

## Evolutionary Algorithms for the Design of Orthogonal Latin Squares based on Cellular Automata

Luca Mariot<sup>1</sup>, Stjepan Picek<sup>2</sup>, Domagoj Jakobovic<sup>3</sup>, Alberto Leporati<sup>1</sup>

<sup>1</sup> Dipartimento di Informatica, Sistemistica e Comunicazione (DISCO)  
Università degli Studi Milano - Bicocca

<sup>2</sup> Cyber Security Research Group  
Delft University of Technology

<sup>3</sup> Faculty of Electrical Engineering and Computing  
University of Zagreb

GECCO 2017 – Berlin, July 15–19, 2017

# Latin Squares

## Definition

A *Latin square* of order  $N$  is a  $N \times N$  matrix  $L$  such that in every row and in every column each number of  $[N] = \{1, \dots, N\}$  occurs exactly once

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

# Orthogonal Latin Squares (OLS)

## Definition

Two Latin squares  $L_1$  and  $L_2$  of order  $N$  are *orthogonal* if their superposition yields all the pairs  $(x, y) \in [N] \times [N]$ .

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

1	4	2	3
3	2	4	1
4	1	3	2
2	3	4	1

1,1	3,4	4,2	2,3
4,3	2,2	1,4	3,1
2,4	4,1	3,3	1,2
3,2	1,3	2,1	4,4

Applications of sets of **Mutually** orthogonal Latin squares (MOLS):

- ▶ Cryptography (secret sharing schemes, authentication codes)
- ▶ Coding theory (MDS codes)

- ▶ Existing constructions of OLS mostly based on **algebraic methods** [Keedwell15]
- ▶ No work addressing the design of OLS via **evolutionary techniques**
- ▶ Possible reasons and challenges:
  - ▶ **Representation issues**: how to encode a pair of Latin squares?
  - ▶ **Variation operators**: how to cross two individuals and still get a pair of Latin squares?
  - ▶ **Search space analysis**: the number of Latin squares/OLS is not even known for generic  $N$

# A different perspective: Cellular Automata (CA)

## Definition

**One-dimensional CA:** triple  $\langle m, n, f \rangle$  where  $m \in \mathbb{N}$  is the number of cells on a one-dimensional array,  $n \in \mathbb{N}$  is the neighborhood and  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is the local rule.

Example:  $m = 8$ ,  $n = 3$ ,  $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$  (Rule 150)

... 

0	1	1	0	0
---	---	---	---	---

 ...

$f(1, 1, 0) = 1 \oplus 1 \oplus 0$

0
---

1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Parallel update  $\Downarrow$  Global rule  $F$

1	0	0	1	1	0
---	---	---	---	---	---

**Notation:**  $\Omega(f)$  is the  $2^n$ -bit string representing the **truth table of  $f$**

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \Rightarrow \Omega(f) = 01101001$$

# Latin Squares through Bipermutive CA (1/2)

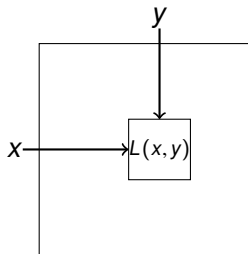
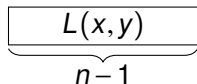
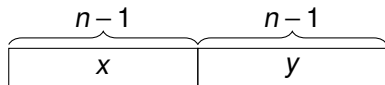
- ▶ **Bipermutive CA**: local rule  $f$  is defined as

$$f(x_1, \dots, x_n) = x_1 \oplus \varphi(x_2, \dots, x_{n-1}) \oplus x_n$$

- ▶  $\varphi : \{0, 1\}^{n-2} \rightarrow \{0, 1\}$ : **generating function** of  $f$

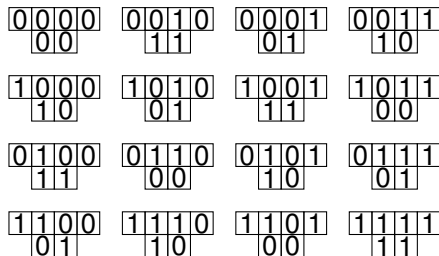
**Lemma ([Eloranta93, Mariot16])**

Let  $\langle 2(n-1), n, f \rangle$  be a CA with bipermutive rule. Then, the global rule  $F$  generates a Latin square of order  $N = 2^{n-1}$



