picek, Rotim, and Cupic developed a new cost function able to reach high nonlinearity values for several different S-box sizes [13]. The authors use a multiple-phase optimization procedure where they optimize for the nonlinearity. Interestingly, they also report the differential uniformity results. While the emphasis of this paper was on the development of a new cost function, the authors also presented results with the multi-objective approach (NSGA-II), where they optimized for nonlinearity and differential uniformity.

4 EXPERIMENTAL SETUP

In this section, we briefly discuss the evolutionary algorithms we use, the solution encodings, and the fitness functions. All the experiments were performed 30 times to ensure that significant results are obtained.

4.1 Encodings

In order to tackle the problem of finding S-boxes with good cryptographic properties, two solution encodings were applied, the permutation and the cellular automata encoding. Those encodings are selected based on the related work, see., e.g., [13, 20].

The permutation encoding represents the S-box as a permutation of numbers between 0 and $2^n - 1$, where n is the S-box size, and thus implicitly ensures the bijectivity of the S-box. For this encoding, three mutation operators and five crossover operators, which are among the most commonly used in practice, were selected. The mutation operators are insert mutation, inversion mutation, and swap mutation. As for the crossover operators, we used partially mapped crossover (PMX), position-based crossover (PBX), order crossover (OX), uniform like crossover (ULX), and cyclic crossover.

The second encoding is based on the fact that an S-box could be represented as a cellular automaton (CA) with defined transitions from the input bits as the current state to the output bits as the following state. One way to define the transitions of the cellular automaton is by using a local update rule, which is simply a Boolean function of at most n bits with a single output bit. The CA local rule defines the next state of a given bit $c_i(t + 1)$ based on the current state of the same bit and adjacent bits: $c_i(t)$, $c_{i+1}(t)$, $c_{i+2}(t)$ etc. The same principle is used in the design of existing S-boxes, such as in the Keccak cipher [1].

Since the S-box is, in this case, represented with a Boolean function, we employ genetic programming to evolve a suitable function in the form of a tree. The input bits of the S-box are used as GP terminals, where the number of terminals equals n. The GP uses the function set that consists of several Boolean primitives: NOT, which inverts its argument, XOR, AND, OR, NAND, and XNOR, each of which takes two input arguments. Finally, we use the function IF, which takes three arguments and returns the second one if the first one evaluates to *true*, and the third one otherwise. A candidate Boolean function obtained with GP is evaluated in the following manner: all the possible 2^n input states are considered, and for each state, the same rule is applied in parallel to each of the bits to determine the next state (S-box output)

The variation operators are simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover [22] (selected at random) and subtree mutation. All our experiments suggest that having a maximum tree depth equal to the size of the S-box is sufficient (i.e., maximum tree depth equals n, which is the number of terminals). Both encodings are applied with the same population size of 500 individuals, whereas for the mutation probability, the permutation encoding uses 0.8, and the cellular automata encoding uses 0.7. Both encodings use the same stopping criterion,

which is either 500 000 function evaluations, or 10 000 iterations without improvement.

4.2 Fitness Functions

In this paper, we consider three properties, namely nonlinearity, differential uniformity, and the number of fixed points, which are optimized in pairs. To optimize the pairs of criteria, we use two scenarios. In the first scenario, a single objective approach, which represents a weighted sum of the two optimized properties is defined in the following way $f = \alpha P_1 + \beta P_2$, where P_1 and P_2 represent the optimized property values, whereas α and β represent the weights of the properties. The value of the previously defined fitness function is maximized.

In the experiments, the fitness function is defined to always favor the optimization of one property, by giving it a larger weight than that of the second property. In that way, the algorithm will mostly focus on optimizing the first property and only fine-tune the second one to achieve a better fitness value. Several experiments with different ratios of property weights were tested. The tuning experiments demonstrate that the best average results over several executions were obtained when the ratio between the property weights is 10:1. Furthermore, both properties are additionally normalized. Nonlinearity and δ are normalized by using the best-known values denoted in Table 1, $\delta_{F,worst}$ represents the worst possible delta value which is equal to 2^n, whereas the number of fixed points is normalized by 2^n, where n is the column number of the S-box. However, depending on whether the property is improved or worsened, the normalization is performed in a way that the best solution has a value of 1 or 0, respectively. This fitness function is optimized by using the steady-state genetic algorithm with the 3-tournament selection.

