EVOLUTION OF CELLULAR AUTOMATA MACHINES

Alan Fedoruk

March 11, 2003

INTRODUCTION

Exploring the use genetic algorithms to evolve rules to program cellular automata

---

- Cellular Automata (CA) have been studied for many years and have many interesting properties including universal computation—von Neumann [7], Conway [1], Wolfram [8].
- von Neumann showed that a CA can implement a Turing machine and therefore exhibits universal computation.
- Wolfram uses simple programs (CAs and others) as models for most real-world processes and is trying to build a *New Kind of Science* around that idea. Complexity does not need a complex model!

- Conway's *Game of Life* has been extensively studied as a model for real-world processes, to implement a universal computer and as a fun diversion.
- 1D CAs are possibly the simplest example of decentralized system which exhibits emergent computation.
- Problem: The huge rule space and the low level of the rules makes it difficult to program CAs for non-trivial computation. One solution: use genetic algorithms to evolve *programs*.
- Today's topics: Work done by Sipper using non-uniform CAs and the work done by the EvCA group at SFI.

FOCUS

- Mitchell, Crutchfield, Das et al, 1994-1999 – The Santa Fe Institute: the Evolving Cellular Automata (EvCA) Group
  - evolve 1D CAs to perform 2 tasks : density and synchronization
  - computation mechanics framework [3] to analyze how global interaction is emerging from local interactions
- Sipper 1997 – *Evolution of Parallel Cellular Machines* [6]
  - uses non-uniform CAs
  - follows up on work at EvCA
  - overall goal – evolvable hardware

TERMINOLOGY

- IC – Initial Condition
- r – radius of neighbourhood (r=2, neighourhood, n = 5)
- N – size of lattice
- M – time iterations to evolve
- Cellular Automata as a Computer
  - not a Turing Machine approach
  - IC is the input, state of the lattice after M iterations is output
  - program emerges as the evolution of the lattice by repeated application of the rules

APPLICATION AREAS

Density and Synchronization Tasks

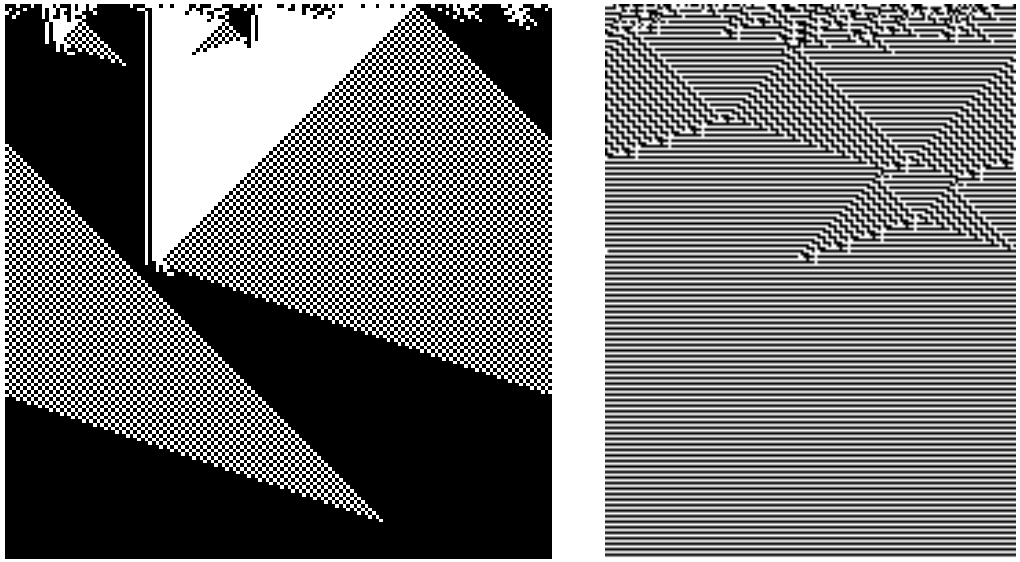


Figure 1: Examples of Density and Synchronization

EvCA Group Work

Use genetic algorithms to evolve rules to program cellular automata to perform the density and synchronization tasks

