An Evolutionary View on Reversible Shift-Invariant Transformations

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

DOI
10.1007/978-3-030-44094-7_8

Publication date
2020

Document Version
Final published version

Published in
Genetic Programming - 23rd European Conference, EuroGP 2020, Held as Part of EvoStar 2020, Proceedings

Citation (APA)
Mariot, L., Picek, S., Jakobovic, D., & Leporati, A. (2020). An Evolutionary View on Reversible Shift-Invariant Transformations. In T. Hu, N. Lourenço, E. Medvet, & F. Divina (Eds.), *Genetic Programming - 23rd European Conference, EuroGP 2020, Held as Part of EvoStar 2020, Proceedings* (pp. 118-134). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 12101 LNCS). SpringerOpen. https://doi.org/10.1007/978-3-030-44094-7_8

An Evolutionary View on Reversible Shift-Invariant Transformations

Luca Mariot[1(✉)], Stjepan Picek[1], Domagoj Jakobovic[2], and Alberto Leporati[3]

[1] Cyber Security Research Group, Delft University of Technology, Mekelweg 2, Delft, The Netherlands
{L.Mariot,S.Picek}@tudelft.nl
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, Zagreb, Croatia
domagoj.jakobovic@fer.hr
[3] DISCo, Università degli Studi di Milano-Bicocca, Viale Sarca 336/14, 20126 Milano, Italy
alberto.leporati@unimib.it

Abstract. We consider the problem of evolving a particular kind of shift-invariant transformation – namely, Reversible Cellular Automata (RCA) defined by conserved landscape rules – using GA and GP. To this end, we employ three different optimization strategies: a single-objective approach carried out with GA and GP where only the reversibility constraint of marker CA is considered, a multi-objective approach based on GP where both reversibility and the Hamming weight are taken into account, and a lexicographic approach where GP first optimizes only the reversibility property until a conserved landscape rule is obtained, and then maximizes the Hamming weight while retaining reversibility. The results are discussed in the context of three different research questions stemming from exhaustive search experiments on conserved landscape CA, which concern (1) the difficulty of the associated optimization problem for GA and GP, (2) the utility of conserved landscape CA in the domain of cryptography and reversible computing, and (3) the relationship between the reversibility property and the Hamming weight.

Keywords: Shift-invariant transformations · Cellular automata · Reversibility · Genetic Programming · Genetic Algorithms

1 Introduction

The property of *shift-invariance* plays an important role in studying and modeling several types of discrete dynamical systems. In particular, 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 the state of the system is described by a finite array, shift-invariant transformations are *cellular automata* (CA), i.e., functions defined by a local update rule which is uniformly applied at

T. Hu et al. (Eds.): EuroGP 2020, LNCS 12101, pp. 118–134, 2020.
https://doi.org/10.1007/978-3-030-44094-7_8

all sites of the array. Due to their simplicity and versatility, CA have been studied as models for simulating a wide variety of dynamical systems (see e.g. [1]).

Reversible shift-invariant transformations, and in particular Reversible CA (RCA) have the additional characteristic of preserving information. Thus, the dynamics of an RCA can be reversed backward in time starting from any state, and the inverse mapping is itself a CA. This makes RCA especially interesting for the design of energy-efficient computing devices since as stated by *Landauer's principle* [2] any *irreversible* logical operation implemented in hardware leads to the dissipation of heat, hence posing a physical lower bound on the miniaturization of devices based on irreversible gates. Another domain of interest is *cryptography*, where RCA can be used to design encryption and decryption algorithms [3].

Despite the extensive body of literature about RCA, up to now only a few classes of reversible CA are known (see [4] for a concise survey). Moreover, although such RCA are characterized in terms of relatively simple combinatorial definitions, there are no straightforward ways to construct them by taking into account further criteria that are of interest for practical applications. In this regard, *Evolutionary Algorithms* (EAs) represent an interesting method to investigate known RCA classes concerning these additional design criteria, since exhaustively searching for all possible RCA becomes unfeasible for large local rule sizes. To the best of our knowledge, this research method has not been pursued before, although some authors employed EA to evolve CA featuring certain properties other than reversibility [5,6].