# Latin Squares through Bipermutive CA (2/2)

- ▶ **Example:** CA  $\langle 4, 3, f \rangle$ ,  $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$  (Rule 150)
- ▶ Encoding:  $00 \mapsto 1, 10 \mapsto 2, 01 \mapsto 3, 11 \mapsto 4$



(a) Rule 150 on 4 bits

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b) Latin square  $L_{150}$

# Motivations and Goals

- ▶ Construction of OLS solved for **linear CA** [Mariot16]
- ▶ MOLS arising from **nonlinear constructions** have relevance in **cheater-immune** Secret Sharing Schemes [Tompa88]

**Goal:** Design OLS based on CA by evolving **pairs of nonlinear bipermutive local rules** through GA and GP

Twofold motivation:

- ▶ **Theoretical:** Understand the mathematical structure of the space of nonlinear CA-based OLS
- ▶ **EC perspective:** Source of new problems for evolutionary algorithms



# Nonlinearity of CA Local Rules (Boolean Functions)

- ▶ **Affine function**:  $l(x_1, \dots, x_n) = a \oplus a_1 x_1 \oplus \dots \oplus a_n x_n$ ,  $a, a_i \in \{0, 1\}$
- ▶ **Nonlinearity** of  $f$ : Hamming distance of  $\Omega(f)$  from the set of all affine functions
- ▶ **Walsh transform** of  $f$ : given  $\omega \in \{0, 1\}^n$ ,

$$W_f(\omega) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus \omega \cdot x}, \text{ where } \omega \cdot x = \bigoplus_{i=1}^n \omega_i \cdot x_i$$

## Definition

Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . The *nonlinearity* of  $f$  is defined as

$$NI(f) = 2^{n-1} - \frac{1}{2} \max_{\omega \in \{0,1\}^n} \{|W_f(\omega)|\}$$

# Search Space Size

- ▶ Number of Boolean functions of  $n$  variables:  $\mathcal{F}_n = 2^{2^n}$
- ▶ Bipermutive rules of size  $n \Leftrightarrow$  Generating functions of size  $n-2$  (which are  $\mathcal{F}_{n-2} = 2^{2^{n-2}}$ )
- ▶ Pairs of bipermutive rules of size  $n$ :  $\mathcal{B}_n = 2^{2^{n-1}} = \mathcal{F}_{n-1}$

$n$	3	4	5	6	7	8
$\mathcal{B}_n$	16	256	65536	$\approx 4.3 \times 10^9$	$\approx 1.8 \cdot 10^{19}$	$\approx 3.4 \cdot 10^{38}$
$N \times N$	$4 \times 4$	$8 \times 8$	$16 \times 16$	$32 \times 32$	$64 \times 64$	$128 \times 128$
#OLS	8	72	1704	533480	?	?

**Remark:** Exhaustive enumeration possible up to  $n = 6$

## Fitness Functions (1/2)

- ▶  $\#rep(L_1, L_2)$ : Number of occurrences of each pair (except the first one) in the superposition of Latin squares  $L_1$  and  $L_2$

1	3	4	2
4	2	1	3
2	4	3	1
2	3	4	1

(a)  $L_1$

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b)  $L_2$

4,1	1,4	2,3	3,2
3,2	2,3	1,4	4,1
1,4	4,1	3,2	2,3
2,3	3,2	4,1	1,4

(c)  $\#rep(L_1, L_2) = 12$

- ▶ Let  $\varphi, \gamma$  be the generating functions of two bipermutive CA, and let  $L_\varphi, L_\gamma$  be the associated Latin squares

