Evolutionary algorithms for designing reversible cellular automata

Mariot, Luca; Picek, Stjepan; Jakobovic, Domagoj; Leporati, Alberto

DOI
10.1007/s10710-021-09415-7

Publication date
2021

Document Version
Final published version

Published in
GENETIC PROGRAMMING AND EVOLVABLE MACHINES

Genetic Programming and Evolvable Machines
https://doi.org/10.1007/s10710-021-09415-7

Evolutionary algorithms for designing reversible cellular automata

Luca Mariot[1] · **Stjepan Picek**[1] · **Domagoj Jakobovic**[2] · **Alberto Leporati**[3]

Abstract
Reversible Cellular Automata (RCA) are a particular kind of shift-invariant transformations characterized by dynamics composed only of disjoint cycles. They have many applications in the simulation of physical systems, cryptography, and reversible computing. In this work, we formulate the search of a specific class of RCA – namely, those whose local update rules are defined by conserved landscapes – as an optimization problem to be tackled with Genetic Algorithms (GA) and Genetic Programming (GP). In particular, our experimental investigation revolves around three different research questions, which we address through a single-objective, a multi-objective, and a lexicographic approach. In the single-objective approach, we observe that GP can already find an optimal solution in the initial population. This indicates that evolutionary algorithms are not needed when evolving only the reversibility of such CA, and a more efficient method is to generate at random syntactic trees that define the local update rule. On the other hand, GA and GP proved to be quite effective in the multi-objective and lexicographic approach to (1) discover a trade-off between the reversibility and the Hamming weight of conserved landscape rules, and (2) observe that conserved landscape CA cannot be used in symmetric cryptography because their Hamming weight (and thus their nonlinearity) is too low.

Keywords Shift-invariant transformations · Cellular automata · Reversibility · Genetic programming · Genetic algorithms

1 Introduction

The shift-invariance property is important when studying and modeling several types of discrete dynamical systems. The property states that any translation of the input state results in the same translation of the output state in a system governed by a shift-invariant transformation. When a finite array describes the state of the system,

✉ Luca Mariot
l.mariot@tudelft.nl

Extended author information available on the last page of the article

Published online: 29 September 2021

shift-invariant transformations correspond to cellular automata (CA), i.e., functions defined by a local update rule uniformly applied at all sites of the array [15]. In fact, in this case, the dependency neighborhood that a cell uses to update its state is upper bounded by the size of the finite array itself, while over infinite arrays, one could have shift-invariant transformation where each coordinate depends on cells that are arbitrarily far. CA have been thoroughly studied both as models for simulating discrete dynamical systems in physics [2, 14, 40], biology [8, 9, 37], ecology [1, 12, 11] and other fields, as well as to design computational devices, for example in symmetric cryptography [13, 25, 35] and fault-tolerant computing [26, 27, 42]. Reversible shift-invariant transformations, particularly Reversible CA (RCA), have the additional characteristic of preserving information. As such, the dynamics of an RCA can be reversed backward in time starting from any state, and the inverse mapping is itself a CA. This characteristic makes RCA especially interesting for designing energy-efficient computing devices, as stated by Landauer's principle [20]. In fact, any irreversible logical operation implemented in hardware leads to heat dissipation, which entails a physical lower bound on the miniaturization of devices based on irreversible gates. One more interesting domain for RCA is cryptography, where they can be used to design encryption and decryption algorithms [25].

Unfortunately, while RCA are characterized by simple combinatorial rules, designing them is a difficult problem when considering additional properties as required by specific applications. This is because there are only a few known classes of RCA [15] and an exhaustive search of all possible RCA is unfeasible for large local rule sizes. Considering these difficulties and the limited number of available theoretical results, heuristics – and, more precisely, evolutionary algorithms (EA) – represent an interesting option for designing RCA.

An interesting class of CA that include reversible ones are *marker CA*, where the local update rule flips the state of a cell if its neighbors take on a set of patterns (also called *flipping landscapes*) that are conserved by the resulting shift-invariant transformation [40]. Evolutionary algorithms like genetic algorithms (GA) and genetic programming (GP) intuitively represent a good fit to evolve the local rules of marker CA since they have a simple description through their generating functions. In particular, the output of a marker CA rule corresponds to the XOR of the cell in the origin of the neighborhood and its generating function evaluated on the neighboring cells. As such, it becomes rather straightforward to formulate the optimization objective for the reversibility property by minimizing the number of compatible flipping landscapes defined by the generating function. An optimal solution, in this context, is a marker CA rule whose flipping landscapes are mutually incompatible, or equivalently a *conserved landscape rule*.

Additionally, the Hamming weight of a generating function in a marker CA represents a good indicator of its *nonlinearity* [3, 39], which is a relevant property of Boolean functions used in domains like sequences [28], telecommunications [29], and cryptography [25]. Consequently, maximizing the Hamming weight of the generating function can be considered an additional optimization objective and also motivates the use of multi-objective evolutionary algorithms to investigate the resulting Pareto fronts.

Our research investigates how difficult it is for evolutionary algorithms to find conserved landscape CA rules, considering their small number as compared to the corresponding search space size. Further, we explore the evolution of rules of larger diameter (i.e., larger neighborhood size), as such rules are relevant from the practical perspective. Finally, we investigate the trade-offs between the reversibility of a marker CA rules and the Hamming weight.

This paper is an extended version of the work "An Evolutionary View on Reversible Shift-Invariant Transformations" [19] presented at EuroGP 2020. With respect to that work, here:

1. We consider one additional evolutionary algorithm in our experiments, namely the lexicographic genetic algorithm. By doing so, we allow a more detailed analysis of the lexicographic paradigm for the evolution of conserved landscape CA.
2. We conduct an extensive tuning phase for all algorithms on the problem instance with diameter d equal to 10. This represents a much larger and more difficult problem than the one considered in [19], where the diameter for tuning was set to 7, thus allowing more meaningful tuning results.
3. We consider more problem instances: while the original paper considered diameter sizes d from 8 to 13, this work investigates diameter sizes ranging from 7 to 15.
4. While in the original paper we allowed the offset ω to be of size $d - 1$, here we set ω equal to 3 for all experiments. By doing so, we aim to explore a more difficult optimization problem, as there will be fewer solutions fulfilling the criteria.
5. Finally, we provide a more detailed experimental analysis by considering Hamming weight distributions and algorithms' convergence.

Besides confirming the observations from [19], the new set of experiments allowed us to discover two additional findings:

- We show that GP manages to find optimal solutions already in the initial population. This indicates that although decreasing ω limits the total number of optimal solutions, it still allows GP to "easily" guess some of those solutions. Thus, having smaller ω makes the problem simpler for GP, but not for GA, where we observed a trend of increasing difficulty similar to the one reported in [19]. Overall, these findings indicate that evolutionary algorithms are not needed to construct conserved landscape CA: a simpler and more effective way is to generate at random Boolean trees until one that maps to a conserved landscape rule is obtained.
- The Pareto fronts obtained with the multi-objective optimization approach indicate not only that the Hamming weight of an optimal solution must necessarily be low concerning the length of its truth table, but also that balanced generating functions are the farthest possible from giving reversible rules. This, in turn, allows us to further explain why for GA, it is extremely unlikely to guess an optimal solution by chance in the initial population, while it is easy for GP.

The rest of this paper is organized as follows. In Sect. 2, we discuss some types of cellular automata, and we provide some relevant definitions and notations. Section 3 presents related works. In Sect. 4, we discuss how to optimize the reversibility of CA, presenting an approach based on conserved landscape rules. Section 5 first presents the result of a preliminary exhaustive search and then provides details on our experimental setting and parameter tuning phase. In Sect. 6 we present the results of our evolutionary experiments, while in Sect. 7 we discuss them with respect to the stated research questions. Finally, Sect. 8 concludes the paper and offers potential directions for future research.

2 Cellular automata (CA)

This section covers background definitions and notions on reversible cellular automata, upon which the rest of the paper is based. We start with some general definitions, followed by discussions on reversible CA and marker CA.

2.1 Basic definitions

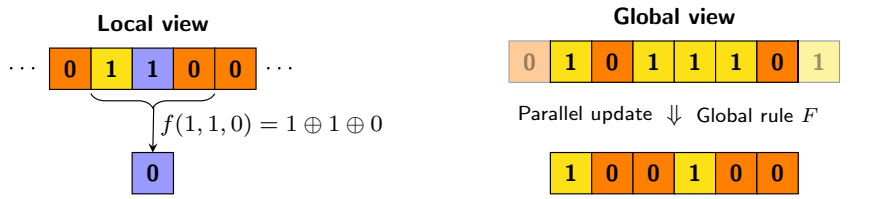
Let us denote by $A^{\mathbb{Z}}$ the set of all *bi-infinite strings* over the finite alphabet A. In the field of *symbolic dynamics* [21], the set $A^{\mathbb{Z}}$ is usually equipped with the *shift operator* σ, which shifts by one place to the left each coordinate of a bi-infinite string. For this reason, $A^{\mathbb{Z}}$ is also called the *full-shift space*. A mapping $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ is called *shift-invariant* if it commutes with, that is,

$$F(\sigma(x)) = \sigma(F(x)), \quad \text{for all } x \in \{0,1\}^{\mathbb{Z}},$$

for all bi-infinite strings $x \in A^{\mathbb{Z}}$.

Cellular Automata (CA) are a particular class of shift-invariant transformations whose output is determined by the parallel application of a single *local update rule* over all components (or *cells*) of a bi-infinite string. Such a rule depends only on a finite number of neighboring cells, also called the *diameter*. The *Curtis-Hedlund-Lyndon (CHL) Theorem* characterizes CA as those mappings $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ that are both shift-invariant and uniformly continuous concerning the Cantor distance [10]. When considering finite arrays with *periodic boundary conditions* (i.e., where the array can be seen as a "ring" in which the first cell follows the last one), instead of bi-infinite strings, the continuity requirement of the CHL theorem can be dropped. In other words, a mapping $F : A^n \to A^n$ is a CA if and only if F is shift-invariant. The only difference is that the shift operator in this context is applied *cyclically*. Thus the first cell will be shifted to the last one when σ is applied. Clearly, finite CA represent the most interesting case for practical applications, and we focus exclusively on them in the rest of this paper. Therefore, in what follows, we use the term CA and shift-invariant transformation interchangeably.

Various CA models can be defined depending on the dimension of the cellular array, the alphabet of the cells, and the boundary conditions. In this work, we focus on *one-dimensional periodic Boolean* CA, defined as follows:



(a) Local application of the local rule on the blue cell.

(b) Global application of the rule with periodic boundary conditions.

Fig. 1 An example of CA with $n = 6$ cells, diameter $d = 3$, offset $\omega = 1$, and local rule defined as $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$, corresponding to Wolfram code 150

Definition 1 A one-dimensional periodic Boolean CA (PBCA) of length n, diameter d, offset ω, and local rule $f : \{0,1\}^d \to \{0,1\}$, is defined as a vectorial function $F : \{0,1\}^n \to \{0,1\}^n$ where for every vector $x \in \{0,1\}^n$ and all $0 \leq i \leq n-1$, the i-th component of the output vector is given by:

$$F(x)_i = f(x_{[i-\omega, i-\omega+d-1]}) = f(x_{i-\omega}, x_{i-\omega+1}, \cdots, x_i, \cdots, x_{i-\omega+d-1}) \tag{1}$$

with all indices being computed modulo n. Function F is also called the *global rule* of the CA.