---

- r=3, rule space = 2^{128}; N = 149; IC randomly selected from 2^{149} possibilities
- 100 rules are randomly generated, fittest 20 (elite) moved to next generation, other 80 generated with crossover and mutation from elite
- 100 new ICs chosen at each generation, run with each generated rule
- fitness: fraction of correct solutions
- run for 100 generations

RESULTS

Evolved *Particle* Rule for the Density Task

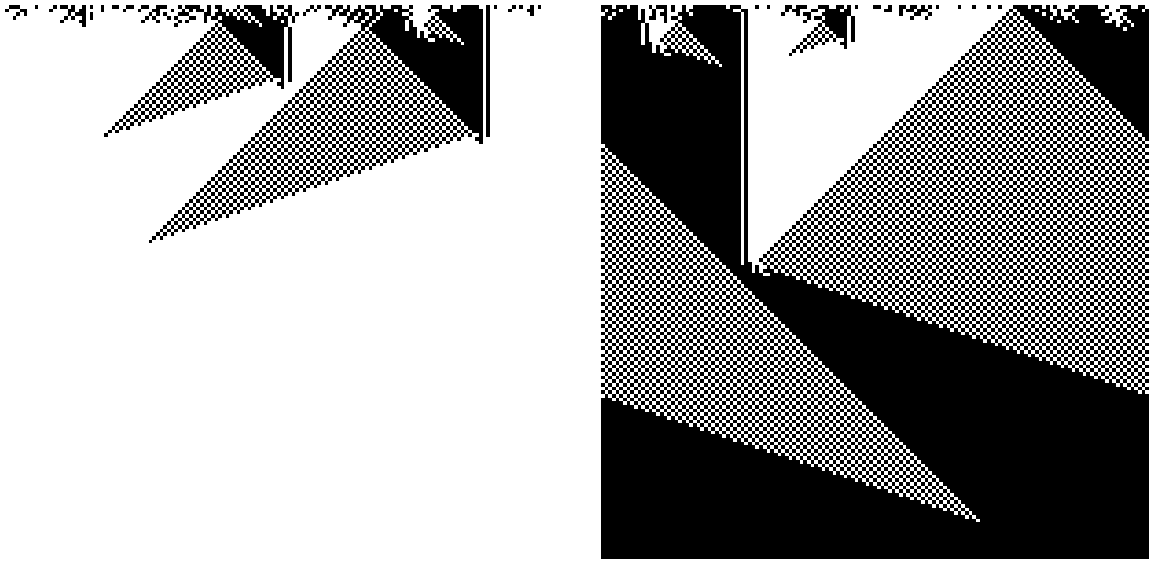


Figure 2: Evolution of *particle* rule with low and high density input. Effectiveness $\approx 95\%$ [5]

RESULTS

Evolved Rule for the Synchronization Task

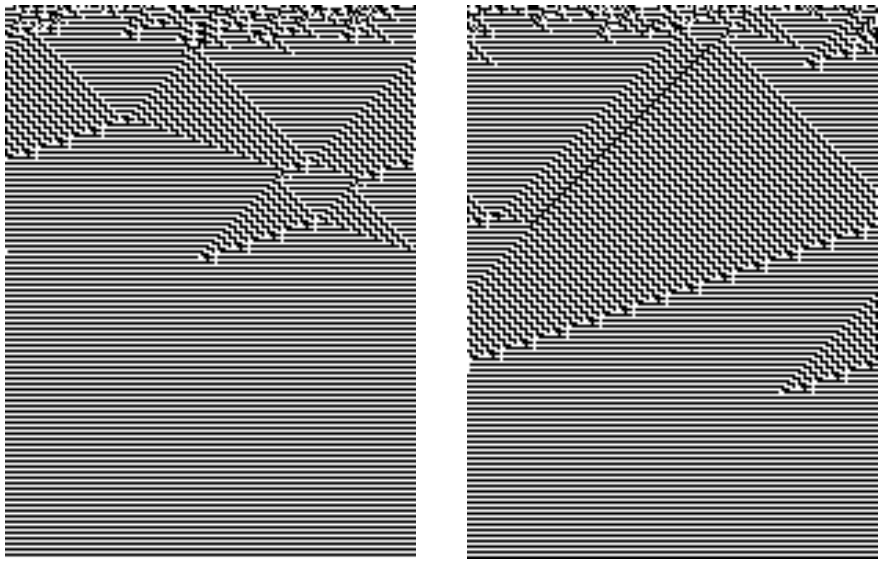


Figure 3: Two runs using evolved rule for synchronization task. Effectiveness = 100% [2]

