Elementary Cellular Automata as an Error Minimized Hash

Daniel McKinley

November 2024

1 Introduction

Elementary cellular automata (ECA) are extensions of logic gate truth tables done linearly in parallel. [2] Here they are explored as a lossy compression algorithm that works by minimizing discrepancies between the input and a codeword's ECA output. General algorithm, specific ECA rule, and aggregate properties relevant to specific applications are discussed. It is implemented in Java at [1].

2 Main Algorithm

The algorithm is ultimately lossy compression implemented via codewords' wrapped ECA output that minimizes discrepancies between the input data and that wrapped output, with an weighted coefficient error sum for each possible codeword.

The algorithm has 3 properties of a good hash algorithm, and 2 that make it more suitable as a signal hash rather than a block data hash [some hash paper citation]. Each input has a unique solution, the solutions are distributed evenly across all possible solutions, and it loses less than half the bits per compression cycle. Small changes to the input do not produce large changes in the solution, and errors in compression are semi uneven which is mitigated in context of a signal.

Random data input with wrap-around columns, it took about 10000 trials * 256 ECA rules to get consistent results in the sorted best-performing lists									
	Input is random binary with equal probability (0,1)								
wrap	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	wrap
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	
	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	Input□□	

All possible input neighborhoods of a given size are computed with each trial neighborhood placed at row 0. 64 bits input = 8 bit neighborhood = 256 possible codewords, b0..b7 = bitN of each possible neighborhood									
	ECA[row][column] = Wolfram[rule, row-1, {column-1,column,column+1}]								
wrap	b0	b1	b2	b3	b4	b5	b6	b7	wrap
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	
	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	ECA□□	

All possible input neighborhoods of a given size are computed with each trial neighborhood placed at row 0. 64 bits input = 8 bit neighborhood = 256 possible codewords, b0..b7 = bitN of each possible neighborhood									
	ECA[row][column] = Wolfram[rule, row-1, {column-1,column,column+1}]								
wrap	b0	b1	b2	b3	b4	b5	b6	b7	wrap
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	
	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	ECA▯▯	

Include graphics for: Input example, Codeword output example, Error map example

2.1 Compression properties

2.1.1 Unique Solutions

Certain ECA rules reduce to unique solutions for any given input. Another overlapping set reduces to a unique solution when the errorScore is calculated with an exponent of 4 instead of 2.

2.1.2 Even distribution of solutions

All possible solutions for any given rule have an equal probability of occuring over enough iterations of random input.
Errors are distributed equally across columns, and with the errors in the last row roughly half that of the first row. In the context of a signal you would have a tendency to linearly lose more in either the high or low frequencies.

2.1.3 Accuracy

Certain ECA rules, including a large overlap with the XOR additive rules [citation] have better than 2/3 accuracy/bit with most rules having better than 1/2 accuracy/bit over all the input bits. The wrapped/non-wrapped tradeoff is 2/3 accuracy and an uneven errors/row w hen done repeatedly or 1/2 accuracy and linear errors/frequency. In the non-wrapped signal version the best performing rules had better than 1/2 accuracy/bit.

2.1.4 Distance between codewords

If you change the input of a given (input, solution) and score the difference the same way you score the error that gets minimized, the mean changeScore per Hamming change in the codeword is roughly 100, corresponding to a one bit difference at any column between row 6 and row 7 where $2^r ow = changeScore/bit$.

2.1.5 Avalanche-ability

Avalanche property

2.1.6 Individual rules' performance

Results are checked for all neighborhoods with stochastic input, roughly 10000 are required for consistent results in some categories. Rules 90,60,102,153 are the best performing rules in terms of accuracy with several class 4 rules performing almost as well as the Pascal triangle set. When the error scoring exponent is changed to 4 instead of 2, several more members of the XOR additive rules perform well.

2.1.7 Wrapped v non-wrapped

Various shapes and sizes of arrays

2.1.8 Differential Equations

CDT of spectral method of solving DEs

2.1.9 Pi Phi ratio errors in 150

Row ratio constants

2.1.10 Various error score coefficients

Various error score coefficients

2.1.11 Voting subfunction to increase accurace

voting() subfunction to increase accuracy

References

[1] Daniel McKinley. github.com/dmcki23/, 2024.

[2] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.