RANDOM NEURAL NETS WITH SMALL CONNECTIVITIES

Nicole M. Mayer[1]

August, 1993

Abstract

This paper uses an experimental approach to analyze the expected cycle lengths of completely random nets with small connectivities. Using a "0 - 1" threshold law, the results gave reason to expect that the transition point between nets with small cycles and nets with large cycles is between the connectivities of 2 and 3 where the probability of finding a long cycle length in a large size net will be 1 for all connectivities greater than the transition point and 0 for all connectivities less than the transition point.

I. Introduction

The topic of neural nets was one of great popularity twenty years ago, and has once again become a hot topic in the scientific community. Much of the present interest is in neural nets as learning devices, but there are also many questions about the dynamics of neural nets which have remained unanswered. This paper will address one of these unanswered questions.

Kauffman(1969,1993) speculated on the basis of some simulations that there is a significant difference between the behavior of neural nets with two inputs per neuron and nets with more than two inputs per neuron. In particular, he suggested that the expected cycle length of state cycles should be about $\sqrt{n}$ for a net with n neurons and connectivity 2, and that the expected cycle length of state cycles should be about $2^{n/2}$ for nets with n neurons and high connectivity. While the expected state cycle length could be calculated for nets with connectivity n as shown by Rubin and Sitgreaves(1954) and confirmed by Cull(1978) and Gelfand(1982), calculations of the expected cycle length for other connectivities has not yet been accomplished.

Based on some results of Fagin(1976), we speculated that the expected cycle length might follow a "0 - 1" threshold law, so that the probability of finding a long cycle might be small (near 0) for low connectivity, and then at some value of connectivity this probability might jump to become nearly 1. We set out to experimentally determine if these speculations were reasonable, and this paper describes what we found and how closely the results matched our speculations. The paper is divided into three main parts: part II consists of the basic definitions needed in understanding the construction of neural nets, part III contains an explanation of how the experimental data was produced, and part IV compares the results of the experiment with the initial conjectures made at the beginning of the experiment. In addition, an appendix at the end of the paper includes one version of the Pascal program used to produce the experimental data.

[1]Supported in part by NSF-REU Grant and Oregon State University. The author is grateful to Dr.Paul Cull for his helpul guidance and encouragement.

II. Terminology and Notations

A *neural net* consists of a finite set of n neurons which are connected so that some neurons give inputs to other neurons and some neurons receive output from other neurons. We are going to be dealing with *autonomous systems* in which there is no external input coming into the net from outside of the net. The *state* of a neuron when there is binary output will be either "on" or "off," denoted by either 1 or 0. In an autonomous net, the state of a neuron at time $(t+1)$ is a function f of the states of the input neurons at time t. Every next state function representing the state of a neuron at time $(t+1)$ can be written in polynomial form. For example, given a net of two neurons with states X_1 and X_2 respectively, the function representing the state of X_1 at time $(t+1)$ can be written as the polynomial

$$X_1(t+1) = f((X_1, X_2)_t) = c_0 + c_1X_1 + c_2X_2 + c_3X_1X_2 \ (mod2) \ \textit{where} \ c_i \in \{0,1\}. \quad (1)$$

Writen this way, the structure of the net is *completely connected* where all of the neurons in the net are inputs into each neuron's next state function. Note that there are 2^{2^n} distinct polynomial functions f in n variables for a binary net. In vector form, we can write the general form for the state S of the completely connected two neuron net at time $(t+1)$ as

$$S_{t+1} = F(S_t) = \begin{pmatrix} f_1((X_1,X_2)_t) \\ f_2((X_1,X_2)_t) \end{pmatrix} = \begin{pmatrix} a_0 + a_1X_1 + a_2X_2 + a_3X_1X_2 \\ b_0 + b_1X_1 + b_2X_2 + b_3X_1X_2 \end{pmatrix} \ (mod2) \quad (2)$$

$$\textit{where} \ a_i, \ b_i \in \{0,1\}.$$

In (2), S_{t+1} is a *state vector* whose components are the states of each neuron in the net at time $(t+1)$. In addition, since there are two possible states for each of the neurons, there will be a total of 2^n state vectors in a net of size n. In the case of a two neuron net, the state vectors $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$ will be $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

The nicest nets to work with are *random completely connected nets* in which each neuron computes a randomly chosen function of the states of all the neurons in the net. Such a net is equivalent to a random mapping from the net's state set to itself. The reason these nets are nice to work with is because the statistics for random completely connected nets can be computed from the statistics of random mappings from the state set to itself. To explain, we can generate a random mapping from the state set to itself by assigning each neuron in the net a next state function that is chosen at random from the 2^{2^n} possible in a net of size n. By generating a finite set of random mappings, we can calculate the statistics for that set of mappings which will in turn give us the expected statistics for the entire set of random completely connected nets. Thus, by looking at several randomly generated nets in a particular class of neural nets, we can find the *expected behavior* of a certain characteristic in that class of nets. Rubin and Sitgreaves(1954) and Gelfand(1982) have computed various statistics for random completely connected nets that characterize such expected behaviors.