**First fitness function:** minimize  $fit_1(\varphi, \gamma) = \#rep(L_\varphi, L_\gamma)$

- ▶ **Remark:**  $fit_1$  does not consider the nonlinearity of  $\varphi$  and  $\gamma$ !
- ▶ Nonlinearity penalty factor:

$$NIPen(\varphi, \gamma) = \begin{cases} 0, & \text{if } NI(\varphi) > 0 \text{ AND } NI(\gamma) > 0 \\ 1, & \text{if } NI(\varphi) = 0 \text{ XOR } NI(\gamma) = 0 \\ 2, & \text{if } NI(\varphi) = 0 \text{ AND } NI(\gamma) = 0 \end{cases}$$

**Second fitness function:** minimize

$$fit_2(\varphi, \gamma) = \#rep(L_\varphi, L_\gamma) + NIPen(\varphi, \gamma) \cdot N^2$$

- ▶ The  $N^2$  scaling factor balances the range of  $\#rep(L_\varphi, L_\gamma)$ , which is  $\{0, \dots, N^2\}$

# GA Encoding: Single Bitstring

- ▶ Let  $\varphi, \gamma : \{0, 1\}^{n-2} \rightarrow \{0, 1\}$  be a pair of generating functions, with  $2^{n-2}$ -bit truth tables  $\Omega(\varphi), \Omega(\gamma)$ , and let  $\|$  denote concatenation

**First GA encoding:**  $enc_1(\varphi, \gamma) = \Omega(\varphi)\|\Omega(\gamma)$

Example:

$$\varphi(x_1, x_2, x_3) = x_1 \oplus x_3 \Rightarrow \Omega(\varphi) = (0, 1, 0, 1, 1, 0, 1, 0)$$

$$\gamma(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \Rightarrow \Omega(\gamma) = (0, 1, 1, 0, 1, 0, 0, 1)$$

$$enc_1(\varphi, \gamma) = (0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1)$$

- ▶ Classic GA variation operators like one-point crossover and bit-flip mutation are applied in this case

- ▶ **Idea:** Keep the generating functions separated and evolve them independently

**Second GA encoding:**  $enc_2(\varphi, \gamma) = (\Omega(\varphi), \Omega(\gamma))$

- ▶ We use the same idea for GP: the genotype is composed of two trees  $T(\varphi)$  and  $T(\gamma)$  representing  $\varphi$  and  $\gamma$

**GP encoding:**  $enc_{GP}(\varphi, \gamma) = (T(\varphi), T(\gamma))$

- ▶ Classic GA and GP variations operators are applied **independently** on each of the two components

## Definition

$f, g : \{0, 1\}^n \rightarrow \{0, 1\}$  are **pairwise balanced** (PWB) if

$$\begin{aligned} |(f, g)^{-1}(0, 0)| &= |(f, g)^{-1}(1, 0)| = \\ &= |(f, g)^{-1}(0, 1)| = |(f, g)^{-1}(1, 1)| = 2^{n-2} \end{aligned}$$

Example:

- ▶  $f(x_1, x_2, x_3) = x_1 \oplus x_3$  (Rule 90)
- ▶  $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$  (Rule 150)

$$\Omega(f) = (0, 1, 0, 1, 1, 0, 1, 0) ,$$

$$\Omega(g) = (0, 1, 1, 0, 1, 0, 0, 1) .$$

Each of the pairs  $(0, 0), (1, 0), (0, 1), (1, 1)$  occurs  $2^{3-2} = 2$  times

# GA Encoding: Balanced Quaternary Strings (2/2)

- ▶ Experimental observations on exhaustive search:
  - ▶ Two bipermutive CA generate OLS  $\Rightarrow$  the local rules are PWB
  - ▶ Generating functions are PWB  $\Rightarrow$  the local rules are PWB

**Third GA encoding:**  $enc_3(\varphi, \gamma)$  is a **quaternary** string of length  $2^{n-2}$  where each number from 1 to 4 occurs  $2^{n-4}$  times