Thus, a PBCA is composed of a one-dimensional vector of n cells that can be either in state 0 or 1, where each cell simultaneously updates its state by applying the local rule f on the neighborhood formed by itself, the ω cells on its left and the $d-1-\omega$ cells on its right. Here, "periodic" refers to the fact that all indices are computed modulo n: in this way, the leftmost ω cells and the rightmost $d-1-\omega$ ones respectively have enough left and right neighboring cells to apply the local rule. Unless ambiguities arise, in what follows we refer to PBCA simply as CA, as the former is the main CA model considered in this work. The *orbit* of a PBCA starting from x is the sequence of vectors $\{x(t)\}_{t \in \mathbb{N}}$ where $x(0) = x \in \mathbb{F}_2^n$ and $x(t) = F^t(x)$ for all $t > 0$ (remark that F^t denotes the iteration of the CA global rule F for t times).

Since the cells of a CA take binary values, the local rule can be seen as a *Boolean function* $f : \mathbb{F}_2^d \to \mathbb{F}_2$ of d variables where $\mathbb{F}_2 = \{0,1\}$ is the finite field of two elements, and thus it can be represented by its *truth table*, which specifies for each of the possible 2^d input vectors $x \in \mathbb{F}_2^d$ the corresponding output value $f(x) \in \mathbb{F}_2$. Assuming that the input vectors of $\mathbb{F}_2^d$ are sorted lexicographically (i.e., $x \leq y$ if and only if $x_i \leq y_i$ where i is the first index such that x_i and y_i differ), one can encode the truth table as a single binary string $\Omega_f \in \mathbb{F}_2^{2^d}$, which is the output column of the table. In the CA literature, the decimal encoding of Ω_f is also called the *Wolfram code* of the local rule f [43]. Figure 1 reports an example of CA with $n = 6$ cells, diameter $d = 3$, offset $\omega = 1$, and local rule defined as $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$, corresponding to Wolfram code 150. Hence, each cell looks at itself and its left and right neighbors to compute its next state through rule 150. The two shaded cells in Fig. 1b represent "copies" respectively of the first and the last cell to help visualize the neighborhoods of the cells at the boundaries. As mentioned above, one can

Table 1 Truth table representation of local rule 150

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

effectively think of the CA array as a ring, bending it so that the leftmost and rightmost cells come close to each other.

As an example, Table 1 reports the truth table associated to the CA F with local rule 150.

2.2 Reversible CA

The property of reversibility is of particular importance in the field of dynamical systems. Stated informally, the orbits of the states of a reversible system are disjoint cycles without transient parts or pre-periods. Consequently, the dynamics of such systems can also be run backward in time since each state has exactly one predecessor, and the inverse system is analogous to the original one. In the context of infinite cellular automata, this property translates to the fact that the global rule F must be bijective to ensure that each global state of the cellular array has exactly one predecessor, and the inverse global mapping must also be a CA, that is, F^{-1} has to be defined by a local rule. If these two requirements are fulfilled, then the corresponding infinite CA is called *reversible*.

Hedlund [10] and Richardson [33] independently proved that an infinite CA is reversible if and only if its global rule is bijective. In other words, bijectivity in a CA is sufficient to grant the property that the inverse global rule F^{-1} is both shift-invariant and continuous. However, this result does not give constructive proof to find the inverse global rule F. Indeed, even characterizing the diameter of the inverse local rule in a reversible CA is still an open problem, as shown by Czeizler and Kari [6].

The relationship between bijectivity and reversibility is less straightforward in the case of finite CA. If we start from a local rule f that generates a reversible infinite CA, then we can conclude that the same rule will give rise to a reversible PBCA for any length $n \in \mathbb{N}$ of the cellular array. This is because the set of spatially periodic configurations is a proper subset of the full-shift space $A^{\mathbb{Z}}$, and it is exactly the subset where PBCA act upon. Conversely, if we know that a local rule f induces a bijective global rule on a PBCA of a certain length $n \in \mathbb{N}$, then the inverse global rule is not necessarily defined by a local rule, nor is it the case that the global rule stays bijective for different lengths of the PBCA under the same local rule.

Local rules that generate bijective global rules only for certain lengths $n \in \mathbb{N}$ of the CA array and whose inverses cannot be described by local rules are also called *globally invertible*. An example is the χ transformation used in the KECCAK sponge construction for hash functions [7], which corresponds to a CA of length $n = 5$ and is defined by the local rule of diameter $d = 3$ with the Wolfram code 210. The offset of this CA is $\omega = 0$, meaning that each cell applies rule 210 over itself and the two cells to its right to update its state. The algebraic expression of χ is:

$$\chi(x_1, x_2, x_3) = x_1 \oplus (x_2(1 \oplus x_3)). \tag{2}$$

In other words, the cell in position 1 flips its state if and only if the logical AND of x_2 and the complement of x_3 is true. Daemen [13] showed that rule 210 is globally invertible since it induces a bijective global rule only for odd lengths of the cellular array. In particular, the inverse mapping can be specified by a sequential algorithm that takes as input a vector of odd length and a "seed" value, which is basically a single component of the preimage. Then, the other components of the preimage are determined by "leaps" of length two by going leftwards with respect to the seed. Since the vector has an odd length and periodic boundary conditions, each of the remaining components can be determined using a single seed. The fact that a preimage seed can always be found for any configuration of odd length shows why the resulting CA is reversible. More details about the inversion procedure with seeds and leaps for rule χ can be found in [13].

On the other hand, a local rule that induces a bijective global function for all finite lengths $n \in \mathbb{N}$ of the cellular array is called *locally invertible*. Using a topological argument that relies on the compactness of the full shift space [15], it can be shown that locally invertible rules induce bijective global functions also on infinite CA. Hence, from the discussion above, it follows that locally invertible rules are exactly those defining reversible CA, where the inverse global rule F^{-1} is determined by a local rule for all lengths $n \in \mathbb{N}$ of the cellular array. In what follows, we consider searching for locally invertible rules as an optimization problem, focusing on the class of marker CA.

2.3 Marker CA

Up to now, only a few classes of reversible CA are known in the literature (see, e.g., [15]). These classes are usually defined in terms of particular properties of the local rule so that a subset of the rules satisfying them can generate a reversible CA. In this section, we describe the class of *marker CA* that are the focus of the main contributions of this paper in later sections.

A *marker CA* (also known as a *complementing landscape CA* [40]) can be defined as a CA having a local rule that always flips the bit of the cell in position ω (i.e., the one whose state is being updated) whenever the cells in its neighborhood take on a particular pattern, or *marker*. Otherwise, the cell stays in its current state. The set of patterns defining a local rule of a marker CA can be formalized through the concept of a *landscape*:

Definition 2 Let $d, \omega \in \mathbb{N}$ with $\omega < d$. A *landscape* of width d and center ω is a string $L = l_0 l_1 \cdots l_{\omega-1} \star l_{\omega+1} \cdots l_{d-1}$ where $l_i \in \{0, 1, -\}$ for all $i \neq \omega$.

The $\star$ symbol in a landscape L indicates the *origin* of the neighborhood in the local rule (i.e., the cell whose state is being updated), and consequently, occurring at position ω. The $-$ symbol represents a "don't care", meaning that the corresponding cell can be either in the state 0 or 1. Thus, landscapes can be considered as a restricted form of regular expressions over the binary alphabet $\{0, 1\}$, where the "don't care" symbol stands for the regular expression $(0 + 1)$ (i.e., both 0 and 1 match).

A local rule of a marker CA is described by one or more landscapes, all having the same width d and center ω. In the multiple landscape case, a cell is flipped if its neighborhood partakes on any of the patterns included in the union $\bigcup_{i=1}^{k} L_i$ of the landscapes $L_1, \cdots, L_k$ defining the local rule. For example, observe that the transformation χ used in Keccak, whose definition is recalled in Eq. (2), is a marker rule. Indeed, it can be seen that the cell x_1 flips its state if and only if x_2 and x_3 are equal to 1 and 0, respectively. Therefore, rule χ is defined by the single landscape $\star 10$.

It is possible to define a *partial order* $\leq_C$ over the set of landscapes. Namely, given two landscapes $L = l_0 \cdots l_{d-1}$ and $M = m_0 \cdots m_{d-1}$ with the same width d and center ω, we define

$$L \leq_C M \Leftrightarrow l_i = m_i \text{ or } l_i \in \{0, 1\} \text{ and } m_i = - \tag{3}$$

for all $0 \leq i \leq d - 1$. Intuitively, this partial order describes the "generality" of a landscape: the more "don't care" symbols it has, the more patterns it contains. The bottom of this partial order is the trivial landscape $\star$, which corresponds to the identity rule (i.e., each cell copies its state without looking at its neighbors). Above this minimal element are the *atomic landscapes*, which do not contain any "don't care" symbols, describing only single patterns. Finally, the top element is the landscape composed only of "don't care" symbols, which includes all possible patterns; the corresponding rule coincides with the complement of the identity, that is, the rule where each cell flips its state no matter what pattern its neighbors partake on. In what follows, we refer to $\leq_C$ as the *compatibility* partial order relation. In particular, we call two landscapes L_1, L_2 with the same width d and center ω *compatible* if $L_1 \leq_C L_2$ or $L_2 \leq_C L_1$. Otherwise, if L_1 and L_2 are not comparable with respect to $\leq_C$, we say that they are *incompatible*. As an example, Fig. 2 reports the diagram of the compatibility relation for $d = 3$ and $\omega = 0$.

The compatibility order relation can be used to characterize a subset of reversible marker CA, namely those of the *conserved landscape* type. In such CA, a cell that is in a particular landscape L defined by the local rule will still be in the *same* landscape upon application of the global rule. This property can be formalized by requiring that the cells in the neighborhood are in landscapes that are incompatible with L, as shown in the following result proved in [40]:

Lemma 1 *Let $f : \mathbb{F}_2^d \to \mathbb{F}_2$ be a local rule of a marker CA defined by a set of k landscapes $L_1, \cdots, L_k$ of width d and center ω. Further, for all $i \in \{1, \cdots, k\}$ let*

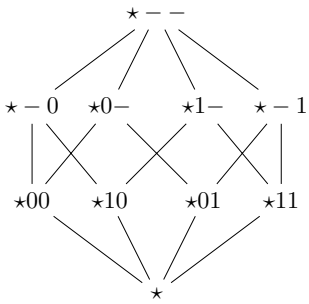


Fig. 2 Hasse diagram for the compatibility poset (partially ordered set) with $d = 3$ and $\omega = 0$. The landscape $\star 10$ defines the rule χ introduced in [13]

$M_{i,0}, \cdots, M_{i,\omega-1}, M_{i,\omega+1}, \cdots, M_{i,d-1}$ be the set of $d-1$ landscapes associated to the neighborhood of L_i. Then, if $M_{i,j}$ is incompatible with all landscapes $L_1, \cdots, L_k$ for all $i \in \{1, \cdots, k\}$ and $j \in \{0, \cdots, \omega-1, \omega+1, \cdots, d-1\}$, rule f induces a locally invertible marker CA.

