Final Year CSAI Project Thesis

- On Pseudo-Rhythmic Asynchronous Random Boolean Networks -

submitted by

Philipp Rohlfshagen
(Candidate number: 20001083)

for the

BSc in Computer Science and Artificial Intelligence Joint,
School of Cognitive Science, University of Sussex,

on

May 06, 2003

Project supervisor: Ezequiel A. Di Paolo

12000 words

Abstract

Recent results obtained from the study of asynchronous random Boolean networks (ARBNs) will be completed and compared with synchronous random Boolean networks (SRBNs) to highlight the intrinsic differences between the generic classes of networks. ARBNs do not naturally exhibit rhythm but instances of pseudo-rhythmic asynchronous networks are found using a simple genetic algorithm (GA) as suggested by Di Paolo (2001). These networks are analysed to determine how rhythm can emerge and be maintained given the indeterminism (randomness) of the updating scheme. It is suggested that ARBNs may be viewed to have strict cyclic attractors given a well defined topology which locally keeps track of time. A general theoretic framework is presented that seemingly fits the data of this and previous studies; mathematical proof has yet to be established. It is assumed the origin of rhythm in pseudo-rhythmic ARBNs is grounded in a carefully constructed physical architecture which only allows deterministic trajectories through the state space. The simplest such architecture has been identified as a chain of nodes with wrapping around at the edges; each node contains relational information about the state to its inputs which ensures the network is locally in order and globally synchronised. Further it will be suggested that the versatility of rhythm is proportional to the quantity of connections found in a network but that only simple architectures are likely to emerge. As the number of connections increases, more sophisticated topologies are possible but the network's complexity grows disproportional and seems to prevent this.

Table of Contents

Introduction

The aim of this project is find the mechanisms underlying the emergence of rhythm in pseudo-rhythmic asynchronous random Boolean networks (ARBNs) and to provide a tool for the analysis and visualisation of these networks. This has been attempted in a variety of approaches, including analytical methods such as single lesion analysis, and behavioural methods including analysis of output and pseudo-periodicity using software specifically developed for this purpose.

The project will first review and extend the work done on the generic properties of ARBNs and compare them to synchronous random Boolean networks (SRBNs). It will be shown that rhythmic behaviour is not a generic property of ARBNs but that networks exhibiting marked rhythm may easily be found using a simple genetic algorithm. These networks are subsequently analysed to determine the cause of their rhythmic behaviour given the non-determinism of the updating scheme. The results of all experiments concurred nicely and lead to the conclusion that rhythm is maintained by the coupling between nodes. A pruning algorithm is suggested that can be used to identify the functional architecture underlying the generation of rhythm. Care has been taken to find as much support as possible, but a mathematical proof remains to be found. The empirical data looks promising, however, and further research along the lines should reveal whether or not this framework scales up.

Dynamical systems

Dynamical systems theory is the most natural way to describe the behaviour of an integrated system (Kauffman, 1993). A dynamical system consists of an initial state and a dynamic law which describes the rate of change of the system's variables and as a result the behaviour of the system as a whole. A dynamical system may be continuous or discrete and it is believed that latter suffices in modelling most biological situations (Nowak and May, 1995). The dynamical law of discrete dynamical systems expresses a constant relationship between the state at time t and $t + 1$. All possible states describe the state space of the system and a succession of states over time defines a trajectory through this space; each initial state will dictate the unique course of the trajectory. These trajectories, however, may converge towards a single state which does not change in time, a steady state. Such a point is called an attractor which can either be a point attractor consisting of only a singe state or a cycle attractor consisting of a (normally small) sub-set of states. A group of states all of whose trajectories lead to such an attractor define the basin of attraction for that particular attractor. Each state situated in this basin will eventually reach the same attractor as every other state lying in the same basin. It can be shown that not all dynamical systems have attractors but if they do the state space can sensibly be divided into their disjoint basins of attraction. Boolean networks are an example of discrete dynamical systems as they have discrete time, space and values (Gershenson, 2002).

