Genetic Boolean Networks with Non-Uniform Delay

Caleb Baker

I. Introduction

Stuart A. Kauffman, in his paper, Metabolic Stability and Epigenesis in Randomly Constructed Boolean Genetic Nets, suggests that “proto-organisms probably were randomly aggregated nets of chemical reactions.” Kauffman believed that these networks could be modeled as boolean networks similar to the logic circuits that make up the control systems of many modern electronics.

Under this view, the “gates” of the circuit represent genes. A gene can be either activated or deactivated, much like a logic gate can put either a high or a low signal on its output wire. This output is determined by the application of a boolean function to the inputs of the gene/gate. A boolean function can be thought of as a truth table which, given a set of inputs, produces an output of either 0 or 1. An example of a boolean function is the AND function. The AND function produces an output of 1 if and only if all of its inputs are 1. If any of the inputs are 0, then the output is 0.

What makes these boolean networks interesting is the fact that the genes are interconnected with one another. The inputs to each of the genes are the outputs of other genes. When one gene changes its output, it causes dozens of other genes to change their outputs in response to changing inputs. This change can propagate through the network, eventually changing the outputs of the genes that serve as inputs into the gene that changed in the first place. This, of course, causes it to have to change its output again.

Since a given network has only a finite number of states that it can be in, it is inevitable that it will eventually fall into a cycle in which it repeats the same set of states over and over again. Kauffman was interested in the lengths of these cycles. If a network fell into short cycle of only one or two states, then it represented a system which was too stable to allow the change that abiogenesis requires. If a network fell into an exceedingly long cycle with up to 10^{30} states, then it represented a system which was too chaotic to allow any kind of well-ordered life to arise. Kauffman believed that a network suitable to abiogenesis would have to have behavior that was “on the edge of chaos.” These are networks which have cycles that are long enough to allow variety, but short enough to enforce order.

Input A	**Input B**	**Output**
0	0	0
0	1	0
1	0	0
1	1	1

The AND function

Input A	**Input B**	**Output**
0	0	0
0	1	1
1	0	1
1	1	1

The OR function

Figure 1. Two Examples of Boolean Functions

Boolean networks are rather easy to create computer simulations of and so simulations tend to be an effective way to study them. Not surprisingly, then, Kauffman created a simulation which he used to study them. One important consideration in constructing such simulations, however, is how to handle time. It is necessary for there to be a delay between a change in a gene's input and the resultant change in its output. If there were no such delay, then a gene would be required to output both 0 and 1 simultaneously, which is not possible.

Kauffman responded to this issue by dividing time into discrete units and assigning identical delays to all genes. This meant that the values of the inputs to a gene at time T determined the value of the gene's output at time T + 1. To give a concrete example, suppose a gene G is defined by the AND function. Further suppose that the inputs into G at time 6 are 0 and 1. Since AND produces 0 for inputs of value 0 and 1, the output of G will have a value of 0 at time 7.

I think that this model of time is more simplistic than the situation allows for. Because of this, I have implemented my own simulation of boolean networks in which I use a continuous model for time. Each gate is assigned a random delay, δ, which is generated using either a normal or a uniform probabilistic distribution. If the inputs to a gene change at time T, then its output will updated to match at time T + δ.

II. Implementation Details

When starting a simulation, the simulator is passed five parameters and an optional distribution flag. The first parameter, n, determines how many genes the simulated network will contain. The second parameter, k, determines how many inputs each gene will have. The third and fourth parameters, α and β, affect the delays of the genes. The fifth parameter, ω, represents the time at which the simulation should be terminated if it has not already ended. The distribution flag, if present, will cause the processing delays of the genes to be determined by a normal distribution with a mean of α and a standard deviation of β. If the distribution flag is not present, then the delays will be determined by a uniform distribution with a minimum of α and a maximum of β.