ANALYSIS

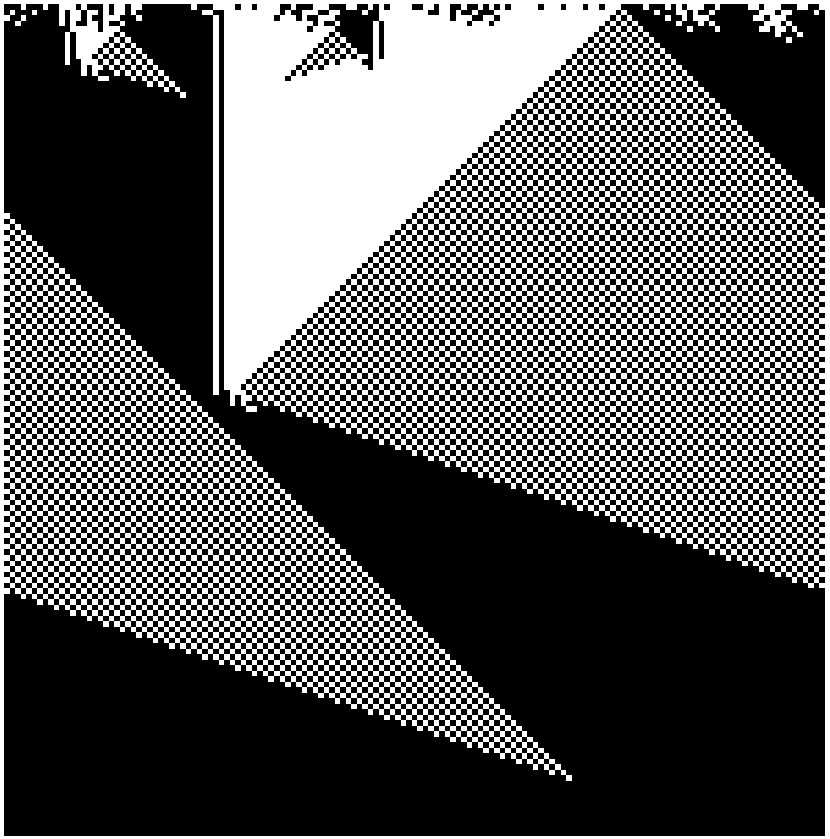


Figure 4: High Density run showing domains, particles and interactions

ANALYSIS (CONTINUED)

- using a GA is an effective means of finding rules in the huge rule space
- the computation mechanics framework [3, 4] models how local interactions can lead to emerging global behaviour
- particles transmit information from one location to another
- a table of domains, particles and particle interactions can be constructed
  - in the example, domains are all $1s$ – black, all $0s$ – white and $(01)s$ – checkered
  - particles are defined by domains they separate and their speed
  - particles can decay, react or annihilate
- the particle model can also be used to predict behaviour [4]

EVOLVING NON-UNIFORM CAS — SIPPER

- tackles the same problems: Density and Synchronization
- uses a non-uniform CA: each cell may contain a different rule
- rather than evolving a global set of solutions: fitness is calculated at the cell level and genetic operations applied to the neighbourhood

EXPERIMENTS

- r=3, N=149, M=150, rule space = $(2^{128})^{149} = 2^{19072}$; IC randomly selected from 2^{149} possibilities
- r=1, N=149, M=150, rule space = $(2^{8})^{149} = 2^{1192}$; IC randomly selected from 2^{149} possibilities

Both experiments used an initial set of randomly chosen rules. 300 ICs were used to calculate fitness.