Boolean networks

Boolean networks are generally classified by the updating scheme used to move the network through the state space; these schemes will be discussed throughout the report. Within such a class, networks are distinguished by the number of nodes n and the number of incoming connections k to each node. The number of inputs to each node is assumed the same for each node albeit a more general case allows k to be an average number of inputs. In the case of random Boolean networks, the connections and Boolean functions are assigned at random without any prior knowledge about possible architectural configurations. If the Boolean function is identical for each node, the network is said to be homogenous. Once connections and Boolean functions are established they normally remain static throughout the networks execution. A Boolean network moves through the state space by changing the values of individual nodes according to their Boolean functions.

The mathematical properties of Boolean networks is summarised in Harvey & Bossomaier (1997) and recited here for completeness:

The truth table is of size 2^k and there are 2^{2^k} such tables in total. The total number of possible arrangements for a network is:

$$\left(\frac{n!}{(n-k)!}\right)^n, \text{ giving a total of } \left(\frac{2^{2^k} n!}{(n-k)!}\right)^n$$

random Boolean networks, but many of them being identical given labelling constraints.

Kauffman (1993) proposes the use of random Boolean networks as simplified models of complex dynamical systems containing rich numbers of coupled variables such as genetic regulatory networks. This abstraction requires many simplifications, two of which have been the Boolean idealisation and the use of synchrony. The Boolean idealisation replaces a continuous range of values with a step function which can only be in a state of 1 or 0, on or off. This abstraction has generally been accepted as it can be shown that coupled systems governed by sigmoidal functions can be sufficiently approximated by the step function. Variables representing physical quantities typically have a floor value of negative saturation and a maximum ceiling value of positive saturation (Harvey & Bossomaier, 1997). Criticism of the synchrony idealisation was raised by Harvey and Bossomaier (1997) and further analysed by Di Paolo (2001) and Gershenson (2002). This will be discussed in great detail later on as it forms the core of this project. Kauffman (1993) lists 7 reasons for the plausibility of Boolean networks and their idealisation, does not, however, discuss the issue of synchrony.

Some terminology will be introduced to highlight different aspects of Boolean networks and to clarify the discussion. The network will be divided into three layers as follows:

- Topological (physical) layer
  - Nodes and links (connections)
- Boolean layer
  - Boolean functions of each node
- Behavioural layer
  - Output patterns given the update scheme.

The network undergoes change over time by producing different states: A network's output is determined by its nodes, either *stationary* (constant value) or *active* (changes values); at each *state transition*, a subset of active nodes, the *changing nodes*, change their values and contribute towards the new state encountered.

Attractors

Several different kinds of attractors have been identified for the general class of Boolean networks. If such an attractor should be reached throughout the history of a network, the run-in is referred to as *transient*. Three attractors relevant for this report are as follows:

- *Point attractor:* The system settles in a single state. This is found in all classes of Boolean networks but with different characteristics (e.g. basin of attraction and number of attractors).

- *Cyclic attractor:* The system loops through a (usually) small subset of states. These cyclic attractors are found in deterministic systems.

- *Loose attractor*: Defined by Harvey and Bossomaier (1997) to describe the state of a system that traverses a subset of states in no specific order. Similar to cyclic attractors but non-deterministic; corresponds to strongly directed components within the network.

Synchrony versus asynchrony

The synchrony idealisation has been criticised by Harvey and Bossomaier (1997) as being biologically implausible; it is indeed unlikely for biological systems to operate in such a coordinated lock-step fashion. Kauffman (1969) expresses doubt that rhythm could be the property of carefully evolved networks with highly ordered circuits. Yet, Kauffman does not seem to question the existence of a perfect timing device dictating the overall behaviour of the network. If one assumes each node in the network has got an internal timing device which dictates the point of update then all the nodes in the network would have to adapt to each other in order to have the exact same time constant (i.e. synchrony). This not only seems implausible but also impossible in some cases where individual connected components are not inter-connected.

