Network Automata and Emergent Phenomena

Richard Foard

**Abstract.** A simple, rule-based graph automaton was defined and simulated. Each of its 722 trillion possible rules specifies a set of local topology and node state changes to be applied iteratively, starting with a random initial graph. Simulation runs were performed using many different rules, each run terminating when its graph collapsed or cycled. Evolving and terminal graph states were analyzed macroscopically, using aggregate statistics, and microscopically, by inspecting graph structures. Results were compared with those from simulations of a machine that iterated by applying random, rather than rule-based, changes. Simple neural networks were unable to predict terminal graph characteristics given the rule that yielded the graph. Using a genetic search procedure to search the rule-space for rules producing target terminal characteristics yielded results dramatically better than those from random searches.

**Introduction.** Since Alan Turing conceived his universal machine in 1936, simple abstract automata have drawn research attention. Interest broadened beyond the academic realm in 1970, when Conway published his Game of Life simulations1 that highlighted the ability of uncomplicated cellular automata to behave in complex ways. Re- searchers in the nascent field of complexity theory began studying similar phenomena, such as the sandpile avalanches first explored by Per Bak2.

In A New Kind of Science3, Stephen Wolfram systematically analyzed a variety of cellular automata types. He found particular inspiration in the behavior of a one-dimensional machine running ”Rule 110.”

Wolfram and others have suggested that some natural processes that were previously thought to evolve by natural selection are instead manifestations of things that nature found ”easy” to accomplish using the same fundamental principles that underlie simple automata.

In this work, we analyze simple, rule-based graph automata using simulations of abstract machines. Our machines operate on the same principles as cellular automata but use a graph, rather than a ”tape” or grid, as a substrate. Where cellular machines define cell adjacency spatially, our machines use graph topology. We discuss the machines’ behavior under varied rules and explore a genetic search algorithm for finding rules that yield terminal graphs with specific characteristics.

**Technical Approach.** Two similar rule-based automata, Machine C and Machine CM, were designed and implemented in simulation. The simulators were run repeatedly, using rules selected at random from a large universe of possibilities. Measurements were recorded as the graphs evolved through each run. The resulting database accumulated data on many thousands of runs. The body of data was analyzed macroscopically, using aggregate statistics, and microscopically, by inspecting statistical and graphic snapshots from individual runs.

Each simulation run begins with a selected rule and a randomly generated graph. It proceeds by iteratively modifying the graph’s edges and node states according to the rule. Each rule is effectively a simple program that specifies, based on the state of each node and its neighbors, how local changes should be applied during an iteration.

The automaton simulator was constructed in C++. It can be used to simulate the rule-based automaton with or without permitting multi-edges in the evolving graphs. It can also simulate operation in  
a random mode, in which node state and topology changes are still applied within each node’s two-hop neighborhood but are determined at random rather than by applying a governing rule. For each execution, command line arguments select the type of automaton, governing rule number, number of graph nodes, depth of cycle-checking, random number seed, and maximum number of iterations.

*Outcome Prediction with Simple Machine Learning*

A simple learning experiment was conducted to gauge whether, given a specified rule, a trained neural network could predict the values  
of the machine’s terminal graph. Training data consisted of (encoded\_rule, statistic\_value) pairs in which statistic\_value was one of:

• residual number of nodes  
• average clustering coefficient  
• number of iterations to reach terminal state • diameter  
• outcome (cycling vs. collapse)

Two types of encoding were used for the encoded\_rule value:

• Rule-parts: A vector of eight integers in the range [0, 72 (6 × 6 × 2)], each representing the local transformation to be applied depending on the compound state of a node and its two neighbors, and,

• Rule-map: A vector of 112 0/1 values encoding the rule in a modified one-hot scheme. Where an edge destination could take values L, R, LL, LR, RL, or RR, for example, the destination RL would be redundantly encoded in the bit sequence 0,0,0,0,1,0, indicating L=0, R=0, LL=0, L=0, RL=1, RR=0.

*The Rule-space Searcher (Searcher)*

The rule-space for machine C is vast and incoherent. It comprises  
7.2 × 1014 rules; numerically similar rules produce very dissimilar behaviors and yield widely varying terminal graph states. We developed a genetic search algorithm6 for finding rules that produce terminal graph states with specific characteristics such as high clustering coefficients, or a large number of (separate) connected components. A ”fitness” function quantifies the relative extent to which a rule pro- duces a terminal graph with the desired characteristics.