Example:  $n = 5, (0, 0) \mapsto 1, (1, 0) \mapsto 2, (0, 1) \mapsto 3, (1, 1) \mapsto 4$

$$\Omega(\varphi) = (0, 1, 0, 1, 1, 0, 1, 0)$$

$$\Omega(\gamma) = (0, 1, 1, 0, 1, 0, 0, 1)$$

$$enc_3(\varphi, \gamma) = (1, 4, 3, 2, 4, 1, 2, 3)$$

- ▶ Balancedness-preserving variation operators for GA:
  - ▶ **Crossover:** use counters to keep track of the multiplicities of the 4 values in the offspring (as in [Mariot15])
  - ▶ **Mutation:** use a swap-based operator



## Common Parameters:

- ▶ Problem instances: rules of  $n = 7$  and  $n = 8$  variables
- ▶ Termination condition: 300000 fitness evaluations
- ▶ Each experiment is repeated over 50 independent runs
- ▶ Selection operator: steady-state with 3-tournament operator

## GA Parameters:

- ▶ Population size: 30 individuals
- ▶ Crossover and mutation probabilities:  $p_c = 0.95$ ,  $p_m = 0.2$

## GP Parameters:

- ▶ Boolean operators: AND, OR, XOR, XNOR, NOT, IF
- ▶ Population size: 500 individuals
- ▶ Mutation probability:  $p_m = 0.5$

# Results

- ▶  $(GA, n, enc_i)$ : GA experiment with CA rules of  $n$  variables and encoding  $enc_i$ , fitness function  $fit_1$
- ▶  $(GP, n, fit_j)$ : GP experiment with CA rules of  $n$  variables and encoding  $enc_{GP}$ , fitness function  $fit_j$

Exp.	avg fit	std fit	#opt	#lin	#nlin
$(GA, 7, enc_1)$	520.32	360.16	12/50	0	12
$(GA, 7, enc_2)$	565.44	389.03	15/50	0	15
$(GA, 7, enc_3)$	392.64	328.47	18/50	0	18
$(GA, 8, enc_1)$	4165.44	604	1/50	0	1
$(GA, 8, enc_2)$	4222.16	125.03	0/50	0	0
$(GA, 8, enc_3)$	4696.48	135.51	0/50	0	0
$(GP, 7, fit_1)$	0	0	50/50	50	0
$(GP, 7, fit_2)$	0	0	50/50	0	50
$(GP, 8, fit_1)$	0	0	50/50	47	3
$(GP, 8, fit_2)$	0	0	50/50	0	50

For GP:

- ▶ GP always manages to converge to an optimal solution
- ▶ ... but under  $fit_1$ , all solutions found are linear!
- ▶ Possible explanation: GP first converges to linear pairs (since it has the XOR operator), then OLS are easily found

On the other hand, for GA:

- ▶ GA converged just once for  $n = 8$  and the performances for  $n = 7$  are worse than GP
- ▶ ... but all solutions found are nonlinear, even under  $fit_1$

# Conclusions and Future Directions






## Wrapping up:

- ▶ We addressed for the first time the problem of designing orthogonal Latin squares based on nonlinear CA through GA and GP
- ▶ We experimented with three encodings for the candidate solutions and two fitness functions
- ▶ The problem seems to be easy for GP up to  $n = 8$  variables, less so for GA (who converges only up to  $n = 7$  variables)

## Future directions:

- ▶ Investigate the behaviour of GP under  $fit_1$  (why does it always find linear solutions?)
- ▶ Design other encodings for the candidate solutions of GA

# References

-  [Keedwell15] Keedwell, A.D., Dénes, J.: Latin squares and their applications. Elsevier (2015)
-  [Eloranta93] Eloranta, K.: Partially Permutive Cellular Automata. Nonlinearity 6(6), 1009–1023 (1993)
-  [Mariot16] Mariot, L., Formenti, E., Leporati, A.: Constructing Orthogonal Latin Squares from Linear Cellular Automata. In: Exploratory papers of AUTOMATA 2016 (2016)
-  [Mariot15] Mariot, L., Leporati, A.: A Genetic Algorithm for Evolving Plateaued Cryptographic Boolean Functions. In: TPNC 2015: 33-45 (2015)
-  [Tomba88] Tompa, M., Woll, H.: How to share a secret with cheaters. J. Cryptology 1(2), 133–138 (1988)