Synchrony has been used for its simplicity and determinism: Determinism implies that two distinct states of the network may converge onto the same successor but a single state may never diverge onto two distinct successors. Gershenson (2002) points out that the characteristics of networks seem to be dictated by the degree of determinism rather than the updating scheme itself. There are several asynchronous updating schemes, some of which are deterministic. This report will solely focus on the non-deterministic scheme which does not assume any prior knowledge of individual time delays. It updates all nodes in a random fashion, allowing for multiple updates of the same node in a single time step. It has been shown (Harvey and Bossomaier, 1997) that this updating scheme results in very different behaviour of the network, most notably a large number of point attractors (see next section).

Kauffman tested several cyclic networks for robustness when disturbed by a single bit flip. This was done to verify how the network would move among different attractors once disturbed. There seems no evidence, however, that Kauffman tested for a low but persistent level of noise in the updating scheme itself, which cast further doubt upon its validity. Random asynchrony on the other hand is inherently noisy and there is no reliable source of order in the sequence of updates. Viewing asynchrony as a noisy updating scheme enables one to use synchrony as an idealised tool of analysis for ARBNs.

It should be clarified that the only difference between distinct classes of Boolean networks is the update scheme. The topological and Boolean layers are equally probable throughout. This stresses the power of the update scheme but also highlights the possibility of comparisons between different update schemes as they function on the same construct.

Review and comparison of Boolean networks

Synchronous Boolean networks have had a long history of statistical analysis and have been used in various models of biological systems, most notably genetic regulatory networks. Only recent criticism about the synchrony idealisation has caused the study of alternative, non-deterministic update schemes. A classification of different networks and update schemes is given by Gershenson (2002), see appendix.

The results that have been obtained from various studies will be compared and completed to gain a better understanding about the intrinsic properties individual classes of networks have. Kauffman tested networks using all possible Boolean functions and networks which omitted both tautology and contradiction. Since there has been no mention about the use of tautology or contradiction for the asynchronous case, it is assumed they have been included.

Synchrony:

Cyclic attractors: The number and length of cyclic attractors for different value of k are as follows:

k	state cycle length	number of cyclic attractors
$k = n$	$0.5 \times 2^{n/2}$	n / e
$k > 5$	0.5×2^{Bn}	$\approx n \left[\frac{\log\left(\frac{1}{0.5 \pm \alpha}\right)}{2} \right]$, $\alpha = p(k) - 0.5$
$k = 2$	$\sqrt{n}$	$\sqrt{n}$
$k = 1$	$\sqrt{(\pi n/2)}$	exponential in n

Table 1: The properties of cyclic attractors in SRBNs given k (Kaufman, 1993; pp 193).

Activity: The number of changing elements at the first state transition is about $0.4n$. This number decreases by a negative exponential with a half-decay of 3-4 states to a minimum activity of 0 - $0.25n$ within a cycle. The number of active nodes is said to be up to 35% in a net with n=100 (Kauffman, 1969).

Transients: The length of the transient seems uncorrelated to the length of the cyclic attractor encountered. The distribution of transients is highly skewed towards short lengths (Kauffman, 1969).

Asynchrony:

Point attractors: The expected number of point attractors, independent of the value of k, is 1 though with a skewed distribution. It has been found that if a point attractor exists, there are often 2 or 3 of them and their basin of attraction covers most of the state space (Harvey and Bossomaier, 1997).

One experiment shows how many networks (n = 8, 16) out of 100 reach a point attractor within 10000 single node updates for different values of k:

k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	50	40	28	34	34	41	41	41

(a)

k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	41	31	28	30	34	32	36	34
k	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
	37	36	37	35	34	30	33	34

(b)

Table 2. Number of point attractors reached for (a) n = 8 and k = 1 to 8; (b) n = 16 and k = 1 to 16 (Harvey and Bossomaier, 1997).

