Natural Computing (2022) 21:377–391
https://doi.org/10.1007/s11047-022-09885-3

Heuristic search of (semi-)bent functions based on cellular automata

Luca Mariot[1] · **Martina Saletta**[2] · **Alberto Leporati**[2] · **Luca Manzoni**[3]

Accepted: 26 April 2022 / Published online: 20 May 2022
© The Author(s) 2022

Abstract
An interesting thread in the research of Boolean functions for cryptography and coding theory is the study of *secondary constructions*: given a known function with a good cryptographic profile, the aim is to extend it to a (usually larger) function possessing analogous properties. In this work, we continue the investigation of a secondary construction based on cellular automata (CA), focusing on the classes of bent and semi-bent functions. We prove that our construction preserves the algebraic degree of the local rule, and we narrow our attention to the subclass of quadratic functions, performing several experiments based on exhaustive combinatorial search and heuristic optimization through Evolutionary Strategies (ES). Finally, we classify the obtained results up to permutation equivalence, remarking that the number of equivalence classes that our CA-XOR construction can successfully extend grows very quickly with respect to the CA diameter.

Keywords Cellular automata · Symmetric cryptography · Bent functions · Nonlinearity · Combinatorial search · Evolutionary strategies

Mathematics Subject Classification 94A60 · 06E30 · 68Q80

1 Introduction

The design of symmetric ciphers traditionally revolves around the concepts of confusion and diffusion, introduced by Shannon (1949) as a general guideline to develop encryption functions that are able to frustrate statistical attacks. Specifically, *confusion* prescribes that the relationship between the ciphertext and the encryption key should be as complicated as possible, while the aim of *diffusion* is to spread the statistical structure of the plaintext over the ciphertext—or equivalently, to make the value of every ciphertext bit depend on many plaintext bits (ideally all of them).

The study of diffusion layers in block ciphers mainly leverage on the theory of error-correcting codes to come up with transformations that propagate the differences of very similar plaintexts in an optimal way (see e.g. the MixColumns operation in AES Daemen and Rijmen 2020, which relies on an MDS matrix). On the other hand, *Boolean functions* (i.e., mappings of the type $f : \{0,1\}^n \rightarrow \{0,1\}$) with specific properties are usually sought to design confusion layers, for instance in the form of *combining* or *filtering* functions in stream ciphers (Carlet 2021). In block ciphers, *S-boxes* (i.e., the vectorial generalization of Boolean functions) are used to design the confusion layer instead, as in the *Substitution-Permutation Network* paradigm (Stinson and Paterson 2018). The rationale behind choosing a particular Boolean function when designing a symmetric cipher usually boils down to check several *cryptographic properties*, which are aimed at countering specific attacks.

For the above reason, research in the past decades focused on methods for the construction of Boolean

✉ Luca Mariot
luca.mariot@ru.nl

Martina Saletta
martina.saletta@unimib.it

Alberto Leporati
alberto.leporati@unimib.it

Luca Manzoni
lmanzoni@units.it

[1] Digital Security Group, Radboud University, PO Bus 9010, 6500 GL Nijmegen, The Netherlands

[2] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336/14, 20126 Milan, Italy

[3] Dipartimento di Matematica e Geoscienze, Università degli Studi di Trieste, Via Valerio 12/1, 34127 Trieste, Italy

functions with good cryptographic properties, about which the reader can find an excellent account in Carlet's recent book (2021). These methods are usually classified in two approaches. In *primary constructions*, one seeks to construct Boolean functions with certain properties starting "from scratch", i.e., by leveraging on other combinatorial objects such as permutations (see the *Maiorana–McFarland construction* McFarland 1973) or finite vector spaces (as in Dillon's *partial spread construction* 1974). On the other hand, a *secondary construction* starts from an existing Boolean function with a certain cryptographic profile, and constructs a new one (usually over a larger number of variables) possessing similar properties. The best known example in this respect is *Rothaus's construction* (1976).

Cellular automata (CA) have been extensively investigated as a computational building block for designing several cryptographic primitives. The most famous examples include Wolfram's pseudorandom number generator for Vernam-like stream ciphers (Wolfram 1986), which was based on the chaotic dynamics of the elementary local rule 30 (later shown to be vulnerable to correlation and approximation attacks Meier and Staffelbach 1991; Koc and Apohan 1997), and the χ nonlinear transformation by Daemen et al. (1994), used as an S-box in Keccak (Bertoni et al. 2011) and other symmetric ciphers. As far as we know, most of these works focused on the study of certain local rules that, when plugged into a CA, would yield good cryptographic functions. Such studies, however, usually rely on ad hoc arguments that mostly depend on the specificities of the single rules, and are hardly generalizable. To the best of our knowledge, there have been no significant attempts to use CA as a general method to define primary or secondary constructions of Boolean functions with good cryptographic properties. The only exception we are aware of is the use of linear bipermutive CA as a primary construction for *bent* Boolean functions belonging to the partial spreads class (Gadouleau et al. 2020), recently proposed by the first author of this paper together with M. Gadouleau and S. Picek.

In this paper, we continue the investigation of a secondary construction of Boolean functions set forth in the paper "*Exploring Semi-bent Boolean Functions Arising from Cellular Automata*" (Mariot et al. 2020b), presented at ACRI 2020. The idea underlying the construction, which we name the *CA-XOR construction*, is quite simple: a CA of n cells is evolved for a single time step, and the XOR of the output cells is taken as the value of a Boolean function of n variables, where the CA initial configuration constitutes the input vector. The properties of the resulting function depend on the underlying local rule used to evolve the CA, which is itself a Boolean function of $d \leq n$ variables. The goal is thus to start from a Boolean function with good cryptographic properties such that, when used as a local rule in the CA-XOR construction, results in a larger function with similar good properties. Hence, we actually obtain a *recursive* secondary construction with our method: indeed, one can define an infinite family of functions by just adding enough cells to the CA. In particular, given a starting function of d variables, it is possible to define a new function for each $n > d$.

The present manuscript extends the work in Mariot et al. (2020b) along several directions:

- We enlarge the scope of our investigation by considering also bent functions, instead of only semi-bent ones as in the original paper. Since bent functions exist only when the number of variables n is even, we focus our attention on those (semi-)bent functions that, when used as a starting point in our CA-XOR construction, always yield bent functions when the number of CA cells n is even, and semi-bent functions when n is odd.
- We refine the computational search approach laid down in Mariot et al. (2020b). By leveraging on some known results on quadratic bent functions, we simplify the combinatorial algorithm by enumerating only the set of *homogeneous* algebraic normal forms (ANFs) of degree 2, i.e., those without linear terms. This allows us to extend our exhaustive search experiments up to diameter $d = 7$.
- For higher diameters where exhaustive search becomes unfeasible, we devise an *Evolutionary Strategies* (ES) optimization algorithm. Exploiting the fact that there exists only one (semi-)bent Boolean function of d variables up to *Extended Affine* (EA)-*equivalence*, we use ES to evolve affine transformations applied to this function. The optimization objective thus becomes to find a suitable affine transformation such that the resulting quadratic function can be extended by the CA-XOR construction up to $n = 16$ cells.
- Finally, we gather all the functions obtained through the exhaustive search and ES optimization experiments, and classify them up to *permutation equivalence* (Carlet 2021). We report the numbers of the obtained equivalence classes, observing that they grow very quickly with respect to the diameter of the CA.

The remainder of this paper is structured as follows. Section 2 recalls the basic definitions and results concerning Boolean functions and CA used throughout the paper. Section 3 gives a general overview of the literature concerning the use of CA in symmetric cryptography, and covers the main works related to the algebraic and heuristic constructions of Boolean functions with good cryptographic properties. Section 4 introduces the CA model considered in this work and defines the CA-XOR construction of Boolean functions, proving some basic theoretical facts about it. Section 5 describes the search algorithm used to exhaustively enumerate quadratic functions with a homogeneous ANF, while Sect. 6 is devoted to the ES optimization algorithms used to evolve quadratic functions on larger diameters. Section 7 presents the results of our experiments, providing a classification of the obtained functions up to permutation equivalence. Section 8 summarizes the key contributions of the paper, and discusses several open problems for future research on the subject.