The aim of this paper is to start the investigation of RCA by means of Genetic Algorithms (GA) and Genetic Programming (GP), focusing in particular on the class of reversible *marker* CA. There, the local update rule flips the state of a cell if its neighbors take on a set of patterns (or *landscapes*), which are *conserved* by the resulting shift-invariant transformation [7]. The motivation of our goal is twofold. First, the local rules of marker CA have a simple description through their *generating functions*, which leads to a natural formulation of the optimization objective for the reversibility property by minimizing the *compatibility* of its flipping landscapes. Second, the *Hamming weight* of a generating function in a marker CA is a good indicator of its *nonlinearity*, a fundamental property in cryptography, as well as of its dynamical behavior, which is relevant in the design of reversible computing devices. Consequently, maximizing the Hamming weight of the generating function can be considered as a further optimization objective in addition to reversibility.

After defining the genotype encodings for GA and GP to represent the candidate marker CA and the fitness function for reversibility, we set up three different research questions which consider the difficulty of the optimization problem for GA and GP, the utility of reversible marker CA for applications, and the relationship between reversibility and Hamming weight. We address this questions by organizing our experiments in three phases. In the first phase, we adopt a *single-objective* approach where only the reversibility of marker CA is optimized. In particular, our results show that both GA and GP always manage to

generate reversible marker CA over all considered problem instances, although with different performances. In the second phase, we consider a *multi-objective* (MO) approach with GP, where we optimize both the reversibility and the Hamming weight of marker CA. The Pareto fronts approximated by our MOGP algorithm clearly show that there is a trade-off between these two properties: the higher is the Hamming weight of a generating function, the lower will be the reversibility of the resulting marker CA. Finally, in the third phase, we use a *lexicographic optimization* strategy, where we first use GP to optimize only the reversibility property, and then maximize the Hamming weight when a reversible solution is found. With this approach, we manage to obtain a better coverage of reversible marker CA in terms of the Hamming weights.

The rest of this paper is organized as follows. Section 2 covers the necessary background notions about shift-invariant transformations and reversible CA. Section 3 defines the optimization problem for reversible marker CA with conserved landscapes, discusses the genotype encodings for GA and GP, and defines the fitness function for the reversibility property. Section 4 briefly reviews the existing literature about the use of evolutionary algorithms to design CA for specific purposes, such as in cryptography. Section 5 presents and discusses the results of our experiments organized into three phases. Finally, Sect. 6 sums up the main findings of the paper and sketches some directions for future work.

2 Background

2.1 Shift-Invariant Transformations and Cellular Automata

Let A be a finite alphabet and $A^{\mathbb{Z}}$ be the *full-shift space* of bi-infinite strings over A. In the field of symbolic dynamics, *shift-invariant transformations* are those mappings $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ that commute with the *shift operator*. *Cellular Automata* (CA) are a particular class of shift-invariant transformations whose output is determined by the uniform application of a single *local update rule* over all components (or *cells*) of a bi-infinite string. In this work, we focus only on shift-invariant transformations over finite arrays, which coincide with finite CA; thus, in what follows we will use the term CA and shift-invariant transformation interchangeably.

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

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

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

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

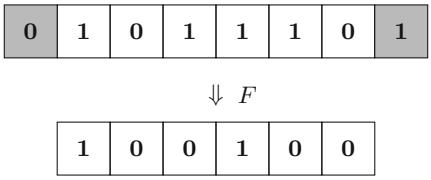


Fig. 1. Example of CA based on rule 150.

In other words, 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 in order to apply the local rule. In particular, the state vector of a PBCA can be seen as a ring, with the first cell following the last one. In the following, we will refer to PBCA simply as CA, since the former is the main CA model considered in this work.

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, one can encode the truth table as a single binary string $\Omega_f \in \mathbb{F}_2^{2^d}$, which is basically 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 [8]. Figure 1 reports an example of CA with $n = 6$ cells, diameter $d = 3$, offset $\omega = 1$, and local rule defined as $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$, which corresponds to Wolfram code 150. Hence, each cell looks at itself and its left and right neighbors in order to compute its next state through rule 150. The two cells shaded in grey in Fig. 1 represent "copies" respectively of the first and the last cell, in order to better visualize the neighborhoods of the cells at the boundaries.

2.2 Reversible CA

Reversibility is a particular property featured by certain dynamical systems where the orbits are cycles without transient parts or pre-periods. In particular, the orbits of a reversible system 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 CA context, this means that the global rule of a reversible CA must be bijective (to ensure that each global state of the cellular array has exactly one predecessor) and its inverse must also be a CA, that is, F^{-1} must be defined by a local rule.