The simulation constructs n gene objects. Each gene has an array of inputs which is assigned to k randomly selected genes from the network. A truth table is constructed for each gene's boolean functions and the output column of the table is randomly filled in. At time 0, the simulation assigns each gene's output to be either 0 or 1, chosen at random. For each gene, its output is calculated based on the output values assigned to all of its inputs. If the computed output value does not match the gene's initial output value then an event is constructed which says which gene's output should change, when it should change, and what it should change to. This event is then thrown into a list of future events (for those familiar with data structures, this "list" is actually a priority queue).

With the initial conditions set, the simulation now begins. As long as the list of events is not empty and the current time has not passed ω, the event with the earliest time is removed from the list and processed. The first step in the processing of an event is to compare the output specified by the event to the current output of the gene specified by the event. If the outputs match, the event is thrown out and the simulation moves on to the next event.

If, however, the outputs do not match, the output of the gene is changed to match that specified by the event. Events are then created for each gene whose input comes from the changing genes output. The values these genes should change to is stored in the events and the events are given times of $t + \delta$, where t is the current time and δ is the delay of the gene that a given event was created for.

An observant reader might note that, under these rules, the simulation would only terminate if the network had a cycle of length 1 (otherwise the event list would never dry up). That is why the simulation includes one other detail. After every event, the state of the network is obtained by looking at the current output of each gene. A history is constructed which records which states were visited and what order they were visited in. This history is analyzed for cycles. If the same pattern has been repeated four times in a row, the network is assumed to be locked into a cycle and so the simulation is terminated. The number of states in one iteration of the cycle is reported along with how many states were visited before the cycle was first entered.

III. Results

Table 1 presents a summary of the data that was collected from running 366 simulations of boolean networks. Each table row represents a set of simulations that were created with identical parameters. The n column represents the number of genes that were in each network. The k column represents the number of inputs that each gene had. The gene delay column represents the amount of delay (δ) associated with each gene. For the first 11 rows, every gene had the same δ value. For the next 4 rows, δ values for each network follow a uniform distribution where the first number given is the minimum δ value and the second number is the maximum. For the last 5 rows, δ values follow a normal distribution where the first number given is the mean and the second is the standard deviation. The remaining columns are self-explanatory, given some qualification. A network is considered frozen if it resolves to a cycle of length one. A network is considered chaotic if the simulation ran for half of an hour or crashed my computer before terminating. Data from "chaotic" networks is not considered in computation of mean and median cycle lengths and mean run-ins. The run-in is the number of states encountered before a cycle was entered.

The first eleven rows act as a control. These rows represent the type of network that Kauffman studied. The data from these rows verifies some of the conclusions that Kauffman came to for networks of this type. In particular, networks where k = 1 tend to fall into very short cycles, whereas networks where k approaches n tend to have very long cycles. Further, it is significant that no network of this type was found to be chaotic, whereas several of the networks with non-uniform delays were chaotic. This gives indication that networks with non-uniform gene delays are more likely than networks with uniform gene delays to have long cycles or long run-ins. Due to the lack of rigour in my definition of chaotic, nothing more precise than this can be surmised from this observation.

Rows 11 through 18 maintain constant n and k parameters with 8 different distributions of delay. Most distributions produced higher average cycle lengths than were produced by uniformly giving all genes the same delay. The two exceptions to this rule are the uniform distribution on a range from 1 to 10 and the normal distribution with a standard deviation of 0.2. It should be noted, however, that both of these distributions produced a sizeable number of chaotic networks (5 and 8 respectively) and so it is possible that their actual means could be larger than what I have calculated.

Despite having higher average cycle lengths, all distributions which allowed non-uniform delay fell into cycles with a length of one or two in more than a third of their tests. That is nearly three times as often as happened in the control tests, of which one found a cycle of length two and four found cycles of length one. The effects of this phenomenon upon median cycle lengths can be observed in figure 3.

One final thing to note before moving on to more detailed sets of data is that distribution of non-uniform delay always produced much higher average run-in times than the control case.

