A Search for Differentially-6 Uniform $(n, n-2)$ Functions

Stjepan Picek* †, Karlo Knezevic‡, Domagoj Jakobovic‡, Claude Carlet† §
*Cyber Security Research Group, Delft University of Technology, Delft, The Netherlands
†LAGA, Department of Mathematics, University of Paris 8 (and Paris 13 and CNRS), France
‡University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
§University of Bergen, PB 7803, 5020 Bergen, Norway

***Abstract*—Finding cryptographic primitives satisfying certain properties is a difficult problem. In this domain, besides the algebraic constructions, researchers often use heuristics. There exists a set of interesting problems related to the notion of differential uniformity for a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. When $n = m$, then the best obtainable differential uniformity equals 2, since it is necessarily positive and even, and since examples of differentially 2-uniform functions are known. Heuristics are able to reach such functions; there is then some intuition that heuristics can be used for other open problems related to differential uniformity. When $n > m > n/2$, differential uniformity is bounded by $2^{n-m} + 2$ from below (when $m = n - 2$, by 6). Unfortunately, we know such functions only for dimensions equal to $n = 4, 5$. In this paper, we explore several evolutionary algorithms and problem sizes in order to find functions having differential uniformity equal to 6. Our results show that several solution encodings are able to find such functions but only in dimensions $(4, 2)$ and $(5, 3)$. Since differentially 6-uniform functions were known for those sizes before, our results can be used as a source of new functions in those dimensions and as an indicator that for $(6, 4)$ such functions either do not exist or that it is extremely difficult to find them.**

I. Introduction

In symmetric key cryptography, all parties in the communication use the same secret key [1]. They combine the information about the secret key and the message (input) in such a way that it is infeasible for an attacker to obtain the message without knowing the key. Functions combining secret keys and input information are called cryptographic algorithms, more commonly known as ciphers. Symmetric key cryptography can be divided into stream and block ciphers. Within the domain of block ciphers, there are several design strategies one could follow: Substitution Permutation Network (SPN), Feistel structure, and ARX structure [1]. Those structures have in common that they use mathematical operations to provide confusion and diffusion effects [2]. The confusion principle means that the cipher output statistics should depend on the cipher input statistics in a manner too complicated to be exploited by the attacker. Diffusion principle relates to the fact that each digit of the input and each digit of the secret key should influence many digits of the output.

To provide confusion, a common way is to use Substitution Boxes (S-boxes), see e.g., [3]. S-boxes, which are mappings between $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$, provide nonlinear relationship between the n input bits and the m output bits in a controllable fashion for a specific secret key. When used in the SPN structure, S-boxes must be bijective (and then, $n = m$). Well-known examples of such ciphers are AES [4] and PRESENT [5]. When used in the Feistel structure, n and m sizes can differ and both directions are possible. For instance, in the DES cipher [6], input dimension equals 6, while the output dimension equals 4. In the CAST cipher [7], input dimension is 8, while the output dimension is 32.

All S-boxes have in common that they need to fulfill a number of criteria so that the cipher using them can resist cryptanalyses. For instance, large nonlinearity will make the cipher more resilient against linear cryptanalysis [8], while small differential uniformity will make the cipher more resilient against differential cryptanalysis [9]. To obtain S-boxes fulfilling those properties (among other relevant ones), we can use algebraic constructions and heuristics. Interestingly, although this problem is an active research domain for several decades, our current knowledge is still limited and we know only a handful of algebraic constructions [3].

To construct bijective S-boxes with the best possible differential uniformity, we currently know only a few algebraic constructions where the most prominent examples are so-called monomial power functions, which are of the form $F(x) = x^d$. The best possible (and known) differential uniformity value then equals 2 for any odd n and also for $n = 6$. For n even and larger than 6, the best differential uniformity of (n, n)-permutations is an open question. Functions having differential uniformity equal to 2 are called Almost Perfect Nonlinear (APN). When considering heuristics, currently we are able to generate APN functions only for dimensions $n = 5, 7$ [10]. The notion of APN function, i.e., differentially 2-uniform function, can be weakened and then we consider differentially δ-uniform (n, m)-functions. Interestingly, such functions are much less explored when $m \neq n$ and we know of even fewer algebraic constructions than for APN functions [11].