The search proceeds by first creating a pool of randomly selected rules, then improving aggregate fitness by iteratively replacing lower- fitness rules with new ones. The replacement rules are synthesized by choosing two existing rules with relatively high fitness and ”crossing”

• residual number of nodes  
• average clustering coefficient  
• number of iterations to reach terminal state • diameter  
• outcome (cycling vs. collapse)

emergence in network automata 6

6 The algorithm design benefitted from the experience of Adonis Bovell, GTRI CIPHER laboratory.

them, possibly mutating one of them before crossing. The process iterates, terminating either when the pool reaches a target fitness  
or when an iteration limit is reached. Successful searches terminate with a pool having significantly higher fitness than that of randomly generated pools.

A rule is mutated by replacing one or more of its components with a new, randomly chosen value. Two rules are crossed by randomly choosing one or more of their components and exchanging them. The likelihood of a mutation occurring is controlled by the probMutation parameter.

The search proceeds in generations. Each successive generation creates a new pool of rules from the previous generation’s by repeatedly choosing pairs of rules from the previous pool, mutating and crossing them, then adding the chosen rules and their new offspring to the new pool until it is full. The process repeats until maxGenerations have been generated.

Rules with relatively high fitness are selected from a pool using a list of rules in order of decreasing fitness. Each entry is accompanied by cumulative fitness, summed from the beginning of the list. A choice is made by choosing a target fitness value randomly.

**Technical Results**

*In-Degree Entropy*

Machine C, on average, generates graphs with a larger maximum in- degree than in the random case of machine R, and also produces a larger number of distinct in-degrees.

Shannon’s entropy8 computed on summary in-degree statistics and normalized,9 can be regarded a measure of a graph’s ”randomness.” As would be expected, entropy is consistently large in the randomly generated initial graphs. Between the R and C machines, entropy in the terminal graphs for C is lower than that for the randomly operating R (Figure 10).

The drop in average entropy between initial graphs and the R machine’s terminal graphs seems surprising on its face, but is accounted for by the restriction that R’s topological changes, like C’s, may only redirect a node’s out-edge within its two-hop neighborhood. The effect is to ”localize” the randomness, abruptly increasing apparent order in the random graph as soon as the first iteration is finished.

*Degree Distribution*

Because of the structure constraint that all nodes have out-degree  
of two, the distribution of out-degrees is always flat at that value. In-degrees, on the other hand, may vary unconstrained as successive iterations transform the graph. The terminal graphs produced by both the C and R machines almost invariably had in-degree distributions characterized by a large plurality of nodes with degree 0 or 1 and a handful of nodes with varied, very large degree. The blue line in Figure 11 shows the typical case; for comparison, the green line plots the distribution for a very rare case exhibiting the power-law distribution that appears for most real-world networks.

*Predicting Outcomes*

Using the KERAS framework, a simple neural network with an input layer, a single hidden layer, and an output layer was constructed; weights were updated using stochastic gradient descent. For both rule-part and rule-map input encodings, with networks trained over half the rules for which data was available, no outcome statistic was predicted with accuracy better than that of random choice. No further attempt was made at predicting outcomes.

*Searching the Rule-space*

The genetic search algorithm described in Methods was applied to the task of finding rules that generated terminal graphs with characteristic statistics falling within specified ranges. For most characteristics, the

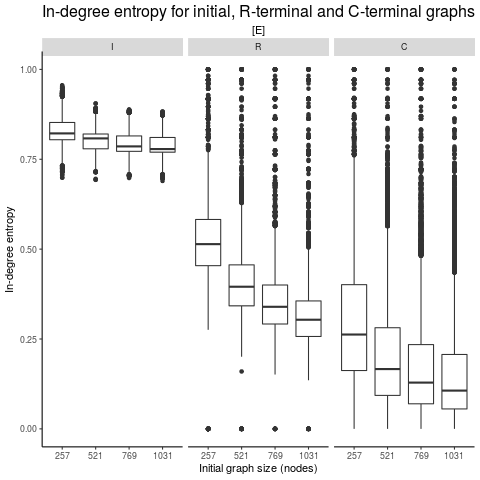


Figure 10: In-degree entropy is largest in initial random graphs, smaller for R’s terminal graphs, and smallest for C’s terminal graphs.