The fitness functions used in experiments are:

- Improving nonlinearity and worsening differential uniformity. $F_1 = 10 * \frac{nl}{nl_{best}} + \frac{\delta}{\delta_{F,worst}}$
- Improving differential uniformity and worsening nonlinearity. $F_2 = \frac{nl_{best} - nl}{nl_{best}} + 10 * \frac{(\delta_{F,worst} - \delta)}{\delta_{F,worst}}$
- Improving nonlinearity and the number of fixed points. $F_3 = 10 * \frac{nl}{nl_{best}} + \frac{fp}{fp_{max}}$, where fp represents the number of fixed points in the solution, and fp_{max} the maximum number of fixed points possible (or, believed to be possible) for the considered S-box size.
- Improving differential uniformity and the number of fixed points. $F_4 = 10 * \frac{(\delta_{F,worst} - \delta)}{\delta_{F,worst}} + \frac{fp}{fp_{max}}$

In the second scenario, the NSGA-II algorithm [7] is applied to perform multi-objective optimization of two criteria. In this case, no normalization is required and depending on whether the criteria need to be improved or worsened, it will simply be maximized or minimized during the evolution process.

5 RESULTS

We conduct two sets of experiments. In the first set of experiments (called the Best-Best Scenario), we employ either single-objective or multi-objective optimization to optimize for two properties. In the second set of experiments (called the Best-Worst Scenario), we concentrate on nonlinearity and differential property, but now, instead of improving both properties, one property we improve. At the same time, the other property we deteriorate (i.e., we are interested in the combinations of best property-worst property). This may seem counter-intuitive as one would assume we aim to improve cryptographic properties (regardless of one or several). Still, with this set of experiments, we aim to obtain insights on whether optimizing for a single property is enough. More precisely, if we optimize for a single property, how bad can the other property be. Then, if the other property is still good (despite our effort to make it as bad as possible), this will show we can optimize for a single property and expect the second property (or several properties) to be good. On the other hand, if we can obtain solutions that are very good in one property and bad in other property, this means that 1) there is a wide variety of solutions one can obtain, and 2) not optimizing for all relevant properties can be problematic (we say can as often, an S-box with one good property has also other property good). Note, in all experiments, the higher, the better the nonlinearity, and the lower, the better the differential uniformity. Finally, note that in Tables 4 until 7, we present the best obtained solutions regardless of their encoding.

Table 2: Best results with heuristics from the literature. The results for 4×4, 5×5, and 7×7 are also the best possible ones. We note that all solutions except 8×8 are obtained with the single-objective approach. For 8×8, both single-objective and multi-objective obtained the same values.

Size	nl	δ
4×4	4	4
5×5	12	2
6×6	24	4
7×7	56	2
8×8	104	8

5.1 Best-Best Scenario

First, in Table 2, we recall on the best results from the related works [19]. Note there are no solutions for 3×3 size, but this is the simplest scenario that we will see does not offer much insight into the S-box optimization.

After introducing the best-known solutions from the literature, we present our results for the multi-objective approach in Table 3. Note that here, we optimize both nonlinearity and differential uniformity. The results for sizes 3×3 and 4×4 are the best possible ones, which is as expected (as the problem is "easy" for such small sizes). We see already for the 5×5 size that the results cannot compare with the single-objective approach. This indicates that concentrating on two properties makes the search easier to get stuck in local optima. For sizes 6×6 up to 8×8, we see the results not being able to compete with the single-objective approach. Still, for 8×8 size, the best-obtained solution is not far from the best-known solutions with heuristics (nonlinearity is not as good as in [14], which indicates there is an additional benefit from more elaborate fitness functions).

In Tables 4, we give results for optimizing either nonlinearity and the number of fixed points or differential uniformity and the

Table 3: Results obtained for the maximization of *nl* and minimization of δ in the multi-objective approach

Size	*nl*	δ
3×3	2	2
4×4	4	4
5×5	10	4
6×6	22	6
7×7	46	8
8×8	100	8

Table 4: Results obtained for optimizing 1) *nl* and fixed points and 2) δ and fixed points. Single-objective approach.