It has been found that k = 3 has the least probability of reaching a point attractor (Harvey and Bossomaier, 1997). These experiments are repeated to verify the workings of the software suite (taking the average of 5 runs over 100 networks each):

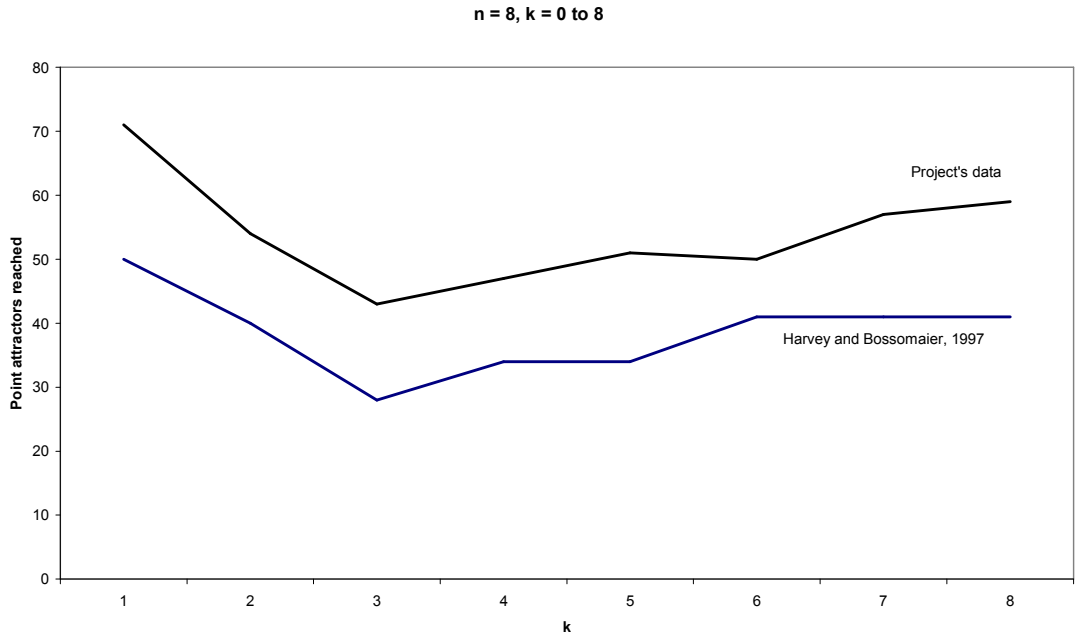
k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	71	54	43	47	51	50	57	59

(a)

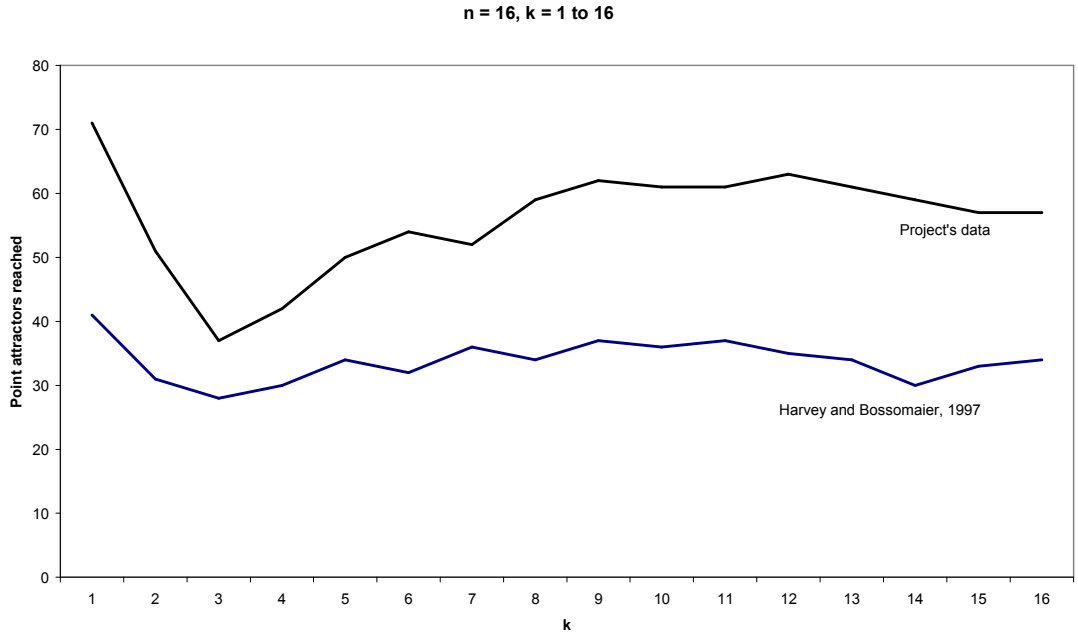
k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	71	51	37	42	50	54	52	59
k	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
	62	61	61	63	61	59	57	57