It is an open problem whether there exist differentially δ-uniform $(n, n-k)$ functions with $k \geq 2$, k significantly smaller than $\frac{n}{2}$, $\delta < 2^{k+1}$, and $n > 5$ [12] (constructions exist for k near $\frac{n}{2}$, see [13]). More concretely, if we set $k = 2$, it is still unsolved whether there exist functions $(n, n-2)$ that have differential uniformity less than 8 when $n > 5$. In this paper, we examine this problem and whether heuristics, more precisely evolutionary algorithms, can be used to find

new functions fulfilling such criteria or at least bring new insights about the problem. To this end, we experiment with 5 representations of solutions and 4 problem sizes. Our experiments show that some of the representations we use are able to find differentially 6-uniform functions in dimensions $(4, 2)$ and $(5, 3)$. Although such functions were known before, we still consider our results relevant. The lack of success for higher dimensions coupled with the unsuccessful results from algebraic constructions could serve as a strong indicator that differentially 6-uniform functions do not exist in dimension $(6, 4)$.

The rest of this paper is organized as follows. In Section II we provide necessary information about S-boxes, the differential uniformity property, and we give an overview of relevant work. Section III discusses our experimental setup. In Section IV, we give results obtained with 5 encodings and for 4 problem sizes. Besides that, we provide a discussion on the relevance of the results and possible future research directions. Finally, in Section V, we briefly conclude the paper.

II. Background

In this section, we first discuss basic notions about S-boxes and the differential uniformity property. Afterwards, we give an overview of related work considering both algebraic constructions and heuristics.

A. Generalities on S-boxes

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 $\mathbb{F}_2$, where $\mathbb{F}_2$ is the Galois field with two elements, and by $\mathbb{F}_{2^n}$ the finite field with 2^n elements. Since 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 "+", as usual in mathematics.

An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. 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.

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:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) + F(x + a) = b\}. \quad (1)$$

The entry at the position (a, b) corresponds to the cardinality of the difference distribution table $D_F(a, b)$ and is denoted as $\delta(a, b)$. The *differential uniformity* δ_F is then defined as [14]:

$$\delta_F = \max_{a \neq 0, b} \delta(a, b). \quad (2)$$

Differential uniformity is always an even number since if x is a solution of the equation $F(x)+F(x+a) = b$ when $a \neq 0$, then also $x+a$ must be a solution. Kaisa Nyberg observed that $\delta_F \geq 2^{n-m}$ only if $n > m$, and $\delta_F \geq 2$ only if $n \geq m$ [14]. Actually, she proved that $\delta_F = 2^{n-m}$ if and only if n is even and $m \leq \frac{n}{2}$. For $n \geq 5$, Nyberg proved that δ_F is bounded below by 6 for $(n, n-2)$ functions. Differentially 8-uniform $(n, n-2)$ functions are easily constructed by composing on the left any APN function by a surjective affine $(n, n-2)$ function. Still, we do not know whether it is actually possible to construct a differentially 6-uniform $(n, n-2)$ when n is larger than 5.

B. Related Work

When considering algebraic constructions, there are results showing bounds for differential uniformity for $(n, n-2)$ functions [14]. Carlet and Alsalami are able to construct $(n, n-1)$ functions that are differentially 4-uniform [11]. They also use the same construction to find $(5, 3)$ functions that are differentially 6-uniform. Unfortunately, their construction does not generalize to higher dimensions and they are not able to find any differentially 6-uniform function for dimension $(6, 4)$. De Meyer and Bilgin conducted an exhaustive search of all quadratic $(6, 4)$ functions [15]. Their results show there are actually no $(6, 4)$ differentially 6-uniform functions with algebraic degree equal to 2.

When considering heuristics, most of the work on S-boxes concentrates on bijective S-boxes. Additionally, the primary goal in these papers is usually the nonlinearity property and only seldom researchers consider differential uniformity. Clark et al. used simulated annealing (SA) and hill climbing algorithm to evolve bijective S-boxes of sizes up to 8×8 with high nonlinearity values [16]. Millan et al. worked with genetic algorithms to evolve S-boxes with high nonlinearity and low autocorrelation value. Additionally, the authors discussed the selection of the appropriate genetic algorithm parameters [17]. Burnett et al. used a heuristic method to generate MARS-like S-boxes [18]. They are able to generate a number of S-boxes of appropriate size that satisfy all the requirements placed on the MARS S-box [19] and they even manage to find S-boxes with the improved nonlinearity values. P. Tesar used a combination of a special genetic algorithm with a total tree searching to generate 8×8 S-boxes with nonlinearity equal up to 104 [20].