Size	*nl*	fixed points	δ	fixed points
3×3	2	4	2	4
4×4	4	8	4	7
5×5	12	7	2	7
6×6	24	8	4	10
7×7	48	2	8	26
8×8	98	45	10	41

number of fixed points with a single-objective approach. When optimizing nonlinearity, for sizes 3×3 and 4×4, we can reach the maximal possible number of fixed points for the optimal nonlinearity (half of the values). For size 5×5, we can reach the maximal nonlinearity, but the number of fixed points is one less than the maximal possible. Larger sizes give relatively good nonlinearity values, but the number of fixed points is far from the maximal possible. Note that for 8×8 size, the nonlinearity value is lower than for our multi-objective results or related work results, which indicates that fixed points restrict the search more than the combination of nonlinearity and differential uniformity. This, in turn, indicates that good values of nonlinearity and differential uniformity appear in the same region of search space while increasing the number of fixed points moves the search away from the region of high nonlinearity. Finally, as we are not able to reach optimal nonlinearity with heuristics, it is expected that adding one more constraint on the number of fixed points makes the search even more difficult. When minimizing differential uniformity, sizes 3×3 up to 5×5 reach the minimal possible differential uniformity, but we notice the number of fixed points to be less than half (except for 3×3 case). Still, we note that there is no known bound for the optimal differential uniformity and the maximal number of fixed points, so these results could also be the optimal ones. Larger S-box sizes give very good differential uniformity values, but the number of fixed points is far from half, which indicates it is a very difficult problem to optimize for differential uniformity and the maximal number of fixed points.

In Table 5, we give results when using the multi-objective approach for optimizing either nonlinearity and the number of fixed points, or differential uniformity and the number of fixed points. For each S-box size, the table includes only the solution that obtained the best result for the cryptographic property that was optimized. Sizes 3×3 up to 5×5 do not reveal anything new when compared

Table 5: Results obtained for optimizing 1) *nl* and fixed points and 2) δ and fixed points. Multi-objective approach.

Size	*nl*	fixed points	δ	fixed points
3×3	2	4	2	4
4×4	4	8	4	7
5×5	12	7	2	7
6×6	24	4	4	10
7×7	46	10	8	26
8×8	98	34	8	32

with the single-objective approach. Larger sizes show somewhat worse results than for the single-objective approach when considering nonlinearity and somewhat better results for differential uniformity. Still, both are far from optimal values for nonlinearity/differential uniformity and what should be the maximal possible number of fixed points for optimal cryptographic properties. As such, we can conclude that for smaller sizes, there is no significant difference between single-objective and multi-objective approaches, but as the problem size goes up, the single-objective approach can reach a better trade-off between properties, especially for nonlinearity and fixed points case.

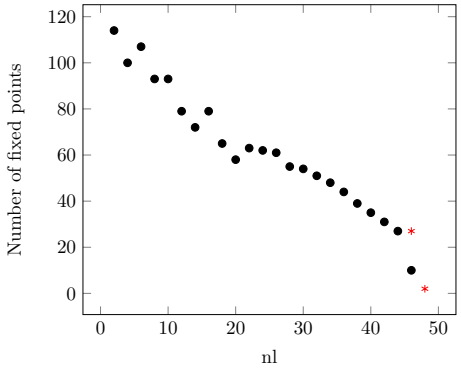
The distribution of solutions obtained by both approaches is presented in Figure 1. It should be noted that these figures do not represent the first Pareto front of solutions, but rather the union of first Pareto fronts from different executions and encodings. Therefore, certain solutions in those figures will be dominated by other solutions. The reason for choosing such a representation is to denote better the distribution of the first Pareto fronts over different runs, and the difference between the fronts obtained by the two encodings. The results are only included for the larger S-box sizes since, for them, a better distribution of results was obtained. For both sizes, it is evident that as the nonlinearity or differential uniformity properties improve, the number of fixed points in the solutions is decreasing. This just confirms the previous observation that improving these two properties moves the search away from solutions that have a large number of fixed points.