(b)

Table 3. Number of point attractors reached for (a) $n = 8$ and $k = 1$ to 8; (b) $n = 16$ and $k = 1$ to 16.



(a)



(b)

Fig. 1: Comparison of data from Harvey & Bossomaier (1997) and this project for (a) $n = 8$ and (b) n = 16 and k = 1 to n.

The general form the graph is very similar, especially the reported low at $k = 3$ is evident. The data from this project, however, has a higher average of reaching a point attractor in general. The reason for this has not been found.

Two other measurements conducted by Kauffman (1967) analyse the activity and the length of transients. The author is unaware of any published work having included such a measure.

k	**1**	**2**	**3**	**4**	**5**	**6**
	1	3	15	62	203	551
k	**7**	**8**	**9**	**10**	**11**	**12**
	911	796	1176	1675	1329	1459
k	**13**	**14**	**15**	**16**		
	1833	1959	1980	1981		

Table 4: Length of transients before a point attractor is reached for $n = 16$ and $k = 1$ to 16.

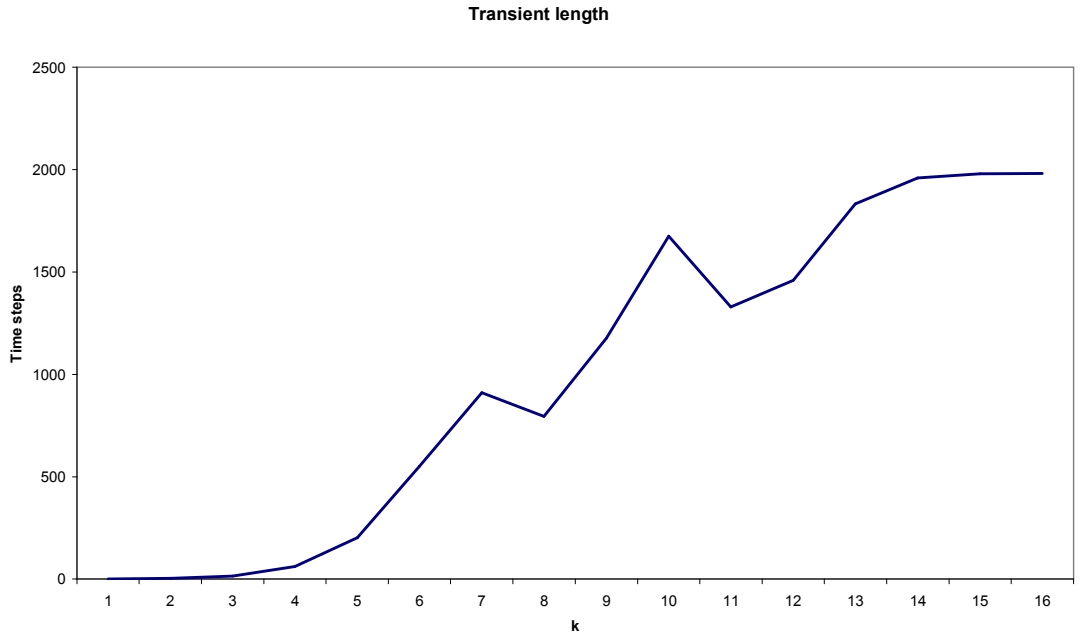


Fig. 2: Transients for $n = 16$ and $k = 1$ to 16.