2 Preliminary definitions

In this section, we gather all background definitions related to Boolean functions and CA used in the following. As a general notation, let $\mathbb{F}_2 = \{0, 1\}$ be the finite field of two elements, where the sum operation is the XOR (denoted by $\oplus$) while multiplication is the logical AND (denoted by simple concatenation). The n-dimensional vector space over $\mathbb{F}_2$ (i.e., the set of all n-bit vectors) is denoted $\mathbb{F}_2^n$. The *support* of $x \in \mathbb{F}_2^n$ is defined as $supp(x) = \{i : x_i \neq 0\}$, while the *Hamming weight* of x is $w_H(x) = |supp(x)|$, i.e., the number of 1s in x. The vector space $\mathbb{F}_2^n$ is usually equipped with the *scalar product* defined as $a \cdot x = \bigoplus_{i=1}^n a_i x_i$ for all $a, x \in \mathbb{F}_2^n$. Further, for all $n \in \mathbb{N}$ we denote by $[n] = \{1, \ldots, n\}$ the set of the first n positive integer numbers.

2.1 Boolean functions

The body of literature devoted to Boolean functions for cryptography and coding theory is quite extensive. In this section, we only recall the essential notions related to the main cryptographic properties of Boolean functions that we will use in the rest of the paper, referring the reader to Carlet (2021) for a more thorough treatment of the topic.

A *Boolean function* of $n \in \mathbb{N}$ variables is a mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$, i.e., a function from the set of n-bit vectors to a single bit. There exist several ways to represent Boolean functions, the simplest one being the *truth table*. Assuming that $\mathbb{F}_2^n$ is endowed with a certain total order (e.g., the lexicographic order), the truth table of a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the 2^n-bit vector $\Omega_f \in \mathbb{F}_2^{2^n}$ that specifies the output value of f for each of the vectors in $\mathbb{F}_2^n$. The *weight* of f is defined as the Hamming weight of its truth table Ω_f. Further, function f is called *balanced* if $w_H(\Omega_f) = 2^{n-2}$, that is, if its truth table is composed of an equal number of 0s and 1s. Balancedness is a fundamental cryptographic property for Boolean functions used in stream and block ciphers.

Although being the simplest representation, most of the cryptographic properties of a Boolean function are difficult to define in terms of its truth table. To this end, another more useful unique representation is the *Algebraic Normal Form* (ANF). Remarking that $x^2 = x$ for all $x \in \mathbb{F}_2$, the ANF of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as the following multivariate polynomial over the quotient ring $\mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2 \oplus x_1, \ldots, x_n^2 \oplus x_n)$:

$$P_f(x) = \bigoplus_{I \in 2^{[n]}} a_I \left(\prod_{i \in I} x_i \right), \tag{1}$$

where $2^{[n]}$ is the power set of $[n] = \{1, \ldots, n\}$. The *algebraic degree* of f is the cardinality of the largest subset $I \in 2^{[n]}$ in its ANF such that $a_I \neq 0$, or equivalently the size of its largest nonzero monomial. As a cryptographic criterion, the algebraic degree should be as high as possible. *Affine functions* are defined as those Boolean functions with degree at most 1. In particular, the ANF of an affine function is defined as a scalar product between a fixed vector $a \in \mathbb{F}_2^n$ and the input vector $x \in \mathbb{F}_2^n$, plus a constant $b \in \mathbb{F}_2$, i.e., $A_{a,b}(x) = a \cdot x \oplus b$. If $b = 0$, then the function is called *linear*. Thus, there exist 2^n linear functions of n variables. In a similar way, *quadratic functions* are the Boolean functions of degree at most 2. The fact that "at most" is used instead of "exactly" is for simplicity of language when talking about the derivatives of Boolean functions (see Carlet 2021, p. 53 for further explanations). Further, the set of all Boolean functions of degree at most d has a vector space structure, since it is closed under sum.

The vector of the ANF coefficients a_I and the truth table of f are related through the *Möbius transform*:

$$f(x) = \bigoplus_{I \in 2^{[n]} : I \subseteq supp(x)} a_I. \tag{2}$$

In particular, the *Möbius transform* is an *involution*, meaning that one can apply the same formula in Eq. (2) to recover the vector of ANF coefficients from the truth table of a function.

A third way to uniquely represent Boolean functions is the *Walsh transform*, which captures several cryptographic properties. Formally, the Walsh transform of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined for all $a \in \mathbb{F}_2^n$ as:

$$W_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus a \cdot x}. \tag{3}$$

A function f is balanced if and only if the Walsh coefficient over the null vector is zero, i.e., $W_f(0) = 0$. More in general, the coefficient $W_f(a)$ measures the *correlation* between f and the linear function $a \cdot x$. Thus, the Walsh transform can be used to compute the *nonlinearity* of a Boolean function f, which is defined as the minimum Hamming distance of (the truth table of) f from the set of all affine functions. More precisely, the nonlinearity of f equals

$$N_f = 2^{n-1} - \frac{1}{2} \cdot \max_{a \in \mathbb{F}_2^n}\{|W_f(a)|\}. \quad (4)$$

The nonlinearity of Boolean functions used in the combiner and filter model of stream ciphers should be as high as possible (Carlet 2021). From Eq. (4), this means that the maximum absolute value of the Walsh transform should be as low as possible. By *Parseval relation*, this can happen only when all Walsh coefficients have the same absolute value $2^{\frac{n}{2}}$, yielding the *covering radius bound*: $N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}$. Functions satisfying this bound are called *bent*, and they exist only when n is even (since the Walsh transform can yield only integer numbers). Such functions are not balanced, since $W_f(0) = \pm 2^{\frac{n}{2}}$, and thus they cannot be used directly in the design of stream and block ciphers. However, there exist several ways to construct highly nonlinear balanced functions from bent ones (such as, for example, XORing the value of the function with an independent additional variable). For n odd, the *quadratic bound* is given by $N_f \leq 2^{n-1} - 2^{\frac{n+1}{2}-1}$. The name of this bound comes from the fact that it can be always achieved by functions of algebraic degree 2. In general, finding the highest possible nonlinearity between the quadratic and the covering radius bound is an open problem for $n > 7$ odd.

Plateaued functions represent an interesting generalization of bent functions, since they can also be balanced while still retaining high nonlinearity. Formally, a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is *plateaued* if its Walsh transform takes only three values, i.e., if $W_f(a) \in \{-\lambda, 0, +\lambda\}$ for all $a \in \mathbb{F}_2^n$. In particular, a plateaued function is *semi-bent* if $\lambda = 2^{\frac{n+1}{2}}$ for n odd and $\lambda = 2^{\frac{n+2}{2}}$ for n even. This means that the nonlinearity of a semi-bent function equals $2^{n-1} - 2^{\frac{n-1}{2}}$ when n is odd and $2^{n-1} - 2^{\frac{n}{2}}$ when n is even. Hence, semi-bent functions reach the quadratic bound for nonlinearity when n is odd.

We conclude this section by giving a short overview of the *equivalence relations* studied in the context of Boolean functions. The size of the space of n-variable Boolean functions is 2^{2^n}, i.e., it is superexponential in n. Therefore, one cannot perform an exhaustive search of all Boolean functions to find those with the desired cryptographic properties for a specific application already for $n > 5$. Using equivalence relations that preserve these properties helps in reducing the search space size, by looking only at the set of equivalence classes. The coarsest equivalence relation used for Boolean functions is *permutation equivalence*. Namely, two functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ are called permutation equivalent if there exists a permutation $\pi : [n] \to [n]$ such that

$$f(x_1, x_2, \ldots, x_n) = g(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}) \quad (5)$$

for all $x \in \mathbb{F}_2^n$. In simpler terms, f and g are permutation equivalent if it is possible to reorder the input variables of g to match the output of f. A more general relation, which we will use extensively in this paper, is affine equivalence. In this case, we say that f and g are affinely equivalent if there exist an invertible $n \times n$ binary matrix M and a vector $v \in \mathbb{F}_2^n$ such that

$$f(x) = g(Mx^\top \oplus v) \quad (6)$$

for all $x \in \mathbb{F}_2^n$. In this case, the two functions are equivalent if by applying the affine transformation $Mx^\top \oplus v$ to the input vector x and then evaluating g on the result, one gets the output of f computed directly on x. When $v = 0$ is the null vector we say that the two functions are *linearly equivalent*. Remark that permutation equivalence is a special case of linear equivalence, since one can define a permutation of the input variables in terms of a *permutation matrix*. Concerning the cryptographic properties, affine equivalence (and therefore, linear and permutation equivalence which are special cases) preserves the algebraic degree and the nonlinearity.