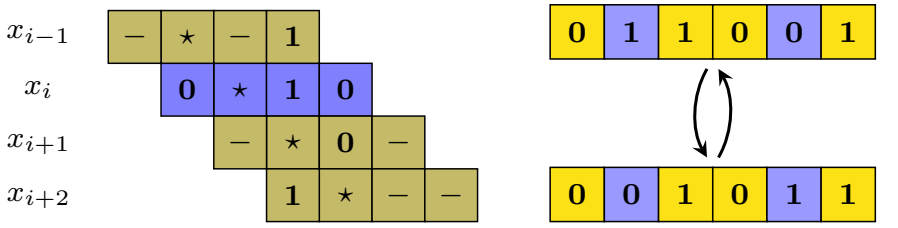
When the conditions of Lemma 1 are fulfilled, f is named a *conserved landscape* rule. Toffoli and Margolus noted that a conserved landscape local rule induces an *involution*, i.e., the global rule of the resulting marker CA is its inverse [40]. This is because any cell being in one of the marker landscapes will still be in the same landscape after applying the local rule. Therefore, after a further application of the local rule, the cell will return to its initial state.

Conserved landscape rules define a particular type of reversible CA since all cycles have a length of 2. Daemen argued that such CA could be useful in those cryptographic applications where both the encryption and decryption functions are implemented in hardware [13]. It is also possible to relax the conditions of Lemma 1 by allowing the landscapes of the local rule to partially *overlap* one another [40]. In this case, a cell in a landscape defined by the local rule will be in any other landscape defined by the local rule after applying the global rule. As a consequence, the resulting marker CA can exhibit more complex behaviors with longer cycle lengths.

To better illustrate the idea, we provide an example of the only single conserved landscape rule of diameter $d = 4$ (up to complement and reflection of the input), originally discovered by Patt [30]:

Example 1 Let $d = 4$ and $\omega = 1$, and let $f : \mathbb{F}_2^4 \to \mathbb{F}_2$ be the local rule defined by the single landscape $L = 0 \star 10$. The tabulation depicted in Fig. 3a shows that all three landscapes of the neighboring cells are incompatible with L. In particular, when x_i is in landscape L, then:

1. Cell x_{i-1} is in landscape $- \star -1$, which is incompatible with $0 \star 10$ as there is a mismatch in position 3.
2. Cell x_{i+1} is in landscape $- \star 0-$, which is incompatible with $0 \star 10$ as there is a mismatch in position 2.
3. Cell x_{i+2} is in landscape $1 \star --$, which is incompatible with $0 \star 10$ as there is a mismatch in position 0.



(a) Landscape tabulation for rule $0 \star 10$. **(b)** Example of cycle of length 2.

Fig. 3 A locally invertible CA defined by the single landscape $0 \star 10$. Fig. 3b displays an example of a cycle starting from the initial state 011001. The two cells in blue are in the landscape $0 \star 10$

3 Related works

As far as we are aware, our work is the first one exploring the application of evolutionary algorithms to evolve reversible CA. Therefore, in this section, we briefly discuss related works related to the use of EA to evolve shift-invariant transformations and related objects for other tasks, such as random number generation. For a somewhat outdated but very detailed overview of works using GA to evolve CA, we refer interested readers to [22].

Bäck and Breukelaar used genetic algorithms to evolve behavior in CA and explored different neighborhood shapes [38]. The authors showed that their approach works for different topologies and neighborhood shapes. Sipper and Tomassini [23] proposed a cellular programming algorithm to co-evolve the rule map of non-uniform CA for designing random number generators. With their approach, the authors managed to evolve good generators that exhibit behaviors similar to those from the previously described CAs. Additionally, the authors reported advantages stemming from a "tunable" algorithm for obtaining random number generators.

Picek et al. demonstrated that GP could be used to evolve CA rules suitable to produce S-boxes (nonlinear elements used in block ciphers) with good cryptographic properties [34]. This approach allowed finding optimal S-boxes for several sizes of practical importance. Interestingly, this is the first time that EA has managed to obtain optimal S-boxes for larger sizes. Next, Picek et al. used genetic programming to demonstrate that the S-boxes obtained from the CA rules could have good implementation properties [35]. The authors concentrated on two S-box sizes, 4×4 and 5×5, and managed to find S-boxes with good latency, area, and power consumption. Subsequently, Mariot et al. conducted a more detailed analysis of the S-boxes based on CA, and they proved the best possible values for relevant cryptographic properties when CA rules of a certain size are used [25]. The authors also used GP to experimentally validate their findings and reverse engineer a CA rule from a given S-box.

Mariot et al. used EA to construct orthogonal Latin squares built from CA [18]. The authors reported that GP could always generate orthogonal Latin squares, where the optimal solutions were mostly linear. On the other hand, when using GA, the results were significantly worse than GP in evolving orthogonal Latin squares, but

the corresponding Boolean functions were always nonlinear. Finally, Mariot et al. investigated the possibility of evolving Reversible Cellular Automata (RCA). The authors considered three optimization strategies and obtained good results [19].

We note that the evolution of CA rules for cryptographic purposes is connected with the evolution of Boolean functions with good cryptographic properties. This direction is rather well-explored, and there are multiple works considering various evolutionary approaches, see, e.g., [41, 17, 31].

4 Optimizing the reversibility of CA

This section cast the search of reversible CA as an optimization problem that can be tackled with evolutionary algorithms, focusing on the class of conserved landscape rules.

Lemma 1 tells us that to find a marker CA that is locally invertible, we need to define a set of landscapes $L_1, \cdots, L_k$, in such a way that their associated neighborhood landscapes are incompatible with them. This suggests the following idea to turn the search of conserved landscape rules into an optimization problem: given the landscape specification of a local rule, *count* the number of compatible landscape pairs, and minimize it. Using the partial order relationship that we defined in Sect. 2.3, this is equivalent to minimize the number of comparable pairs of landscapes. An optimal solution is a set of landscapes that are all mutually incompatible (including the neighborhood landscapes), or equivalently an *antichain* of elements in the poset induced by $\leq_C$. These observations lead to the following optimization problem:

Problem 1 Let $d, \omega \in \mathbb{N}$ with $0 < \omega < d - 1$. Find a set of landscapes $L_1, \cdots, L_k$ of width d and center ω, such that for all $i \in \{1, \cdots, k\}$ and for all $j \in \{0, \cdots, d-1\}$, the neighborhood landscape $M_{i,j}$ associated to L_i is incompatible with all other landscapes $L_1, \cdots, L_k$, that is $M_{i,j} \not\leq_C L_t$ and $L_t \not\leq_C M_{i,j}$ for all $t \in \{1, \cdots, k\}$.

In the rest of this section, we will first address how to obtain the landscape representation of a local rule from its truth table, and then we will define the fitness function to be minimized for Problem 1.

4.1 Genotype representation for marker CA

The first question arising from Problem 1 is how to represent local rules of marker CA so that they can be evolved by the variation operators of GA and GP. In particular, GA usually works on a bitstring encoding of the candidate solutions of an optimization problem, while GP relies on a tree representation. Hence, directly using the landscape specification of a marker CA rule does not seem a natural choice for encoding the genotype.

Recall from Eq. (2) that the χ rule was defined as the XOR of the leftmost cell (which also coincides with the cell being updated) with the AND between the second

cell and the complement of the third cell. This observation can be generalized to any local rule of marker CA as follows. Let $L_1, \cdots, L_k$ be a set of landscapes of diameter d and center ω defining a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. Additionally, let $\mathcal{L} = \bigcup_{i=1}^{k} L_i$ be the union of the landscapes. Then, a cell x_i in a marker CA equipped with rule f will flip its state if and only if the neighborhood $x_{i-\omega} \cdots x_{i-1} \star x_{i+1} \cdots x_{i+d-1-\omega}$ belongs to $\mathcal{L}$. Excluding the origin $\star$ of the neighborhood, we obtain a vector of $d-1$ variables that describes the states of the cells surrounding x_i. Consider now all 2^{d-1} possible assignments to this vector, and let $g : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ be the Boolean function defined as:

$$g(x_{i-\omega} \cdots x_{i-1} x_{i+1} \cdots x_{i+d-1-\omega}) = \begin{cases} 1, & \text{if } x_{i-\omega} \cdots x_{i-1} \star x_{i+1} \cdots x_{i+d-1-\omega} \in \mathcal{L} \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

for all $x_{i-\omega} \cdots x_{i-1} x_{i+1} \cdots x_{i+d-1-\omega} \in \mathbb{F}_2^{d-1}$. In other words, function g outputs 1 if and only if the configuration featured by the cells surrounding x_i belongs to the union of landscapes $\mathcal{L}$, when the origin $\star$ is inserted at position ω. Then, it follows that the local rule f can be expressed as

$$f(x_{i-\omega} \cdots x_{i-1} x_i x_{i+1} \cdots x_{i+d-1-\omega}) = x_i \oplus g(x_{i-\omega} \cdots x_{i-1} x_{i+1} \cdots x_{i+d-1-\omega}) \quad , \tag{5}$$

for all configurations of $x_{i-\omega} \cdots x_{i-1} x_i x_{i+1} \cdots x_{i+d-1-\omega} \in \mathbb{F}_2^d$. Hence, the algebraic form of the local rule of a marker CA can be expressed as the XOR of the cell in the origin with the *generating function* g computed on the surrounding cells. Indeed, g evaluates to 1 if and only if the neighborhood takes on any of the landscapes in $\mathcal{L}$, and in this case x_i will flip its state.

Consequently, we can reduce the representation of the local rule f of a marker CA to its generating function g, since we can compute f by simply XORing the output of g with the value of x_i. Since g can be any Boolean function of $d-1$ variables, it follows that we can represent the genotype of a candidate solution to our optimization problem with the commonly used Boolean genotype encodings for GA and GP. In particular:

- For GA, the genotype of a candidate solution is a bitstring of length 2^{d-1}, representing the output of the truth table of g.
- For GP, the genotype is a tree where the terminal nodes represent the input variables of g (i.e., the state of the cells surrounding the origin of the neighborhood), while the internal nodes are Boolean operators combining the values received from their child nodes and propagating their output to their parent node. The output of the root node will be the output of the whole generating function g.

4.2 Fitness functions

At the beginning of this section, we informally introduced the idea to steer the search of conserved landscape rules by counting the number of compatible pairs of landscapes. However, given that the genotype handled by GA and GP is an encoding of the generating function g, we first need to translate this representation to the landscape specification.