Furthermore, as the number of fixed points gradually increases, both properties deteriorate quite fast. An additional observation from Figures 1c and 1d is that one part of solutions is dislocated and achieve better nonlinearity and differential uniformity values. This set of solutions was obtained by using the permutation representation, and thus demonstrates that this representation is better suited for larger S-boxes (which is also in line with the results from the related works).

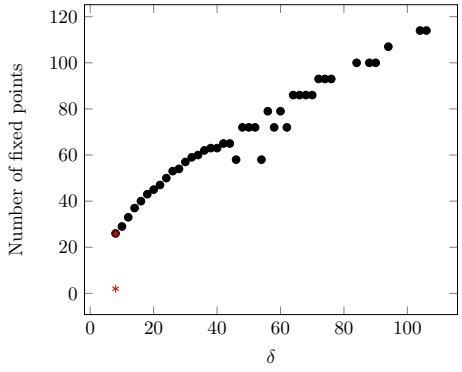
5.2 Best-Worst Scenario

In these experiments, we aim to obtain the solutions that have the best possible value for one property and the worst possible value for the other property. As an example, in Table 6, when we present results for maximization, we aim to obtain the best possible value for nonlinearity (as higher is better) but the worst possible value for differential uniformity (as lower is better).

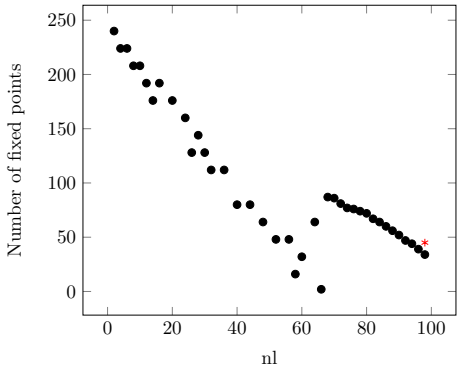
In Table 6, we present results for the single-objective approach. For size 3×3, we see it is not important whether we aim to improve nonlinearity or differential uniformity, as, for the other property,



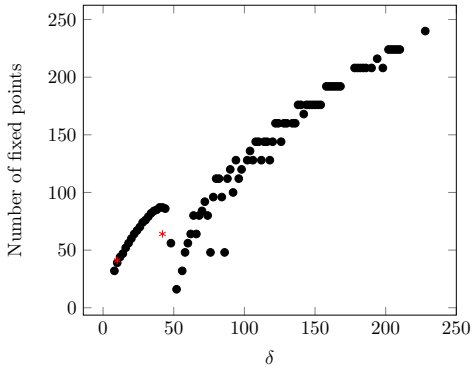
(a) Optimization of nl and fixed point count for S-box size 7×7



(b) Optimization of δ and fixed point count for S-box size 7×7



(c) Optimization of nl and fixed point count for S-box size 8×8



(d) Optimization of δ and fixed point count for S-box size 8×8

Figure 1: Solution distributions. Black points represent solutions obtained with the multi-objective approach, while red stars denote several best solutions obtained with the single-objective approach.

the worst possible value is also the best possible value. When considering this size, there is no need to optimize for both properties. Already for the 4×4 size, the results are much more interesting. First, we see it is not the same whether we improve nonlinearity and degrade differential uniformity or vice versa. Indeed, when maximizing nonlinearity, differential uniformity can differ significantly from the optimal value. At the same time, minimizing differential uniformity results in suboptimal nonlinearity but only for a single step (remember, for permutations, nonlinearity, and differential uniformity values change in steps of two). These results indicate that when maximizing for nonlinearity, the search space of possible values for differential uniformity will be easier to explore than in the opposite case. Next, the S-box size 5×5 gives very interesting results. We see that reaching the best possible value for one property also results in the best possible value for the other property (despite trying to degrade the other property). These results indicate that 1) it is enough to optimize for one property only, and 2) there does not seem to be any solutions that are optimal in one property and suboptimal in another property. Similar results are obtained for 6×6 size, where regardless of what property we maximize/minimize, we obtain the same values. Note, these values are also equal to the best-known values from literature; they are better than those obtained when using multi-objective approach (Table 3, but are worse than the best-known solutions (to the best of our knowledge, no heuristic technique ever reached the best possible value for this S-box size).