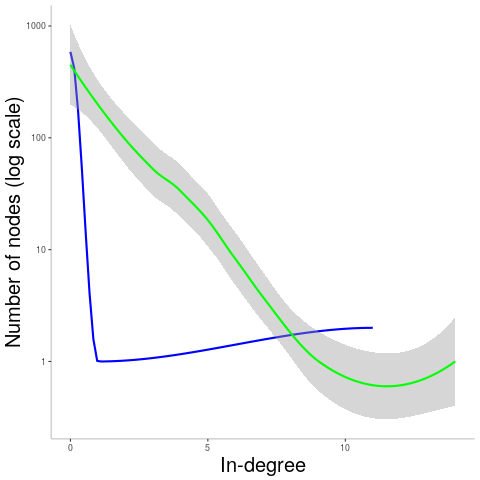


Figure 11: In-degree distribution excerpts for the typical case (blue) and a rare outcome (green) in which a power-law distribution was generated

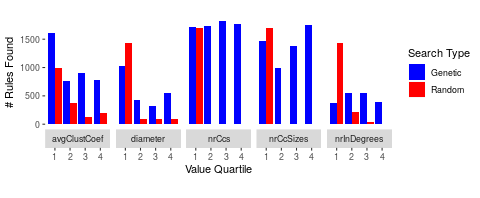


Figure 12: Search performance. Number of rules found with terminal graph statistics in the target quartile of the range after examining 1700 rules chosen using a random (red) or genetic (blue) search strategy.

genetic search algorithm performed dramatically better than random selection. Results are summarized in Figure 12.

**Conclusions.** 40% of terminal graphs produced by Machine C using randomly chosen rules did not collapse to empty. As a group, they had widely vary- ing configurations and degree distributions. Only a handful, though, showed the power-law, ”scale-free” distribution that is a signature of most real-world networks. It is possible that, under the correct trans- formation, scale-free structures would be found to be encoded at a coarser level of detail than that of the final graphs – this is a possible area for further investigation.

A pruning process was applied to Machine C’s evolving graphs in order to maintain structure constraints. Pruning typically had the effect of reducing node counts in an irregularly cascading fashion reminiscent of Per Bak’s sandpile avalanches. Further research specifically focused on this phenomenon could illuminate interesting dynamics.

The failure to predict outcomes using simple neural nets suggests that Machine C behavior is not reducible to a simpler means of generation.

The effectiveness of genetic search in finding rules producing specific values in terminal graph statistics, on the other hand, may indicate the presence of regularities in the seemingly incoherent rule-space. It may be useful to explore the possibility more thoroughly, to see if it is feasible to discover rules that give rise to useful functional characteristics in evolving or terminal graph topologies.

The correspondence of graph structures and their dynamics with mathematical constructs was not investigated in this work, and could conceivably point out parallels that open new lines of inquiry.

In this study, practical considerations dictated the selection of a single type of automaton and accompanying graph topology restrictions. Examining other, similar automata types could give insight into strategies for seeking other abstract machines with behaviors of theoretical or practical interest.

**References and Notes.**

1 Games, M., 1970. The fantastic combinations of John Conway’s new solitaire game “life” by Martin Gardner. Scientific American, 223, pp.120-123.

2 Bak, P., 2013. How nature works: the science of self-organized criticality. Springer Science & Business Media.  
3 Wolfram, S., 2002. A new kind of science (Vol. 5, p. 130). Champaign, IL: Wolfram media.

4 Erdos, P. and Renyi, A., 1959. On random graphs I. Publ. Math.

5 Black is interpreted as zero, white as one. In all cases, ”neighbor” is used to indicate a node at the destination end of one of a node’s out-edges.

6 The algorithm design benefitted from the experience of Adonis Bovell, GTRI CIPHER laboratory.

7 The number of possible states for an N-node graph with out-degree restricted to 2 is 2^N \* (N choose 2)^N.

8 Shannon, C.E., 1948. A mathe- matical theory of communication. Bell system technical journal, 27(3), pp.379-423.

**PLEASE NOTE. This article is a heavily excerpted and extensively reformatted version of the original. The author apologizes for the gaps and missing references that may have resulted. The original is available upon request.**