Hedlund [9] showed that an infinite CA is reversible if and only if its global rule is bijective. On the other hand, the relationship between bijectivity and reversibility is more complicated in the case of finite CA. In particular, if we know that a local rule f induces a bijective global rule on a CA of a certain length $n \in \mathbb{N}$, then the inverse global rule is not necessarily defined by a local rule, nor it is the case, in general, that the global rule remains bijective for different lengths of the CA using 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 [10], which corresponds to a CA of length $n = 5$ and it is defined by the local rule of diameter $d = 3$ with Wolfram code 210. The offset of this CA is $\omega = 0$, which means that each cell applies rule 210 over itself and the two cells to its right to update its state. Daemen [11] showed that rule 210 is globally invertible, since it induces a bijective global rule only for odd CA lengths.

On the other hand, a local rule that induces a bijective global function for all finite lengths $n \in \mathbb{N}$ of the CA array is called *locally invertible*. In this case, the inverse mapping is also defined by a local rule, possibly of a different diameter, and thus the resulting CA is reversible. In what follows, we will consider the search of locally invertible rules as an optimization problem, focusing on the class of marker CA.

2.3 Marker CA

A *marker CA* (or *complementing landscape CA* [7]) is defined by a local rule that always complements the bit of the cell whose state is being updated whenever the cells in its neighborhood form a particular pattern (or *marker*, hence the name). Otherwise, the cell keeps its current state. The set of patterns defining a local rule of a marker CA can be conveniently formalized through the concept of *landscape*, which we define below:

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 is used to indicate the *origin* of the neighborhood in the local rule (that is, the cell whose state is being updated), and thus it occurs at position ω. The $-$ symbol represents a "don't care", meaning that the corresponding cell can be either in state 0 or 1. Hence, 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 can be described by one or more landscapes, all having the same width d and center ω. In particular, 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 $L_1, \cdots, L_k$ that define the locale rule. Observe that 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

$$L \leq_C M \Leftrightarrow l_i = m_i \text{ or } l_i \in \{0,1\} \text{ and } m_i = - \tag{2}$$

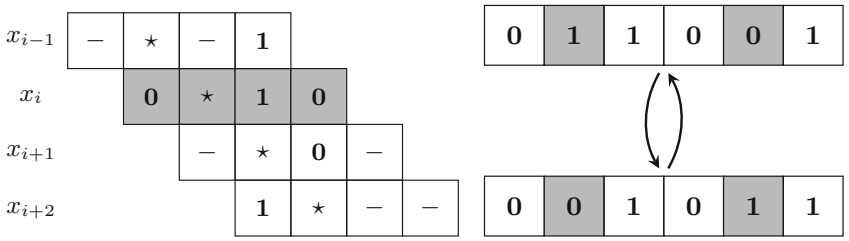
for all $0 \leq i \leq d-1$. Intuitively, this partial order describes the "generality" of a landscape: the more don't care symbols it has, the more patterns it contains. The extreme cases are the *atomic landscapes* that do not contain any don't care symbol, which describe only single patterns, and the landscape composed only of don't cares, which includes all possible patterns. In what follows, we will refer to $\leq_C$ as the *compatibility* partial order relation. In particular, we will 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 the partial order relation $\leq_C$, we will say that they are *incompatible*.

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 [7]:

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 $M_{i,0}, \cdots, M_{i,\omega-1}, M_{i,\omega+1}, \cdots, M_{i,d-1}$ 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, we also say that f is a *conserved landscape* rule. As noted in [7], a conserved landscape local rule induces an *involution*, i.e., the global rule of the resulting marker CA equals its own inverse. This is due to the fact that any cell being in one of the marker landscapes will still be in the same landscape after applying the local rule. After a further application of the local rule, the cell will go back to its initial state.

Hence, conserved landscape rules define a particular type of reversible CA, since all cycles have length 2. Daemen [11] argued that such CA can be useful in those cryptographic applications where both the encryption and decryption functions must be implemented in hardware. As noted in [7], one can relax the conditions of Lemma 1 by allowing the landscapes of the local rule to partially *overlap* one another. In this case, a cell that is in a landscape defined by the local rule will be in any of the other landscapes 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.



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

Fig. 2. A locally invertible CA defined by the single landscape $0 \star 10$.