Two further equivalence relations are usually studied in the context of Boolean functions, namely *Extended Affine* (EA) equivalence and *Carlet–Charpin–Zinoviev* (CCZ) equivalence. However, we will not cover them here, since CCZ equivalence coincides with EA equivalence for single-output Boolean functions (i.e., it is relevant only for vectorial Boolean functions, or S-boxes), while EA-equivalence does not preserve the algebraic degree. In particular, we are interested in the theorem below, whose proof can be found in Carlet (2021):

Theorem 1 *The following results hold*:

- If n is even, every bent quadratic function of n variables is affinely equivalent to the function
$$b(x) = x_1x_2 \oplus x_2x_3 \oplus \cdots \oplus x_{n-1}x_n. \quad (7)$$
- If n is odd, every semi-bent quadratic function of n variables is affinely equivalent to the function
$$s(x) = x_1x_2 \oplus x_2x_3 \oplus \cdots \oplus x_{n-2}x_{n-1} \oplus x_n. \quad (8)$$

Therefore, there exists only one (semi-)bent function of n variables up to affine equivalence, whose ANF is the sum of all separate binomials in increasing order (excluding the last variable which is summed independently for semi-bent functions on an odd number of variables).

2.2 Cellular automata

CA are one of the oldest nature-inspired computational models studied in the literature. Indeed, they were initially considered by Ulam (1952) and Von Neumann (1966) in the 1950–1960s to investigate self-replication phenomena. Nowadays, CA are applied both as a simulation tool for complex systems and as a computing device in the most disparate domains, ranging from physics to ecology. In essence, a CA is characterized by a regular lattice of *cells* which can be in a discrete set of values. The global state of the system is updated by the synchronous application of a *shift-invariant transformation*. In other words, each cell applies the same *local rule* in parallel to decide its next state. Such rule is evaluated on the configuration of states formed by the cell itself and its neighboring cells. The size of the neighborhood is determined by the *diameter* of the CA.

Usually, research on CA focuses on the study of their *long-term* dynamic behavior, which emerges from repeatedly applying the local rule on each cell for multiple time steps. On the contrary, in this work we adopt a different viewpoint: since we are mainly interested in CA as a layer to implement meaningful cryptographic primitives, we focus only on their *short-term* behavior. This implies, in particular, analyzing the algebraic properties of the functions induced by the application of the CA local rule over a *single* time step. Although there are approaches in the literature that consider the synthesis of Boolean circuits by evolving the CA for multiple steps (see e.g. Hazari et al. 2018), we decided to employ this setting for two reasons: first, it greatly simplifies the theoretical analysis of the cryptographic primitives defined by CA. Second, several examples of symmetric ciphers that exploits CA adopt this design direction. In particular, past experience seems to indicate that the best way to employ CA in a symmetric cipher is as nonlinear transformations applied for a single time step, and combining their output with that of other non-CA components to improve diffusion. The best known example of this design pattern is the χ transformation used in Keccak (Bertoni et al. 2011), which is now part of the SHA-3 standard for cryptographic hash functions.

We now introduce the basic CA model considered in the rest of this work.

Definition 1 A *No-Boundary Cellular Automaton* (NBCA) of length n and diameter $d \leq n$ is a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ defined for all $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$ as:

$$F(x_1, x_2, \ldots, x_n) = (f(x_1, \ldots, x_d), \ldots, f(x_{n-d+1}, \ldots, x_n)), \tag{9}$$

where $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is a Boolean function of d variables called the CA *local rule*.

Hence, a NBCA can be seen as a lattice of n cells arranged over a line, where each cell $i \in [n-d+1]$ computes its output state by applying the local rule f on itself and the $d-1$ right neighbors $\{i+1, \ldots, i+d-1\}$. The rightmost $d-1$ cells are effectively lost after the application of the global rule F, since they do not have enough right neighbors to compute their next state. Although there are methods in the literature to address this issue (e.g. with *null* or *periodic* boundary conditions), in this work we are not interested in keeping the number of cells constant, since as we argued above we only consider a single application of the CA global rule F. Remark that the NBCA model as stated in Definition 1 has also been studied for other cryptographic applications, namely in Mariot et al. (2019) for CA-based S-boxes, and in Mariot et al. (2020a) for mutually orthogonal Latin squares.

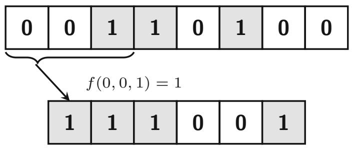
Since the local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is a Boolean function, it can be defined by a truth table Ω_f of 2^d bits. In the CA literature, the truth table of a local rule is usually represented by its *Wolfram code* (1983), which corresponds to the decimal encoding of Ω_f. Figure 1 reports an example of CA of length 8 and diameter d. The local rule, defined as $f(x_i, x_{i+1}, x_{i+2}) = x_i \oplus x_{i+2}$, has Wolfram code 90.

3 Related works

As we mentioned in the Introduction of this paper, CA have a long history of being investigated for cryptographic applications. The reason for this interest is grounded in two main observations. First, the *shift-invariance* property that characterizes CA allows for uniform and efficient hardware implementations: as we recalled in Sect. 2.2 the global rule of a CA is completely determined by the parallel application of the same local rule at all sites of the cellular array. Second, depending on the underlying local rule, the dynamic evolution of a CA can be quite complex and unpredictable. To a first approximation, it seems interesting to exploit the dynamics of certain CA when designing a symmetric cipher that follows the confusion and diffusion principles.

However, relying only on the dynamical properties of CA is usually not sufficient to realize a sound cryptographic primitive, since reasonable levels of security often require much more stringent criteria than those studied in the field of dynamical systems, such as sensitivity to initial conditions (Kurka 2003). This has been the case with the first proposal to employ CA for cryptographic applications, namely Wolfram's pseudorandom generator (1986).

Fig. 1 Example of CA of $n = 8$ cells equipped with rule 90 of diameter $d = 3$, defined as $f(x_i, x_{i+1}, x_{i+2}) = x_i \oplus x_{i+2}$



(a) CA 90 evaluated over 00110100.

x_i, x_{i+1}, x_{i+2}	$f(x_i, x_{i+1}, x_{i+2})$
000	0
100	1
010	0
110	1
001	1
101	0
011	1
111	0

(b) Truth table of rule 90.

Pseudorandom numbers and sequences are crucial in many cryptographic applications, particularly in the case of *Vernam-like* stream ciphers (Stinson and Paterson 2018). In this type of cipher, a short secret key is stretched through a pseudorandom generator, which produces a keystream of the same length as the plaintext. Then, the encryption simply amounts to the bitwise XOR of the plaintext and the keystream. Decryption works symmetrically by computing the XOR between the ciphertext and the keystream, which the receiver can re-create by running again the pseudorandom generator using the same secret key as a seed. Clearly, the security of this system entirely relies on the properties of the pseudorandom generator, with the assumption that it cannot be efficiently predicted by an attacker. Wolfram proposed to use a periodic-boundary CA equipped with the local rule 30 of diameter $d = 3$ to implement such pseudorandom generator. Specifically, the idea was to set the secret key as the initial configuration of the CA, and then to evolve it for several iterations. The *trace* of the central cell (that is, the sequence of states assumed by the cells through different time steps) was then used as a keystream to be XORed with the plaintext.

Wolfram argued the security of this generator on the basis of several system-theoretic and statistical criteria. Indeed, rule 30 belongs to the so-called "class 3" of local rules in Wolfram's taxonomy, which induce CA with a chaotic dynamics. Moreover, CA equipped with rule 30 are also known to satisfy Devaney's definition of topological chaos (Leporati and Mariot 2014). Still, Wolfram's pseudorandom generator was later found to be vulnerable against two serious attacks. The first one was demonstrated by Meier and Staffelbach (1991), who showed that one can mount a *correlation attack* that is able to efficiently recover the CA initial configuration. Koc and Apohan (1997) further showed that rule 30 can be easily approximated by affine functions, and this allows to efficiently invert the CA iteration (and thus again, to easily recover the initial configuration). Martin (2008) later showed that the vulnerabilities of Wolfram's generator can be explained in terms of the cryptographic properties of rule 30, when interpreted as a Boolean function of three variables. Specifically, Meier and Staffelbach's attack succeeds because rule 30 is not *first-order correlation immune*, while Koc and Apohan's attack is related to the low nonlinearity of rule 30.

For the reasons above, more recent works focused on improving Wolfram's generator by searching for rules with a larger diameter and a better trade-off of cryptographic properties. Formenti et al. (2014) used the DIEHARD statistical test suite (Marsaglia 1996) to investigate the quality of pseudorandom sequences produced by CA with local rules of diameter 5 of the best nonlinearity and correlation immunity. Leporati and Mariot (2014) focused on the class of *bipermutive* rules (which are known to induce chaotic CA) and performed a combinatorial search of highly nonlinear correlation-immune rules of diameter 5 and 7. The three best rules of diameter 5 that passed all tests of the NIST suite (Bassham et al. 2010) in that work were later selected to design the nonlinear components of the CARPenter (Lakra et al. 2018) and Pentavium (John et al. 2020) CA-based stream ciphers.