We will now look closer at the cycle lengths of networks with particular distributions of delay. Figure 5 contains a histogram of cycle lengths for networks with uniform delay. For all histograms in this paper, positions on the x-axis represent cycle-length and the height of the vertical bar at a position represents the number of simulations that resulted in cycles of that length.

n	k	Gene delay	Number of tests	Number of "frozen" networks	Number of "chaotic" networks	Mean cycle length	Median cycle length	Mean run-in
26	25	1	16	0	0	5820.375	3495	4067.312
16	15	1	1	0	0	279	279	12
8	7	1	1	0	0	5	5	6
100	1	1	11	10	0	1.091	1	5.364
8	1	1	1	1	0	1	1	2
16	1	1	1	1	0	1	1	2
50	1	1	7	3	0	10.666	3	4.875
64	1	1	8	7	0	1.625	1	6
1024	1	1	4	3	0	1.75	1	9
1000	2	1	21	7	0	17.238	4	19
400	2	1	32	4	0	139.563	11.5	199
400	2	0.8 to 1.2	32	5	5	457.926	2	2485981
400	2	0.999 to 1.001	32	4	0	493.581	42	164465.5
400	2	1 to 10	33	8	5	92.889	2	1091063
400	2	1 to 100	32	5	10	1406.409	2	2106070
400	2	μ: 1 σ: 0.001	32	1	0	561.156	76	163801.4
400	2	μ:1 σ: 0.2	33	4	8	65.4	2	2622727
400	2	μ: 1 σ: 0.5	32	4	7	777.12	2	2104766
1000	2	μ: 1 σ: 0.2	32	6	9	1766.696	2	1035450
400	3	μ: 1 σ: 0.2	5	0	5	--	--	--

Table 1. Data Collected From Simulations

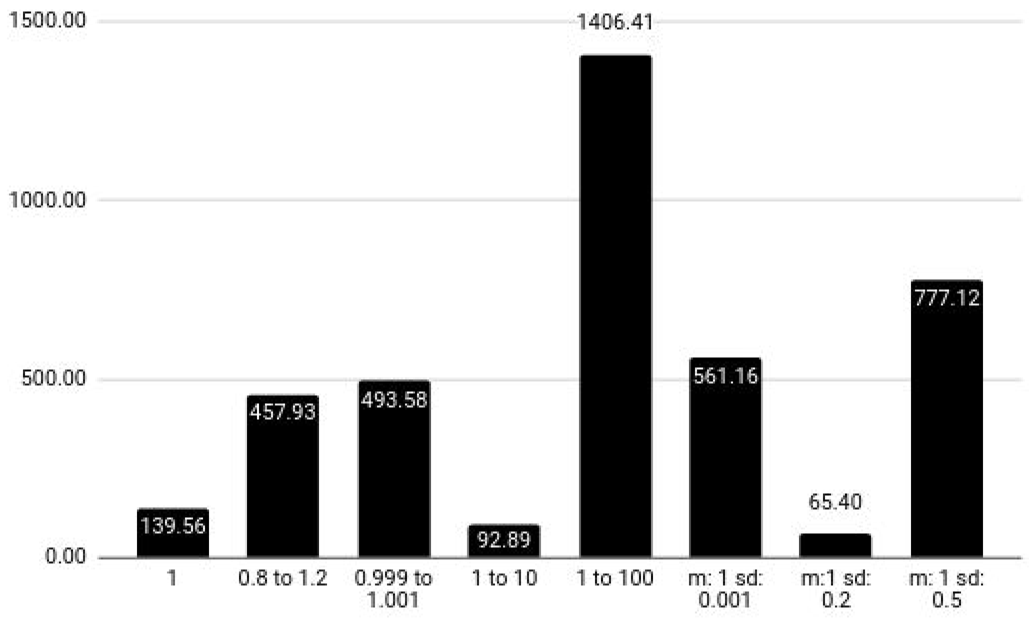

Figure 2. Mean Cycle Lengths of n=400, k=2 Networks with Various Distributions of Delay

