Deriving Smaller Orthogonal Arrays from Bigger Ones with Genetic Algorithms

Luca Mariot[1][0000−0003−3089−6517]

Cyber Security Research Group, Delft University of Technology,
Mekelweg 2, Delft, The Netherlands
l.mariot@tudelft.nl

Abstract. We consider the optimization problem of constructing a binary orthogonal array (OA) starting from a bigger one, by removing a specified amount of lines. In particular, we develop a genetic algorithm (GA) where the underlying chromosomes are constant-weight binary strings that specify the lines to be cancelled from the starting OA. Such chromosomes are then evolved through balanced crossover and mutation operators to preserve the number of ones in them. The fitness function evaluates the matrices obtained from these chromosomes by measuring their distance from satisfying the constraints of an OA smaller than the starting one. We perform a preliminary experimental validation of the proposed genetic algorithm by crafting the initial OA as a random permutation of several blocks of the basic parity-check array, thereby guaranteeing the existence of an optimal solution.

Keywords: orthogonal arrays · genetic algorithms · balanced crossover.

1 Introduction

Orthogonal Arrays (OA) are combinatorial structures that have several applications in cryptography and coding theory, such as secret sharing schemes, stream ciphers and MDS codes [4]. Formally, given a finite set X of $s \in \mathbb{N}$ symbols, an orthogonal array of parameters (N, k, s, t) is a rectangular $N \times k$ matrix with entries from X such that, for any subset of t columns out of k, each of the possible s^t tuples of t bits appears the same number of times $\lambda = N/s^t$. The parameters t and λ are also called respectively the *strength* and the *index* of the OA. Further, an OA is called *simple* if there are no repeated rows in it.

A very interesting application of orthogonal arrays to cryptography is for implementing *masking countermeasures* to side-channel attacks (SCA) [13]. There, the strength parameter t of the OA is related to the order of the SCA that the masking countermeasure can resist. For efficiency reasons, one needs to use an OA as small as possible, i.e. with the smallest possible number of rows N for a given number of columns k and strength t.

From a theoretical standpoint, determining the minimum number of rows admissible for an OA given its strength t is an open problem. The best lower bound in this respect is Delsarte's *linear programming bound* [2], which however is not known to be tight in the general. For example, the question of finding a binary OA of $k = 11$ columns and strength $t = 4$ meeting Delsarte's bound of $N = 96$ rows was open until very recently (see e.g. problem 2.14 in [3]), with Wang [17] proving that the best known example of $N = 128$ rows is the lowest number of admissible rows for that case. Consequently, there is both a theoretical and practical interest in constructing *small* orthogonal arrays. Most of the approaches proposed in the literature, either based on algebraic methods or heuristic techniques, usually aim at constructing an OA from scratch.

On the other hand, in this paper we investigate the opposite direction: *starting from an existing OA, try to derive a smaller one by removing some of its rows.* Clearly, the number of removed rows must be a multiple of s^t, to satisfy the constraint $\lambda = N/s^t$. To this end, we cast the problem in terms of optimization and design a genetic algorithm (GA) for tackling it. The GA takes in input an $\mathrm{OA}(N, k, s, t)$ A and tries to generate a smaller $\mathrm{OA}(N', k, s, t)$ B with $N' < N$ rows. In particular, B is obtained by *cancelling* $p = N - N'$ rows from the original OA A. Hence, each individual I in the GA population represents a possible list of T rows to be cancelled from A. The fitness function to be optimized, taken from [10], measures the deviation from being an $\mathrm{OA}(N', k, s, t)$ of the matrix resulting by cancelling the rows specified by I from A. An optimal solution is thus a list of p rows that, when cancelled from A, results in a $N' \times k$ matrix that satisfies the definition of $\mathrm{OA}(N', k, s, t)$, where $\lambda' = N'/s^t = \lambda - p/s^t$. The GA encodes the candidate solutions by representing them as N-bit strings with t ones, that indicate the rows to be removed from the original matrix. Clearly, this strategy implies that the number of ones in the chromosomes must be kept constant, for which we employ a *balanced* crossover operator investigated in [6] and a simple swap-based mutation operator.