Let's look at a simple example of a random completely connected net: let $n = 2$, so neurons X_1 and X_2 comprise the net as before. To define a next state function for the net, randomly choose two functions out of the 16 possible in 2 variables such that

$$S_{t+1} = F\left(\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}_t \right) = \begin{pmatrix} X_1 + X_2 + 1 \\ X_1 X_2 \end{pmatrix}. \tag{3}$$

Now pick an arbitrary state vector and follow its trajectory using the next state function F. Let $S_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Then we have $\begin{pmatrix} 0 \\ 1 \end{pmatrix}_0 \longrightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}_1 \longleftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}_2$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}_3$. As you can see, $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_1$, and S_3 is mapped to itself.

One characteristic to look at in a random completely connected net is the *average cycle length* in the net, which will be defined slightly different than the traditional notion of an average. In the previous example, if you start at random in any one of the three states S_0, S_1, or S_2, you will find a cycle of length 2, and you will find a cycle of length 1 only if you start in state S_3. So, the "average" cycle length will be $2 \cdot \frac{3}{4} + 1 \cdot \frac{1}{4} = \frac{7}{4} = 1.75$. Therefore, the average cycle length is dependent on the number of cyclic states in each cycle, as well as the number of states we choose to start from. Note that the average cycle length is not 1.5, which is calculated by the more traditional method for finding averages.

The average cycle length in a net is directly dependent on the *connectivity* (K) of the net, the number of neurons on which the next state function depends. From (3), you can see that for our net of two neurons, the connectivity K is equal to the size of the net, 2, since the next state functions for X_1 and X_2 each depend on two inputs. One result which shows the dependence of the expected cycle length of a random completely connected net on its connectivity is that as $K \rightarrow n$ (i.e. the next state function for each neuron depends on inputs from all of the other neurons), the *expected cycle length*$\rightarrow 2^{n/2}$ or more precisely, $\frac{2^{n/2}}{\sqrt{2\pi}}$, and in large size nets the ECL is of order $2^{n/2}$ (Cull 1978) and (Gelfand 1982).

So given a net of size 30 with 2^{30} state vectors, as $K \rightarrow 30$, the expected cycle length approaches $\frac{2^{15}}{\sqrt{2\pi}} \approx 10^4$. A computer simulation from Cull(1978) verifying the asymptotic expected cycle length behavior for a net of size 31 when K approaches n is shown in figure(1), which shows the cycle lengths beginning to level off near $K = 15$.

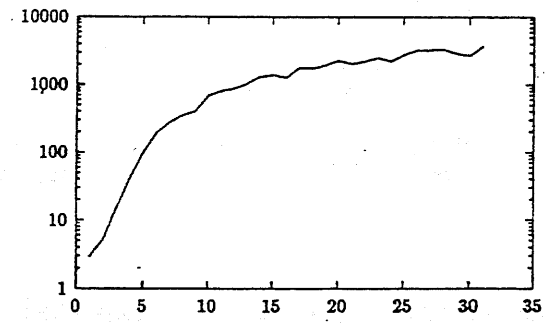

Figure 1 Expected cycle length as a function of connectivity for nets of 31 neurons.

Although this result says a lot when working with large connectivities, it does not give any information about the expected cycle length when dealing with small connectivities. It has, however, been suggested that for two-input binary nets of size n with connectivity 2, the median cycle length will follow a "square root law" where the median cycle length $\approx \sqrt{n}$ (Kauffman 1969). Nevertheless, when small connectivities (in comparison to the net size) are looked at, there appears to be a sharp jump in the expected cycle length for the net. In particular, Kauffman suggests that this jump occurs between the connectivities of 2 and 3. Walker(1984) found the results in figure(2) which show the median cycle length as a function of connectivity for nets of size 20. Note that his data considers nets with more than binary outputs. One can see from his data that the expected cycle length when K lies in [2,3] may in fact signify the transitional area between a net with small cycle lengths and a net with large cycle lengths of order $2^{n/2}$.

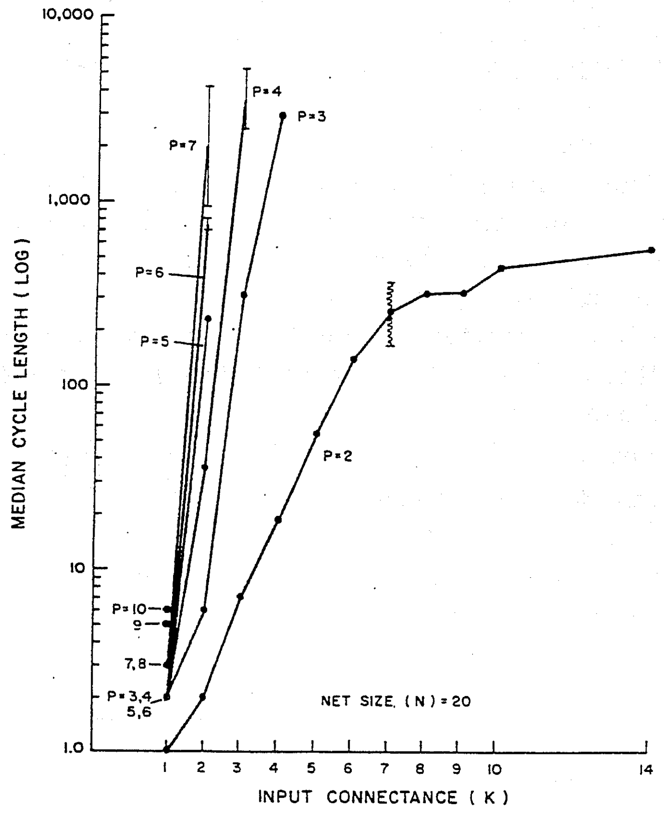

Figure 2 Median cycle length for nets of size 20 as a function of connectivity for constant output varieties (P).