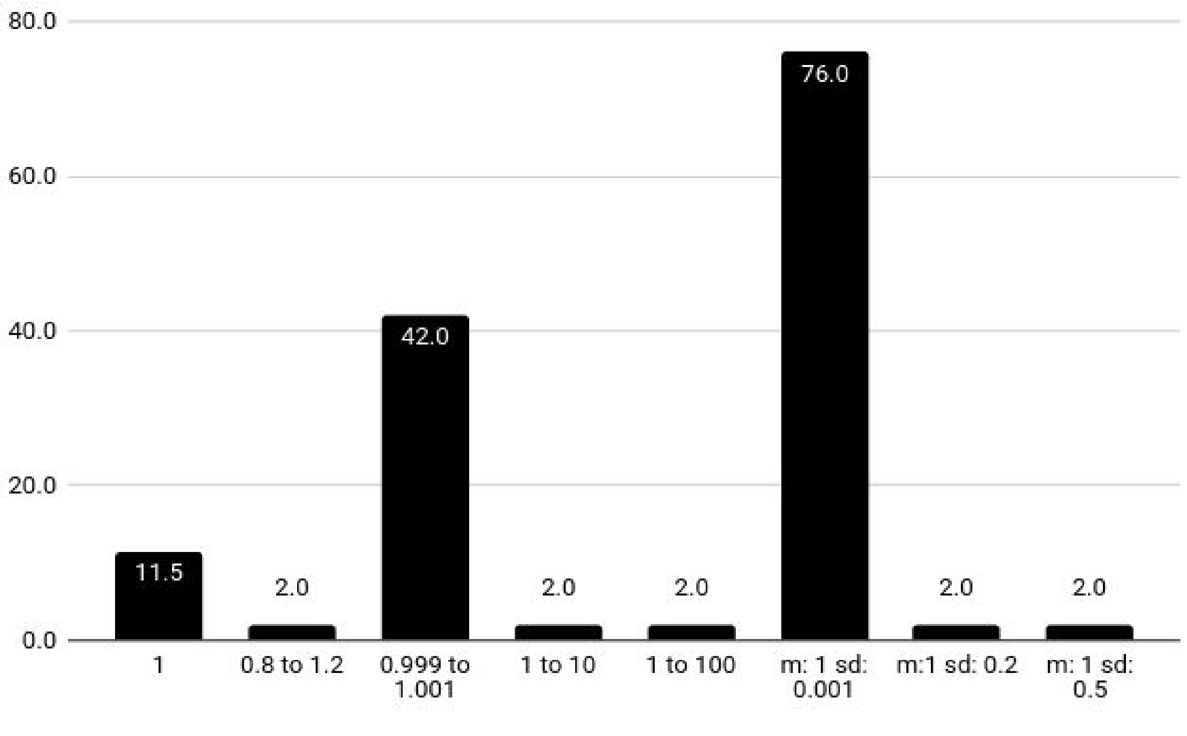

Figure 3. Median Cycle Lengths of n=400, k=2 Networks with Various Distributions of Delay