Other works considered different directions to design pseudorandom generators based on CA. Sipper and Tomassini (1996) were the first to propose the use of *non-uniform* CA to generate pseudorandom sequences, i.e., CA where each cell may use a different local rule to update its state. In particular, they proposed a *cellular programming* technique where a co-evolutionary algorithm is used to evolve the set of rules to be used in the CA. Tomassini and Perrenoud (2001) and Seredynski et al. (2004) subsequently applied the cellular programming approach also to the case of two-dimensional CA for designing strong pseudorandom generators.

A further research direction examined CA-based pseudorandom number generators where the cells have non-binary states. The main motivation underlying this approach is of combinatorial nature: for a given diameter, there are many more local rules on a non-binary alphabet than in the classic Boolean case, which gives more possibilities to find interesting local rules for pseudorandom generation. Clearly, this also brings the problem of actually implementing multi-states CA in hardware, an issue which can be easily addressed in the binary case. Bhattacharjee et al. (2017) considered CA rules on a ternary alphabet with diameter $d = 3$, finding one rule that was able to pass several statistical and empirical tests for randomness. Later, Bhattacharjee and Das (2019) extended the scope of this investigation to CA with a decimal alphabet, finding a few rules whose randomness can compete with some of the best pseudorandom generators proposed in the literature, both based on CA and not.

Finally, a few recent works contemplated additional CA models for pseudorandom generation. For instance, Manzoni and Mariot (2018) performed a statistical investigation through the NIST test suite of a Wolfram-like generator based on *asynchronous CA*, where the cells do not update all at the same time. Interestingly, the authors of that work noticed that the quality of the pseudorandom sequences generated by some local rules actually improves by adding a certain amount of asynchrony. Further, the first author of this manuscript investigated a new model of pseudorandom number generators grounded on the use of *orthogonal CA* (OCA; Mariot 2021). OCA have been recently introduced in the literature as a method to generate orthogonal Latin squares, with the original purpose of designing secret sharing schemes (Mariot and Leporati 2018; Mariot et al. 2020a). The rationale for considering OCA also for pseudorandom number generation is that they guarantee a minimum amount of diffusion, since orthogonal Latin squares are related to MDS codes. Moreover, since orthogonal Latin squares represent a permutation over the Cartesian product of their support set, the dynamics of the resulting generator is also reversible. In particular, Mariot (2021) describes a combinatorial search algorithm to enumerate all linear OCA pairs that can generate sequences with maximum period, a property which is usually sought in good pseudorandom generators.

4 The CA-XOR construction

Almost all works discussed in the previous section have one aspect in common: the preeminence of the statistical approach to assess the quality of pseudorandom generators designed by CA. However, it is widely known that the suitability of a pseudorandom generator for cryptographic applications cannot be established only by means of statistical tests (Carlet 2021; Stinson and Paterson 2018). In particular, statistical tests are useful to discard bad generators, but are in general not sufficient to conclude that certain specific attacks will not break them. Few papers such as Formenti et al. (2014), Leporati and Mariot (2014) and Mariot (2021) partially address this issue by considering also the cryptographic properties of the CA local rules, in combination with the statistical quality of the pseudorandom sequences that they produce. Nevertheless, such works are a minority in this research field. Moreover, cryptographic criteria of Boolean functions like balancedness, algebraic degree and nonlinearity were introduced for pseudorandom generators models that are different from those based on CA. Except for correlation-immunity and nonlinearity, which have been shown to be linked respectively to the Meier–Staffelbach and the Koc–Apohan attacks (Martin 2008; Leporati and Mariot 2014), there is no systematic way to translate the relevance of a cryptographic property for Boolean functions in a CA-based pseudorandom generator model.

In this paper we follow the opposite direction: instead of searching for Boolean functions that are embedded in a CA-based pseudorandom generator, here we use CA to synthesize Boolean functions with good cryptographic properties. The rationale is that such Boolean functions defined by CA can then be used in more established designs of pseudorandom generators (such as the combiner or the filter model), and whose security is better understood. In particular, we focus on defining a *secondary construction* of Boolean functions, where a CA is used to expand a known function f of d variables used as a CA local rule, and to generate a new function f^* of $n \geq d$ variables.

We now give the formal definition of our *CA-XOR construction* for Boolean functions, using the no-boundary CA model introduced in Sect. 2.2.

Definition 2 Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ be a CA of length $n \geq d$ equipped with the local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ of diameter $d \in \mathbb{N}$. Then, the Boolean function generated by f through the CA F is the n-variable function $f^* : \mathbb{F}_2^n \to \mathbb{F}_2$ defined for all $x \in \mathbb{F}_2^n$ as:

$$f^*(x) = \bigoplus_{i=1}^{n-d+1} f(x_i, \ldots, x_{i+d-1}) = f(x_1, \ldots, x_d) \oplus \cdots \oplus f(x_{n-d+1}, \ldots, x_n). \quad (10)$$

In other words, the CA-XOR construction works by first applying the CA vectorial function F induced by the local rule f to the input vector $x \in \mathbb{F}_2^n$ of n variables; then, the value of the resulting function f^* is obtained by computing the XOR of all the $n - d + 1$ output cells of the CA. To illustrate this idea, Fig. 2 gives a schematic depiction of how the CA-XOR construction.

Secondary constructions are mainly employed to generate new Boolean functions from old ones with analogous cryptographic properties. For example, Rothaus's construction (1976) starts from three bent functions of n variables, whose sum is also bent, and produces a new bent

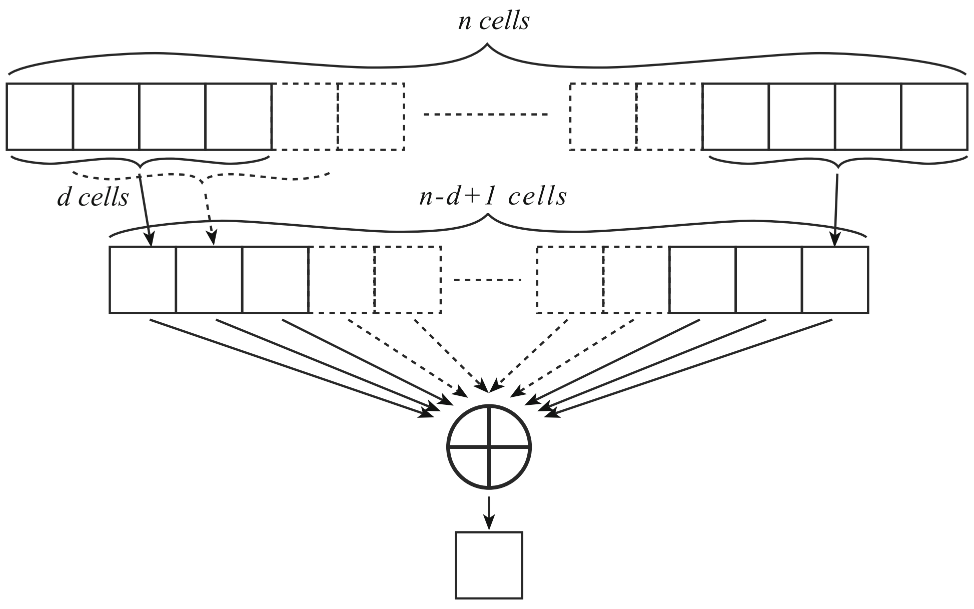


Fig. 2 Representation of our CA-based construction for Boolean functions

function of $n+2$ variables. We thus need to analyze which properties are preserved by our construction. The next lemma shows that the algebraic degree is one such property:

Lemma 1 *Let $f: \mathbb{F}_2^d \to \mathbb{F}_2$ be a Boolean function of d variables. For any $n \geq d$, the function f^* defined by the CA construction of Eq. (10) has the same algebraic degree of f.*

Proof The result is clearly true when $n = d$, since in that case $f \equiv f^*$. We thus only consider the case where $n > d$.