Conjecture: As the size of the net increases, the expected cycle length (ECL) in the net grows at a much faster rate when $K = 3$ than when $K = 2$ such that when $K = 3$, $ECL \approx c2^{n/2}$, and when $K = 2$, $ECL \approx \hat{c}\sqrt{n}$. In addition, there exists some value $\hat{K}$ between 2 and 3 which will determine the transition point between a net with small cycles and a net with large cycles. So the questions to be answered are: *1) How much faster does* *the ECL for a net grow when $K = 3$ than when $K = 2$?* and *2) As the size of the net increases, does there exist a value $\hat{K}$ between 2 and 3 at which the transition point between large and small cycle lengths occurs?.*

III. Experimental Procedure for Finding the Average Cycle Length of a Net

In order to examine the cycle lengths of random nets between the connectivities of 2 and 3, we have to look at the average cycle length of different size nets when K lies in the interval [2,3]. We can find the expected cycle length in a net of size n with connectivity K by calculating one cycle length for each of several randomly generated nets of size n and then taking the average of those cycle lengths. This process can most efficently be carried out using a computer program to simulate random nets, and this is exactly what was created to produce the data. Given a net of size n and a connectivity K, there are a few main tasks this program had to accomplish:

I. Randomly generate a set of n next state functions f_i which correspond to the n neurons in the net.

II. Once there is a random next state function $F = \begin{pmatrix} f_1 \\ : \\ f_n \end{pmatrix}$ such that $F(S_t) = S_{t+1}$, it has to find an arbitrary state vector that is known to be cyclic, and follow its trajectory while at the same time calculating the cycle length for this trajectory.

III. Run steps I and II again to produce another random next state function F and thus another random net to find another cycle length for the same net size n and connectivity K.

IV. Take the average of these cycle lengths to find the average cycle length for a net of size n with connectivity K.

While steps III and IV are self explanatory, steps I and II deserve more explanation within the context of the computer program. To address the problem of randomly assigning next state functions to neurons in the net, it is first necessary to discuss the structure of a randomly connected net in more detail. The net will consist of n neurons, each of which will be randomly assigned a next state function f_i, $i: 1 \to n$. Because we are dealing with "fractional" connectivities that range between 2 and 3, each neuron in the net will have probability $(K - 2)$ that its next state function will need three input values from other neurons in the net. For instance, if $K = 2.37$, each neuron in the net will have probability .37 that its next state function will need three inputs and probability .63 that its next state function will need two inputs. A random number R_i will be generated by the computer to decide whether a next state function f_i will have three input values or two. Thus, a next state function for neuron X_i will be

$$f_i = \begin{cases} c_0 + c_1X_p + c_2X_q + c_3X_pX_q & \text{if } R_i >(K-2) \\ c_0 + c_1X_p + c_2X_q + c_3X_pX_q + c_4X_r + c_5X_pX_r + c_6X_qX_r + c_7X_pX_qX_r & \text{if } R_i \leq (K-2) \end{cases} \tag{4}$$

$$\textit{where } c_j \in \{0,1\}, \quad \textit{and } p,q,r \in \{1,..,n\}.$$

Note that if R_i is greater than $(K-2)$, then f_i will be assigned a connectivity of 2, otherwise it will have connectivity 3. These next state functions f_i will be fixed once random coefficients c_j and random input neurons p,q,r are equiprobably picked for each of the f_i by the computer in a similar manner as before. Thus, the entire net structure will be fixed once next state functions are assigned to all of the neurons by this process. The function $F = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}$ will then describe the transition from state vector to state vector in the net where $F(S_t) = S_{t+1}$.

For step II, it is first necessary to quickly find an arbitrary cyclic state vector in the net. This can be achieved by letting the computer pick an arbitrary state X, and iterating the next state function F such that at the first iteration, $A = F(X)$ and $B = F(F(X))$, and following iterations, $A_{k+1} = F(A_k)$ and $B_{k+1} = F(F(B_k))$. A cyclic state S_0 will be found when A_{k+1} is equal to B_{k+1}. Given S_0, the computer can easily count how many iterations of F it takes to return to S_0, and this number of iterations determines the cycle length for this particular initial cyclic state in this particular net. Creating several random net structures for the same net size n will produce several cycle lengths from which to obtain a better average, as outlined in steps III and IV.

In summary, by following the steps previously outlined, the program created will find the average cycle length for 200 randomly generated nets for a net of size n with connectivity K. For even better averages, 10 cycle lengths were found and averaged from each of the 200 randomly generated nets. Thus, a total of 2000 cycle lengths were averaged for each n and K combination. In addition, the variance from the average cyclelength was calculated at each experimental point. Finding out how fast the average cycle length grows between $K = 2$ and $K = 3$, as well as finding a value $\hat{K}$ for which the transition between large and small cycles will occur, is now just a matter of using the program to test nets of different sizes and varying connectivities within the interval [2,3]. The complete program can be found in the appendix.