Picek et al. explored how to generate S-boxes of size 8×8 with a better resistance against side-channel attacks as measured with the transparency order and modified transparency order properties [21], [22]. Picek, Rotim, and Cupic developed a new cost function able to reach high nonlinearity values for a number of different S-box sizes (they consider only $n = m$ case) [23]. With this fitness function, they eliminated the need for several parameters that usually required an extensive tuning phase. Picek, Knezevic, and Jakobovic used evolutionary computation in order to evolve bent (n, m)-functions but they do not consider differential uniformity [24].

III. Experimental Setup

A. Fitness Function

In order to obtain the target differential uniformity value, the evaluation of each potential solution includes only the calculation of the differential uniformity property. Based on

that, the fitness function to be minimized is simply the absolute distance from the desired value of 6:

$$fitness_1 = |6 - \delta_F|. \quad (3)$$

We noticed that when using $fitness_1$, the used algorithms had difficulties in finding the optimal solutions. Since the values in the difference distribution table (Eq. (1)) depend on each other, looking only at the maximal value can be counterintuitive. Indeed, minimizing the maximal value can easily result in increasing some other value in the table. Consequently, we also conducted experiments with the second fitness function defined as:

$$fitness_2 = \sum_{a \in \mathbb{F}_2^n} \sum_{b \in \mathbb{F}_2^m} |6 - \delta(a, b)|. \quad (4)$$

The motivation for the fitness function defined as in Eq. (4) was to minimize the number of values different from 6 occurring in the difference distribution table, or alternatively, to make as much as possible values close to 6.

B. Solution Representation

We used several representations to obtain more reliable results. Some of the representations, such as genetic programming, integer and floating-point, cover all of the possible search space. Additionally, we employ representations that only map to a portion of the search space, such as permutation based, in the hope of increasing the probability of finding the optimal solution. All the implementations are based on the Evolutionary Computation Framework [25]. We opted to use multiple genetic operators with every encoding since our previous experience showed such a strategy giving the best results. Additionally, since there are no previous works considering the problem of finding differentially 6-uniform $(n, n-2)$ functions, we had no point of reference to select a single best operator.

1) Genetic Programming - Tree Encoding: As the first encoding, we use tree-based genetic programming (GP) in which a Boolean function is represented by a tree of nodes [26]. The function set for GP in all the experiments is OR, NOT, XOR, AND, XNOR, AND with one input inverted, and IF, which takes three arguments and returns the second one if the first evaluates to true, and the third one otherwise. The terminals correspond to n Boolean variables. Note, Boolean functions can be represented only in XOR and AND operators but it is quite easy to transform the function from one notation to the other. We use $n-2$ independent trees in an individual to represent a single solution. The crossover is performed with five different tree-based crossover operators selected at random: a simple tree crossover with 90% bias for functional nodes, uniform crossover, size fair, one-point, and context preserving crossover [27]. The mutation operators are subtree, shrink, hoist and permutation, applied at random for each mutation operation.

2) Integer Encoding: In this representation, we use the truth table of the underlying $(n, n-2)$ function; the truth table is represented with an integer array of length 2^n with each element in the range $[0, 2^{n-2} - 1]$. Before the fitness calculation, it can be easily transformed into the binary form and evaluated accordingly. In this encoding, the mutation and crossover operators are based on their binary counterparts: the mutation selects a random gene and modifies its value. The crossover operators are single point and two-point crossover, which only concatenate parts of different individuals, and average crossover for which the child's genes assume mean values of parents' genes.