Let t be the algebraic degree of f. Each summand in Eq. (10) has degree t, since it always corresponds to the local rule f applied on a different neighborhood. We thus have to show that not all terms of degree t cancel each other out. Consider the first summand $f(x_1, \ldots, x_d)$, and let $S_t = \{I \subseteq 2^{[d]} : |I| = t, a_I \neq 0\}$ be the set of monomials of degree t in the ANF of f. Further, denote by $I_{min} \in S_t$ the minimum element of S_t with respect to the lexicographic order, that is, if $I_{min} = \{i_1, \ldots, i_t\}$ and $J = \{j_1, \ldots, j_t\}$ is any other set of S_t, it holds $i_k < j_k$ for some $k \in [t]$ and $i_h = j_h$ for all $h \in [k-1]$. This monomial cannot be cancelled by any other monomial in the ANF of the subsequent summands, since by Eq. (10) their neighborhoods are shifted by at least one coordinate with respect to that of the first summand. Indeed, if we take the lth summand $f(x_l, \ldots, x_{l+d-1})$ for $l \in \{2, \ldots, n-d+1\}$, and we denote by I_{min}^l its minimum monomial of degree t in lexicographic order, we have that $I_{min}^l = (i_1 + l, \ldots, i_t + l)$, which is distinct from $(i_1, \ldots, i_t) = I_{min}$. Hence, the variables in the monomial I_{min}^l cannot overlap completely those of I_{min}, which means that the two terms do not cancel each other out. Similarly, the monomial I_{min} cannot be canceled by any non-minimal monomial of degree d in the lth summand. Hence, the monomial corresponding to I_{min} appears in the ANF of (10), which proves that the algebraic degree of f^* is also t.

Remark that the result above does not hold if one considers our construction with *periodic boundary* CA, where no cells are lost upon application of the local rule. Indeed, the proof of Lemma 1 relies on the presence of non-overlapping neighborhoods, that do not cancel each other out in the ANF of the function f^*. On the contrary, with periodic boundary conditions one can find several examples where this property is not verified, and where the algebraic degree of the function generated by the CA-XOR construction is strictly less than the degree of the original local rule. This is an additional reason why we sticked to the NBCA model in this paper, and we did not consider evolving the CA for multiple time steps. This not a huge limitation however, since our CA-XOR construction is actually recursive and gives rise to an *infinite family* of Boolean functions: f^* can be defined over any number of variables $n \geq d$ by simply adding n cells to the CA.

5 Exhaustive search

We now investigate the effectiveness of the CA-XOR construction in building Boolean functions with interesting properties from a cryptographic standpoint. In particular, we are interested in finding, and possibly characterising, a family of Boolean functions from which it is possible to build, by means of our construction, other functions with similar cryptographic properties but with an arbitrary number of variables.

As remarked in Sect. 2.1, there are many properties that need to be taken into account when choosing Boolean functions to be used in the combiner or the filter model. Referring to nonlinearity, bent functions are of particular interest, since they reach the covering radius bound. Nonetheless, given that bent functions in n variables exist only if n is even, and that they are not balanced, in this study we also consider *semi-bent* functions, since they exist also for odd number of variables, they reach the quadratic bound of nonlinearity, and they can also be balanced. More precisely, in the rest of this paper we focus on the search of (semi-)bent Boolean functions that, when plugged into our CA-XOR construction, result in a bent function f^* when the number n of CA cells is even, and in a semi-bent function when n is odd. In this section we present a combinatorial algorithm to exhaustively search for such functions up to diameter 7, while in the next one we will use evolutionary algorithms to search them for larger diameters.

Lemma 1 states that the functions yielded by the CA-XOR construction in Eq. 10 have the same algebraic degree of the initial function. This remark is especially interesting when considering the case of quadratic functions. Indeed, quadratic functions are a proper subset of plateaued functions (Carlet 2021), which in turn include both bend and semi-bent functions, as mentioned in Sect. 2.1.

Moreover, Theorem 1 shows that there exists only a single (semi-)bent quadratic function up to affine equivalence. Therefore, it is of interest to classify which quadratic functions can be successfully extended by the CA-XOR construction under coarser equivalence relations.

On account of Lemma 1, in Mariot et al. (2020b) we devised an algorithm that exhaustively visits all ANFs of degree d, retrieve their truth tables via the Mӧbius transform, and then apply our construction to see which of these functions always produce semi-bent functions up to $n = 20$ cells. Considering the case of quadratic functions, the resulting search space explored by the Search-ANF algorithm had the following size with respect to the diameter of the starting local rule:

$$S_{d,2} = \left(2^{\binom{d}{2}} - 1 \right) \cdot 2^{\binom{d}{1}} = \left(2^{\frac{d(d-1)}{2}} - 1 \right) \cdot 2^d. \tag{11}$$

In our previous work (Mariot et al. 2020b), we applied this algorithm up to diameter $d = 6$, where there are approximately 2 million quadratic functions to explore. Here, we refine our search approach by leveraging on Theorem 1. As one can see from Eqs. (7) and (8), the unique (semi-)bent quadratic function that exists up to affine equivalence is *homogeneous*, meaning that its ANF does not have any linear term. Thus, it makes sense to limit our exhaustive search only to homogeneous quadratic ANFs, since this reduces the search space size. Clearly, linear terms do not influence the nonlinearity of the functions resulting from our CA-XOR construction, which is the main property one aims to preserve when considering (semi-)bent functions. In particular, the nonlinearity of a quadratic function is determined only by the monomials of degree 2 in its ANF.

Our search algorithm is based on the ANF representation. In general, given a target algebraic degree t, the 2^d-bit vector of the ANF coefficients can be easily constrained to yield only Boolean functions of degree t: it suffices to set *at least one* of the coefficients a_I such that $|I| = t$ to 1, while all coefficients a_J with $|J| > t$ must be set to 0. Moreover, since we consider only *homogeneous functions*, also the coefficients related to monomials of degree lower degree than t can be set to 0. Then, by using the *Mӧbius Transform* recalled in Eq. (2), one can recover the truth table starting from its ANF coefficients, and check if the corresponding function is bent or semi-bent by computing its Walsh spectrum. In this case, we can finally test if our construction generates quadratic bent or semi-bent functions (depending on whether the number of variables n is even or odd) up to a specified number of variables.

Algorithm 1 reports the pseudocode of the simplified search procedure, adapted to the enumeration of quadratic homogeneous ANFs.

Algorithm 1 Pseudocode of the simplified exhaustive search algorithm.

```
SEARCH-QUAD-ANF($d, n$)
  Initialization:
    1. Build the family $\mathcal{I}_2 = \{I \subseteq [d] : \#I = 2\}$ of monomials of degree 2
    2. Set all $2^d$ ANF coefficients of $f$ to 0
  Loop: For all subsets $\mathcal{T} \subseteq \mathcal{I}_2$ (except the empty set), do:
    1. Reset all 2-degree terms in the ANF to 0
    2. For all $T \in \mathcal{T}$, set the ANF coefficient $a_T$ to 1
    3. Set $f$ to the Möbius Transform (Equation (2)) of the ANF
    4. Compute the Walsh transform (Equation (3)) of $f$
    5. If $f$ is (semi-)bent, then apply the CA-XOR construction up to $n$ cells
    6. If $f$ satisfies the target, print it
```

The SEARCH-QUAD-ANF algorithm takes as input only the diameter d of the starting local rule and the number of cells n up to which to test the CA-XOR construction. The initialization step consists in constructing the family of all subsets of 2 variables out of d, and then set all coefficients in the ANF vector to zero. Next, the main loop performs the enumeration of all $2^{\binom{d}{2}} - 1$ non-empty subsets of of two variables. For each of them, the algorithm set the corresponding ANF coefficients to 1, then recover the truth table of the function via the Möbius Transform and checks whether the function is bent or semi-bent, depending on the parity of d. If d is even, then the target is to check that f is a bent function, and that by applying the CA-XOR construction for all $d < i \leq n$ the resulting function is bent when i is even, and semi-bent when i is odd. If this is the case, then the starting function f is printed.

Remark that SEARCH-QUAD-ANF is simpler than the SEARCH-ANF algorithm described in Mariot et al. (2020b), since only quadratic monomials are considered. More precisely, SEARCH-ANF takes as input the number of variables d and the target degree t, and then performs two nested loops over all possible sets of monomials having degree *at most* $t - 1$ (inner loop), and all possible non-empty sets of monomials of degree *exactly* t (outer loop). Dealing with homogeneous functions, only the outer loop is needed for our purposes. Since functions in d variables have at most $\binom{d}{2}$ terms of degree 2 in their ANFs, such loop is executed $2^{\binom{d}{2}} - 1$ times, and thus it is feasible for functions up to 7 variables. For comparison, Table 1 displays the size of the spaces $\mathcal{Q}_d$ and $\mathcal{H}_d$, respectively visited by the SEARCH-ANF and the SEARCH-QUAD-ANF algorithms.