As a preliminary validation, we test our GA on a very simple problem instance, which guarantees by construction the existence of an optimal solution. In particular, we start from the *parity-check array* of order t, which is an $OA(2^t, t+1, 2, t)$ and then repeat it for λ blocks, thereby obtaining an $OA(\lambda 2^t, t+1, 2, t)$. Then, we randomly shuffle the rows of the resulting OA. Given a new index $\lambda' < \lambda$, the task of the GA is therefore to discover a set of $(\lambda - \lambda') \cdot 2^t$ rows so that the resulting array is a shuffled repetition of λ' blocks of the parity-check array. We perform our experiments for strength $t = 4$, observing a steep increase in the difficulty of the problem already for relatively small starting sizes. This preliminary finding prompts for interesting questions to be addressed in future research, such as performing a more systematic parameter tuning phase and analyzing the associated fitness landscape, which could help in tackling also larger problem instances.

The rest of this paper is structured as follows: Section 2 briefly overviews the related work on the construction methods for OA. Section 3 formulates the search of smaller OA by removing rows as an optimization problem. Section 4 details the GA developed for this optimization problem, while Section 5 presents the preliminary experimental investigation performed to validate it. Section 6 concludes the paper and points out some directions for further research.

2 Related Work

Traditional methods for the construction of orthogonal arrays usually rely on the use of *error-correcting codes* or other related algebraic methods. Indeed, the rows of an $\mathrm{OA}(N, k, s, t)$ can be seen as codewords of an error-correcting codes, where the minimum distance is related to the strength of the OA. An excellent survey of the existing constructions based on these approaches is the book by Hedayat et al. [4]. On the other hand, the literature concerning the use of heuristic optimization techniques for constructing OA, as well as other types of combinatorial designs, is much more limited.

To the best of our knowledge, Safadi and Wang [16, 18] were the first to propose the use respectively of genetic algorithms and simulated annealing for designing *mixed-level* OA, where each column can have entries from sets of different size. The authors of [15] proposed a memetic algorithm for constructing *covering arrays*, which are a generalization of OA where each t-uple must occur *at least* λ times in each subset of t columns. Mariot et al. [9] considered the problem of evolving orthogonal Latin squares (which are equivalent to OA of strength 2) defined by cellular automata rules using genetic algorithms and genetic programming (GP). Later, the same authors in [10] addressed the design of binary orthogonal arrays with GA and GP.

Binary OA are also equivalent to *correlation-immune* Boolean functions, which play an important role in symmetric cryptography [1]. Hence, all works considering the design of correlation-immune Boolean functions with evolutionary algorithms can be seen as addressing the same problem as evolving binary OA. This includes for instance the work by Picek et al. [13], where the authors employed GA and various breeds of GP to evolve correlation-immune functions of high order and minimal Hamming weight. Other works such as Mariot and Leporati [8, 7] and Picek et al. [12, 14] tackled the design of Boolean functions satisfying several cryptographic properties of interest, among which correlation-immunity, using various evolutionary and swarm intelligence methods including GA, discrete particle swarm optimization (PSO) and Cartesian GP.

3 Removing OA Rows as an Optimization Problem

We now define the task of removing rows from an initial OA to obtain a smaller one as an optimization problem. In what follows, we will denote an OA as a set of N vectors over the set X of length k each. Indeed, the order of the rows in an OA is not important, since it does not influence the balancedness constraint. Let $A = \{r_1, \cdots, r_n\}$ be an $\mathrm{OA}(N, k, s, t)$ where $r_i \in X^k$ for all $i \in \{1, \cdots, N\}$, and let $\lambda = N/s^t$ be the index of A. Given a smaller index $\lambda' < \lambda$ and a set of p rows $T = \{i_1, \cdots, i_p\}$, with $i_j \in \{1, \cdots, N\}$ for all $j \in \{1, \cdots, p\}$, define the new array B as follows:

$$B = A \setminus \{r_{i_1}, \cdots, r_{i_p}\} \ . \tag{1}$$

In other words, B is the array obtained by removing the rows specified by T from A. Although being an $N' \times k$ binary array with $N' = \lambda' \cdot s^t$, in general B

will not satisfy the property of an $\mathrm{OA}(N', k, s, t)$. Therefore, we can state our optimization problem of interest as follows:

Problem 1. Let A be an $\mathrm{OA}(N, k, s, t)$ with $\lambda = N/s^t$, and let $\lambda' < \lambda$. Find a set $T = \{i_1, \cdots, i_p\}$ of $p = (\lambda - \lambda') \cdot s^t$ rows such that the array B as defined in Eq. (1) is an $OA(N', k, s, t)$, with $N' = \lambda' s^t$.

To measure the fitness of a candidate solution T to Problem 1, we employ the same fitness function defined in [10] for evolving binary OA. The idea is to compute the *Minkowski distance* of the vector of occurrences of each t-uple from the vector $(\lambda', \lambda', \cdots, \lambda')$. In particular, the array B resulting from the removal of the rows specified by T will be an $\mathrm{OA}(N', k, s, t)$ if and only if such distance is 0. Therefore, the optimization objective is to *minimize* the fitness function. Due to the lack of space, we refer the reader to [10] for the formal definition of the fitness function. In all our experiments, we used the Minkowsky distance with exponent 2, which basically corresponds to the Euclidean distance.

4 Genetic Algorithm

In order to design a genetic algorithm for tackling Problem 1, one first needs to define the chromosome encoding of the candidate solutions, which in our cases are lists of rows to be removed from the original OA. In set-theoretic terms, the most straightforward way is to use a binary vector that represents the characteristic function of the set $T = \{i_1, i_2, \cdots, i_p\}$. Thus, given the orthogonal array A of N rows, the chromosome C_T of a candidate solution T is a binary string of length N whose coordinates are defined as:

$$C_t[i] = \begin{cases} 1, & \text{if } i \in T \ , \\ 0, & \text{otherwise} \end{cases} , \tag{2}$$

for all $i \in \{1, \cdots, N\}$. In particular, each chromosome has a constant *Hamming weight*, that is the number of ones in it is always fixed to p, which corresponds to the size of the candidate solution T. Clearly, the constraint above raises the question of how to make the genetic algorithm preserve the number of ones in the offspring chromosomes, so that they always represent a valid list of p rows to be removed from the original OA. To this end, we respectively used a *balanced operator* for crossover and a *swap-based operator* for mutation.

Such operators have been mostly investigated in the literature related to the optimization of Boolean functions with good cryptographic properties. Since one of the basic properties that such functions must satisfy to be used in a stream or block cipher is to be *balanced* (i.e., having the same number of ones and zeros in their truth tables), several researchers investigated the design of ad-hoc crossover operator to ensure this constraint, in order to reduce the size of the search space explored by a GA. In particular, Millan et al. [11] proposed a crossover operator where two counters are used to keep trace of the multiplicities of zeros and ones while the offspring chromosome is being created from the parents. Once one of the counters reaches the prescribed threshold, the offspring is filled with the complementary value in the remaining positions. Other works [7, 9, 5] considered variations of this operator over non-binary strings and for other problems. Manzoni et al. [6] performed a systematic comparison of three different balanced crossover operators, observing that the *map-of-ones* operator usually performs better. For this reason, we adopted it for our GA. The idea of this crossover is to represent the two parent strings in terms of their map of ones, that is the list of positions in the strings where the ones occur. Then, the offspring map is constructed by randomly copying from the parents' maps, checking consistency to avoid that duplicate positions are inserted in the offspring. In this way, starting from two parents that have the same Hamming weight (that is, map of ones of the same length), the child chromosome is guaranteed to inherit the same weight.

Regarding the mutation phase, instead, we opted for the simple operator used in [6], which randomly selects a pair of positions in the bitstrings holding different values and swap them.