3) Permutation Encoding: In order to restrict the search space and make it easier for the algorithm to form a solution, a permutation encoding is employed rather than an array of arbitrary values. In this representation, we use the permutation array of size 2^n with elements in range $[0, 2^n - 1]$ where each value occurs exactly once. Before evaluation, we decode it into a $(n, n-2)$ function by truncating each value with modulo 2^{n-2}; that way it will represent a $(n, n-2)$ function truth table. For instance, an example input permutation of size 2^n with beginning elements $[7, 0, 9, 11, 2, \ldots]$ will be transformed into $[3, 0, 1, 3, 2, \ldots]$. After that, the individual is evaluated as in the previous encoding. The permutation genotype uses the crossover operators OX, PBX, PMX and cyclic crossover, whereas the mutation operators are insert, inverse and toggle. For each crossover and mutation, a single operator is chosen at random.

4) Quadruple Permutation: The permutation encoding poses constraints on the size of the search space but for larger function sizes it is still very large. Consequently, we limit it even further by employing the representation in which each individual is encoded with four permutations, each of size 2^{n-2}. This encoding uses the same genetic operators as the previous one. The total length of the individual genotype is the same as in the previous case but this time the crossover and mutation operators are performed separately on each of the containing four permutations - by default, an individual will have only one of them affected with mutation or crossover.

5) Floating-point Encoding: Finally, we use the floating-point encoding in the following manner: an individual is an array of floating-point values in range $[0, 1]$. Each floating-point value represents one or more integer values in range $[0, 2^{n-2} - 1]$ as in the integer encoding. Based on the number of different integer values (2^n) and the number integers per single floating-point (k), the range $[0, 1]$ is divided into $(2^n)^k$ intervals. Depending on the interval in which each floating-point value falls, we decode k elements of the truth table from a single floating-point variable. In our experiments we use $k = 1, 2, 4$ so the size of the floating-point array is either the same (for $k = 1$), two times or four times smaller than the corresponding integer array size. For instance, if $k = 2$ and we are searching for the $(4, 2)$ function, the individual is an array of 8 floating-point values where each gene presents two integer values. If the first few floating-point numbers are $[0.54, 0.41, 0.96, \ldots]$ they will get decoded into

integer array $[2, 0, 1, 2, 3, 3, \ldots]$ of size 16. This way, we are able to use a wide variety of floating-point based crossover and mutation operators, which are normally not applicable in the discrete domain. In our experiments we used the arithmetic, heuristic, average, one point and SBX crossover; the mutation consists of a single operator which changes a randomly selected gene.

Considering the expressiveness of the above representations, we note that the GP, integer, and floating-point array allow mapping to all the possible solutions, while the permutation and quadruple permutation map only a portion of the search space.

C. Algorithm and Parameters

Regardless of the representation, all the variants use the same evolutionary algorithm, which is a steady-state process based on tournament selection for individuals to eliminate, with the tournament size equal to 3: the algorithm selects 3 individuals at random and eliminates the worst among them. The remaining tournament survivors are then used as parents to create a new individual via crossover (presented in Algorithm 1).

Algorithm 1 Steady-state 3 tournament selection

```
randomly select 3 individuals;
remove the worst of 3 individuals;
child = crossover (best two of the tournament);
perform mutation on child, with given individual mutation probability;
insert child into population;
```

In all the experiments, the number of independent trials for each configuration is 30 and the stopping criterion for all algorithms equals 10^6 evaluations or reaching the differential uniformity equal to 6. In order to evaluate the differences between different representations, we conducted a tuning phase. For each encoding, we selected the most influential parameters (based on our previous experience) and run the tests with different parameter values on the problem size of $(5, 3)$. Unfortunately, the problem at hand provided no clear guidelines, since in most cases all the experiments converge to the same value in all the repetitions. In other words, for most of the configurations there were no statistically significant differences between different parameter values. In cases where the difference was not visible we have kept the initial parameter values, whereas the different final parameter values are shown in boldface. The results of the tuning phase are shown in Tables I till V.

The tuning phase was conducted with both $fitness_1$ and $fitness_2$ functions. The results showed that $fitness_1$ significantly outperforms $fitness_2$ and we consequently present the results for only the first fitness function.

IV. Results

In total, the search space size of possible (n, m)-functions equals 2^{m2^n}. When considering $(n, n-2)$ functions, when $n = 4$ the search space size equals 2^{32}, which is possible to exhaustively search. Already for $(5, 3)$ functions, the search space size equals 2^{96}, which is far beyond current computing capabilities.