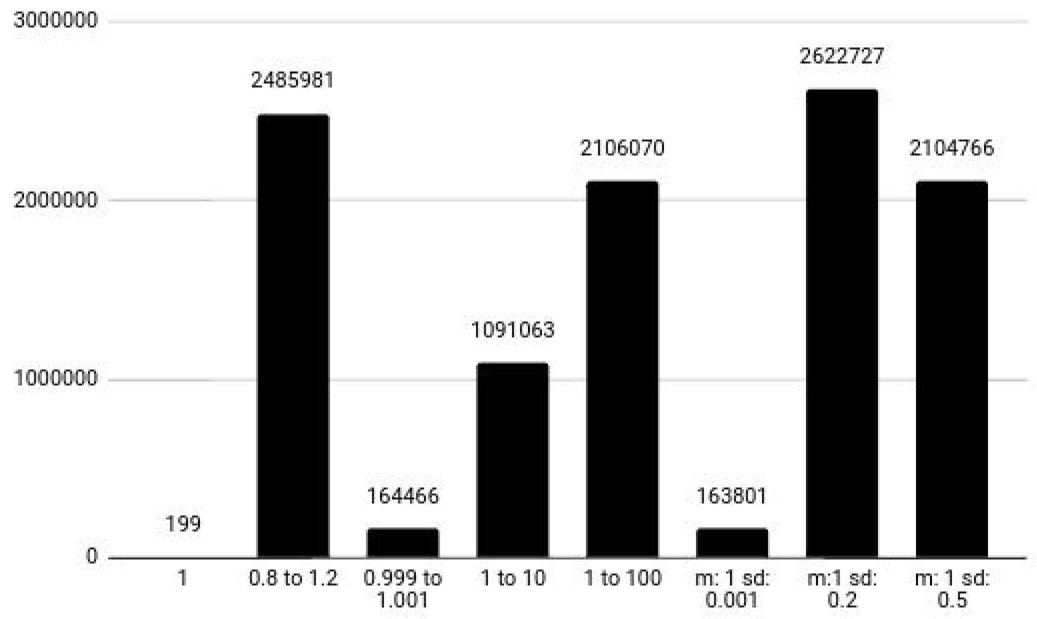

Figure 4. Mean Run-In Times of n=400, k=2 Networks with Various Distributions of Delay

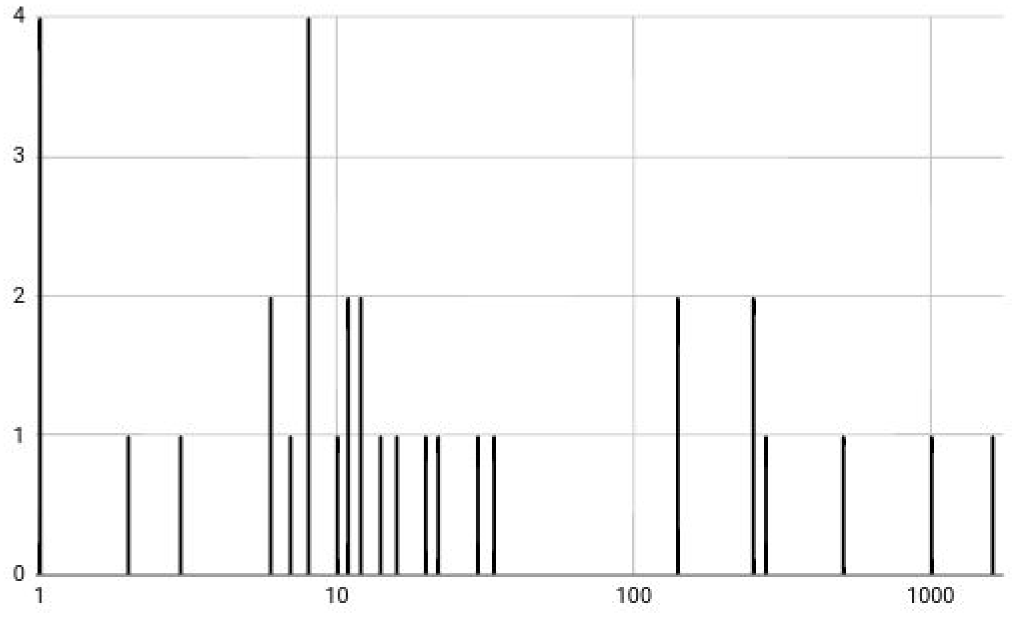

Figure 5. Histogram of Cycle Lengths for an n=400, k=2 Network with Uniform Delay

Because this is the distribution which Kauffman studied, it is not surprising that figure 5 looks very similar to some of the histograms that appear in Kauffman's paper. Much of the data represents cycles with lengths between 6 and 34. This particular clustering of data is indicative of the "short stable cycles" that Kauffman believes to be necessary for life. Although there are outliers that break from the pattern, such as the one eighth of the simulations that resulted in frozen networks, the data seems to support Kauffman's conclusions regarding the behavior of this type of network.