The graph almost resembles a sigmoidal curve; the transient increases proportionally to k, meaning longer transients with increasing complexity of network architectures.

The activity of ARBNs is more difficult to establish given the non-determinism of the updating scheme. In cases where a point attractor is reached, the number of changing nodes per state transition does decrease, but in a non-uniform matter. An average is difficult to obtain given the different transients.

Another attempt to investigate the generic properties of ARBNs addressed the length of transients. Very long running times have been allowed before checking for rhythm using the measure of pseudo-rhythm. This approach made the assumption that asynchrony has a longer ordering phase due to its indeterminism. The networks have been tested for pseudo-rhythm after 30 000 time steps have elapsed without success.

The data that has been obtained verifies the workings of the software suite and completes the picture of the generic class of ARBNs. It is evident that the generic class of ARBNs does not exhibit rhythmic properties.

Pseudo-periodicity

Pseudo-periodicity was defined by Di Paolo (2001) in order to show relational properties of a state sequence in non-deterministic networks. It indicates the degree to which a given state of *n* nodes approximately recurs after approximately *p* x *n* single node updates. One step in time is defined a *n* random updates and the similarity between states at time *t* and *t+1* is defined as:

$$C(t,t+1) = \frac{1}{N}\sum_{i=1}^{N} s_i^*(t)s_i^*(t+1),$$

where $s_i^*(t)$ is the scaling of $s_i(t)$ into [-1, 1]. A global measure of state correlation is obtained by an average correlation between a state and its successors over *M* successive states:

$$AC(k) = \frac{1}{M}\sum_{j=1}^{M} C(t,t+k),$$

with $k = 0,1,2,\ldots$ and *M* being sufficiently large. This measure, called autocorrelation, will return an estimate of how well each state is correlated to any state occurring *k* time steps afterwards. In the case of rhythmic networks, there will be a distinct peak at values *k* close to *p*.

It should be made clear, that the correlation is an approximation and identical scores of similarity need not mean identical pairs of states. This holds true in all cases except 1 and -1 which indicate identical and opposite states respectively. The pseudo-rhythmic scale ranges from 1 to -1 with 0 being the state of neutrality as exactly half the nodes are different. For the purpose of discussion, we shall call the upper half positive correlation and the lower half negative correlation with the two states of 1 and -1 as identity and inverse identity respectively.

Pseudo-periodicity provides a measure of similarity and as such gives an indication about the Hamming distance between states; the state of neutrality for example is a distance of *n*/2 from the initial state, independent of how the individual nodes differ.

Di Paolo (2001) describes an example network which scores high on pseudo-rhythmic scale. This network will be discussed briefly as it serves as an illustrative example later on in this project and also help to visualise the concept of pseudo-periodicity. The network is homogenous and each node is connected to itself, the previous and succeeding node with wrapping around at the end. The network as a whole thus represents a chain of nodes. If any of the nodes is in the state 0, it remains there unless the preceding node is 1 and the succeeding node is 0. If a node is in state 1 it only changes if there is a 0 in the previous node and a 1 und the following node. This network produces travelling waves with period $p = n$ when updated synchronously and pseudo-periodic waves at $p \approx n$ when updated asynchronously. The start is all nodes being 1, only one node being 0.