TABLE I: Parameters for GP representation

Parameter	Initial value	Tested values	Final value
Max tree depth	4	{4, 5, 6}	**5**
Population size	200	{100, 200, 300}	200

TABLE II: Parameters for quadruple permutation

Parameter	Initial value	Tested values	Final value
Population size	200	{100, 200, 500}	200
Mutation rate	0.5	{0.1, 0.3, 0.5, 0.7, 0.9}	**0.9**

A. Exhaustive Search in $(4, 2)$ Dimension

We conduct the exhaustive search in $(4, 2)$ dimension to investigate the properties of the functions with differential uniformity equal to 6. We find 0.38% of functions with such differential uniformity. Only 1.367% of those functions have the same number of occurrences of each value (e.g., 4 values 0, 4 values 1, etc.). The distribution of S-box values is uniform, i.e., we notice every possible value on every possible position. Next, we investigate the number of functions in which, at least, one value is missing. There are 1.366% of functions with that property. When considering a single value repeating itself a number of times consecutively, we find the longest such subset to be of length 6. Finally, when considering the results that could be obtained with the quadruple permutation encoding, out of 0.38% solutions with differential uniformity equal to 6, there are 6.70% that can be represented with quadruple permutations.

B. Heuristic Results

We conduct experiments with 5 representations and 4 different problem sizes: $(4, 2)$, $(5, 3)$, $(6, 4)$, and $(7, 5)$. The performances of different representations are shown as box-plots for each problem size in Figures 1 until 4. The combinations in which we were able to obtain the optimal solution (with differential uniformity 6) are denoted with the fitness value of zero.

In the most simple $(4, 2)$ case, the GP, integer, and floating-point representations are always able to reach the optimal

TABLE III: Parameters for permutation encoding

Parameter	Initial value	Tested values	Final value
Population size	200	{100, 200, 500}	200
Mutation rate	0.5	{0.1, 0.3, 0.5, 0.7, 0.9}	0.5

TABLE IV: Parameters for integer encoding

Parameter	Initial value	Tested values	Final value
Population size	200	{100, 200, 500}	200
Mutation rate	0.5	{0.1, 0.3, 0.5, 0.7, 0.9}	0.5

TABLE V: Parameters for floating-point encoding

Parameter	Initial value	Tested values	Final value
Population size	50	{50, 100, 200}	50
Mutation rate	0.5	{0.3, 0.5, 0.7, 0.9}	0.5
Integers encoded with single FP value	1	{1, 2, 4}	**2**

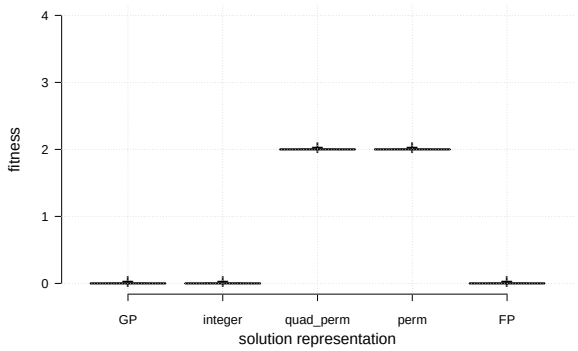

Fig. 1: Results for $(4, 2)$ S-box size

solution (see Figure 1). On the other hand, the permutation and quadruple permutation encoding were unsuccessful in every run. This is an interesting behavior since we know from the exhaustive search it is possible to obtain differentially 6-uniform functions encoded with quadruple permutations. Continuing to that fact, it is intriguing that only the quadruple permutation encoding succeeded in obtaining the optimal solution for the $(5, 3)$ size.

When considering $(5, 3)$ dimension as given in Figure 2, we see that GP encoding works the worst, while quadruple permutation is the only one reaching the global maximum. Floating-point, integer, and permutation encoding all reach the same value and there is no statistically significant difference in their behavior.

In Table VI, we give results for $(5, 3)$ size and quadruple permutation representation, with population size equal to 200 and mutation of 0.9.

For the $(6, 4)$ size we depict results in Figure 3. Here, the integer encoding is the only one reaching differential uniformity equal to 8, while all the other encodings are stuck on differential uniformity of 10.