Table 1 Search space sizes for the set of all quadratic functions $\mathcal{Q}_d$ and the set of all homogeneous quadratic functions $\mathcal{H}_d$ of d variables

d	$\mathcal{Q}_d$	$\mathcal{H}_d$
3	56	7
4	1008	63
5	32,736	1023
6	2,097,088	32,767
7	268,435,328	2,097,151
8	68,719,476,480	268,435,455

6 Evolutionary search

As it can be seen from Table 1, the size of the search space for the set of quadratic functions quickly grows for small sizes already, even by considering only the subset of homogeneous ANFs. For this reason, we enlarged our empirical search of quadratic (semi-)bent functions that our CA-XOR construction can extend by using *Evolutionary Algorithms* (EA).

EA and other nature-inspired optimization heuristics such as swarm intelligence algorithms have been extensively used to optimize the cryptographic properties of Boolean functions. Examples include *Genetic Algorithms* (GA; Millan et al. 1998; Mariot and Leporati 2015a; Manzoni et al. 2020), *Genetic Programming* (GP; Castro et al. 2003; Hrbacek and Dvorak 2014; Picek et al. 2015) and *Particle Swarm Optimization* (PSO; Saber et al. 2006; Mariot and Leporati 2015b). Most of these works usually focus on the use of EA and swarm intelligence algorithms as alternative constructions of Boolean functions with good cryptographic properties. In particular, such methods do not work like primary or secondary constructions, since they blindly search only on the basis of a *fitness function* to be optimized, which measures a combination of some properties of interest. Consequently, it is difficult to compare the functions obtained with these methods with those produced by a primary or secondary construction. Moreover, directly employing EA to find quadratic functions of diameter $d > 7$ that our CA-XOR construction can extend presents another practical problem: namely, one should adopt a *two-stage* optimization strategy where first EA are used to evolve (semi-)bent functions. Then, in the second phase the EA should tweak the evolved functions (without decreasing their nonlinearity) so that when plugged into a no-boundary CA, the CA-XOR construction can successfully extend them up to a large number of variables.

6.1 Candidate solutions encoding

Consequently, it would be useful to find a representation of candidate solutions such that the EA is constrained to explore only (semi-)bent quadratic functions, in order to focus the optimization effort only on maximizing the number of cells in the CA-XOR construction. To this end, recall from Theorem 1 that there exists only one bent (respectively, semi-bent) quadratic function of d variables for all $d \in \mathbb{N}$, which we defined as $b(x)$ in Eq. (7) (respectively, $s(x)$ in Eq. (8)). Since each bent quadratic function can be obtained by applying a suitable affine map to b(x) (or s(x) if we consider the semi-bent case), our basic idea to apply EA to this problem is to *evolve a population of affine transformations*, rather than evolving a population of specific Boolean functions as done in the majority of the works in this area.

Given the number of variables d of the starting Boolean function, the genotype of an individual in the EA population is thus a pair $c = (M, v)$, where M is a $d \times d$ invertible Boolean matrix, while $v \in \mathbb{F}_2^d$ is a vector of d bits. Then, the phenotype corresponding to c is the function $f : \mathbb{F}_2^d \to \mathbb{F}_2$ defined as follows:

$$f(x) = \begin{cases} b(Mx^\top \oplus v), & \text{if } d \text{ is even,} \\ s(Mx^\top \oplus v), & \text{if } d \text{ is odd,} \end{cases} \tag{12}$$

where the parity of d is used to understand whether we need to start from a bent function or semi-bent function when d is even or odd, respectively.

6.2 Mutation operator

Beside the encoding of the candidate solutions, an EA needs also to specify a set of *variation operators* that are used to create new solutions from the current populations. In this case, we chose to employ *Evolutionary Strategies* (ES), since they only need to define a *mutation operator*. As a matter of fact, other EA such as genetic algorithms also define *crossover operators*, which take in input two individuals in the population and output one or more *offspring* solutions. However, defining a crossover operator that preserves our encoding for the candidate solutions is not straightforward. Indeed, while the vectors v defining the translation of an affine map is easy to handle with classic operators (such as one-point crossover), the same cannot be said for the linear transformation part. In fact, to define a crossover operator between two parents $p_1 = (M, v)$ and $p_2 = (M', v')$, one would need to devise a method to combine the matrices M, M' such that the result is still an invertible matrix. For this reason, we decided to focus only on mutation, which applies a random small tweak to the genotype of a single individual, rather than combining those of two parents.

Mutation of the translation vector v is easy to perform: one can apply the classic mutation operators developed in the EA literature for bitstring-based encodings. In particular, we adopted a simple *bit-flip* operator which, for all positions $i \in [d]$, complements the bit v_i with mutation probability $p_{p_\mu} \in (0, 1)$. The mutation of the linear map part of an individual, namely the invertible matrix M, starts from the following observation. Let M_i be the $d \times d - 1$ matrix obtained by removing column $i \in [d]$ from an invertible $d \times d$ Boolean matrix M. Since M is invertible, M_i must have rank $d - 1$. Hence, the linear transformation $L : x \mapsto Mx^\top$ generates the whole vector space $\mathbb{F}_2^d$, while the map $L' : x \mapsto M_i x^\top$ spans a subspace $S_i \subseteq \mathbb{F}_2^d$ of dimension $d - 1$. Let $C_i = \mathbb{F}_2^d \setminus S_i$ be the complementary set of the subspace C_i. By definition of subspace spanned by the matrix M_i, each vector in C_i will be linearly independent with each vector in S_i. Therefore, if we take any vector $c \in C_i$ and add it to M_i, we obtain an invertible $d \times d$ matrix.

In other words, our mutation operator works by replacing with probability p_μ each column of a matrix with a random vector chosen from the complementary set of the span generated by the remaining columns. In this way, the rank (and thus the invertibility) of the matrix is preserved. Algorithm 2 summarizes our operator as high-level pseudocode.

Algorithm 2 Pseudocode of the mutation operator used in our ES algorithm.

```
MUTATION-AFFINE($M, v, d, p_\mu$)
  Vector part:
    Loop: for all $i \in [d]$ do:
      1. Draw a random number $r \in (0,1)$
      2. If $r < p_\mu$, then flip the bit $v_i$
  Matrix part:
    Loop: for all $i \in [d]$ do:
      1. Draw a random number $r \in (0,1)$
      2. If $r < p_\mu$, then:
         (a) Remove the $i$-th column of $M$
         (b) Generate the span $S_i$ of the remaining columns
         (c) Pick a random vector $c$ from $C_i = \mathbb{F}_2^d \setminus S_i$ and add it to $M$
```

6.3 Overall ES algorithm structure

There exist two main types of ES (Luke 2015): in (μ, λ) ES, a population of λ initial individuals is first generated at random, and then their fitness is assessed. The μ best individuals are selected for reproduction. In particular, each of the μ selected individuals will generate λ/μ new individuals through mutation, thereby producing a new population of size λ. The ES process is then iterated by assessing the fitness of the new λ individuals, selecting the μ best ones, and applying the mutation operators on them, and so on. The $(\mu + \lambda)$ ES differs only in the fact that the μ best individuals from the old population are added to the new population of λ individuals generated through mutation. Therefore, in this case the parents directly compete also with their children.

For our experiments, we employed both (μ, λ) and $(\mu + \lambda)$ ES. Since the encoding of the candidate solutions described in Sect. 6.1 guarantees that the phenotypes of the individuals are all (semi-)bent quadratic functions, the fitness function used in our ES optimizes another criterion. In particular, each function f in the population is used as a local rule of diameter d of a no-boundary CA. Given a target number of cells $N \in \mathbb{N}$, the fitness function applies the CA-XOR construction for all CA lengths $d < n \leq N'$, where $N' \leq N$ is the maximum number of cells up to which the construction always yields bent functions when n is even, and semi-bent functions when n is odd. The fitness value of f is then $N' - d$, i.e., the number of cells over which the CA-XOR construction successfully works when using f as a local rule.