The Genetic Algorithm

The genetic algorithm (GA) used to search for pseudo-rhythmic ARBNs has been implemented as suggested by Di Paolo (2001). It had been attempted to use real numbers in hope to increase the efficiency of the algorithm but with only little success. The original implementation uses a genome of length

$$G_L = n\left(k\log_2 n' + 2^k\right)$$

where *n'* is the first power of 2 greater or equal to *n*. For each node, the *k* inputs and corresponding Boolean function are expressed as binary strings. The genetic algorithm is one of the reasons not to simulate networks with non-uniform values for *k* as genotypes of increasing length would be needed. Also, the number of nodes *n* corresponds to a power of 2 (e.g. 8, 16, 32) as that makes the implementation inherently easier. Otherwise illegal encodings would be possible which either require careful validation or longer execution times.

Each network is simulated for around 500 – 1000 time steps over 4 – 10 trials, measuring the autocorrelation after each trial. The autocorrelation is taken for $k = 0, 1, 2, \ldots, 2n-1$ and

averaged over all but the last $2n$ states of the run. The fitness is determined by the difference between the autocorrelation and the target function and is averaged over the number of trials. A standard deviation is deducted to benefit networks with a low variance between trials. The size of the population is normally around 90 and the rate of mutation is kept around 20% per genotype. The cross-over is uniform and the entire population is replaced at once.

The target correlation

The target correlation as used by Di Paolo (2001) has been defined using steps between 0 and 1: The values of 1 are used around the chosen pseudo-period p and its multiples such that the network will show high correlation around that period. Values of 1 are assigned to values of k in $[np - e, np + e]$, with $n = 0, 1, 2, \ldots$. The value of 0 is assigned to all other value of k (Di Paolo, 2001). It has been found that the search process is very sensitive to the target correlation such that a small bias towards 1 or 0 may render the search useless. The width of the steps has to be well balanced for networks to evolve rhythm. The fitness of a network is given by $1 - D$ where D is the normalised distance between the networks autocorrelation and the target function. The subtraction of a standard deviation is used as networks may have variable pseudo-rhythmic outputs from trial to trial given different initial states. As it will be shown later, different target correlations have been identified causing different network attributes to emerge. A target function compromised only of 0's may be used to evolve de-correlated networks that have non-stationary, non-rhythmic output. These networks will be used later for reasons of comparison.

Single lesion analysis

One of the initial proposals for this project was to implement a multi-lesion analysis to determine the functional distribution of the network's behaviour. An algorithm is given by Aharonov, Segev, Meilijson and Ruppin (2002) which the authors successfully applied to neural networks to determine the contributions of individual nodes. This idea had to be abandoned, however, given the complexity of the mathematical framework required for a proper implementation. Nevertheless, some ideas of the paper have been carried forward and have been used to verify certain results. A network may have a single node lesioned and the difference in performance may be analysed using the numerical or graphical output or the measure of pseudo-periodicity. This has been done despite the criticism by the above authors on the use of single-lesion analysis and has given results that did confirm other experimental data as described later.

There are several different methods that may be employed in lesioning a single element in the network. One attempt is to cut off individual input / output links to from / to other nodes or to lesion the node itself. The latter approach has been taken and a node is lesioned by replacing its output with a random value of 1 and 0 of equal probability. It is not advisable to replace the output with a stationary value as that had an affect on neighbouring nodes as well, spreading the lesion itself. This had also been pointed out by Aharonov et al. (2002).

The software

The software lab has been constructed to evolve and analyse Boolean networks. Despite the initial planning, large proportions have been added during the time of this project according to new insights gained. It has been tried to maintain a structural approach which allows for extension of further modules and methods in the future.