For 7×7 size, we cannot reach optimal solutions. We see that when maximizing, nonlinearity is high, but so is the differential uniformity. Consequently, optimizing for nonlinearity only can result in having bad differential uniformity. Still, the nonlinearity value is lower than the best-obtained nonlinearity value (Table 2, which means that increasing the nonlinearity value above 48 would probably result in improving differential uniformity. As such, single-objective optimization could suffice, if one uses good fitness function. On the other hand, we see that when minimizing differential uniformity and nonlinearity, differential uniformity is quite good but at the expense of having a linear function. We postulate this happens as it was enough to make one of the coordinate functions linear (which made the whole S-box linear), and this only doubled the differential uniformity value. Finally, for the 8×8 size, when maximizing nonlinearity, we can reach very good values. Unfortunately, the differential uniformity property is also very high, which means there is a significant number of attainable solutions for one to explore (and only a small part of it contains good solutions). When minimizing differential uniformity, we see similar behavior as for the 7×7 size, where we can obtain very good differential uniformity value but at the expense of having a linear

Table 6: Results obtained for nl and δ for single-objective approach.

Size	Maximization		Minimization	
	nl	δ	nl	δ
3×3	2	2	2	2
4×4	4	8	2	4
5×5	12	2	12	2
6×6	24	4	24	4
7×7	48	16	0	8
8×8	100	54	0	10

Table 7: Results obtained for nl and δ for multi-objective approach.

Size	Maximization		Minimization	
	nl	δ	nl	δ
3×3	2	2	2	2
4×4	4	8	2	4
5×5	12	2	12	2
6×6	16	32	4	6
7×7	48	16	2	8
8×8	100	38	24	8

function (which means that the search space is huge as it includes both linear/affine functions as well as those that are nonlinear).

Finally, in Table 7, we give the results for the multi-objective approach and the best-worst scenario. For sizes 3×3 up to 5×5, there is no difference in the best-obtained results when compared with the single-objective approach. For 6×6 size, we see an interesting trade-off: when maximizing nonlinearity, differential uniformity is very high. At the same time, nonlinearity also does not reach good values. When minimizing differential uniformity, both differential uniformity and nonlinearity cannot compare with the solutions for single-objective optimization. For 7×7, when maximizing nonlinearity, the best-obtained solution is the same as for the single-objective approach. When minimizing differential uniformity, we can reach the same value as for the single-objective, but now, we obtained a function that is not linear. While this is better (from a cryptographic perspective), it also tells us that the multi-objective approach is more restrictive as it was not able to look in the search space part with linear functions. Similarly, for the 8×8 size, the solutions with a multi-objective approach do not have an as wide set of values as for the single-objective approach. Again, this gives us more confidence that the multi-objective approach cannot encompass as large solution space as a single-objective approach.

Figure 2 shows the distribution of solutions when improving one criterion and worsening the other one. These distributions demonstrate that by improving one criterion, the second one is also implicitly improved. This is especially true when improving nonlinearity, as differential uniformity can be seen to improve as well steadily. On the other hand, when improving differential uniformity, a very interesting phenomenon can be observed. Namely, with the improvement of the differential uniformity property, the nonlinearity of solutions is improved as well, but only until a certain point. For the two smallest achieved differential uniformity values, smaller nonlinearity values than those of several previous solutions were obtained. The reason why this happened is that these two solutions dominate all the other solutions, and thus the algorithm could not explore the Pareto front more exhaustively. Finally, these two solutions with the smallest differential uniformity values were obtained by using the permutation encoding, wheres the other solutions were obtained by using the cellular automata encoding, which additionally demonstrates the superior performance of the permutation encoding for larger S-box values.