When examining $(7, 5)$ dimension, we observe even worse results than in the previous cases. This behavior is somewhat

TABLE VI: Results for $(5, 3)$ size and quadruple permutation encoding.

Min	Max	Average	Std dev
0	2	1.68	0.74

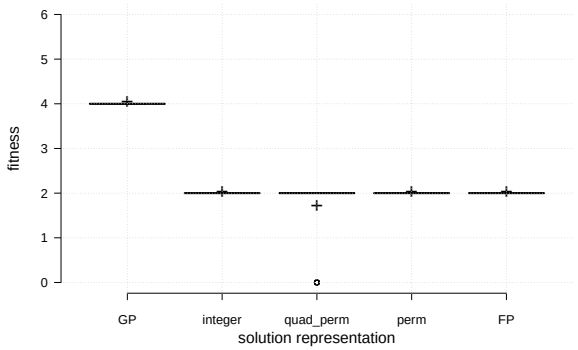

Fig. 2: Results for $(5, 3)$ S-box size

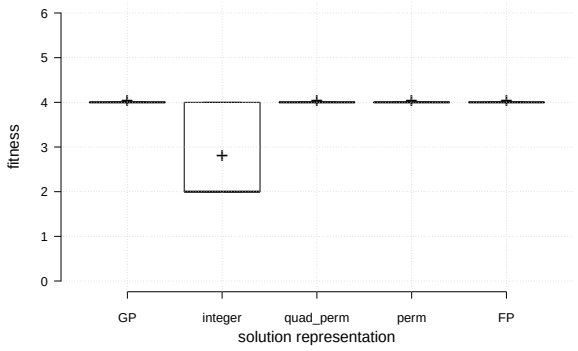

Fig. 3: Results for $(6, 4)$ S-box size

expected since even for the smaller dimension we could not find optimal solutions. Interestingly, again integer encoding gives the best results with differential uniformity going down to 10. Quadruple permutation, permutation, and floating-point behave the same and reach differential uniformity of 12. Finally, GP encoding works the worst with differential uniformity solutions in the range $[12, 14]$. Figure 5 gives the convergence results for the $(7, 5)$ dimension when using the permutation encoding where we depict the averaged Min, Max, and Average values over all experimental runs. As it can be seen, despite having relatively rough grained fitness values, the evolution process still needs more than 150 000 evaluations to find the best solution and more than 800 000 evaluations before is starts stagnating.

The best obtained solutions over all the experiments are then given in Table VII. When a certain encoding is able to reach the global optimum of 0, we depict it in bold style.

The relation between the representations and the obtained

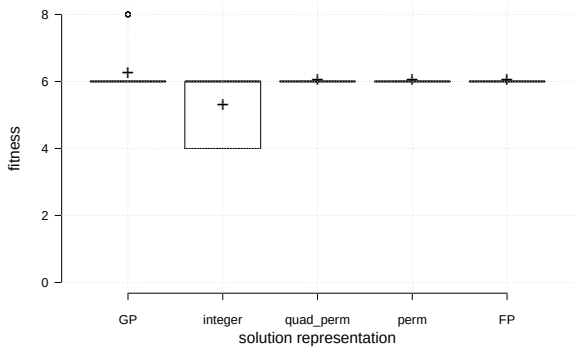

Fig. 4: Results for $(7, 5)$ S-box size

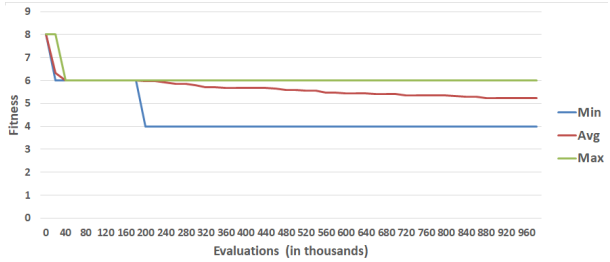

Fig. 5: Convergence for $(7, 5)$ dimension with permutation encoding

differentially 6-uniform solutions is illustrated in Figure 6. Note that the sizes are not given in the actual ratio.