Suppose that we have the truth table of the generating function g. In the GA case, this corresponds exactly to the genotype of an individual. For GP, we can easily recover it by evaluating the Boolean tree of an individual over all possible input vectors $x \in \mathbb{F}_2^{d-1}$. Let $supp(g) = \{x \in \mathbb{F}_2^{d-1} : g(x) \neq 0\}$ be the *support* of g, i.e., the set of input vectors over which g evaluates to 1. By construction, the elements of $supp(g)$ coincide with all the patterns that the cells surrounding the origin must feature to flip the state of the central cell. Thus, to obtain the list of atomic landscapes, it suffices to insert the origin symbol $\star$ in position ω to each vector of the support. Of course, some of these patterns could be described in a more "compact" way with more general landscapes that also use the "don't care" symbol. For example, if $supp(g) = \{101, 111\}$ and the center is $\omega = 1$, then the two atomic landscapes $1\star01$ and $1\star11$ can be described by the single landscape $1\star-1$, where we substituted the central variable with a "don't care" symbol[1].

However, the set of atomic landscapes obtained from the support suffices to check if a rule is of the conserved landscape type or not. It is not difficult to see that two landscapes containing "don't care" symbols are incompatible if and only if all the atomic landscapes that they describe are incompatible between themselves. This means that we can directly use the support of the generating function to *count* the number of pairs of compatible landscapes. Given that we want to minimize such a number in order to get a conserved landscape rule, we define the following objective function:

Definition 3 Let $g : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ be a generating function of a marker CA rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ of diameter d and offset ω, and let $supp(g)$ be its support. Further, let $L_1, \cdots, L_k$ be the set of atomic landscapes obtained by adding the origin symbol $\star$ in position ω to each vector in $supp(g)$, and for each $i \in \{1, \cdots, k\}$ let $M_{i,0}, \cdots, M_{i,\omega-1}, M_{i,\omega+1}, \cdots, M_{i,d-1}$ be the set of neighborhood landscapes associated to L_i, obtained through the tabulation procedure. Then, the fitness function value of g is defined as follows:

$$obj_1(g) = \sum_{i \in [k], j \in [d-1]_\omega} \sum_{t \in [k]} comp(M_{i,j}, L_t) \quad , \tag{6}$$

where $[k] = \{1, \cdots, k\}$, $[d-1]_\omega = \{0, \cdots, \omega - 1, \omega + 1, \cdots, d-1\}$, and the function $comp(\cdot, \cdot)$ returns 1 if the two landscapes passed as arguments are compatible, and 0 otherwise.

Hence, the objective function loops over all neighborhood landscapes $M_{i,j}$ induced by each atomic landscape L_i, compares each of these neighborhood landscapes with all atomic landscapes $L_1, \cdots, L_k$ through the function $comp(\cdot, \cdot)$, and

[1] This method can be generalized using the following greedy procedure. Let $supp(g)$ be the support of the generating function, and remove $x, y \in supp(g)$ such that their Hamming distance is 1. Then, insert in $supp(g)$ the landscape L that has the same symbols as x and y, except for the single position in which they differ, where L has a "don't care" symbol $-$. Repeat this procedure until no further replacements can be performed (i.e., all pairs of landscapes in $supp(g)$ are at Hamming distance higher than 1).

adds 1 whenever a compatible pair is found. Therefore, the function obj_1 measures the degree of compatibility of a set of atomic landscapes induced by the support of a generating function g. The optimization objective is thus to *minimize* obj_1, with $obj_1(g) = 0$ corresponding to an optimal solution where all neighborhood landscapes are incompatible with the atomic landscapes, and thus the latter define a conserved landscape rule.

Secondly, a good indicator of the complexity of the dynamical behavior of a marker CA is the *Hamming weight* of its generating function g, i.e., the cardinality of its support. This metric can also be used as a proxy for the utility of a marker CA in cryptography since it is related to the *nonlinearity* of the resulting vectorial Boolean function [3, 39]. Given a generating function g, we thus define a second optimization objective function as follows:

$$obj_2(g) = |supp(g)|. \tag{7}$$

In the optimization of these objectives, we experimented using three optimization scenarios. The first one, which we denote as the *single-objective* scenario, included only the minimization of the reversibility objective. The fitness function for the first scenario is then simply defined as:

$$fit_1(g) = obj_1(g), \tag{8}$$

where the optimization goal is minimization.

As it became apparent quite early in our experiments that this goal is very easily attainable with both representations, we modified the fitness function so the evolution could generate more distinct solutions with different Hamming weights. This is made possible simply by maximizing the Hamming weight value, but only for solutions that already obtained a conserved landscape solution, i.e., those for which the first objective is already minimized. At the same time, whenever an algorithm reaches a solution with a higher Hamming weight, every such individual is added to a set of distinct solutions reported at the end of each run.

Therefore, in the second scenario, which is denoted as *lexicographic optimization*, we are interested in maximizing the Hamming weight while retaining an optimal value of obj_1. For this reason, we define a second fitness function for this particular case as follows:

$$fit_2(g) = \begin{cases} obj_1\,, & \text{if } obj_1 > 0, \\ -obj_2\,, & \text{if } obj_1 = 0. \end{cases} \tag{9}$$

Stated otherwise, with the second fitness function, we still minimize obj_1 until we reach a reversible rule, and after that, we minimize the opposite of the Hamming weight (thus, equivalently, we are maximizing obj_2).

Finally, we included a multi-objective approach to investigate the interaction between the reversibility of a marker CA rule and the Hamming weight of its generating function. In the *multi-objective* scenario, we *minimized* the reversibility objective obj_1 and *maximized* the Hamming weight as defined by obj_2 in Equation (7).

5 Experimental evaluation

In this section, we present the experimental setting and results obtained by applying GA and GP on Problem 1. We start by performing an exhaustive exploration of all conserved landscape rules up to diameter $d = 6$, which is still computationally feasible. Next, we use the findings obtained from the exhaustive search to formulate our research questions and lay down our experimental settings. Finally, we present the results of our parameter tuning and discuss them in light of our research questions.

5.1 Preliminary exhaustive search

As noted in Sect. 4.1, the local rule of a marker CA of diameter d can be identified with its generating function g of $d-1$ variables, computed on the neighborhood cells surrounding the origin since the state of the central cell is XORed with the result of g. Given a diameter $d \in \mathbb{N}$, this means that we can define the phenotype space as the set $\mathcal{P}(d) = \{g : \mathbb{F}_2^{d-1} \to \mathbb{F}_2\}$ of all Boolean functions of $d-1$ variables. The genotype space, on the other hand, will correspond to the set of all binary strings of length 2^{d-1} specifying the truth tables Ω_g of the generating functions in $\mathcal{P}(d)$. For GP, it will be the space of all Boolean trees whose terminals represent the $d-1$ input variables, and the internal nodes represent Boolean operators.

Since the number of Boolean functions of $d-1$ variables is $2^{2^{d-1}}$, the phenotype space $\mathcal{P}(d)$ can be exhaustively searched for reversible marker CA rules up to diameter $d = 6$, since there are at most $2^{32} \approx 4.3 \cdot 10^9$ generating functions to check for the conserved landscape property. As far as we know, an exhaustive search of reversible marker CA rules has been carried out only by Patt [30], who considered diameters up to $d = 4$. For completeness, Table 2 reports the numbers of conserved-landscape rules we found by exhaustively searching the sets of generating functions up to $d = 6$ for each possible value of ω, along with the length of the truth table (2^{d-1}), the size of the phenotype space ($\#\mathcal{P}(d)$), and the observed Hamming weights. Recall that the Hamming weight of the generating function corresponds to the number of atomic landscapes over which a cell flips its state. We excluded from the count the identity rule, which copies the state of the central cell since it is trivially reversible for any diameter. Further, we halved the numbers of the remaining rules since, if a rule is of the conserved landscape type, then its complement is too. As a general remark, one can see from Table 2 that the number of conserved landscape rules is much smaller than the size of the whole generating function set for any offset ω. Another interesting observation is that the highest numbers of conserved landscape rules are always found when ω corresponds to the center of the neighborhood or its immediate left or right (if d is even). Indeed, the extreme cases are $\omega = 0$ and $\omega = d$, where no conserved landscape rules exist. As noted in [13], if the offset is on either the leftmost or rightmost cell of the neighborhood, then any landscape is always compatible with at least another one. Also, the fact that the distributions of conserved landscape rules are symmetrical to the center of the neighborhood is backed by the results proved in [40], where reversible marker rules in different offsets are shown to be

Table 2 Numbers of conserved landscape rules found by exhaustive search, up to equivalence by complement and excluding the trivial identity rule. Last column reports the approximate time (in seconds) needed to span the entire search space for each ω

d	2^{d-1}	$\#\mathcal{P}(d)$	ω	#REV	Weights	Time
4	8	256	0	0	–	~0.1s
			1	1	1	
			2	1	1	
			3	0	–	
5	16	65 536	0	0	–	~120s
			1	2	1	
			2	5	1, 2	
			3	2	1	
			4	0	–	
6	32	$4.3 \cdot 10^9$	0	0	–	$\sim 2.3 \cdot 10^5$s
			1	8	1, 2	
			2	23	1, 2, 3	
			3	23	1, 2, 3	
			4	8	1, 2	
			5	0	–	

symmetric under rotations and reflection. Further, the number of the observed Hamming weights is quite limited since, for the largest considered instance of diameter $d = 6$, we only found reversible rules defined by at most three landscapes, which are thus not very useful for cryptographic and reversible computing purposes. Finally, a remark on time required to span the space of marker rules completely is in order. We carried out exhaustive search experiments on a Linux machine with an AMD Ryzen 7 1800X Eight-Core Processor running at a base clock of 3.6 GHz. As reported in the last column of Table 2, the time required to enumerate all conserved landscape rules grows very rapidly, starting from only 0.1 seconds for diameter $d = 4$ up to more than two days for $d = 6$. This completely rules out the possibility to perform an exhaustive search for larger diameters, since, for $d = 7$, the corresponding search space is already composed of $2^{64} \approx 1.84 \cdot 10^{19}$ rules. Even taking into account that the invertibility of a single rule of diameter $d = 7$ can be checked in about 10^{-5} seconds (based on our current implementation), it would still take several million years to exhaustively visit the search space. Thus, in this case, not even a massive parallelization of the enumeration algorithm would help.

5.2 Research questions

The empirical observations obtained from the exhaustive search experiments presented in the previous section prompted us with three research questions:

- **RQ1:** Does the limited number of conserved landscape rules with respect to the search space size imply a difficulty for evolutionary algorithms to find them?

- **RQ2:** Do there exist conserved landscape rules of a larger diameter that are useful for cryptographic and reversible computing applications, i.e., having larger Hamming weights with respect to the size of the generating function truth table?
- **RQ3:** Is there a trade-off between the reversibility of a marker CA rule (as measured by the objective function obj_1 defined in Sect. 4.2) and its Hamming weight (as defined by the second objective obj_2)?

Although these research questions are inherited from the conference version of this work [19], we emphasize that here they are explored from a different perspective, especially concerning the first two. In particular, in our previous conference paper, the offset ω was fixed to the neighborhood center, i.e., $\omega = \lfloor (d-1)/2 \rfloor$. The reason for that choice was that, as shown in Table 2, most of the reversible rules are found when the offset is closer to the center.