Table 8 represents the maximum, average, and standard deviation of the fitness values, which represent the weighted sum of the two considered properties, based on the 30 performed runs. Additionally, the Mann-Whitney test was performed between the results for the same S-box size between the different encodings, and the result is denoted in the column "p-val". The bolded values in this column denote that the specific encoding performed significantly better, where the results are considered significantly different if the p-value is below 0.05. The table shows that both encodings achieve the same performance for the smallest S-box values, with the CA encoding usually achieving significantly better results. On the other hand, for the larger S-box sizes, the permutation encoding performs better. This is evident from the fact that the permutation encoding achieved a better average value, as well as a smaller standard deviation, which means that the obtained results in all runs are comparable. The statistical tests confirm the previous conclusion. This is usually manifested that for the property that was improved, the algorithm obtained similar values in all runs. The largest differences between the runs were in the values of the property that had to be worsened.

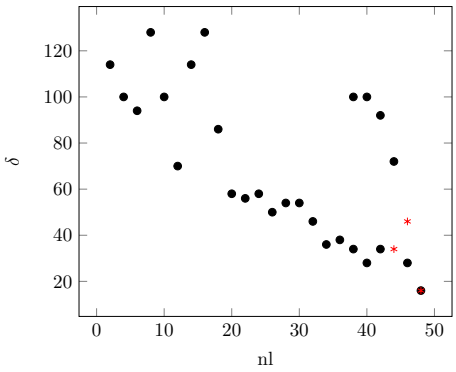
Table 8: Statistical results for the single-objective optimization.

	Size	Maximization				Minimization			
		max	avg	std	p-val	max	avg	std	p-val
Permutation	3×3	10.25	10.25	0	1	7.5	7.5	0	1
	4×4	10.5	10.5	0	10^{-5}	8	8	0	$\mathbf{10^{-5}}$
	5×5	8.58	7.7	0.30	10^{-5}	9.42	9.25	0.07	10^{-5}
	6×6	8.95	8.88	0.04	$\mathbf{60^{-5}}$	9.98	9.82	0.07	$\mathbf{10^{-5}}$
	7×7	8.57	8.52	0.03	$\mathbf{10^{-5}}$	10.38	10.34	0.02	$\mathbf{10^{-5}}$
	8×8	9.17	9.13	0.03	$\mathbf{10^{-5}}$	10.61	10.58	0.02	$\mathbf{10^{-5}}$
Cellular automata	3×3	10.25	10.25	0	1	7.5	7.5	0	1
	4×4	10.38	10.38	0	$\mathbf{10^{-5}}$	7.5	7.5	0	10^{-5}
	5×5	10.06	9.82	0.56	$\mathbf{10^{-5}}$	9.58	9.36	0.13	$\mathbf{10^{-5}}$
	6×6	10.06	7.79	1.22	60^{-5}	9.39	8.86	0.44	10^{-5}
	7×7	8.87	6.97	1.04	10^{-5}	9.66	9.16	0.23	10^{-5}
	8×8	7.04	5.96	0.48	10^{-5}	9.21	8.69	0.31	10^{-5}

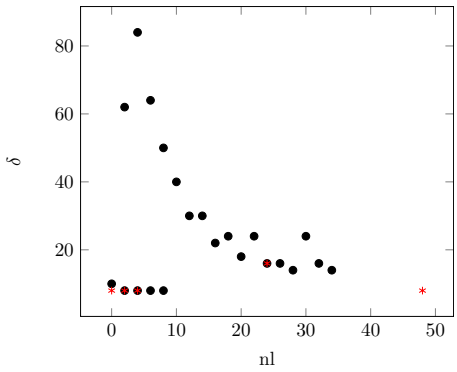
6 DISCUSSION

Based on the results from the previous section, we can give a set of guidelines when optimizing S-boxes. Informally, when discussing smaller S-boxes, we consider sizes up to $n = 5$, while larger S-boxes are any size where $n > 5$.