IV. Results

In order to verify that there does in fact exist a jump in the average cycle length in a net between the connectivities of 2 and 3 as Kauffman suggests, the first set of experimental data produced the average cycle lengths for nets of size 2 through 70 both with $K = 2$ and $K = 3$. Due to increasingly long run times, while all net sizes between 2 and 40 were run, net sizes larger than 40 were only tested at n = 45, 50,...,65, and 70. The results of this first run of tests are shown in figure(3), in which the data was plotted on both normal scales and a log scale to clarify the difference between the average cycle lengths for the varying net sizes at $K = 2$ and $K = 3$.

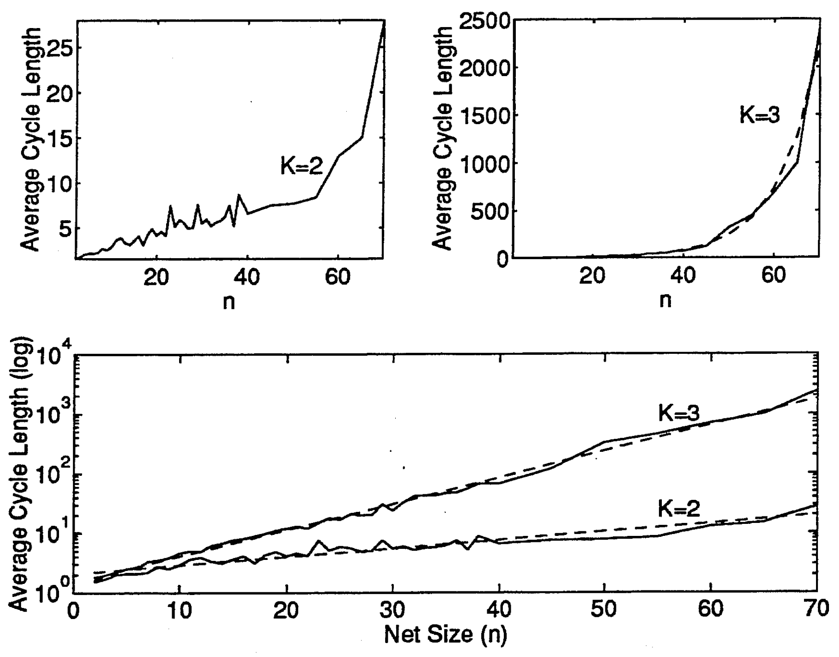

Figure 3 Average cycle lengths for varying net sizes at the connectivites of 2 and 3.

The graphs in figure(3) illustrate the rates of growth when $K = 2$ and when $K = 3$ of the average cycle length in a net in relation to the net size n. Close approximations were made for these graphs whenever possible and represented by the superimposed dashed lines. The inconsistencies in the smoothness of the graphs are a result of sampling error, as large variances from the average cycle lengths in the nets were quite common. Other reasons for a lack of smoothness are small net sizes and the possible effect of the primeness of n (or the number of factors of n) in determining the connections and thus the cycle lengths of the net. Discounting these inconsistencies, it is possible to analyze the general nature of these curves. It is important to note that these approximations were made solely on the experimental data available, and that the average cycle length of a net in relation to the net size may behave slightly differently if nets larger than size 70 are analyzed.

It turned out that the data does not directly support the initially conjectured approximations for the ECL when $K = 2$ and when $K = 3$. When $K = 3$, we know that the ECL will approach $c2^{n/2}$ as n grows large where $0 < c < 1$. Our data shows that the average cycle lengths were approaching $2^{n/6}$ for net sizes up to 70, which is most likely because the net sizes tested were not large enough to confirm the expected $2^{n/2}$ rate of growth. On the other hand, the initial conjecture for the ECL when $K = 2$ was that it approached $\hat{c}\sqrt{n}$, which came from Kauffman's suggestion that this will occur when n gets large enough. However, the data showed that the average cycle length when $K = 2$ for nets up to size 70 is increasing faster than $\sqrt{n}$, but still at a much slower rate than when $K = 3$. Walker(1984), who was also unable to confirm Kauffman's conjecture, suggests that the net sizes tested should at least be carried out to size 1000. In our case, since the data for $K = 2$ tended to be more random than when $K = 3$, it is not clear whether an approximating function for the average cycle length when $K = 2$ is of an exponential or quadratic form. In either case, as $n \rightarrow 70$, the average cycle length of the net when $K = 2$ is so much smaller than when $K = 3$, it is not a great loss that we cannot accurately approximate the behavior of the ECL when $K = 2$. Thus, although the data confirmed the large jump in the average cycle length between the connectivities of 2 and 3, our approximations for the average cycle length of a net when $K = 2$ and $K = 3$ differ from the predicted ECL's at these two connectivities. The approximations for the average cycle lengths of the various net sizes that the data did help to produce are good for the net sizes we are dealing with, yet it is reasonable to assume that as n increases, the ECL will behave like our original predictions.