We can observe that evolving differentially 6-uniform $(n, n-2)$ functions is an extremely difficult problem when $n > 5$. What is more, this could be impossible since we have no proofs that such functions even exist. Unfortunately, as is the case with all heuristic techniques, failure to produce optimal results does not mean there are no such results. From that perspective, our experiments did not yield any new information since we still do not know such functions nor do they exist. Since the differential uniformity property can have only even values (i.e., it changes in jumps of two), it is easy to observe that there is not much gradient that could lead the search process.

Still, for the $(5, 3)$ S-box size, we are able to find a number of differentially 6-uniform functions. Since the corpus of

TABLE VII: Best obtained results for all encodings

Encoding \S-box size	(4,2)	(5,3)	(6,4)	(7,5)
Genetic programming	**0**	4	4	6
Permutation	2	2	4	6
Quadruple perm.	2	**0**	4	6
Integer array	**0**	2	2	4
Floating-point array	**0**	2	4	6

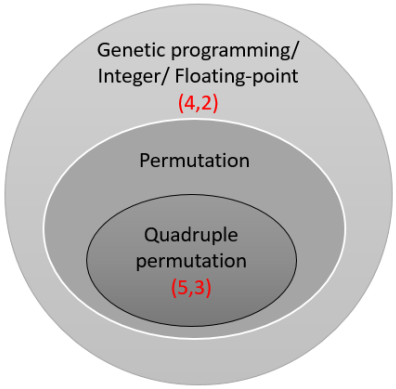

Fig. 6: Illustration of the mapping between representations and problem space

currently known such functions is very limited, it could be expected that at least some of the found functions are new (i.e., not equivalent to the previously known functions).

C. Future Work

There are several options one could follow as research directions. The first, obvious one, is to consider different optimization techniques. Still, since we examined here a number of different representations, we do not expect some other algorithm to perform significantly better. One interesting option would be to use the approach of Picek et al. where they use evolutionary algorithms to evolve cellular automata rules that then map to S-boxes [28]. Since they report unmatched results when considering heuristic design of S-boxes, it is possible the same approach would help here. The problem with their approach is that they essentially consider only a subset of search space that can be described by a single repeated Boolean function. There is no guarantee that within that search subspace there exist differentially 6-uniform $(n, n-2)$ functions. Finally, going to a similar problem, it would be interesting to try to evolve differentially 4-uniform $(n, n-1)$ functions.

V. Conclusion

In this paper, we investigate whether heuristics can be used to generate differentially 6-uniform $(n, n-2)$ functions. The limited theoretical work we have on this topic points us to the conclusion that this problem is of extreme difficulty. Our results confirm the problem to be difficult and evolutionary algorithms to be of only limited success.

We are able to obtain global optimums for only dimensions $(4, 2)$ and $(5, 3)$. Interestingly, different encodings are successful for those two dimensions. Since there exist algebraic constructions able to reach the same results for those two dimensions, we cannot report any new finding. Finally, as it can be observed from the presented box plots, there is not much variety in the obtained solutions. Unfortunately, since the differential uniformity values are always even valued integers, obtaining a more fine-grained fitness function seems to be

difficult. This could indicate that evolutionary algorithms are not the best option for this problem, but still the results are highly competitive with the state-of-the-art. The failure of evolutionary approach for larger dimensions cannot of course be considered as a proof that such functions do not exist when $n \geq 6$. Still, considering that the algebraic constructions also cannot find differentially 6-uniform when $n \geq 6$, we believe it could serve as a strong indication of nonexistence of such functions.

ACKNOWLEDGMENTS

This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882.

REFERENCES

[1] L. R. Knudsen and M. Robshaw, *The Block Cipher Companion.*, ser. Information Security and Cryptography. Springer, 2011.

[2] C. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[3] C. Carlet, "Vectorial Boolean Functions for Cryptography," in *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 1st ed., Y. Crama and P. L. Hammer, Eds. New York, USA: Cambridge University Press, 2010, pp. 398–469.

[4] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[5] 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 *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 450–466.

[6] "FIPS 46-3, Data Encryption Standard (DES)," National Institute for Standards and Technology (NIST), Gaithersburg, MD, USA, 1999.