The next question to ask is, "What happens when the delays are allowed to be non-uniform?" We will start with only small variation. Specifically we will look at a normal distribution with a standard deviation that is 0.1% of the mean. This means that 68% of delays will be within 0.1% of the mean and 95% of delays will be within 0.2% of the mean. Figure 6 shows a histogram of cycle lengths for networks of this type.

Despite only allowing small deviations from the average delay, the effects on the distribution of cycle lengths were significant. Only three networks of this type fell into cycles with lengths between 2 and 108. 11 out of 32 of these networks fell into cycles of length 2 or less. Those that fell into longer cycles typically fell into much longer cycles. These networks, then, are mostly not of the type that Kauffman believed could produce life, but there are some that are.

Now that we have looked at networks with small amounts of variation in delay, we will look at networks with a larger amount of variation in the delay. Figure 7 shows a histogram of cycle lengths for networks with a standard deviation for node delay set 20% of the mean. The data follows the same trend for these networks as it did for those with a standard deviation that was 0.1% of the mean, but these trends were taken to greater extremes. 17 out of 33 tests found cycles of length 2 or less and 8 tests resulted in chaotic networks. 4 tests resulted in cycles with lengths between 10 and 84, inclusive.

Finally, we will look at networks with delays distributed according to a uniform distribution. Figure 8 presents data relating to networks with a uniform distribution of delay between 0.8 and 1.2. The same bimodal tendencies from the normal distribution tests are seen once again. 14 tests resulted in cycles of one or two states and 5 tests resulted in chaotic networks. Only two tests found cycles with lengths between 10 and 100.

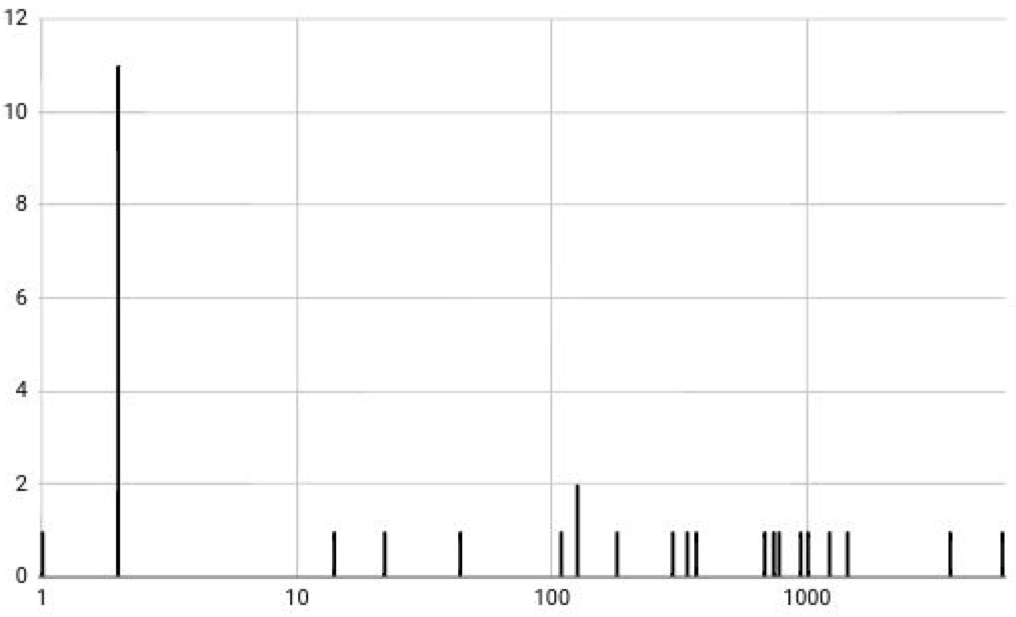

Figure 6. Histogram of Cycle Lengths for an n=400, k=2 Network with a Normal Distribution of Delay (Standard Deviation is 1/1000 of Mean).