Now that it is at least confirmed that there does exist a big gap in the average cycle lengths of a net when $K = 2$ and when $K = 3$, we can start looking at the average cycle lengths for varying net sizes at connectivites between 2 and 3. The next set of data tested varying net sizes with connectivities between 2 and 3 in increments of $\frac{1}{10}$. The results are shown in figure(4) and as before, inconsistencies in the smoothness of the graph are a result of sampling error and other factors.

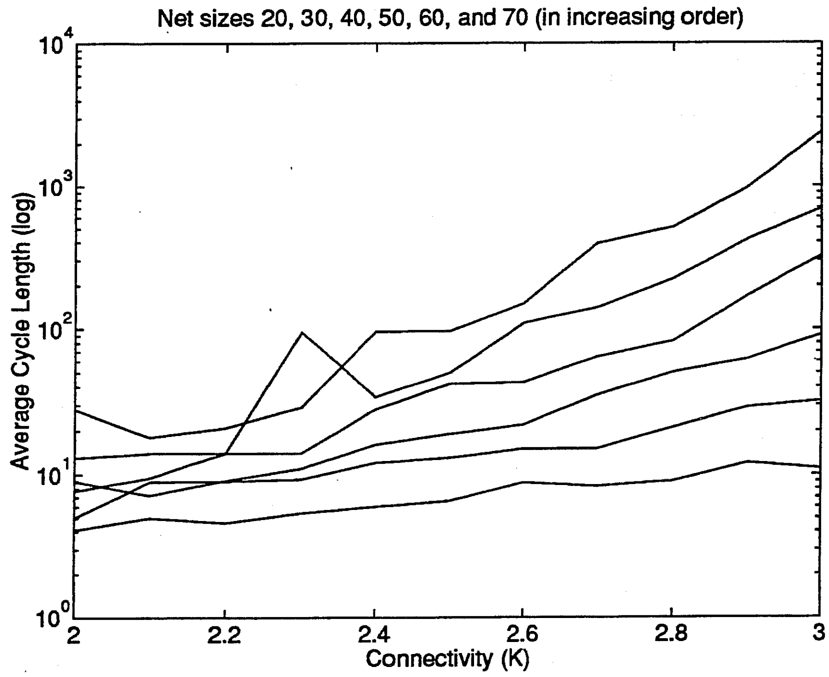

Figure 4 Average cycle lengths for varying net sizes between the connectivities 2 and 3.

As one can see, the curves for all net sizes start to smooth into straight lines when K is roughly around the value of 2.7. This indicates that a "critical" value $\hat{K}$ for which the average cycle length of a net starts to grow exponentially large may occur around this connectivity value. However, it first needs to be shown that there actually does exist a $\hat{K}$ between 2 and 3 that satisfies this property. Intuitively, we need to show that given a large value L, as the size of the nets increase, the probability that a cycle length for the net at $\hat{K}$ will be greater than or equal to L is 1. In other words, we need to show that a "0 - 1" law holds in large nets with a range of connectivities between 2 and 3. If a critical connectivity $\hat{K}$ exists, we want the probability that a cycle length is "large" for all connectivites $\hat{K}$ and larger to be 1 and 0 for all connectivities less than $\hat{K}$. For our purposes, we can use the following theorem proved by Fagin in 1978:

Theorem. "0 - 1" Law for Finite Models (i.e. Random graphs)

Let $\mathcal{P}$ be a finite set of (nonlogical) predicate symbols, or properties P. By a $\mathcal{P}$-structure, we mean a relational structure appropriate for $\mathcal{P}$ (i.e. $(Px.)$, where $(Px.)$ is an atomic sentence meaning the individual constant x satisfies property P). Let $A_j(\mathcal{P})$ be the set of all $\mathcal{P}$-structures with universe $\{1, ..., j\}$, or in other words, with j individual constants to be used in the $\mathcal{P}$- structures $(Px.)$. For each first-order $\mathcal{P}$-sentence σ (with equality), let $u_j(\sigma)$ be the fraction of members of $A_j(\mathcal{P})$ for which σ is true, then $u_j(\sigma)$ converges to 0 or 1 as $j \rightarrow \infty$.

In order to make this theorem work for our problem, lets first define $\mathcal{P}$. There will only be one property we need as an element of $\mathcal{P}$, and that property will define what a "long" cycle length is. Based on the information that the expected cycle length will approach $2^{n/2}$, we will want to consider the value "long" to be some function of 2^n (i.e. $2^{n/2}$, $2^{n/6}$, etc.).

Definition 1. Given a net of size n and its next state function F, let

$$\begin{aligned} LONG_{FUN(2^n)} = {} & [\exists S_1 \cdots \exists S_{FUN(2^n)}] \wedge [(S_2 \neq S_1) \wedge (S_2 = F(S_1))] \wedge [(S_3 \neq S_2) \wedge (S_3 \neq S_1) \\ & \wedge (S_3 = F(S_2))] \wedge \cdots \wedge [(S_{FUN(2^n)} \neq S_{FUN(2^n)-1}) \wedge \cdots \wedge (S_{FUN(2^n)} \neq S_1) \wedge \\ & (S_{FUN(2^n)} = F(S_{FUN(2^n)-1}))] \end{aligned}$$

where the S are state vectors with n components, and $LONG$ is the property that a cycle has a length of $FUN(2^n)$.