To better illustrate the idea, we conclude this section by showing an example of a single conserved landscape rule of diameter $d = 4$, originally discovered by Patt [12]:

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. 2a shows that all three landscapes of the neighboring cells are incompatible with L. In particular, when x_i is in landscape L, then:

- Cell x_{i-1} is in landscape $-\star-1$, which is incompatible with $0\star10$ since there is a mismatch in position 3.
- Cell x_{i+1} is in landscape $-\star0-$, which is incompatible with $0\star10$ since there is a mismatch in position 2.
- Cell x_{i+2} is in landscape $1\star--$, which is incompatible with $0\star10$ since there is a mismatch in position 0.

Figure 2b displays an example of cycle starting from the initial state 011001. The two cells shaded in grey are in the landscape $0 \star 10$.

3 Optimizing Landscapes

3.1 Genotype Representation for Marker CA

Lemma 1 states that a conserved landscape CA can be constructed by searching for a set of landscapes $L_1, \cdots, L_k$ such that their associated neighborhood landscapes are incompatible with them. To perform such a search through Evolutionary Algorithms such as GA and GP, the first question is how to encode the genotype of the candidate solutions. In particular, since GA usually works on a bitstring encoding of the candidate solutions while GP relies on a tree representation, directly using the landscape specification of a marker CA rule does not seem a straightforward choice for encoding the genotype.

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, and define $\nu = x_{i-\omega} \cdots x_{i-1} x_{i+1} \cdots x_{i+d-1-\omega}$ be the vector of $d-1$

variables describing the states of the cells in the neighborhood of the cell at position i. Consider now the *generating function* $g : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ that outputs 1 if and only if the pattern formed by inserting the origin symbol $\star$ in vector ν at position ω belongs to $\mathcal{L}$. Then, the local rule of the marker CA can be defined as:

$$f(x_{i-\omega} \cdots x_{i-1} x_i x_{i+1} \cdots x_{i+d-1-\omega}) = x_i \oplus g(x_{i-\omega} \cdots x_{i-1} x_{i+1} \cdots x_{i+d-1-\omega}) \tag{3}$$

for all neighborhood configuration $x_{i-\omega} \cdots x_{i-1} x_i x_{i+1} \cdots x_{i+d-1-\omega} \in \mathbb{F}_2^d$. Hence, the algebraic expression 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. This is due to the fact that g evaluates to 1 if and only if the neighborhood takes on any of the landscapes in $\mathcal{L}$.

Consequently, we can reduce the representation of the local rule f of a marker CA to its generating function g. In particular, for GA we take the 2^{d-1}-bit string of the truth table Ω_g as encoding for the candidate solution. For GP, we use a tree where the terminal nodes represent the input variables of g, 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.

3.2 Fitness Functions

We can now define the fitness function used to drive the search of conserved landscape CA rules. Suppose that we have the truth table of a generating function g, and 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 patterns that the cells surrounding the origin must feature to flip the state of the central cell. To obtain the list of atomic landscapes, it just suffices to insert the origin symbol $\star$ in position ω to each vector of the support. The set of atomic landscapes obtained from the support can be used to check if a rule is of the conserved landscape type or not. In fact, it is not difficult to see that two landscapes with don't care symbols in them 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 landscapes that are compatible.

Given that we want to minimize such number to get a conserved landscape rule, we define the following fitness function. 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 *reversibility fitness* value of g is defined as:

$$fit_1(g) = \sum_{i,t \in [k], j \in [d-1]_\omega} comp(M_{i,j}, L_t), \tag{4}$$

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 fitness 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 adds 1 whenever a compatible pair is found. Therefore, the fitness function fit_1 measures the degree of compatibility of a set of atomic landscapes induced by the support of a generating function g. Consequently, the optimization objective is to *minimize* fit_1, with $fit_1(g) = 0$ corresponding to an optimal solution where all neighborhood landscapes are incompatible with the atomic landscapes, and thus the latter define a conserved landscape rule.

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 can be used both as a utility measure of a marker CA in cryptography (where it is related to the *nonlinearity* of the CA) and in designing reversible computing circuits. Given a generating function g, we thus define a second optimization objective by maximizing the following fitness function:

$$fit_2(g) = |supp(g)|. \tag{5}$$

4 Related Work

As already stated, this work is the first to use Evolutionary Algorithms to evolve reversible shift-invariant transformations. As such, there are no related works on the topic. Still, we mention several characteristic works where EAs are used to evolve shift-invariant transformations or related objects.