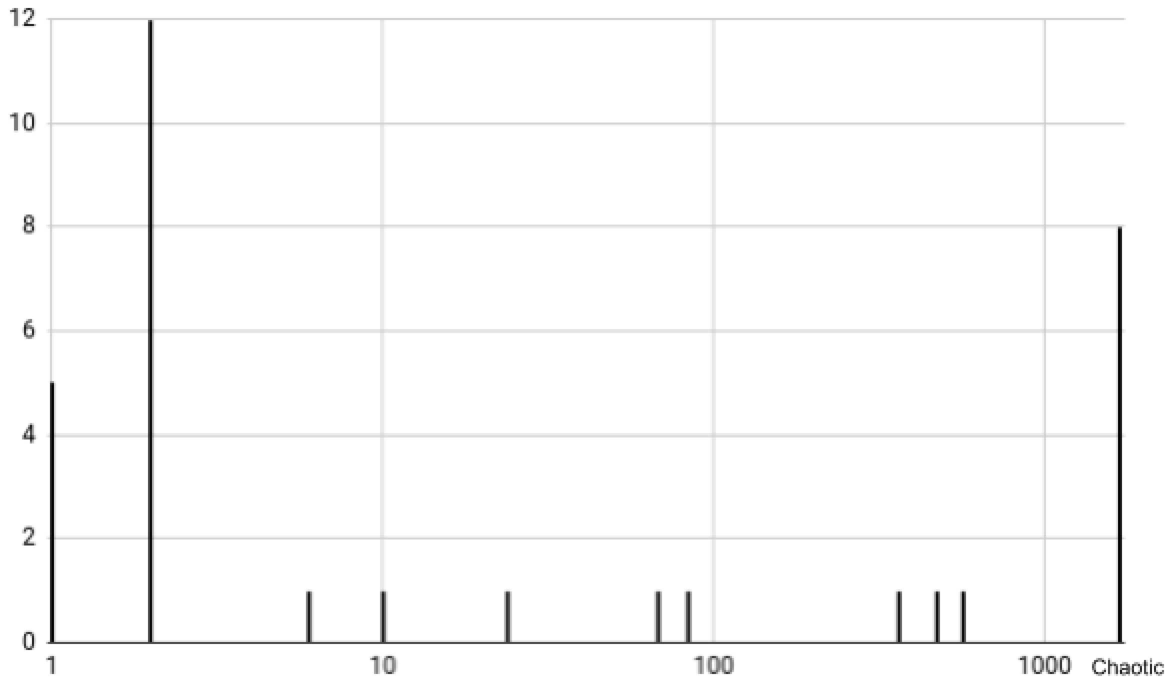

Figure 7. Histogram of Cycle Lengths for an n=400, k=2 Network with a Normal Distribution of Delay (Standard Deviation is 1/5 of Mean)