Definition 2. Given a net of size n, and $LONG_{FUN(2^n)}$, let

$$LONGER = LONG_{FUN(2^n)} \vee LONG_{FUN(2^n)+1} \vee LONG_{FUN(2^n)+2} \vee \cdots \vee LONG_{2^n}$$

where $LONGER$ is the property that a cycle has a length between $FUN(2^n)$ and 2^n, the maximum cycle length in a net of size n.

Thus, $LONG_{FUN(2^n)}$ is a sentence which defines a "long" cycle length as being one of length $FUN(2^n)$. It is important to note that there does exist a problem with this sentence $LONG$, as it is not necessarily written in first order logic as the theorem requires. The problem lies with the sentence's use of the next state function F, which is not written in terms of first order logic. However, this problem can most likely be avoided with some clever rewriting, and therefore we will disregard it for the remainder of the argument.

As a result of the "0 - 1" Law, we know that the probability that a found cycle length in a net satisfies the property $LONGER$ at a fixed k will be either 1 or 0 as $n \to \infty$. Unfortunately, this theorem does not guarantee there will be one breakpoint connectivity over the entire range of connectivities in [2,3]. In other words, it does not guarantee that the probabilities will be monotonic in certain intervals in [2,3] such that if there does exist a breakpoint connectivity $\hat{K}$, the probability that a cycle in a net of size n is of length $FUN(2^n)$ or longer will be 0 in [2,$\hat{K}$] and 1 in [$\hat{K}$,3]. On the other hand, the experimental data shows that the average cycle length in relation to the connectivity is an increasing function for all net sizes tested (see figure(4)), which gives us the belief that the probability will in fact be monotonically spread over the intervals [2,$\hat{K}$] and [$\hat{K}$,3] for some critical connectivity $\hat{K}$. Therefore, we should expect that testing over a range of connectivities between 2 and 3 as n gets large should zero in on the breakpoint connectivity for which the probability a cycle in a net of size n is of length $FUN(2^n)$ or larger is 1 for all k greater than the breakpoint, and 0 for all k less than the break point.

It is important to note that this theorem holds true for any function we wish to determine a "long" cycle length. Ultimately, we would like this function to be of order $2^{n/2}$ since this is how the cycle length grows for large size nets when the connectivity is large. As we have seen from the experimental data for nets up to size 70, the net sizes will have to be much larger before the cycle lengths will be of length $2^{n/2}$. Once the net size is increased, we should expect to see a greater break in the growth rate of the average cycle length over the interval [2,3].

For the set of data we have, it could be useful to use $2^{n/6}$ as the "long" cycle length value, and assign probabilities 1 and 0 to all of the 2000 cycle lengths found at each (n, k) combination which can be done by modifying the computer program used to produce the original data. In other words, if a found cycle length in a random net of size n at connectivity k has a cycle length greater than $2^{n/6}$, then it will be assigned a probability of 1, and if it is less than $2^{n/6}$, it will be assigned a probability of 0. Then a plot could be made for varying net sizes between 2 and 70 of the probability that a cycle length will be of length $2^{n/6}$ for each $k \in [2, 3]$. From this graph, it should be more evident of where the critical value $\hat{K}$ lies, but the most accurate approximation of $\hat{K}$ will be found only when cycles in net sizes larger than 70 are looked at and the "long" cycle length considered is of order $2^{n/2}$. Unfortunately, this requires lengthy computer run times, but Fagin's theorem gives us reason to expect that once n is increased, a value $\hat{K}$ will be found for which the probability that a cycle length is long at a connectivity k is 1 for all $k > \hat{K}$ and 0 for all $k < \hat{K}$.

V. Conclusion

While the data produced by the experiment confirmed the comparitively large difference in average cycle lengths between the connectivities 2 and 3, the size of the nets able to be tested were neither large enough to confirm the expected cycle length of order $2^{n/2}$ near the connectivity of 3 nor confirm Kauffman's "square root law" for the average cycle length when the connectivity is 2. On the other hand, it was able to be shown via Fagin's "$0-1$" law that there is reason to believe there exists a critical connectivity in [2,3] which will signify the transition point between a net with small cycles and a net with large cycles, provided the net sizes analyzed are large enough. It is possible that by defining a value for a "long" cyclelength smaller than $2^{n/2}$, this critical connectivity can be approximated by modifying the computer program used for this experiment so that rather than finding the average cyclelength at each (n, k) combination, it could assign probabilites to each of the 2000 cycle lengths found at each experimental point. Therefore, although the existence of this crictical connectivity between 2 and 3 is strongly expected, the problem of finding it still remains open, and in order to find it, more experiments need to be done with nets of very large sizes.

References

Carnap, R. 1950. *Logical Foundations of Probability.* Chicago: University of Chicago Press.

Cull, P. 1978. "A Matrix Algebra for Neural Nets." *Applied General Systems Research.* New York: Plenum.

Fagin, R. 1976. "Probabilities on Finite Models." *The Journal of Symbolic Logic* **41**, 50-58.