On the other hand, in this work, we consider the situation where the offset is fixed to $\omega = 3$ for the experiments described in the next sections. The reason is twofold: first, by keeping the offset to a fixed value, one could reasonably expect that the difficulty for evolutionary algorithms to converge to an optimal solution increases even more by considering larger diameters than by placing ω near to the center. Indeed, increasing the diameter while keeping ω fixed means that the origin of the landscape rules gets farther from the center. Consequently, as experimentally observed through an exhaustive search, the number of optimal solutions becomes smaller, and this, in turn, likely affects the answers to RQ1 and RQ2 as discussed in our conference paper [19]. Further, in principle, one may assume that the trade-off between the compatibility fitness and the Hamming weight could change by considering an offset far from the center of the neighborhood, potentially affecting the answer to RQ3. Finally, the second reason for choosing a fixed ω in our investigation is more of a practical nature: in this way, we can adopt a more uniform experimental setting, especially concerning the parameter tuning phase described in Sect. 5.4.

5.3 Experimental settings

We utilized a genetic algorithm with truth table encoding and genetic programming with a tree-based representation to investigate the stated research questions. Both representations use the same selection scheme, a steady-state elimination tournament; in each iteration, three individuals are randomly selected from the population. A new solution is generated by applying crossover to the best two individuals from the tournament. The new individual undergoes mutation, subject to a predefined individual mutation rate, which is an algorithm parameter. Finally, the new individual replaces the worst one from the tournament, and the process is repeated. Each iteration produces one new individual and performs a single fitness evaluation. Apart from the described method, we also experimented with an evolutionary strategy-based scheme, in which a number of offspring is generated using mutation only; however, preliminary experiments showed that this selection method produced inferior results for both representations. In the multi-objective approach, we used the well-known NSGA-II algorithm [4].

For the truth table binary representation (GA), we employed one-point, two-point, and uniform crossover operators selected at random at each iteration. The mutation operator was a single bit-flip on a randomly selected position. Five crossover operators are used at random in the tree-based representation (GP): simple subtree crossover, uniform crossover, size fair, one-point, and context preserving crossover. As in the GA case, a single mutation type was used, the subtree mutation, with a fixed mutation probability of 0.5 [32].

The function set used in the tree-based encoding included the binary operators AND, OR, XOR, XNOR, AND with the second input complemented, and the unary operator NOT. Additionally, we included the ternary function IF, which returns the second argument if the first one is true and the third one otherwise. We performed a tuning phase to investigate which subset of these functions provides the best results.

For each considered optimization scenario (single-objective where fit_1 is minimized, multi-objective where obj_1 and obj_2 are respectively minimized and maximized, and lexicographic optimization where fit_2 is minimized) we performed our experiments on the spaces of marker CA rules with diameter $7 \leq d \leq 15$. Therefore, with respect to our previous results reported in [19], we extended our investigation with two additional diameter values. Each experiment was repeated for 50 independent runs to obtain statistically reliable results, and each run was given a budget of 500 000 evaluations, which is the same as adopted in [19]. Indeed, as it will be clear in the next sections, such a budget proved to be more than sufficient to investigate our research questions, and we deemed unnecessary a larger one.

5.4 Parameter tuning

To set up the different parameters of the evolutionary algorithms employed for our experiments, we performed a tuning phase on the instance of marker CA rules with diameter $d = 10$ and $\omega = 3$. Recall, in our previous experiments presented in [19], we carried out this phase on $d = 7$ and $\omega = 3$. While we already elaborated in Sect. 5.2 why we chose an asymmetric offset for all our experiments, we tuned our evolutionary algorithms on a larger problem instance mainly for robustness reasons. Indeed, $d = 7$ is the smallest instance where it makes sense to tune an evolutionary algorithm for this problem since, for smaller values, the search space is limited enough that the problem can be easily solved by exhaustive search, as discussed in Sect. 5.1. Moreover, in this case, $\omega = 3$ corresponds to the center of the neighborhood. Hence, we selected $d = 10$ as a sufficiently representative instance of our new experimental setting since the offset is far enough from the peak of the distribution of optimal solutions occurring in the center.

We tuned the population size p and the mutation probability μ in the GA case. In particular, the population size ranged among the values $\{100, 200, 500\}$, while the mutation probability was in the range $\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

For GP, besides the population size in the same range as GA, we tuned the maximum depth of the trees, considering the values in $\{5, 7, 9, 11\}$. The motivation is that in our experiments in [19] we always set the maximum depth equal to $d - 1$, mainly following a heuristic adopted in previous works on the optimization of Boolean

functions [41, 17, 31]. However, one could argue that if using these methods for large diameters such as $d \geq 10$, one could end up with very large trees, eventually making the investigation of the evolved solutions for interpretability more difficult. Likewise, one could also argue that a larger maximum depth could be beneficial to converge more rapidly on an optimal solution. For this reason, we experimented with both smaller and greater maximum depth with respect to the initial value $d - 1$.

Finally, the third parameter that we tuned for GP is the set of Boolean operators used in the internal nodes of the trees. In [19], we used a function set composed of four binary operators (AND, OR, XOR, and XNOR), one unary operator (NOT), and one ternary operator (IF). Again, the motivation for this choice was the previous experience with optimization problems related to Boolean functions solved using GP [41, 17, 31]. However, as remarked by one of the reviewers of [19], such a set could easily induce the GP trees to bloat since, for example, XNOR is equivalent to the composition of NOT and XOR. Although bloat was already controlled in our previous experiments by adopting the maximum depth parameter, we decided to investigate this question more thoroughly by tuning the set of GP operators. Remark that a composition of operators equivalent to the identity function is not the only source of bloat in GP. Several theories about the origins of bloat have been proposed during the past decades, most of which are not directly related to the underlying set of terminals and functionals (see e.g. [36] for a review of them). However, it is reasonable to assume that having a redundant set of operators could contribute further to the bloat of evolved individuals. Hence it makes sense to search for a minimal set that still gives good results.

To this end, we started with a minimal set of operators such that a combination of them can express any Boolean function, i.e., AND, OR, and NOT. Then, we added to this minimal set the combinations of XOR, XNOR, AND with the second input complemented, and IF, retaining only the combinations that significantly improved the results.

For both GA and GP tuning, each parameter combination was tuned with a fitness budget of 100 000 evaluations, repeated in 30 independent runs for statistical significance purposes. In particular, we used a smaller budget of fitness evaluations and independent runs than in the main experimental evaluations presented in the next section, since here, we are considering only the tuning of the parameter. After each run, the fitness value of the best individual was recorded, thus obtaining a sample of 30 observations that approximated the distribution of the best fitness for a particular parameter combination. Moreover, to select the parameter combinations to be used in our subsequent experiments, we performed a two-stage statistical analysis with non-parametric tests. First, we used the *Kruskal-Wallis* test [16] to compare a group of parameter combinations all at once, using a significance level of $\alpha = 0.05$. If no significant differences were observed, then another criterion for selecting the parameter combination to be used among those in the group was adopted (i.e., highest median). On the other hand, if the distributions were detected to be significantly different, we employed the *Mann-Whitney U* test [24] to perform pairwise comparisons and determine the best parameter combination. In particular, the null hypothesis for the test was that the random variable of the fitness represented by the first distribution was better than that of the second distribution. Here, the definition of "better"

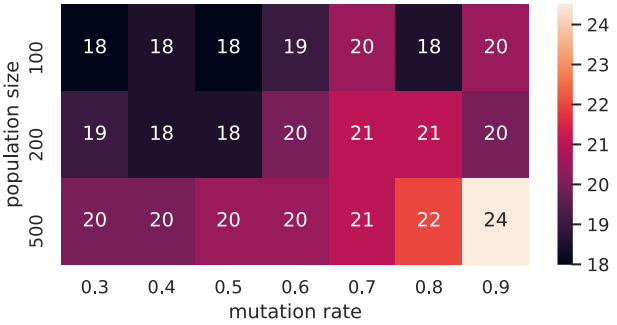


Fig. 4 Heatmap for the tuning phase of LEXGA. The numbers inside the cells refer to the median fitness obtained by the best individual across all experimental runs

depends on the context: when we performed the tuning for the single-objective versions of our algorithms where only the reversibility fitness function is used, then better corresponds to lower fitness values. On the other hand, for the lexicographic optimization approach, the objective is to maximize the Hamming weight while retaining reversibility. Hence, in this case, it better corresponds to the higher Hamming weight values. The significance level was again set to $\alpha = 0.05$, applying *Bonferroni correction* [5] since we performed multiple comparisons.

5.4.1 GA tuning

For the GA tuning, we performed a complete sweep across all $3 \times 7 = 21$ parameter combinations for population size and mutation rate, considering both the single-objective case (SOGA), where only fit_1 is minimized, and the lexicographic optimization approach (LEXGA), where fit_2 is minimized. Concerning SOGA, *no differences were detected* during the parameter sweep: indeed, for each considered parameter combination, the best solution always reached an optimal fitness in all 30 experimental runs. For this reason, we focused only on the lexicographic optimization approach, adopting the same parameter combination selected for LEXGA also for SOGA.

Figure 4 depicts the heatmap of the median best fitness obtained by LEXGA across all 21 parameter combinations of population size and mutation probability. We only show the Hamming weight being maximized as the second objective since the first objective was optimal in every case. The color gradient already indicates an advantage in using large populations and high mutation rates. Indeed, after performing the Kruskal-Wallis test for all 21 distributions, significant differences were detected. For this reason, we proceeded by performing pairwise comparisons through the Mann-Whitney U test. As a criterion to select the best parameter combination, we used a ranked tournament: each distribution was compared against all others, and if the Mann-Whitney U test rejected the null hypothesis (that is, the obtained p-value was below the corrected significance level), then a +1 was scored by the distribution, and the distribution scoring the highest number of points was then selected as a winner. This resulted

in the combination $p = 500$ and $\mu = 0.9$, since it achieved 20 points (i.e., it had significant differences against all other combinations), while the second-best ones reached a consistently lower score equal to 7. Incidentally, this analysis also confirmed the result suggested by the heatmap, although only the median best fitness was considered there. Therefore, both for LEXGA and SOGA, we selected a population size of 500 individuals and a mutation probability of 0.9.

5.4.2 GP tuning

The number of parameter values to test for GP was 3 for the population size, 4 for the maximum depth, and 7 for the subsets of operators. Checking all 84 parameters combinations resulting from a grid search, as in the case of GA, would have implied a too large computational effort. Therefore, we decided to opt for a lexicographic tuning approach: first, we determined the best maximum depth among $\{5, 7, 9, 11\}$ by keeping the population size fixed to 100 individuals and using the minimal operators set of AND, OR, NOT. Then, we used the best maximum depth values and the same set of operators to tune the population size. Finally, we tuned the operators set by using the selected best population size and maximum depth.

Similar to the GA tuning, in the single-objective scenario (SOGP), no differences were observed since, in all the configurations, GP always obtained the optimal solution in every algorithm run. Therefore, we used the lexicographic scenario (LEXGP) to estimate the appropriate set of parameters.