Bäck and Breukelaar used genetic algorithms to evolve behavior in CA where the authors explored different neighborhood shapes [6]. Sipper and Tomassini [13] proposed a cellular programming algorithm to co-evolve the rule map of non-uniform CA for designing random number generators. For a somewhat outdated, but very detailed overview of works using GA to evolve CA, we refer readers to [5]. Picek et al. demonstrated that GP can be used to evolve CA rules that then produce S-boxes with good cryptographic properties [14]. Next, Picek et al. used the same technique to further demonstrate that the S-boxes obtained from the CA rules have good implementation properties [15]. Mariot et al. conducted a more detailed analysis of the S-boxes based on CA where they also proved what are the best possible values for relevant cryptographic properties if one uses CA rules of a certain size [3]. There, the authors used GP to experimentally validate their findings but also to reverse engineer a CA rule from a given S-box. Mariot et al. used EA to construct orthogonal Latin squares based on CA [16]. Finally, the evolution of CA rules for cryptographic purposes is connected with the evolution of Boolean functions with good cryptographic properties. There, there are several works considering various evolutionary approaches, see for example [17,18].

5 Experiments

5.1 Research Questions and Experimental Setting

As noted in Sect. 3.1, the local rule of a marker CA of diameter d can be identified with its generating function g of $d-1$ variables which is computed on the neighborhood cells surrounding the origin, since the state of the central cell is simply 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 $\mathcal{P}(d)$, while 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. In what follows, we will assume that the offset ω is always fixed to $\lfloor (d-1)/2 \rfloor$, i.e., when d is odd the neighborhood origin will be the middle cell, while for d even it will be the left middle cell. This does not hinder the scope of our investigation since as shown in [7] reversible marker rules in different offsets are symmetric under rotations and reflection.

Note that, 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 are aware, an exhaustive search of reversible marker CA rules has been carried out only by Patt [12], who considered diameters up to $d = 4$. For completeness, Table 1 reports the numbers of conserved-landscape rules we found by exhaustively searching the sets of generating functions up to $d = 6$, 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. Further, we excluded from the count the identity rule which simply copies the state of the central cell, since it is trivially reversible for any diameter. As a general remark, one can see from Table 1 that the number of conserved landscape rules is much smaller than the size of the whole generating function set. Moreover, the number of 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 3 landscapes, which are thus not very useful for

Table 1. Numbers of conserved landscape rules found by exhaustive search.

d	2^{d-1}	$\#\mathcal{P}(d)$	#REV	Weights
3	4	16	0	–
4	8	256	1	1
5	16	65 536	10	1, 2
6	32	$4.3 \cdot 10^9$	46	1, 2, 3

cryptographic and reversible computing purposes. Nevertheless, these findings prompt us with three interesting 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 landscapes rules of a larger diameter which 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 fitness function fit_1 defined in Sect. 3.2) and its Hamming weight?

We employed Genetic Algorithms (GA) and Genetic Programming (GP) to investigate the three questions above, by optimizing the fitness functions fit_1 and fit_2. The reason for comparing GA and GP was to assess whether the representation of the solutions as bitstrings or trees affected the convergence to an optimal solution on this particular problem. We considered the spaces of marker CA rules of diameter between $d = 8$ and $d = 13$ as problem instances for our experiments, using the $d = 7$ case for tuning our evolutionary algorithms. Both our GA and GP employed a steady-state tournament selection operator, which randomly samples three individuals from the populations. Next, the crossover is applied to the best two individuals of the tournament to produce a child candidate solution, which is then mutated and inserted into the population by replacing the worst individual of the tournament. For GA, we employed one-point, two-point, and uniform crossover operators (selected at random at each iteration), while we adopted a classic bit-flip operator for mutation. In the case of GP, we used a function set for the Boolean trees composed of the binary operators AND, OR, XOR, XNOR, 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. Although this function set is redundant, since any Boolean function can be formulated with a smaller set, the choice of these elements is based on our previous experience evolving Boolean expressions with GP that define CA [3]. To avoid bloat, we observed through preliminary experiments that setting the maximum tree depth equal to the number of variables of the generating functions $(d-1)$ was a good choice in terms of GP performance. Further, for crossover in GP, we employed five different operators, namely simple subtree crossover, uniform crossover, size fair, one-point, and context preserving crossover, again selected at random at each iteration. Analogously to GA, for mutation we adopted a single operator, namely subtree mutation. Concerning the population size and the mutation probability, we performed a tuning phase over the $d = 7$ problem instance, which resulted in population sizes of 25 and 500 individuals for GA and GP, respectively, and a mutation probability of 0.8 and 0.5 for GA and GP. Similarly to previous works on related optimization problems [16,18], we set a budget of 500 000 fitness

evaluations for both GA and GP, and we performed 30 runs for each considered problem instance.