Gelfand, A.E. 1982. "A Behavioral Summary for Completely Random Nets." *Bulletin of Mathematical Biology* **44**, 309-320.

Kauffman, S.A. 1969. "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets." *Journal of Theoretical Biology* **22**, 437-467.

Kauffman, S.A. 1993. *The Origions of Order.* New York: Oxford University Press.

Rubin, H. and Sitgreaves, R. 1954. "Probability Distributions Related to Random Transformations on a Finite Set." Technical Report 19A, Applied Math and Statistics Lab., Stanford University.

Walker, C.C. 1984. "Beyond the Binary Case in Random Nets." *Bulletin of Mathematical Biology* **46**, 845-847.

Appendix

The following is a version of the Pascal computer program used to calculate the average cycle length for a net of size n with connectivity K. For a constant value K, this program finds the average cycle length for nets of size 2 through 40. Several adjustments were made to this program in order to decrease runtime when testing large size nets (for example, function calls were replace by in-line code). This program was also modified to test more than one connectivity at a time for each net size.

```
program cycles(output);
     {Given a net size n and a connectivity K, this program randomly
      generates 200 nets finding 10 cycles in each net.  It then finds
      the average cycle length for the net size by taking the average of the
      2000 total cycle lengths it finds.  Specifically, given one connectivity
      value, average cycle lengths are found for net sizes 2 through 40.}


     const

            N = 1073741824;  {N is the upperbound for the number  of
                               state vectors to be used in the program}

            K = 3.0; {connectivity}

     type
            COEFF = array[0..7] of integer;
            CONSTANTVECTORS = array [1..40] of COEFF;
            ONETWOTHREE = array [1..3] of integer;
            INPUTVECTOR  = array [1..40] of ONETWOTHREE;
            NEURONVECTOR = array [1..40] of integer;
            RANDOMNUM = array [1..40] of real;


       var
            cyclelnth: array[1..200,1..10] of integer;
            c : CONSTANTVECTORS;
            inp : INPUTVECTOR;
            X, V, A : NEURONVECTOR;
            n, v, w, x, i, clength : integer;
            R : RANDOMNUM;
            variation, CL, sum : real;
```

```
{Functions and Procedures}
function F2(c : CONSTANTVECTORS; i, first, second : integer): integer;
     {F2 is a next state function in polynomial form assigned to a neuron
      which has probability 1-K-2 that it will have two inputs that
      determine its next state.}

     var  G : integer;

     begin

          G := c[i,0] + (c[i,1] * first) + (c[i,2] * second)
               + (c[i,3] * first * second);

          F2 := G mod 2

     end;


function F3(c: CONSTANTVECTORS; i, first, second, third : integer): integer;
     {F3 is a next state function in polynomial form assigned to a neuron
      which has probability K-2 that it will have three inputs that
      determine its next state.}

     var  G : integer;

     begin

          G := c[i,0] + (c[i,1] * first) + (c[i,2] * second)
               + (c[i,3] * first * second) + (c[i,4] * third) +
               (c[i,5] * first * third) + (c[i,6] * second * third)
               + (c[i,7] * first * second * third);

          F3 := G mod 2

     end;


function F(K:real; R:RANDOMNUM; c:CONSTANTVECTORS; i, first, second,
          third : integer): integer;
     {F is the function which determines whether the next state function
      of a neuron will have connectivity 2 or 3.}

     begin
```

```
        if R[i] <= K-2
           then F := F3(c, i, first, second, third)
           else F := F2(c, i, first, second)

    end;


procedure randconst(n:integer);
    {randconst generates n random coefficient vectors with 8 components
     of constants.  These are the coefficient vectors which will be
     associated with each of the n next state functions for the n neurons.
     Note:  F2 needs 4 constants and F3 needs 8 constants.}

    var   p, l, x : integer;
          r : real;


    begin
          x := 0;
          for l := 1 to n do
               for p := 0 to 7 do
                 begin
                   r := random(x);
                   if r <= 0.5 then
                     c[l,p] := 0
                     else c[l,p] := 1
                 end
    end;


procedure inputneurons(i, n :integer);
    {This procedure generates three random distinct input neurons for one of
     the n next state functions:  one "input" vector is created for one
     neuron's next state function in this procedure.}

    label stop;

    var   p, k, x : integer;
           r : real;

    begin
```

```
          x := 0;
          for p := 1 to 3 do
               begin
                   r := random(x);
                    if (0 <= r) and (r <= (1/n))
                         then inp[i,p] := 1
                         else

                               for k := 1 to (n-1) do
                                 if ((k/n) < r) and (r <= ((k+1)/n))
                                       then
                                          begin
                                             inp[i,p] := k+1;
                                             goto stop
                                          end;
                stop:
                end


     end;


procedure n_inputvectors(n:integer);
     {In this procedure, n_inputvectors generates three random distinct
      input neurons for each of the n next state functions.}

     var  i : integer;

     begin

          for i := 1 to n do
              inputneurons(i, n)

     end;


procedure initialstatevector(n:integer; var X: NEURONVECTOR);
     {This procedure randomly picks one state vector to be used to find