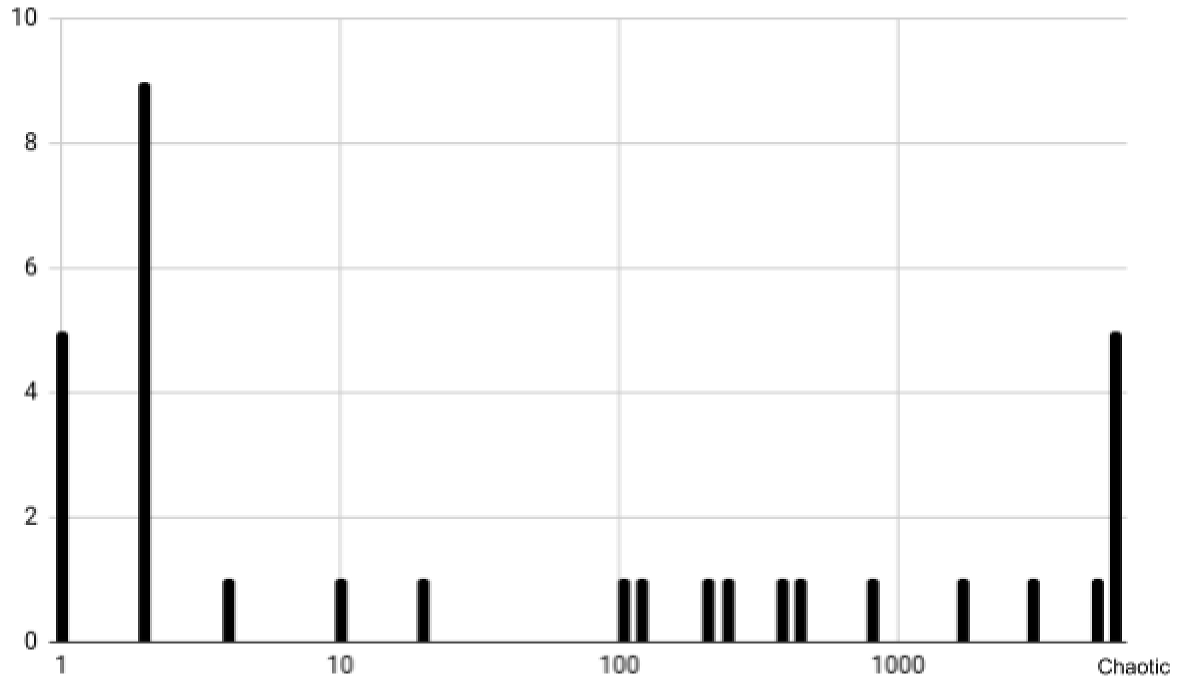

Figure 8. Histogram of Cycle Lengths for an n=400, k=2 Network with a Uniform Distribution of Delay (Range from 0.8 to 1.2).

IV. Conclusions

Kauffman, through his own experiments, fully demonstrated that, given a certain set of assumptions, his model produces the results that he wanted. In particular, it gives order for free, indicating the inevitability of the emergence of life. What has now been demonstrated is the necessity of the assumptions that Kauffman made in order for his model to work.

The assumption that my research was primarily focused on challenging is the assumption that all genes have identical delay. There are very few things in life that possess a high enough degree of precision for such an assumption to be valid. The importance of this is powerfully demonstrated by the case where the node delay was distributed according to a normal distribution with a standard deviation that was 0.1% of the mean. Even the tiny amount of variation allowed by those constraints produced huge differences in the resulting simulations. Kauffman's order for free requires perfect uniformity. Whether the variation in delay is slight or major, any variation will take away the order for free and replace it with order by lots. By this I mean that well behaved networks are still possible, but they are no longer the expected outcome.

Another assumption of Kauffman's is that, in genetic networks, no gene is affected by more than two other genes and very few genes are affected by less than two other genes. This seems like an assumption that would be unlikely to be valid. If one were to imagine a soup of genes interacting with each other, it seems natural to assume that each gene would be influenced by all genes that it is adjacent to and that a given gene would be adjacent to more than two other genes. If this is the case, then Kauffman's own results demonstrate that networks with more edges produce far more chaotic behavior. For this reason, it is critical for Kauffman's results that each gene is influenced by two other genes and not more than two.

A third assumption is that short, stable cycles necessitate life. Here it is worth noting that Kauffman observed several other properties of these cycles which I lack the time to address in this paper. Such properties include things like basins of influence and the ability of cycles to recover from the corruption of data. Regardless of what properties these cycles have, the question of what constitutes life remains more of a philosophical question than a scientific one. I, personally, fail to see how a stable cycle in a finite automata which has interesting mathematical properties can be considered alive or capable of producing life.

A final assumption is that this type of model is even a relevant way to model this scenario. To my knowledge, there is no sufficiently convincing argument for why this type of model would be valid, but neither is there a sufficiently convincing argument for why this type of model would be invalid. This question is worth contemplating, but is well beyond the scope of this paper.

Kauffman, in his research, discovered many interesting mathematical properties of boolean networks. I believe that I have found a few more interesting properties. How these networks relate to the origin of life, however, remains an open question. If they are, in fact, a relevant and accurate model, then the possibility of an inevitable emergence of life seems highly unlikely. This model, if valid, seems to allow for two possibilities. Either events proceeded by chance and we are incredibly lucky that life arose or the network was intentionally designed by an outside force in such a way that it would produce life.