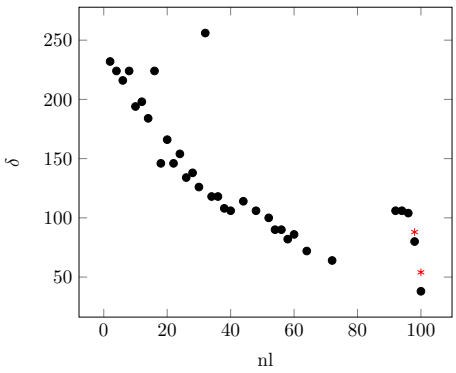
- If optimizing small S-box sizes and nonlinearity/differential uniformity, there is no influence of the selection of the encoding. Next, single-objective and multi-objective approaches



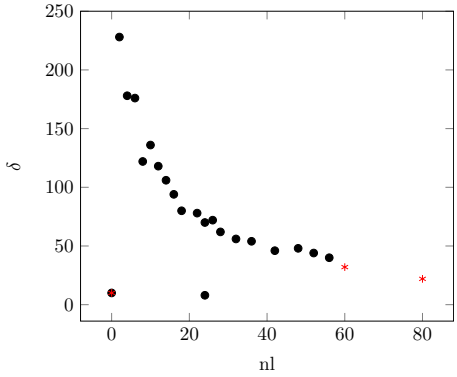
(a) Maximization of nl and δ for S-box size 7×7



(b) Minimization of nl and δ for S-box size 7×7



(c) Maximization of nl and δ for S-box size 8×8



(d) Minimization of nl and δ for S-box size 8×8

Figure 2: Solution distributions. Black points represent solutions obtained with the multi-objective approach, while red stars denote several best solutions obtained with the single-objective approach.

result in the same best-obtained solutions. Optimizing for a single property is enough, as the other property is either very good or even optimal.

- For larger S-box sizes, the permutation encoding gives better results (also, statistical indicators are more stable over several runs).
- For larger S-box sizes, single-objective is better than multi-objective, but there is a limit. 8×8 S-box size gives similarly good results for single-objective and multi-objective approaches. This means that the single-objective approach has a more sudden drop in the performance between 7×7 and 8×8 sizes.
- Nonlinearity and differential uniformity tend to go inline: having good one property makes the other property also good. Still, optimizing for nonlinearity gives on average better results for differential uniformity than vice versa.
- Maximizing for the number of fixed points poses more restrictions on nonlinearity and differential uniformity for larger S-box sizes (or vice versa).
- Single-objective optimization results in a wider spread of solutions for the best-worst scenario. As such, this leaves more room for random behavior (for instance, some properties with extremely bad values) if optimizing for a single property but interested in several properties.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we explore various trade-offs for S-boxes stemming from two encodings, two algorithms, and three cryptographic properties. Our results indicate that for small S-box size, there is little importance in the selection of encodings, algorithms, or cryptographic properties. For larger sizes, the permutation encoding is better than the one based on cellular automata. Next, the single-objective approach gives solutions with better cryptographic properties but also leaves more room for a bad property value if not explicitly optimized in the fitness function.

In future work, we plan to explore the influence of the granularity of the fitness function to the performance. Indeed, for now, we considered only the property final values, but one could also consider the whole Walsh-Hadamard spectrum or all the values in the DDT. Besides this, as the single-objective approach results in better solutions, it would be interesting to see how to tune the multi-objective approach to reach the same performance. Alternatively, to understand at what point in the optimization process, the multi-objective approach concentrates on the local optima region.

REFERENCES

[1] Guido Bertoni, Joan Daemen, Michäel Peeters, and Gilles Van Assche. 2011. The Keccak reference. (January 2011). http://keccak.noekeon.org/.

[2] Eli Biham and Adi Shamir. 1991. Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '90)*. Springer-Verlag, London, UK, UK, 2–21. http://dl.acm.org/citation.cfm?id=646755.705229

[3] Claude Carlet. 2010. Vectorial Boolean Functions for Cryptography. In *Boolean Models and Methods in Mathematics, Computer Science, and Engineering* (1st ed.), Yves Crama and Peter L. Hammer (Eds.). Cambridge University Press, New York, USA, 398–469.

[4] Claude Carlet, Annelie Heuser, and Stjepan Picek. 2017. Trade-Offs for S-Boxes: Cryptographic Properties and Side-Channel Resilience. In *Applied Cryptography and Network Security*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.). Springer International Publishing, Cham, 393–414.