5.2 Single-Objective Approach

As a first attempt to investigate the research questions stated in the previous section, we employed a single-objective approach where GA and GP only minimized the reversibility fitness function fit_1 as an optimization criterion, analyzing the Hamming weights of the best solutions in a second moment. The motivation was to address research question RQ1, i.e., investigate how difficult it is for GA and GP to optimize fit_1, especially considering the scarcity of conserved landscape rules assessed by our exhaustive search experiments.

The first remarkable finding is that *GA and GP achieved a full success rate over all considered problem instances*, i.e., both algorithms always converged to a reversible rule in all 30 experimental runs for each diameter between $d = 8$ and $d = 13$. In particular, using the fitness function fit_1 as defined in Eq. (4), GP always converged to the trivial solution 0, which corresponds to the identity rule. For this reason, we slightly tweaked fit_1 for our GP experiments by adding a penalty factor that punishes a candidate solution having a null Hamming weight. After this modification, GP again obtained a full success rate over all problem instances, thus finding *non-trivial* conserved landscape rules. Interestingly, this finding is analogous to what was observed in [16] for the optimization of orthogonal Latin squares based on cellular automata, where GP always converged to "simple" solutions – which in that context were represented by *linear* local rules – when optimizing only the orthogonality constraint.

The remarks above seem to answer question RQ1 in negative: the limited number of conserved landscape rules as compared to the size of the search space does not seem to pose a problem for GA and GP to converge to an optimal solution. However, the comparison shown in Fig. 3 on the number of fitness evaluations performed by GA and GP tells us a more precise story. As can be seen, the number of fitness evaluations required by GA to find a reversible rule scales exponentially with respect to the rule diameter (note that we adopted a logarithmic scale in Fig. 3 for the sake of comparison). In particular, the median number of fitness evaluations performed by GA approximately doubles every time the diameter increases by 1. On the contrary, GP features a much more stable and slower growth in the number of fitness evaluations that are necessary for converging to an optimal solution. Further, this number is always smaller by at least one order of magnitude than the number of GA fitness evaluations over all problem instances. This observation indicates that GP is a better suited heuristic than GA for this optimization problem, for which reason we employed only GP in our subsequent experiments on multi-objective and lexicographic optimization. The superiority of GP with respect to GA is also reflected in the Hamming weights of the optimal solutions found by the two heuristics: although GA provides a better coverage of distinct weights over all 30 experimental runs for each instance, the maximum weight found by GP is always consistently greater than the maximum achieved by GA. We refer the reader to Table 2 for a comparison of the Hamming weights found by all optimization approaches.

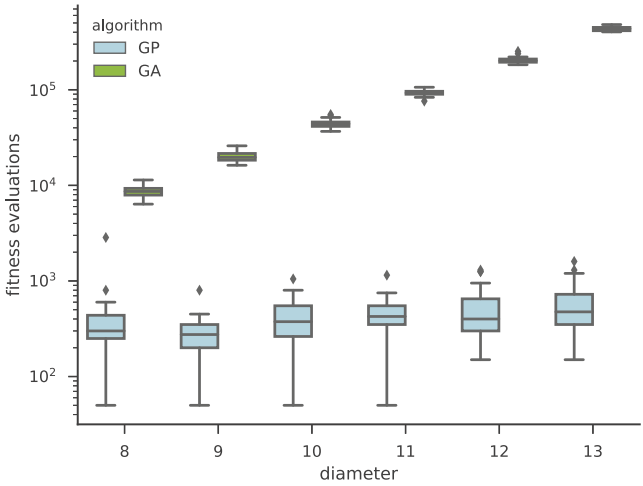


Fig. 3. Comparison of fitness evaluations performed by GA and GP.