5 Experiments

As a preliminary assessment of our GA for Problem 1, we adopted the following experimental setup considering only the case of binary OA (hence, with $s = 2$ and $X = \mathbb{F}_2$). For the initial OA, we started with the *parity-check* array as a basic building block, also called the zero-sum array in [4]. The parity-check array P of order t is defined as a $(2^t) \times (t+1)$ binary matrix where the first t columns holds all 2^t binary vectors in $\mathbb{F}_2^t$ (for example, in lexicographic order). The last column, on the other hand, contains the results of the XOR of the bits in the previous columns. It is rather easy to prove that such an array P is an $\mathrm{OA}(2^t, t+1, 2, t)$. Then, given the desired starting index λ, we constructed a $(\lambda 2^t) \times (t+1)$ binary array by repeating λ times the OA P, randomly shuffling its rows at the end. Obviously, the resulting array is an $\mathrm{OA}(\lambda \cdot 2^t, t+1, 2, t)$, which is not simple since each rows occurs exactly λ times.

For our experiments, we considered the case of strength $t = 4$, with the index λ for the initial OA ranging between 2 and 4. Hence, the smallest problem instance consisted in starting from an $\mathrm{OA}(32, 5, 2, 4)$ ($\lambda = 2$) and finding a subset of 16 rows such that the reduced matrix is an $\mathrm{OA}(16, 5, 2, 4)$ ($\lambda' = 1$). In this case, the size of the search space is $\binom{32}{16} = 6.32 \cdot 10^8$, which in principle is still amenable to exhaustive search, but nonetheless provides an interesting case to gauge the performances of our GA. The largest instance, on the other hand, was to start from an $\mathrm{OA}(64, 5, 2, 4)$ ($\lambda = 4$) and find a subset of 32 rows to erase in order to obtain an $\mathrm{OA}(32, 5, 2, 4)$ ($\lambda' = 2$). In this case the search space size is $\binom{64}{32} \approx 1.82 \cdot 10^{18}$, which cannot be exhaustively searched.

Regarding the parameters of our GA, we drew upon those used in [10], which also targeted the construction of binary OA: steady-state selection with tournament size 3, population size of 500 individuals, mutation probability 0.2, and a fitness budget of 100 000 evaluations. Finally, we repeated each experiment for 30 independent runs to obtain statistically significant results.

λ, λ'	1	2	3
2	(24/30, 0.0)	–	–
3	(9/30, 7.07)	(4/30, 7.07)	–
4	(8/30, 7.07)	(0/30, 7.07)	(8/30, 7.07)

Table 1. Number of optimal solutions and median fitness over all problem instances.

Table 5 reports, for each of the considered 6 problem instances, the number of optimal solutions found over the 30 independent runs and the median fitness value of the best solution evolved by the GA. It can be observed that there is a steep degradation in performances as soon as one leaves the smallest problem instance with $\lambda = 2$ and $\lambda' = 1$. Indeed, while in this case the GA almost always converges to an $\mathrm{OA}(16, 5, 2, 4)$ starting from an $\mathrm{OA}(32, 5, 2, 4)$, the situation worsens already for $\lambda = 3$ with only 9 and 4 optimal solutions found respectively for $\lambda' = 1$ and $\lambda' = 2$. Over the largest problem instance, i.e. $\lambda = 4$ and $\lambda' = 2$, the GA never converges to an optimal solution. It is also interesting to note that, except for the smallest problem instance, the median fitness is always the same. In fact, we found that the fitness distribution for the final best individual is actually bi-modal over all problem instances tackled by our GA, with the only observed values being 0.0 and 7.07.

6 Conclusions

In this paper, we proposed a genetic algorithm to evolve orthogonal arrays with small index starting from bigger ones. The basic idea is to represent a candidate solution as a set of rows to be removed from the original arrays, which are represented by bitstrings with a constant number of ones. The GA then evolves such bitstrings by using ad-hoc crossover and mutation operators that preserve the Hamming weight of the strings.

The preliminary results gathered in our investigation show that this optimization problem seems to be extremely difficult for GA: indeed, already for small instances such as removing 32 rows from an $\mathrm{OA}(64, 5, 2, 4)$, our GA was not able to produce any optimal solution. Also, we noticed that all obtained distributions of the best fitness are bi-modal. These empirical observations indicate that, in future research, two directions should be particularly considered: first, analyzing the fitness landscape of this optimization problem might help in understanding why the GA gets stuck in the local optima with fitness value 7.07. As a matter of fact, it would be interesting to analyze the solutions to which our GA converges, to verify whether it is always the same, or if several different solutions with the same sub-optimal fitness exist. Second, we suspect that a more systematic parameter tuning phase could benefit the performances of our GA on the larger problem instances.