[5] Florent Chabaud and Serge Vaudenay. 1995. Links between differential and linear cryptanalysis. In *Advances in Cryptology — EUROCRYPT '94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, 1994 Proceedings*, Alfredo De Santis (Ed.). Springer Berlin Heidelberg, 356–365.

[6] John A. Clark, Jeremy L. Jacob, and Susan Stepney. 2005. The design of S-boxes by simulated annealing. *New Generation Computing* 23, 3 (Sept. 2005), 219–231. http://dx.doi.org/10.1007/BF03037656

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Trans. Evol. Comp* 6, 2 (April 2002), 182–197. https://doi.org/10.1109/4235.996017

[8] Lars R. Knudsen and Matthew Robshaw. 2011. *The Block Cipher Companion*. Springer. I–XIV, 1–267 pages.

[9] Luca Mariot, Stjepan Picek, Alberto Leporati, and Domagoj Jakobovic. 2019. Cellular automata based S-boxes. *Cryptography and Communications* 11, 1 (01 Jan 2019), 41–62. https://doi.org/10.1007/s12095-018-0311-8

[10] Mitsuru Matsui and Atsuhiro Yamagishi. 1993. A new method for known plaintext attack of FEAL cipher. In *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques (EUROCRYPT'92)*. Springer-Verlag, Berlin, Heidelberg, 81–91. http://dl.acm.org/citation.cfm?id=1754948.1754958

[11] Kaisa Nyberg. 1991. Perfect Nonlinear S-Boxes. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings (LNCS)*, Vol. 547. Springer, 378–386. https://doi.org/10.1007/3-540-46416-6_32

[12] Kaisa Nyberg. 1993. On the construction of highly nonlinear permutations. In *Advances in Cryptology - EUROCRYPT' 92*, RainerA. Rueppel (Ed.). LNCS, Vol. 658. Springer Berlin Heidelberg, 92–98.

[13] Stjepan Picek, Marko Cupic, and Leon Rotim. 2016. A New Cost Function for Evolution of S-Boxes. *Evolutionary Computation* 24, 4 (2016), 695–718. PMID: 27482748.

[14] Stjepan Picek, Marko Cupic, and Leon Rotim. 2016. A New Cost Function for Evolution of S-Boxes. *Evolutionary Computation* 24, 4 (2016), 695–718.

[15] Stjepan Picek, Bariş Ege, Lejla Batina, Domagoj Jakobovic, Lukasz Chmielewski, and Marin Golub. 2014. 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)*. ACM, New York, NY, USA, 13–18.

[16] Stjepan Picek and Domagoj Jakobovic. 2019. On the Design of S-Box Constructions with Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 395–396. https://doi.org/10.1145/3319619.3322040

[17] S. Picek, K. Knezevic, and D. Jakobovic. 2017. On the evolution of bent (n, m) functions. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2137–2144. https://doi.org/10.1109/CEC.2017.7969563

[18] Stjepan Picek, Luca Mariot, Alberto Leporati, and Domagoj Jakobovic. 2017. Evolving S-Boxes Based on Cellular Automata with Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 251–252. https://doi.org/10.1145/3067695.3076084

[19] Stjepan Picek, Luca Mariot, Alberto Leporati, and Domagoj Jakobovic. 2017. Evolving S-boxes Based on Cellular Automata with Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. ACM, New York, NY, USA, 251–252.

[20] Stjepan Picek, Luca Mariot, Bohan Yang, Domagoj Jakobovic, and Nele Mentens. 2017. Design of S-boxes Defined with Cellular Automata Rules. In *Proceedings of the Computing Frontiers Conference (CF'17)*. ACM, New York, NY, USA, 409–414.

[21] Stjepan Picek, Julian F. Miller, Domagoj Jakobovic, and Lejla Batina. 2015. Cartesian Genetic Programming Approach for Generating Substitution Boxes of Different Sizes. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion '15)*. Association for Computing Machinery, New York, NY, USA, 1457–1458. https://doi.org/10.1145/2739482.2764698

[22] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. 2008. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. (With contributions by J. R. Koza).

[23] Petr Tesař. 2010. A New Method for Generating High Non-linearity S-Boxes. *Radioengineering* 19, 1 (April 2010), 23–26.