RESULTS

- r=3 density task: found a non-uniform rule with $\approx 92\%$ effectiveness; the best uniform showed $\approx 95\%$
- r=1 density task: for uniform CAs best possible is 83% effectiveness, non-uniform showed $\approx 93\%$
- r=1 synchronization task: for uniform CAs best possible is 84%, non-uniform showed 100%

ANALYSIS

- Non-uniform CAs can attain high performance on non-trivial computation.
- Coevolution can be used to perform the computation — results in quasi-uniform system.
- Non-uniform CAs have lower connectivity requirements i.e. lower r with same results.

ANALYSIS

Sipper also performed experiments with different application areas using 1D and 2D CAs

- Random Number Generators – produce random digits
- Rectangle – identify a rectangle in a grid
- Thinning – find thin representations of rectangles
- Ordering – Move all 0s to the left
- Density – 2D
- Synchronization – 2D

How Does Evolution Work

Both research groups tried to examine how the evolution occurred.

- EvCA group found evolution happened in leaps. They dubbed these epochs.
- Sipper examined how the various genes interacted and how they affected the resulting evolved program.

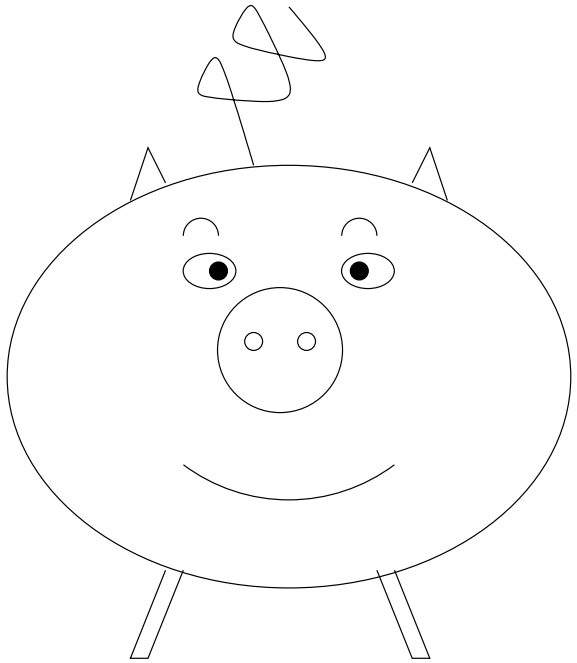
SUMMARY

- CAs perform distributed, parallel computation with only local interaction.
- Genetic algorithms are an effective way to program CAs, whether using non-uniform or uniform CAs.
- Global behaviour can emerge; information is transported through the grid which can be modelled via particles and domains.
- Models processes occurring in natural systems

FURTHER READING

- The Santa Fe Institute, Evolving Cellular Automata Group: `http://www.santafe.edu/projects/evca/`
- Moshe Sipper Webpage: `http://www.moshesipper.com/`
- Stephen Wolfram: `http://www.stephenwolfram.com/`

Discussion



References

[1] E. Berlekamp, J. H. Conway, and R. Guy. *Winning Ways for Your Mathematical Plays, vol. 2*. Academic Press, New York, 1982.

[2] R. Das, J. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, 1995.

[3] J. E. Hanson and J. P. Crutchfield. The arractor-based protrait of a cellular automaton. *Journal of Statistical Physics*, 66(5/6):1415–1462.

[4] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-particle computation in evolved cellular automata. In T. Toffoli, M. Biafore, and J. Lea, editors, *Physics and Computation '96 (Pre-proceedings), New England Complex Systems Institute*, pages 153–158, 1996.

[5] M. Mitchell, J. P. Crutchfield, and R. Das. 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)*, 1996.

[6] M. Sipper. *Evolution of Parallel Cellular Machines: The Celluar Programming Approach*. Springer, 1997.

[7] J. von Neumann and A. W. Burks. *Theory of Self- Reproducing Automata*. Univ. of Illinois Press, Urbana, IL, 1966.

[8] S. Wolfram. *A New Kind of Science*. Wolfram Media, Inc., 2002.