Concerning the first phase (maximum depth tuning), significant differences were detected with the Kruskal-Wallis test on the set $\{5, 7, 9, 11\}$. Using the Mann-Whitney U test with a ranked tournament as in the case of LEXGA, the best values for this parameter were 7 and 9, with no significant differences between them. For this reason, we kept them both for the next phase, where we analyzed all combinations of parameters for maximum depth in $\{7, 9\}$ and population size in $\{100, 200, 500\}$. Once again, significant differences resulted from applying the Kruskal-Wallis tests on such distributions. Using the ranked tournament approach for the pairwise comparisons with the Mann-Whitney U test, we obtained 4 remaining combinations, each achieving the same score. Among these 4 remaining combinations, we selected the one with the highest median best fitness, i.e., 500 individuals for the population size and maximum depth of 9. Finally, for the last phase, where the tuning was performed by adding operators to the minimal set, no significant differences arose from the Kruskal-Wallis test. Hence, we again selected the combination with the highest median fitness. The final parameters combination selected for both LEXGP and SOGP was $p = 500$, $d = 9$ and operator set including AND, OR, NOT, and AND with second input complemented. In particular, since the selected maximum depth turned out to be equal to 9 while the tuning diameter was 10, we kept $d - 1$ as a maximum depth for all other instances in the subsequent experiments, as done in our previous work [19].

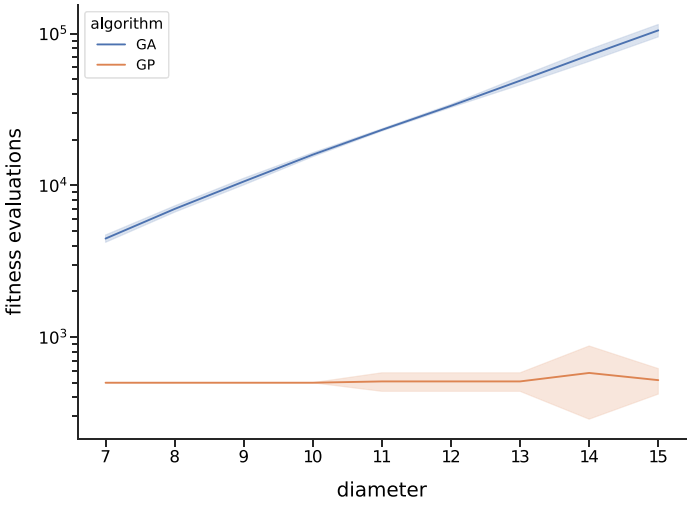


Fig. 5 Comparison of fitness evaluations performed by SOGA and SOGP, in logarithmic scale. The error bands represent the standard deviation

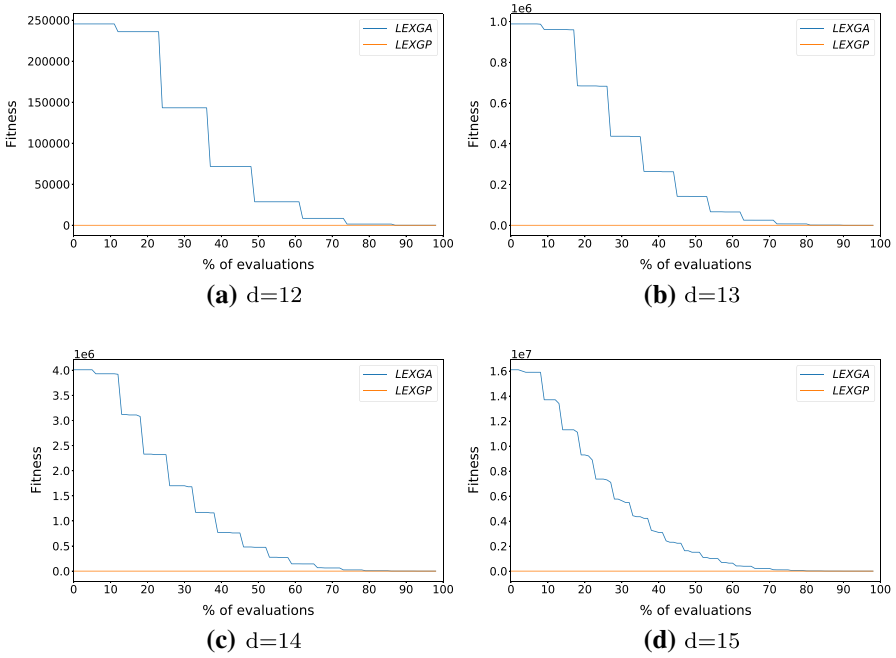
6 Results

In this section, we present the results emerging from our experimental evaluation. First, we discuss the results for single-objective optimization, followed by those on multi-objective and lexicographic optimization. Finally, we analyze the diversity of the CA rules obtained in our experiments by using several metrics related to the number of unique solutions and the different Hamming weights.

6.1 Single-objective optimization results

Figure 5 gives results for the single-objective GA and GP considering the number of evaluations needed to reach the optimal value. For GP, we reach optimal fitness value for each dimension already in the initial population, making the results less interesting. A possible reason GP shows such a behavior is that it is easier to guess a generating function that results in a reversible marker CA rule with a random algebraic expression than with a random string of bits, as in the case of GA. Still, we require somewhat more evaluations for larger dimensions, indicating that it becomes slightly more difficult to guess optimal solutions randomly. We address this phenomenon and its consequences more in detail in Sect. 7.

For GA, we observe an exponential increase in the number of required fitness evaluations concerning the diameter sizes (remark that the fitness evaluations axis in the plot is in logarithmic scale). This indicates that larger problem instances are more difficult, but there should be no reason why GA would not work well on even larger diameters. A similar trend was also observed in our previous



(a) d=12 **(b)** d=13 **(c)** d=14 **(d)** d=15

Fig. 6 Single-objective convergence plots

investigation [19], although there, a much smaller population was used. Indeed, the number of fitness evaluations required with our current setting is consistently smaller than in our previous one, requiring less than 100 000 fitness evaluations to converge for $d = 13$. This seems to indicate that using a larger population is beneficial for GA.

Next, in Fig. 6, we display the convergence plots for the GA and GP single-objective optimization algorithms. We plot the median best fitness results, focusing only on diameter size d from 12 to 15, as smaller sizes show similar trends, but the optimization process becomes easier. Notice that for GP, all cases show that the random initial population contains optimal solutions. On the other hand, GA starts with large fitness values but continuously improves them and reaches the optimal value after using around 70% of the fitness evaluation budget allowed.

6.2 Multi-objective optimization results

Figure 7 depicts the Pareto fronts approximated by MOEA when minimizing the compatibility score (i.e., obj_1) and maximizing the Hamming weight (i.e., obj_2). For the sake of readability, we only report the fronts for $d = 9, 10, 11$. The scale difference on the Hamming weight axis between one diameter size d and the next one is so large that displaying all fronts between $d = 7$ and $d = 15$ would only make the larger ones visible, rendering indiscernible the smaller ones. However, this is not

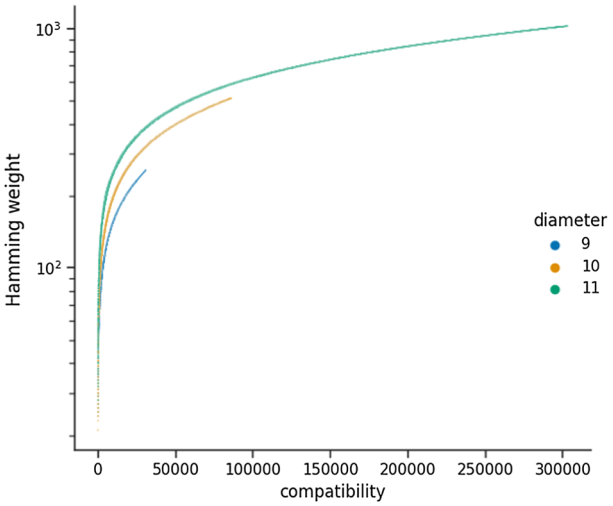


Fig. 7 Pareto fronts for $9 \leq d \leq 11$ approximated by MOEA

a serious issue since all fronts obtained in our experiments follow similar shapes. Hence, the three fronts reported in the figure are enough to draw conclusions about RQ2 and RQ3.

The curves in Fig. 7 corroborate the previous findings reported in [19]: the closer a CA marker rule is to be of the conserved landscape type, the lower the Hamming weight of its generating function must be. The first extreme case occurs when the rule achieves an optimal compatibility score of 0 (i.e., the rule is reversible), with very small Hamming weights observed (see also Sect. 7 for an overview of the possible Hamming weights when adopting a lexicographic optimization approach). On the other side, one can see that the compatibility fitness reaches its highest values when the Hamming weight is maximal, and in particular, it is about half the length of the generating function truth table. Hence, this indicates that marker CA rules with *balanced* generating functions (whose truth tables are composed of an equal number of 0s and 1s) are the farthest possible from being reversible under the conserved landscape definition.

6.3 Lexicographic optimization results

In Fig. 8, we depict convergence plots for the lexicographic optimization approach. As before, we depict the median of the best fitness value obtained over all experimental runs. Recall that in this case, the optimization objective is the minimization of fit_2, where the compatibility objective is first minimized to get a reversible rule, and then the opposite of obj_2 is minimized in order to maximize the Hamming weight of the generating function. Notice also that we cut off the fitness values larger than 100 as GA starts with very large fitness values while GP starts with values close to 0, making the final differences between GA and GP not noticeable. Considering GA, we observe around the same percentage of the evaluation required

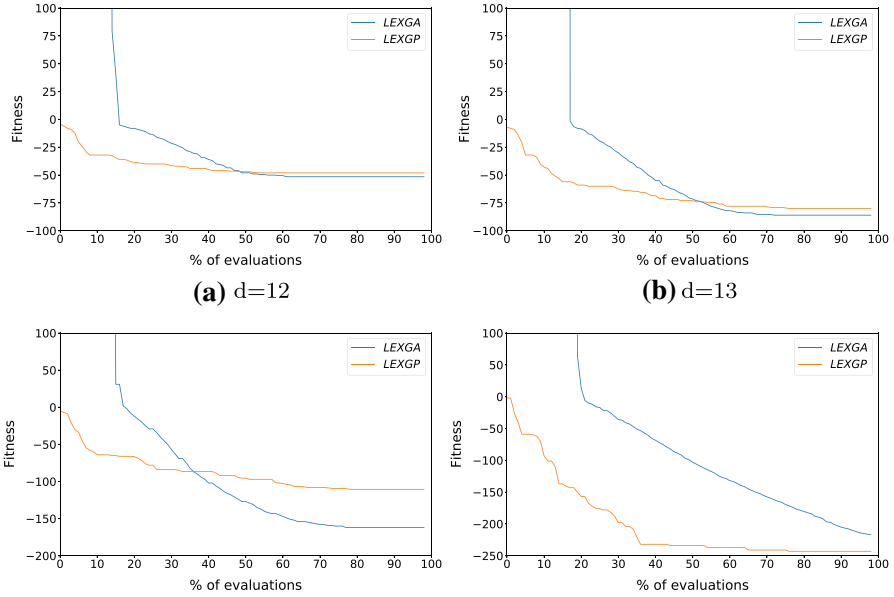


Fig. 8 Lexicographical optimization convergence plots

to reach fitness 0 as for the single-objective case. Afterward, it manages to optimize further the Hamming weight making the results comparable with GP. On the other hand, GP again starts with solutions around 0 and slowly improves the fitness value by maximizing the Hamming weight. For sizes up to 14, GA finds better final solutions than GP, where the difference is especially noticeable for $d = 14$. Interestingly, for $d = 15$, GP finds better final solutions but shows no improvement after around 40% of the fitness budget is used. On the other hand, GA improves the fitness values consistently throughout the evolution process, indicating that GA could probably reach the performance level of GP with more evaluations.