[7] C. M. Adams, "Constructing symmetric ciphers using the cast design procedure," *Designs, Codes and Cryptography*, vol. 12, no. 3, pp. 283–316, Nov 1997. [Online]. Available: https://doi.org/10.1023/A:1008229029587

[8] M. Matsui and A. Yamagishi, "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*, ser. EUROCRYPT'92. Berlin, Heidelberg: Springer-Verlag, 1993, pp. 81–91. [Online]. Available: http://dl.acm.org/citation.cfm?id=1754948.1754958

[9] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '90. London, UK, UK: Springer-Verlag, 1991, pp. 2–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=646755.705229

[10] L. Mariot, S. Picek, A. Leporati, and D. Jakobovic, "Cellular automata based s-boxes," Cryptology ePrint Archive, Report 2017/1055, 2017, https://eprint.iacr.org/2017/1055.

[11] C. Carlet and Y. Alsalami, "A new construction of differentially 4-uniform $(n, n-1)$-functions," *Advances in Mathematics of Communications*, vol. 9, p. 541, 2015. [Online]. Available: http://aimsciences.org//article/id/c152a42b-4fdc-49f9-b487-20d028044b61

[12] C. Carlet, "Open questions on nonlinearity and on apn functions," in *Arithmetic of Finite Fields*, Ç. K. Koç, S. Mesnager, and E. Savaş, Eds. Cham: Springer International Publishing, 2015, pp. 83–107.

[13] C. Carlet, X. Chen, and L. Qu, "Constructing Infinite Families of Low Differential Uniformity (n, m)-Functions with $m > n/2$," 2018, preprint.

[14] K. Nyberg, "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*, ser. Lecture Notes in Computer Science, vol. 547. Springer, 1991, pp. 378–386.

[15] L. D. Meyer and B. Bilgin, "Classification of balanced quadratic functions," Cryptology ePrint Archive, Report 2018/113, 2018, https://eprint.iacr.org/2018/113.

[16] J. A. Clark, J. L. Jacob, and S. Stepney, "The design of S-boxes by simulated annealing," *New Generation Computing*, vol. 23, no. 3, pp. 219–231, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1007/BF03037656

[17] W. Millan, L. Burnett, G. Carter, A. Clark, and E. Dawson, "Evolutionary Heuristics for Finding Cryptographically Strong S-Boxes," in *Information and Communication Security*, ser. Lecture Notes in Computer Science, V. Varadharajan and Y. Mu, Eds. Springer Berlin Heidelberg, 1999, vol. 1726, pp. 263–274.

[18] L. Burnett, G. Carter, E. Dawson, and W. Millan, "Efficient Methods for Generating MARS-Like S-Boxes," in *Proceedings of the 7th International Workshop on Fast Software Encryption*, ser. FSE '00. London, UK, UK: Springer-Verlag, 2001, pp. 300–314. [Online]. Available: http://dl.acm.org/citation.cfm?id=647935.740914

[19] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O'Connor, M. Peyravian, D. Safford, and N. Zunic, "The MARS Encryption Algorithm," 1999.

[20] P. Tesař, "A New Method for Generating High Non-linearity S-Boxes," *Radioengineering*, vol. 19, no. 1, pp. 23–26, Apr. 2010.

[21] 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*, ser. CS2 '14. New York, NY, USA: ACM, 2014, pp. 13–18.

[22] S. Picek, B. Mazumdar, D. Mukhopadhyay, and L. Batina, "Modified Transparency Order Property: Solution or Just Another Attempt," in *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, 2015, pp. 210–227.

[23] S. Picek, M. Cupic, and L. Rotim, "A New Cost Function for Evolution of S-Boxes," *Evolutionary Computation*, vol. 24, no. 4, pp. 695–718, 2016.

[24] S. Picek, K. Knezevic, and D. Jakobovic, "On the evolution of bent (n, m) functions," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, June 2017, pp. 2137–2144.

[25] D. Jakobovic and et al., "Evolutionary computation framework," 2017. [Online]. Available: http://ecf.zemris.fer.hr/

[26] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

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

[28] S. Picek, L. Mariot, B. Yang, D. Jakobovic, and N. Mentens, "Design of s-boxes defined with cellular automata rules," in *Proceedings of the Computing Frontiers Conference*, ser. CF'17. New York, NY, USA: ACM, 2017, pp. 409–414. [Online]. Available: http://doi.acm.org/10.1145/3075564.3079069