5.3 Multi-objective Approach

To investigate the interaction between the reversibility of a marker CA rule and the Hamming weight of its generating function, we adopted a multi-objective strategy as a second optimization approach. In particular, we considered only a multi-objective version of GP (MOGP), since in the single-objective approach we observed that GP outperformed GA in terms of fitness evaluations. The MOGP approach used the well-known NSGA-II algorithm, where we *minimized* the reversibility fitness value fit_1 and *maximized* the Hamming weight as measured by fit_2. For each considered problem instance, we run the MOGP algorithm with the same experimental parameters adopted for the single-objective setting described in Sect. 5.1, and at the end of each run, we recorded all Pareto optimal solutions in the population and added them to a list. Figure 4 plots the Pareto front approximated by MOGP for the instance $d = 8$. The main observation one can draw from the plot is that there is a clear trade-off between reversibility and the Hamming weight, thereby providing an empirical answer to research question RQ3: *the closer a marker CA rule is to being of the conserved landscape type, the lower the Hamming weight of its generating function must be.* Incidentally, the left tail of the Pareto front also provides some hints with respect to research question RQ2. Indeed, there are only a few points aligned on the $fit_1 = 0$ value, all having small Hamming weights with respect to the truth table size of the generating function (which for $d = 8$ equals 128). This finding seems to hinder the applicability of conserved landscape rules in cryptography, and in particular in the design of S-boxes based on CA [3], since a local rule with a low Hamming weight will induce an S-box with low nonlinearity.

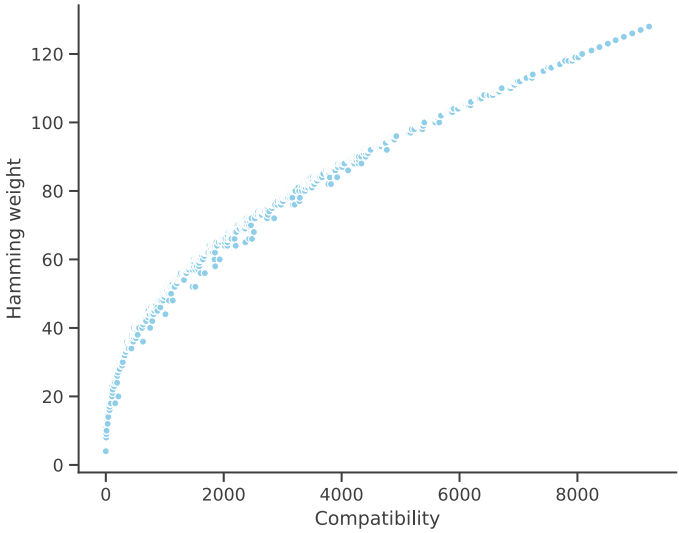


Fig. 4. Pareto front for $d = 8$.

5.4 Lexicographic Optimization

Our third experiment consisted of a *lexicographic optimization* approach, to assess whether a better coverage of the Hamming weights of conserved landscape rules could be obtained. In the first optimization stage, GP minimized only the reversibility fitness value fit_1. After obtaining a conserved landscape solution, in the second stage GP maximized fitness fit_2, logging each new solution that was still reversible and with a higher Hamming weight.

Table 2 compares in terms of solutions diversity the four optimization approaches adopted in our experiments – single-objective GA (SOGA) and GP (SOGP), multi-objective GP (MOGP), and lexicographic GP (LEXGP). In particular, each entry of the table is a triplet of the form (*UHW*, *MHW*, *USol*) where *UHW* denotes the number of distinct Hamming weights found, *MHW* is the maximum Hamming weight observed, and *USol* is the number of distinct optimal solutions found. For single-objective GA and GP, we report only the data of the best solutions found over all 30 experimental runs, while for MOGP and lexicographic GP we consider the whole populations after finishing the 30 runs. In particular, one can see that LEXGP is the method achieving the best trade-off in terms of distinct weights coverage, maximum weight, and uniqueness of solutions produced. SOGA is the heuristic that reaches the widest diversity of distinct Hamming weights but its maximum weights are the lowest among all four methods. SOGP, on the other hand, reaches higher maximum weights than GA, but with a quite low variety of the Hamming weights. MOGP further improves on the maximum weights and have similar coverage of distinct weights to SOGP, but the number of distinct solution is quite low (recall that with MOGP we recorded all Pareto optimal solutions in the population

Table 2. Diversity of the solutions produced by all optimization methods.

d	SOGA			SOGP			MOGP			LEXGP		
	UHW	MHW	USol	UHW	MHW	USol	UHW	MHW	USol	UHW	MHW	USol
8	5	6	30	4	8	27	4	10	24	5	10	47
9	6	7	30	4	16	29	2	20	22	8	20	60
10	7	11	30	3	16	30	4	32	48	6	28	65
11	9	15	30	3	32	29	6	56	40	6	56	64
12	11	23	30	4	64	30	4	72	29	7	80	71
13	12	29	30	2	64	29	4	128	50	7	160	73

over all runs). Finally, LEXGP is the one obtaining a good coverage of distinct weights, although not as good as SOGA. This is compensated by the fact that LEXGP achieved the highest maximum weights among all four methods (except for the case $d = 10$ where it was outperformed by MOGP). Moreover, LEXGP generated more distinct reversible solutions than MOGP.