6.4 Diversity analysis

In Table 3, we provide various diversity metrics for the results obtained for all considered problem instances and algorithms, excluding MOEA. We used the multi-objective optimization approach to investigate a different research question not related to the diversity of the solutions. In particular, for each of the four algorithms (SOGA, SOGP, LEXGA, and LEXGP) and diameter $7 \leq d \leq 15$, we report the number of unique Hamming weights found (UHW), the minimum and maximum Hamming weights observed (respectively mHW and MHW), and the number of unique solutions found (USol). The numbers in bold are the highest values across all methods for each considered diversity metric and diameter.

Table 3 Diversity metrics for the solutions produced by all optimization methods (excluding MOEA) over all considered diameters

Algorithm	Metric	d								
		7	8	9	10	11	12	13	14	15
SOGA	UHW	**5**	4	4	6	6	6	6	7	2
	mHW	1	1	1	1	1	1	1	1	1
	MHW	5	4	4	6	6	6	6	8	3
	USol	37	46	**50**	**50**	**50**	**50**	**50**	49	39
SOGP	UHW	**5**	**6**	9	8	8	7	9	15	13
	mHW	1	1	1	1	1	1	1	1	1
	MHW	5	8	10	12	16	12	16	19	32
	USol	31	34	46	47	49	49	**50**	**50**	**50**
LEXGA	UHW	2	5	**10**	**16**	**19**	**26**	33	15	21
	mHW	**6**	**8**	10	15	**25**	**37**	**65**	**126**	**206**
	MHW	**7**	**12**	19	30	45	**74**	116	**191**	313
	USol	23	34	46	**50**	**50**	**50**	**50**	**50**	**50**
LEXGP	UHW	2	4	5	12	16	19	**34**	**28**	**27**
	mHW	**6**	**8**	**16**	**16**	24	32	32	48	95
	MHW	**7**	**12**	**25**	**30**	**48**	68	**128**	170	**344**
	USol	**45**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	49

Considering the single-objective algorithms, notice that GP finds more unique Hamming weights, where the differences are small for smaller diameter sizes but become even an order of magnitude larger for greater diameters. The minimal Hamming weight equals 1 for both algorithms and all diameter sizes, which is not surprising as single-objective algorithms do not aim to maximize the Hamming weight. The maximal Hamming weights are slightly larger for GP, especially for larger sizes. Again, this is not unexpected as GP finds optimal solutions already in the initial population, while GA required a significant number of evaluations to reach that performance level. On the other hand, if we consider the number of unique solutions found, we observe that GA works better (i.e., it found more diverse solutions). This result is aligned with our previous discussion as GP finds optimal solutions from the beginning, but then it is intuitive that some of those solutions could repeat. Indeed, syntactically different GP trees could map to the same truth table, thus giving rise to the same reversible rule. Going to larger diameter sizes gives good diversity results for GP too, since then, more solutions are optimal.

Next, considering the lexicographic optimization, the number of unique Hamming weights is similar for both GA and GP. The minimal Hamming weight for larger diameters is smaller for GP than GA and similar for smaller diameters. This indicates that the evolution process works better on average for GA, as a larger part of the population exhibits good behavior. For the maximal Hamming weights, we see that GP reaches better results for large diameters, but this is to be expected. Indeed, as GP has solutions with fitness equal to 0 already in the initial population, it can "use" the whole evolution process to optimize the Hamming weights. On the other hand, GA requires more than half of evaluations to reach a compatibility score

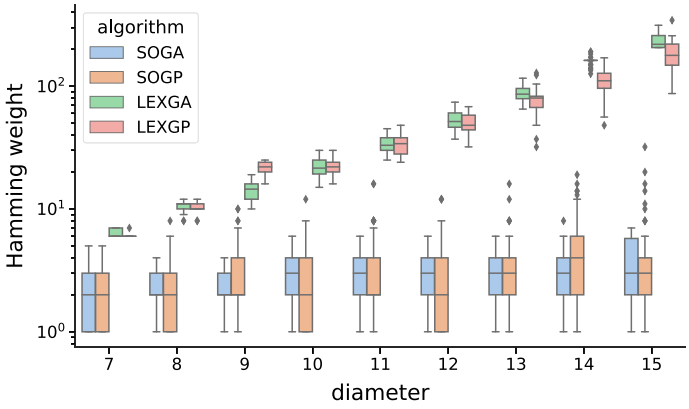


Fig. 9 Hamming weight distributions across all compared algorithms and diameters

of 0, which means it has much fewer evaluations available to maximize the Hamming weights. Still, the convergence plots indicate that GA progresses well throughout the evolution process. Possibly, adding more evaluations would allow GA to (at least) reach the Hamming weight values obtained with GP.

Finally, Fig. 9 presents the Hamming weight distributions for all considered algorithms and diameter sizes. First, we can recognize two natural groupings of the distributions, i.e., those related to single-objective optimization and lexicographic optimization. Since in the single-objective optimization, the goal is to reach a fitness value of 0 (i.e., we do not try to maximize the Hamming weight), we can observe that both GA and GP perform similarly and the increase in the Hamming weight value happens only due to a larger diameter (and thus, problem instance). On the other hand, GP performs better on smaller sizes for the lexicographic optimization scenario, which is expected as the initial population already reaches fitness equal to 0, and the obtainable Hamming weight values are relatively close to 0. Considering larger diameters, GA shows slightly better behavior on average. Still, considering the extreme values, we notice that GP performs better for sizes 13 and 15. Again, this is not surprising as GP has a better "starting position", so a greater portion of the evolution process can be used to maximize the Hamming weight. We believe adding more evaluations would resolve this problem and make GA a better performing algorithm, considering the best-obtained values.

7 Discussion

We now discuss the results obtained from our experimental evaluation applied on Problem 1 concerning the three research questions stated in Sect. 5.2.

Concerning RQ1, our experiments in the single-objective optimization scenario give somewhat counterintuitive results. In fact, despite the exiguous number of conserved landscape rules compared to the huge size of the search space, both GA and GP always converged to an optimal solution. This remark is in line with the previous

finding reported in [19]. However, notice that here we are dealing with an even smaller optimal set in our current setting than the one adopted in [19], since we set $\omega = 3$ instead of $\omega = \lfloor d - 1 \rfloor / 2$. Nevertheless, this choice made the problem *easier* for GA and GP, rather than harder as we expected in our hypotheses.

There is, moreover, an important distinction to observe on this statement. While the difficulty for GA to find a reversible rule increases as the diameter gets bigger, GP almost always finds an optimal solution already in the initial population, without even needing to start the evolution process. As mentioned in Sect. 6.1, the likely reason for this substantial difference in performances lies in the underlying genotype representations. Arguably, the chances of guessing at random a bitstring of length 2^{d-1} that maps to a conserved landscape rule of diameter d are quite low due to the very small number of such rules observed in our exhaustive search experiments. Since the GA population is initialized exactly in this way, it is thus very unlikely that the initial population will already include an optimal individual. Moreover, a random bitstring will likely have the Hamming weight close to half of its length, or equivalently it will be close to being balanced. As we remarked in Sect. 6.2, balanced bitstrings occur on the top right limit of the Pareto front. Hence they always have the highest possible value concerning the compatibility fitness that one seeks to minimize.

Contrarily, the maximum depth allowed for the trees evolved by GP is linear in the diameter of the local rule, so it is much smaller than the length of the corresponding truth table, which is instead exponential in the diameter. Consequently, it seems reasonable that a random GP tree will map to a truth table with a small Hamming weight. A further explanation of this phenomenon is that we did not use the XOR and XNOR in our experiments since they were filtered out during the tuning phase. This reduces the probability that the truth table obtained from the evaluation of a GP tree will be balanced, and thus that it will have a large Hamming weight.

Considering the arguments above, we can finally conclude that *there is no need to use evolutionary algorithms to construct conserved landscape reversible CA*. Indeed, the fact that an optimal solution is almost always found by GP already in the initial population suggests that a more efficient way to obtain a conserved landscape CA rule is the following:

1. Set the diameter d of the local rule f and $d - 1$ as the number of variables of the generating function g
2. While a conserved landscape rule has not been found do:

   (a) Generate at random a Boolean tree T with operator set {AND, OR, NOT, AND2}, where AND2 represents the AND with the second input complemented, and with maximum depth $d - 1$
   (b) Evaluate the truth table of g generated by T
   (c) Construct the local rule f as:

   $$f(x_1, x_2, x_3, \cdots, x_d) = x_3 \oplus g(x_1, x_2, x_4, \cdots, x_d) \quad .$$

   (d) Check if f is a conserved landscape rule.

3. Return f.

From our experiments in the single-objective optimization approach, the random generation method described above should likely succeed in the first few hundred guesses.

Regarding RQ2, our results obtained with the lexicographic optimization approach are also in line with our findings presented in [19]. Although LEXGA and LEXGP could find higher Hamming weights than in our previous experiments, they are nonetheless too low to be of any use for cryptographic applications. In particular, we can say something more precise in this respect: as mentioned in Sect. 1, the Hamming weight is a good proxy for the nonlinearity of a Boolean function, which is a measure of its distance from the set of affine functions. Ideally, Boolean functions of d variables used in stream and block ciphers should have a nonlinearity as high as possible, in the order of 2^{d-1} (we refer the reader to [3] for the reason why this is the case). Moreover, Cusick [39] showed that the nonlinearity of a d-variable Boolean function *coincides* with its Hamming weight if the latter is sufficiently small, namely if it is less than 2^{d-2}. In our case, all generating functions evolved by GA and GP have a Hamming weight which is significantly below 2^{d-2}, so their nonlinearity corresponds to their weight. Therefore, our results rule out the possibility of using conserved landscape CA in the design of symmetric ciphers components such as filter functions or S-boxes.

Finally, concerning RQ3, the results obtained by our multi-objective optimization experiments further corroborate our previous findings in [19]. In particular, in the case of a fixed offset ω far from the center of the neighborhood, the Pareto fronts approximated by MOEA show a clear trade-off between the reversibility of a marker CA rule under the conserved landscape definition and its Hamming weight. Moreover, the shapes of the fronts are quite similar to those obtained in [19] where the offset was placed at the center. This further suggests that the relationship between the compatibility objective function and the Hamming weight is independent of the cell's position that gets updated in the neighborhood.