7 Experiments

We applied the exhaustive search algorithm described in Sect. 5 on quadratic functions of $3 \leq d \leq 7$ variables, while we applied the ES approach for functions of $8 \leq d \leq 12$ variables. In the case of exhaustive search, we tested our CA-based construction up to $N = 20$ cells. On the other hand, with ES the fitness function applied the CA-XOR construction to a maximum of $N = 16$ cells. This is due to the fact that testing the construction up to 20 cells for a single starting function of more than seven variables takes approximately 20 s on the server that we used to perform our experiments, which is equipped with a Core i7 processor running at 2.8 GHz. Since the ES algorithm needs to test several thousands of these functions during the evolution process, the amount of time required to finish a single optimization run would have been to large. For the other parameters related to the ES, we set $\lambda = d$ and $\mu = \lceil \lambda/3 \rceil$. Therefore, the diameter of the local rules determines the number of individuals generated at each iteration. The mutation probability was set to $p_\mu = 1/\lambda$, a common choice in the field of GA and ES (Luke 2015). Further, for each considered diameter $8 \leq d \leq 12$ we performed 100 independent runs of (μ, λ) and $(\mu + \lambda)$ ES, giving a budget of 1000 generations for each run.

To investigate the results of our exhaustive search and ES experiments, we classified the obtained functions up to permutation equivalence. As a matter of fact, we already know by Theorem 1 that there exists only a single (semi-)bent quadratic function up to affine equivalence, hence a classification based on this equivalence relation would not give us any insight into the functions found in our search experiments. Consequently, we needed to a coarser relation to discriminate different groups of functions that our CA-XOR construction can successfully extend.

Table 2 Results obtained with the exhaustive search experiments

d	$\mathcal{H}_d$	$\mathcal{C}_{16}$	$\mathcal{C}_{20}$	classes
3	7	0	0	0
4	63	3	3	1
5	1023	22	22	7
6	32,767	318	318	44
7	2,097,151	15,656	10,974	797

We denote with $\mathcal{H}_d$ the total number of homogeneous quadratic functions in d variables, and with $\mathcal{C}_n$ the number of functions for which the construction applied up to n cells always produces bent or semi-bent functions (for n even or odd, respectively)

7.1 Exhaustive Search Results

The results for the exhaustive enumeration algorithm Search-Quad-ANF are summarized in Table 2. In particular, for each considered diameter d, we report in column $\mathcal{H}_d$ the total number of homogeneous quadratic functions in d variables, that is $2^{\binom{d}{2}} - 1$. Columns $\mathcal{C}_{16}$ and $\mathcal{C}_{20}$ show the total number of functions over which our construction works up to 16 and 20 cells respectively, namely:

- bent functions in an even number of variables that always produces bent functions when an even number of variables is added to the CA and semi-bent functions when an odd number of variables is added to the CA;
- semi-bent functions in an odd number of variables that always produces semi-bent functions when an even number of variables is added to the CA and bent functions when an odd number of variables is added to the CA.

In other words, we are looking for Boolean functions that always yield bent functions when n is even and semi-bent functions when n is odd. The last column reports the number of distinct classes represented among the functions for which the construction works up to $n = 20$ cells, up to permutation equivalence.

One can see from the table that the number of distinct permutation classes grows at increasing diameter. Namely, one starts from the smallest instance $d = 3$ where no quadratic semi-bent functions can be extended by the CA-XOR construction up to $n = 16$ cells (let alone $n = 20$ cells), while for diameter $d = 7$ one already obtains 797 distinct classes. It is also of interest to notice that for $d = 7$ the number of functions for which the construction works up to 20 cells is lower than for 16 cells, while for lower diameters it is always equal. This might indicate that, as the diameter grows, there are more (semi-)bent quadratic functions for which the CA-XOR construction works up to a certain point, but then fails for all larger CA lengths.

7.2 Evolutionary strategies results

The results obtained by our (μ, λ) and $(\mu + \lambda)$ ES are somehow surprising. The first remarkable finding is that, for all considered diameters $8 \leq d \leq 12$, both ES algorithms *always managed to converge to an optimal solution in all 100 experimental runs*. In other words, the two algorithms always find (semi-bent) quadratic functions that the CA-XOR construction can successfully extend up to $N = 16$ cells. This is especially interesting considering the limited computational budget of 1000 generations given to both algorithms.

Consequently, for each diameter we obtained a set of 200 functions that can be extended up to 16 cells. We filtered these solutions by checking which among them could also be extended to 20 cells under the CA-XOR construction. For each diameter, this step always resulted in an average of 30 functions, which is in line with the observation done for diameter $d = 7$ in the previous section: for larger diameters, there exist many (semi-)bent quadratic functions that can be extended up to a certain point, after which our construction always fails.

Finally, we classified the filtered functions of diameter $d = 8$ and 9 up to permutation equivalence. As a matter of fact, to check if two functions of d variables are permutation equivalent one needs to find a suitable permutation among all the $d!$ possible ones. Considering that we had an average of 30 functions to classify, this task was feasible only up to $d = 9$ variables. After the classification step, we observed the remarkable fact that *each function belongs to a different equivalence class*. This remark seems to confirm what we already observed in the previous section with the exhaustive search results, namely that the number of permutation classes grows quickly with respect to the CA diameter. Moreover, our ES algorithms seem to be able to uniformly sample among the distribution of such classes, instead of getting stuck in the same local optima. This is a reasonable question, since having an evolutionary algorithm that always converges over all experimental runs could indicate that it always finds the same solution. However, this doubt is cleared by the fact that we obtained as many permutation equivalence classes as many functions in our classification.

8 Conclusions

In this paper, we continued the investigation of a secondary construction of Boolean functions based on CA that was initially introduced in Mariot et al. (2020b). The CA-XOR construction uses a Boolean function $f : \mathbb{F}_2^d \rightarrow \mathbb{F}_2$ as a local rule of diameter d in a no-boundary CA of $n \geq d$ cells, and defines a new function $f^* : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ of n variables by computing the XOR of the CA output cells once the local rule f has been evaluated over the input vector. A preliminary theoretical investigation shows that such construction preserves the algebraic degree of the starting function used as a local rule, which motivates the focus on quadratic functions. Indeed, quadratic functions have a well-known structure, and it is known that there exist only one (semi-)bent quadratic function up to affine equivalence. Therefore, it is interesting to find which quadratic functions can be successfully extended by the CA-XOR construction, using coarser equivalence relations such as permutation equivalence. To this end, we refined the combinatorial algorithm proposed in Mariot et al. (2020b) to enumerate only homogeneous quadratic ANFs, which allowed us to exhaustively search all (semi-)bent quadratic functions that our construction can extend up to diameter $d = 7$. Further, we devised a (μ, λ) and $(\mu + \lambda)$ ES algorithm to evolve functions that can be extended by the CA-XOR construction for diameters $8 < d \leq 12$. The results of our search experiments show that the number of permutation equivalence classes quickly grows with respect to the diameter of the CA local rule. In particular, beside observing a growing number of distinct classes in the exhaustive search experiments, this observation is also corroborated by the fact that the functions found by our ES algorithms all belong to different classes.

The present work is far from exhausting all aspects related to the CA-XOR construction, and there are several avenues for further research on the topic. The first direction worth exploring is a deeper investigation of the equivalence classes found in our heuristic search experiments. In particular, it would be interesting to give a theoretical characterization of the equivalence classes that the CA-XOR can extend *recursively*, i.e., no matter the CA length one always obtain a (semi-)bent function. A possible way to approach this problem is to use the *graph representation* described in Tokareva (2015). In particular, the graph of a quadratic bent function is defined as the graph where the nodes are the input variables, and two nodes are connected by an edge if and only if the multiplication of the corresponding variables occurs as a quadratic monomial in the ANF of the function. A theoretical characterization of the functions which can be extended by the CA-XOR construction might then be inferred by the properties of such graphs.

A second idea to explore concerns the generalization of the CA-XOR construction. In particular, one could consider variations where not all output cells in the CA are XORed to define the constructed function, but only a subset of them. Notice that this is equivalent to study the *component functions* of the S-box defined by the no boundary CA. As a matter of fact, the interest for studying the CA-XOR construction originated from the need of characterizing the cryptographic properties of S-boxes defined by CA. The authors of Mariot et al. (2019) recently showed an upper bound on the nonlinearity of such S-boxes in terms of the nonlinearity of the underlying local rule. The CA-XOR construction studied in this paper can be seen as a single component of a CA-based S-box, namely the linear combination that sums all output coordinates. Consequently, studying under which conditions a generalized CA-XOR construction yields Boolean functions with a specific value of nonlinearity could help in deriving new bounds for CA-based S-boxes.