6 Conclusions and Future Work

In this paper, we used GA and GP to study a particular class of reversible shift-invariant transformations – namely CA defined by conserved landscape rules – using three different optimization approaches. We now sum up the main findings of our experiments and suggest some possible future developments in the context of the three research questions that we posed.

Regarding the first research question, the results obtained with the single-objective approach seems to indicate that evolutionary algorithms, and in particular GP, can find relatively easily conserved landscape CA rules, despite the limited size of the optimal solutions set. Although this makes the associated optimization problem unsuitable for benchmark purposes, it would be interesting to investigate the performance difference between GA and GP, for example by analyzing the fitness landscapes induced by fit_1 on the two genotype spaces.

For the second research question, our findings show that the relevance 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. Nevertheless, 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, in particular with the lexicographic optimization method that proved to be the best performing one.

References

1. Chopard, B.: Cellular automata and lattice Boltzmann modeling of physical systems. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 287–331. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_9
2. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961)
3. Mariot, L., Picek, S., Leporati, A., Jakobovic, D.: Cellular automata based S-boxes. Cryptogr. Commun. **11**(1), 41–62 (2018). https://doi.org/10.1007/s12095-018-0311-8
4. Kari, J.: Reversible cellular automata: from fundamental classical results to recent developments. New Gener. Comput. **36**(3), 145–172 (2018). https://doi.org/10.1007/s00354-018-0034-6
5. Mitchell, M., Crutchfield, J.P., Das, R., 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 1996), vol. 8 (1996)
6. Bäck, T., Breukelaar, R.: Using genetic algorithms to evolve behavior in cellular automata. In: Calude, C.S., Dinneen, M.J., Păun, G., Pérez-Jímenez, M.J., Rozenberg, G. (eds.) UC 2005. LNCS, vol. 3699, pp. 1–10. Springer, Heidelberg (2005). https://doi.org/10.1007/11560319_1
7. Toffoli, T., Margolus, N.H.: Invertible cellular automata: a review. Phys. D **45**(1–3), 229–253 (1990)
8. Wolfram, S.: Statistical mechanics of cellular automata. Rev. Mod. Phys. **55**(3), 601 (1983)
9. Hedlund, G.A.: Endomorphisms and automorphisms of the shift dynamical systems. Math. Syst. Theory **3**(4), 320–375 (1969). https://doi.org/10.1007/BF01691062
10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak reference (2011)
11. Daemen, J.: Cipher and hash function design strategies based on linear and differential cryptanalysis. Ph.D. thesis, Doctoral Dissertation, KU Leuven, March 1995
12. Patt, Y.: Injections of neighborhood size three and four on the set of configurations from the infinite one-dimensional tessellation automata of two-state cells. Technical report, Army Electronics Command Fort Monmouth, NJ (1972)
13. Sipper, M., Tomassini, M.: Co-evolving parallel random number generators. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 950–959. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61723-X_1058
14. Picek, S., Mariot, L., Leporati, A., Jakobovic, D.: Evolving S-boxes based on cellular automata with genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017, pp. 251–252 (2017)
15. Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N.: Design of S-boxes defined with cellular automata rules. In: Proceedings of the Computing Frontiers Conference, CF 2017, pp. 409–414 (2017)
16. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 306–313 (2017)

17. Picek, S., Carlet, C., Guilley, S., Miller, J.F., Jakobovic, D.: Evolutionary algorithms for Boolean functions in diverse domains of cryptography. Evol. Comput. **24**(4), 667–694 (2016)
18. Mariot, L., Jakobovic, D., Leporati, A., Picek, S.: Hyper-bent Boolean functions and evolutionary algorithms. In: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (eds.) EuroGP 2019. LNCS, vol. 11451, pp. 262–277. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16670-0_17