8 Conclusions and future works

This paper considered the search of locally invertible cellular automata defined by conserved landscape rules as a combinatorial optimization problem, using GA and GP to solve it. We based our experimental investigation around three research questions stemming from exhaustive search experiments. We adopted three optimization approaches to investigate them – a single-objective, a multi-objective, and a lexicographic optimization approach. After performing a thorough parameter tuning phase, we evaluated the spaces of marker CA rules with diameters between 7 and 15, therefore expanding the experiments presented in [19] with three additional problem instances. In general, the results obtained from this new set of experiments corroborate the findings of our previous work. In particular, in this new set of experiments, the main new finding is that we fixed the rule offset ω to 3 for all problem instances instead of setting it at the center of the neighborhood. Contrary to our

initial assumption, where we hypothesized that this choice would make the optimization problem harder, it turned out to be simpler, especially in the GP case. On the other hand, similar trends of increasing difficulty were observed for GA, although with smaller magnitudes than in the results presented in [19]. As argued in Sect. 7, this difference is most likely caused by the different genotype representations used by GA and GP. The main conclusion that we can draw from these results is that evolutionary algorithms are not needed to construct conserved landscape CA when other properties such as the Hamming weight are not considered. Rather, a more efficient way is to generate random Boolean trees until an optimal solution is found. Further, the Pareto fronts obtained through our multi-objective optimization experiments not only confirm that the closer a marker CA rule is to be of the conserved landscape type, the lower its Hamming weight must be, but also the converse. Balanced generating functions with maximal Hamming weight are also the farthest possible from inducing a reversible rule. This gave us an additional insight because GP finds an optimal solution already in the initial population since the maximum depth enforced on the GP trees is sufficiently small that the resulting truth table will likely have a small Hamming weight.

Several avenues for future research remain to be explored on this subject. Regarding the first research question, the fact that the number of fitness evaluations required for GA to find a conserved landscape rule increases exponentially in the diameter seems to indicate that the difficulty of Problem 1 can be easily tuned for optimization algorithms with a bitstring-based representation. This could have, in turn, potential interesting applications for benchmark purposes. Further, it would be interesting to study this problem from the perspective of runtime analysis. Possibly, one could derive upper bounds on the number of fitness evaluations necessary for a simple evolutionary algorithm to converge on a conserved landscape rule. Likewise, although optimizing only the reversibility property is a trivial problem for GP, it could still be interesting to formally investigate the probability of guessing a tree at random that maps to an optimal solution from a theoretical point of view.

For the second research question, our new findings corroborate that the utility of conserved landscape CA for cryptography and reversible computing is quite limited since their Hamming weights are too low concerning the truth table size of their generating functions. Nonetheless, as remarked in Sect. 2.3, one can easily relax the definition of conserved landscape rules by allowing partial overlapping of the landscapes and obtain a larger class of reversible CA with more complex behaviors. A possible idea worth exploring in this direction would be to adapt the fitness function fit_1 to allow for this partial overlapping and use GP to investigate the Hamming weights of the resulting reversible CA, particularly with the lexicographic optimization method that proved to be the best performing one.

Finally, for the third research question, as discussed above, the Pareto fronts approximated by MOEA showed a clear trade-off between the reversibility of marker CA rules and the Hamming weights of their generating functions. As far as we know, there are no results in the CA literature addressing this aspect of conserved landscape CA. It would thus be interesting to exploit our experimental observation for formally proving an upper bound on the Hamming weight that a conserved landscape CA can achieve.

Author contributions All authors contributed equally.

Funding No funding was received to assist with the preparation of this manuscript.

Data availability The experimental data are available at https://github.com/rymoah/EvoRevCA.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Code availability The code is available at https://github.com/rymoah/EvoRevCA.

References

1. B. Breckling, G. Pe'er, Y.G. Matsinos, Cellular automata in ecological modelling. In: Modelling complex ecological dynamics, pp. 105–117. Springer (2011)
2. B. Chopard, Cellular automata and lattice boltzmann modeling of physical systems. In: handbook of natural computing, pp. 287–331. Springer (2012)
3. C. Carlet, *Boolean Functions for Cryptography and Coding Theory* (Cambridge University Press, 2021)
4. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
5. O.J. Dunn, Multiple comparisons among means. J. Am. Stat. Assoc. **56**(293), 52–64 (1961)
6. E. Czeizler, J. Kari, A tight linear bound on the neighborhood of inverse cellular automata. In: L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (eds.) Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings, *Lecture Notes in Computer Science*, vol. 3580, pp. 410–420. Springer (2005)
7. G. Bertoni, J. Daemen, M. Peeters, G.V. Assche, The Keccak reference (2011). http://keccak.noekeon.org/
8. D. Green, Cellular automata models in biology. Math. Comput. Model. **13**(6), 69–74 (1990)
9. H. Hatzikirou, D. Basanta, M. Simon, K. Schaller, A. Deutsch, 'go or grow': the key to the emergence of invasion in tumour progression? Mathematical medicine and biology: A J. IMA **29**(1), 49–65 (2012)
10. G.A. Hedlund, Endomorphisms and automorphisms of the shift dynamical systems. Math. Syst. Theory **3**(4), 320–375 (1969)
11. P. Hogeweg, Cellular automata as a paradigm for ecological modeling. Appl. Math. Comput. **27**(1), 81–100 (1988)
12. J. García-Duro, L. Manzoni, I. Arias, M. Casal, O. Cruz, X.M. Pesqueira, A. Muñoz, R. Álvarez, L. Mariot, S. Bandini, O. Reyes, Hidden costs of modelling post-fire plant community assembly using cellular automata. In: Cellular Automata - 13th International Conference on Cellular Automata for Research and Industry, ACRI 2018, Como, Italy, September 17-21, 2018, Proceedings, pp. 68–79 (2018)

13. J. Daemen, Cipher and hash function design strategies based on linear and differential cryptanalysis. Ph.D. thesis, Doctoral Dissertation, March 1995, KU Leuven (1995)
14. J.L. Guisado, F. Jiménez-Morales, J.M. Guerra, F.F. de Vega, K.A. Iskra, P.M.A. Sloot, D.L. Gonzalez, Laser dynamics modelling and simulation: an application of dynamic load balancing of parallel cellular automata. In: F.F. de Vega, E. Cantú-Paz (eds.) Parallel and Distributed Computational Intelligence, *Studies in Computational Intelligence*, vol. 269, pp. 321–347. Springer (2010)
15. J. Kari, Reversible cellular automata: from fundamental classical results to recent developments. New Generation Comput. **36**(3), 145–172 (2018)
16. W.H. Kruskal, W.A. Wallis, Use of ranks in one-criterion variance analysis. J. Am. Stat. Assoc. **47**(260), 583–621 (1952)
17. L. Mariot, D. Jakobovic, A. Leporati, S. Picek, Hyper-bent Boolean Functions and Evolutionary Algorithms. In: Genetic Programming, pp. 262–277 (2019)
18. L. Mariot, S. Picek, D. Jakobovic, A. Leporati, Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, pp. 306–313 (2017)
19. L. Mariot, S. Picek, D. Jakobovic, A. Leporati, An evolutionary view on reversible shift-invariant transformations, in *Genetic Programming*. ed. by T. Hu, N. Lourenço, E. Medvet, F. Divina (Springer International Publishing, Cham, 2020), pp. 118–134
20. R. Landauer, Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961)
21. D. Lind, B. Marcus, *An Introduction to Symbolic Dynamics and Coding* (Cambridge University Press, London, 2021)
22. M. Mitchell, J.P. Crutchfield, R. Das, et al.: Evolving cellular automata with genetic algorithms: a review of recent work. In: Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96), vol. 8 (1996)
23. M. Sipper, M. Tomassini, Co-evolving parallel random number generators. In: Parallel Problem Solving from Nature - PPSN IV, Berlin, Germany, September 22-26, 1996, Proceedings, pp. 950–959 (1996)
24. H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947)
25. L. Mariot, S. Picek, A. Leporati, D. Jakobovic, Cellular automata based S-boxes. Cryptogr. Commun. **11**(1), 41–62 (2019)
26. M. McCann, N. Pippenger, Fault tolerance in cellular automata at high fault rates. J. Comput. Syst. Sci. **74**(5), 910–918 (2008)
27. H. Nishio, Y. Kobuchi, Fault tolerant cellular spaces. J. Comput. Syst. Sci. **11**(2), 150–170 (1975)
28. J. Olsen, R. Scholtz, L. Welch, Bent-function sequences. IEEE Trans. On Information Theory **28**(6), 858–864 (1982)
29. K. Paterson, On Codes With Low Peak-to-Average Power Ratio for Multicode CDMA. IEEE Trans. Inf. Theory **50**, 550–559 (2004)
30. Y. Patt, *Injections of Neighborhood Size Three and Four on the Set of Configurations from the Infinite One-Dimensional Tessellation Automata of Two-State Cells* (Tech. rep, Army Electronics Command Fort Monmouth NJ, 1972)
31. S. Picek, C. Carlet, S. Guilley, J.F. Miller, D. Jakobovic, Evolutionary algorithms for boolean functions in diverse domains of cryptography. Evol. Comput. **24**(4), 667–694 (2016)
32. R. Poli, W.B. Langdon, N.F. McPhee, J.R. Koza, A field guide to genetic programming. Lulu. com (2008)
33. D. Richardson, Tessellations with local transformations. J. Comput. Syst. Sci. **6**(5), 373–388 (1972)
34. S. Picek, L. Mariot, A. Leporati, D. Jakobovic, Evolving S-boxes based on cellular automata with genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17, pp. 251–252 (2017)
35. S. Picek, L. Mariot, B. Yang, D. Jakobovic, N. Mentens, Design of s-boxes defined with cellular automata rules. In: Proceedings of the Computing Frontiers Conference, CF'17, pp. 409–414 (2017)
36. S. Silva, E. Costa, Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genet. Program Evolvable Mach. **10**(2), 141–179 (2009)
37. G.C. Sirakoulis, I. Karafyllidis, C. Mizas, V.A. Mardiris, A. Thanailakis, P. Tsalides, A cellular automaton model for the study of DNA sequence evolution. Comp. in Bio. and Med. **33**(5), 439–453 (2003)

38. T. Bäck, R. Breukelaar, Using Genetic algorithms to evolve behavior in cellular automata. in: unconventional computation, pp. 1–10. Springer Berlin Heidelberg (2005)
39. T.W. Cusick, Weight = nonlinearity for all small weight boolean functions. CoRR **abs/1710.02034** (2017)
40. T. Toffoli, N.H. Margolus, Invertible cellular automata: a review. Physica D **45**(1–3), 229–253 (1990)
41. Toward more efficient heuristic construction of boolean functions, Appl. Soft Comput. **107**, 107327 (2021)
42. H. Umeo, N. Kamikawa, M. Maeda, G. Fujita, State-efficient realization of fault-tolerant FSSP algorithms. Nat. Comput. **18**(4), 827–844 (2019)
43. S. Wolfram, Statistical mechanics of cellular automata. Rev. Mod. Phys. **55**(3), 601 (1983)

Authors and Affiliations

Luca Mariot[1] · Stjepan Picek[1] · Domagoj Jakobovic[2] · Alberto Leporati[3]

Stjepan Picek
s.picek@tudelft.nl

Domagoj Jakobovic
domagoj.jakobovic@fer.hr

Alberto Leporati
alberto.leporati@unimib.it

[1] Cyber Security Research Group, Delft University of Technology, Mekelweg 2, Delft, The Netherlands

[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, Zagreb, Croatia

[3] DISCo, Università degli Studi di Milano-Bicocca, Viale Sarca 336/14, Milano, Italy