Finally, a very interesting research direction would be to investigate the CA-XOR construction with functions of higher algebraic degree, such as cubic bent functions. In this regard, we performed a preliminary test on the cubic semi-bent functions of 6 variables found in Mariot and Leporati (2015a) through a genetic algorithm. However, none of them could be successfully extended by our CA-XOR construction. This remark calls for a broader and more systematic evaluation, by repeating the exhaustive search experiments performed in this paper also for cubic functions, up to a number of variables where this is still feasible.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Bassham LE III, Rukhin AL, Soto J, Nechvatal JR, Smid ME, Barker EB, Leigh SD, Levenson M, Vangel M, Banks DL et al (2010) SP 800–22 Rev. 1a: a statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST, Gaithersburg

Bertoni G, Daemen J, Peeters M, Assche GV (2011) The Keccak reference, January 2011. http://keccak.noekeon.org/

Bhattacharjee K, Das S (2019) Random number generation using decimal cellular automata. Commun Nonlinear Sci Numer Simul 78:104878

Bhattacharjee K, Paul D, Das S (2017) Pseudo-random number generation using a 3-state cellular automaton. Int J Mod Phys C 28(06):1750078

Carlet C (2021) Boolean functions for cryptography and coding theory. Cambridge University Press, Cambridge

Castro JCH, Isasi P, and del Arco-Calderón CL (2003). Finding efficient nonlinear functions by means of genetic programming. In: Palade V, Howlett RJ, Jain LC (eds) Proceedings of KES 2003, Part I, LNCS, vol 2773. Springer, pp 1192–1198

Daemen J, Rijmen V (2020) The design of Rijndael—the advanced encryption standard (AES), 2nd edn. Information security and cryptography. Springer, Berlin

Daemen J, Govaerts R, Vandewalle J (1994) Invertible shift-invariant transformations on binary arrays. Appl Math Comput 62(2–3):259–277

Dillon JF (1974) Elementary Hadamard difference sets. PhD Thesis

Formenti E, Imai K, Martin B, Yunès JB (2014) Advances on random sequence generation by uniform cellular automata. In: Calude CS, Freivalds R, Kazuo I (eds) Computing with new resources. Springer, Cham, pp 56–70

Gadouleau M, Mariot L, Picek S (2020) Bent functions from cellular automata. IACR cryptology ePrint archive, p 1272

Hazari R, Kundu S, Bhardwaj M, Das S (2018) ECA 184 can implement any logic circuits. J Cell Autom 13(4):359–371

Hrbacek R, Dvorak V (2014) Bent function synthesis by means of Cartesian genetic programming. In: Proceedings of PPSN 2014, LNCS, pp 414–423

John A, Nandu BC, Ajesh A, Jose J (2020) PENTAVIUM: potent trivium-like stream cipher using higher radii cellular automata. In: Gwizdalla TM, Manzoni L, Sirakoulis GC, Bandini S, Podlaski K (eds) Cellular automata—14th international conference on cellular automata for research and industry, ACRI 2020, Proceedings, Lodz, Poland, 2–4 December 2020, LNCS, vol 12599. Springer, pp 90–100

Koc CK, Apohan A (1997) Inversion of cellular automata iterations. IEE Proc Comput Digit Tech 144(5):279–284

Kurka P (2003) Topological and symbolic dynamics. Société mathématique de France

Lakra R, John A, Jose J (2018) Carpenter: a cellular automata based resilient pentavalent stream cipher. In: Mauri G, Yacoubi SF, Dennunzio A, Nishinari K, Manzoni L (eds) Cellular automata—13th international conference on cellular automata for research and industry, ACRI 2018, proceedings, Como, Italy, 17–21 September 2018, LNCS, vol 11115. Springer, pp 352–363

Leporati A, Mariot L (2014) Cryptographic properties of bipermutive cellular automata rules. J Cell Autom 9(5–6):437–475

Luke S (2015) Essentials of metaheuristics, 2nd edn. Lulu, Raleigh

Manzoni L, Mariot L (2018) Cellular automata pseudo-random number generators and their resistance to asynchrony. In: Mauri G, Yacoubi SE, Dennunzio A, Nishinari K, Manzoni L (eds) ACRI 2018, LNCS, vol 11115. Springer, pp 428–437

Manzoni L, Mariot L, Tuba E (2020) Balanced crossover operators in genetic algorithms. Swarm Evol Comput 54:100646

Mariot L (2021) Hip to be (Latin) square: maximal period sequences from orthogonal cellular automata. CoRR, abs/2106.07750

Mariot L, Leporati A (2015a) A genetic algorithm for evolving plateaued cryptographic Boolean functions. In: Dediu A, Magdalena L, Martín-Vide C (eds) Proceedings of TPNC 2015, LNCS, vol 9477. Springer, pp 33–45

Mariot L, Leporati A (2015b) Heuristic search by particle swarm optimization of Boolean functions for cryptographic applications. In: Silva S, Esparcia-Alcázar AI (eds) Companion proceedings of GECCO 2015. ACM, pp 1425–1426

Mariot L, Leporati A (2018) Inversion of mutually orthogonal cellular automata. In: Mauri G, Yacoubi SE, Dennunzio A, Nishinari K, Manzoni L (eds) Cellular automata—13th international conference on cellular automata for research and industry, ACRI 2018, proceedings, Como, Italy, 17–21 September 2018, LNCS, vol 11115. Springer, pp 364–376

Mariot L, Picek S, Leporati A, Jakobovic D (2019) Cellular automata based S-boxes. Cryptogr Commun 11(1):41–62

Mariot L, Gadouleau M, Formenti E, Leporati A (2020a) Mutually orthogonal Latin squares based on cellular automata. Des Codes Cryptogr 88(2):391–411

Mariot L, Saletta M, Leporati A, Manzoni L (2020b) Exploring semi-bent Boolean functions arising from cellular automata. In: Gwizdalla TM, Manzoni L, Sirakoulis GC, Bandini S, Podlaski K (eds) Cellular automata—14th international conference on cellular automata for research and industry, ACRI 2020, proceedings, Lodz, Poland, 2–4 December 2020, LNCS, vol 12599. Springer, pp 56–66

Marsaglia G (1996) Diehard: a battery of tests of randomness. http://www.stat.fsu.edu/-geo/diehard.html

Martin B (2008) A Walsh exploration of elementary CA rules. J Cell Autom 3(2):145–156

McFarland RL (1973) A family of difference sets in non-cyclic groups. J Comb Theory A 15(1):1–10

Meier W, Staffelbach O (1991) Analysis of pseudo random sequence generated by cellular automata. In: Davies DW (ed) Advances in cryptology—EUROCRYPT '91, workshop on the theory and application of cryptographic techniques, proceedings, Brighton, UK, 8–11 April 1991, LNCS, vol 547. Springer, pp 186–199

Millan W, Clark AJ, Dawson E (1998) 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, proceedings, Espoo, Finland, 31 May–4 June 1998, LNCS, vol 1403. Springer, pp 489–499

Picek S, Guilley S, Carlet C, Jakobovic D, Miller JF (2015) Evolutionary approach for finding correlation immune Boolean functions of order t with minimal Hamming weight. Proc TPNC 2015:71–82

Rothaus OS (1976) On "bent'' functions. J Comb Theory A 20(3):300–305

Saber Z, Uddin MF, Youssef AM (2006) On the existence of (9, 3, 5, 240) resilient functions. IEEE Trans Inf Theory 52(5):2269– 2270

Seredynski F, Bouvry P, Zomaya AY (2004) Cellular automata computations and secret key cryptography. Parallel Comput 30(5–6):753– 766

Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28(4):656–715

Sipper M, Tomassini M (1996) Co-evolving parallel random number generators. In: Voigt H, Ebeling W, Rechenberg I, Schwefel H (eds) Parallel problem solving from nature—PPSN IV, international conference on evolutionary computation. The 4th international conference on parallel problem solving from nature, proceedings, Berlin, Germany, 22–26 September 1996, LNCS, vol 1141. Springer, pp 950–959

Stinson DR, Paterson M (2018) Cryptography: theory and practice. CRC Press, Boca Raton

Tokareva N (2015) Bent functions: results and applications to cryptography. Academic, New York

Tomassini M, Perrenoud M (2001) Cryptography with cellular automata. Appl Soft Comput 1(2):151–160

Ulam S (1952) Random processes and transformations. In: Proceedings of the international congress on mathematics, vol 2, pp 264–275

Von Neumann J (1966). In: Burks AW (ed) Theory of self-reproducing automata. University of Illinois Press, Champaign

Wolfram S (1983) Statistical mechanics of cellular automata. Rev Mod Phys 55(3):601

Wolfram S (1986) Cryptography with cellular automata. In: Williams HC (ed) CRYPTO '85, LNCS, vol 218, pp 429–432

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.