      a cyclic state in the net.}

     var
          m, x : integer;
          r : real;
     begin

          x := 0;
          for m := 1 to n do
               begin
                 r := random(x);
                 if r <=  0.5 then
                    X[m] := 0
                    else X[m] := 1
               end

     end;


procedure firstsecondthird(i: integer; var first, second, third : integer;
                           inp:INPUTVECTOR; var V:NEURONVECTOR);
     {This procedure finds the values of the first, second, and third input
      neurons to be used in evaluating a next state function.}


     begin

          first := V[inp[i,1]];
          second := V[inp[i,2]];
          third := V[inp[i,3]]

     end;



procedure findcyclicstate(K:real; n, N: integer; var first, second,
                           third : integer;
                          c:CONSTANTVECTORS; inp:INPUTVECTOR;
                          R:RANDOMNUM; var X: NEURONVECTOR);
     {Given the random initial state vector X, this procedure uses two
      functions: F(X)=A and F(F(X))=B.  When A=B, then A  will in fact
      be a cyclic state which will be used later to calculate a cycle
      length.}

     label found;

     var  i, f : integer;
```

```
    B, Astorage, B1 : NEURONVECTOR;


begin
    for i := (n+1) to 40 do
      begin
        A[i] := 0;
        B[i] := 0;
        Astorage[i] := 0;
        B1[i] := 0
      end;
    for i := 1 to n do      {A = F(X)}
        begin
           firstsecondthird(i, first, second, third, inp, X);
           A[i] := F(K, R, c, i, first, second, third)
        end;
    for i := 1 to n do      {B = F(F(X)) = F(A)}
        begin
           firstsecondthird(i, first, second, third, inp, A);
           B[i] := F(K, R, c, i, first, second, third)
        end;
    for f := 1 to N do     {Reiterate these functions s.t. A=F(A)
                            and B=F(F(B)) until A=B, indicating a
                            cyclic state.}
        begin
           for  i := 1 to n do   {A = F(A)}
              begin
                firstsecondthird(i, first, second, third, inp, A);
                Astorage[i] := F(K, R, c, i, first, second, third)
              end;
          A := Astorage;
          for i := 1 to n do     {B1 =F(B)}
              begin
                firstsecondthird(i, first, second, third, inp, B);
                B1[i] := F(K, R, c, i, first, second, third)
              end;
           for i := 1 to n do    {B=F(B1)=F(F(B))}
              begin
                firstsecondthird(i, first, second, third, inp, B1);
                B[i] := F(K, R, c, i, first, second, third)
              end;
           if A = B
              then goto found
```

```
            end;

          found:
      end;


procedure cyclelength(K:real; v, w, n, N : integer; c:CONSTANTVECTORS;
                     inp:INPUTVECTOR; R:RANDOMNUM; var X:NEURONVECTOR);
        {Procedure cyclelength finds a cyclic state S1 in the
        net and iterates the next state function F until a j is
        found such that at the j'th iteration of S2=F(S2), S2 = S1,
        resulting in a cyclelength j for this particular cycle in the net.}


        label finish;

        var  first, second, third, i,j : integer;
             S1, S2, S2storage : NEURONVECTOR;

        begin

            for i := (n + 1) to 40 do
             begin
                S1[i] := 0;
                S2[i] := 0;
                S2storage[i] := 0
              end;
            first :=  0;
            second := 0;
            third  := 0;
            findcyclicstate(K, n, N, first, second, third, c, inp, R, X);
            for i := 1 to n do
               S1[i] := A[i];
            for i := 1 to n do
               begin
                  firstsecondthird(i, first, second, third, inp, S1);
                  S2[i] := F(K, R,  c, i, first, second, third)
               end;
            if S1 = S2
               then cyclelnth[v,w] := 1
               else
                  for j := 2  to N do
                     begin
```

```
                      for i:= 1 to n do
                        begin
                           firstsecondthird(i, first, second, third,
                                            inp, S2);
                           S2storage[i] := F(K, R, c, i, first, second,
                                             third)
                         end;
                       S2 := S2storage;
                       if S2 = S1
                           then
                               begin
                                  cyclelnth[v,w] := j;
                                  goto finish
                               end
                    end;
          finish:

       end;


begin  {***MAIN PROGRAM***}

  for n := 2 to 40 do   {n is the net size}
    begin
       for v := 1 to 200 do   {generate 200 random nets}
         begin
           x := 0;
           for i:= 1  to n do
              R[i] := random(x);
           randconst(n);
           n_inputvectors(n);
           for w := 1 to 10 do   {find 10 cycles in each random net}
              begin
                initialstatevector(n,X);
                cyclelength(K, v, w, n, N, c, inp, R, X)
              end
         end;
      clength :=0;
      for v := 1 to 200 do
            for w := 1 to 10 do
                     clength := clength + cyclelnth[v,w];
      CL := clength/2000;
      writeln('(',n:2,',',K:1:2,',',CL:2,')');
```

```
     sum := 0;
     for v := 1 to 200 do   {calculate the variation}
         for w := 1 to 10 do
             sum := sum + ((cyclelnth[v,w] - CL) * (cyclelnth[v,w] -
                           CL));
     variation := (1/1999) * sum;
     writeln(variation:3)
   end

end.    {***MAIN PROGRAM***}
```