References

1. Carlet, C.: Boolean functions for cryptography and coding theory (2021)
2. Delsarte, P.: An algebraic approach to the association schemes of coding theory. Philips Res. Rep. Suppl. **10**, vi+–97 (1973)
3. Gorodilova, A., Agievich, S., Carlet, C., Hou, X., Idrisova, V., Kolomeec, N., Kutsenko, A., Mariot, L., Oblaukhov, A., Picek, S., Preneel, B., Rosie, R., Tokareva, N.N.: The fifth international students' olympiad in cryptography - NSUCRYPTO: problems and their solutions. Cryptologia **44**(3), 223–256 (2020)
4. Hedayat, A.S., Sloane, N.J.A., Stufken, J.: Orthogonal arrays: theory and applications. Springer Science & Business Media (2012)
5. Manzoni, L., Mariot, L., Tuba, E.: Does constraining the search space of GA always help?: the case of balanced crossover operators. In: López-Ibáñez, M., Auger, A., Stützle, T. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019. pp. 151–152. ACM (2019)
6. Manzoni, L., Mariot, L., Tuba, E.: Balanced crossover operators in genetic algorithms. Swarm Evol. Comput. **54**, 100646 (2020)
7. Mariot, L., Leporati, A.: A genetic algorithm for evolving plateaued cryptographic boolean functions. In: Dediu, A., Magdalena, L., Martín-Vide, C. (eds.) Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9477, pp. 33–45. Springer (2015)
8. Mariot, L., Leporati, A.: Heuristic search by particle swarm optimization of boolean functions for cryptographic applications. In: Silva, S., Esparcia-Alcázar, A.I. (eds.) Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings. pp. 1425–1426. ACM (2015)
9. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: Bosman, P.A.N. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017. pp. 306–313. ACM (2017)
10. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary search of binary orthogonal arrays. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, L.D. (eds.) Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11101, pp. 121–133. Springer (2018)
11. Millan, W., Clark, A.J., Dawson, E.: Heuristic design of cryptographically strong balanced boolean functions. In: Nyberg, K. (ed.) Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. Lecture Notes in Computer Science, vol. 1403, pp. 489–499. Springer (1998)
12. Picek, S., Carlet, C., Guilley, S., Miller, J.F., Jakobovic, D.: Evolutionary algorithms for boolean functions in diverse domains of cryptography. Evol. Comput. **24**(4), 667–694 (2016)
13. Picek, S., Guilley, S., Carlet, C., Jakobovic, D., Miller, J.F.: Evolutionary approach for finding correlation immune boolean functions of order t with minimal hamming weight. In: Dediu, A., Magdalena, L., Martín-Vide, C. (eds.) Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9477, pp. 71–82. Springer (2015)

14. Picek, S., Jakobovic, D., Miller, J.F., Batina, L., Cupic, M.: Cryptographic boolean functions: One output, many design criteria. Appl. Soft Comput. **40**, 635–653 (2016)
15. Rodriguez-Tello, E., Torres-Jimenez, J.: Memetic algorithms for constructing binary covering arrays of strength three. In: International Conference on Artificial Evolution (Evolution Artificielle). pp. 86–97. Springer (2009)
16. Safadi, R., Wang, R.: The use of genetic algorithms in the construction of mixed multilevel orthogonal arrays. Tech. rep., Olin Corp Cheshire CT Olin Research Center (1992)
17. Wang, Q.: Hadamard matrices, d-linearly independent sets and correlation-immune boolean functions with minimum hamming weights. Des. Codes Cryptogr. **87**(10), 2321–2333 (2019)
18. Wang, R., Safadi, R.: Generating mixed multilevel orthogonal arrays by simulated annealing. In: Computing Science and Statistics, pp. 557–560. Springer (1992)