Similar Programs

A few software suites are available that deal with Boolean networks or, more general, discrete dynamical systems: DDLab from Wuensche (1994) and RBN-Lab from Gershenson (2002) are two such programs, both publicly accessible. They can be used for the study and

analysis of Boolean networks whereas the first one covers a wide rage of dynamical systems including cellular automata. RBN-Lab deals explicitly with Boolean networks and several different update schemes including deterministic and non-deterministic asynchronous ones. The program provides several methods for basic analysis with emphasis on attractors and also provides visualisations of network structure and output. There are no possible modes of real time interaction or extensive statistical methods for detailed analysis. In some aspects, the software developed for this project could be seen as an extension of RBN-Lab and will mainly focus on the internal workings of asynchronous networks. Some aspects, such as the dynamical graphics or the possibility to simulate the output in single steps have been inspired by RBN-Lab; the code is, however, independently written. Several features found in RBN-Lab have been omitted due the computational requirements and long execution times (e.g. finding all attractors for a particular network). Despite several similarities between these two software suits, it should be pointed out, however, that they are both tools of analysis for rather different studies and as such should not be compared directly.

Users

The software provides a very specific tool of analysis for a restricted area of dynamical systems and has been written primarily for this project. Nevertheless, it has been tried to make the program as accessible as possible to other users. The target user group is expected to be a small number of researchers with background in biology, chemistry, mathematics, computer science or artificial intelligence. A familiarity with the topic may be assumed, computer literacy, however, may not. Comparing DDLab and RBN-Lab, for example, it appears that latter one is more intuitive to use given its graphical user interface (GUI). A GUI may introduce a large overhead and may introduce additional errors, especially caused by parsing user supplied numerical values. Nevertheless, a GUI has been provided, using a clean and simple design and intuitive layout to make the programs functions accessible to the user.

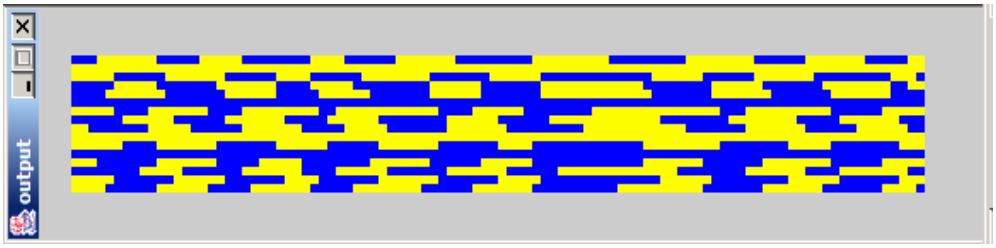
How the program works

The entire software suite has been written in Java 1.4 using Swing as the Graphical User Interface (GUI) design component. There has been a hybrid approach also, attempting to combine the use of Java and Mathworks Matlab but without success: The interface between the languages is still too pre-mature to establish a stable suite. Therefore, almost all analytical and behavioural methods have been implemented in Java except the Fast Fourier Transform (FFT) analysis which has been done in Matlab. The main emphasis of this project was the research aspect using software as a tool of analysis and not to provide a industrial piece of software such as an office tool. Nevertheless, care has been taken to ensure modularity and extensibility such that new aspects may be added with ease as the research area grows. This gave reason to a floating panel approach as that utilises the integration of new components without the need to change the entire graphical layout. A central controller class has been established which serves as interface between all classes but a few, establishing a two way communication that enables any class to use whatever methods provided by other classes. The design is not the most efficient which raises doubt, especially in such computational demanding tasks such as a genetic algorithm. This approach, however, easily allows for a feature that has been thought very important: the user is able to change update schemes, apply lesions or noise on the fly while the network executes. This enables the user to observe the effect of new attributes much more explicitly. Another important aspect has been the emphasis on visualisation aspects as complex dynamical systems are often best understood on an observational basis. Unfortunately there was no time for real-time interaction modes such as the visual construction of networks (in a drag and drop manner). Also, it had been attempted to implement an optimal network layout design component (equal spread between nodes with minimal amount of crossing edges) using simulated annealing. This had not been completed in time but considering the pruning algorithm described later, this is not thought of major significance.

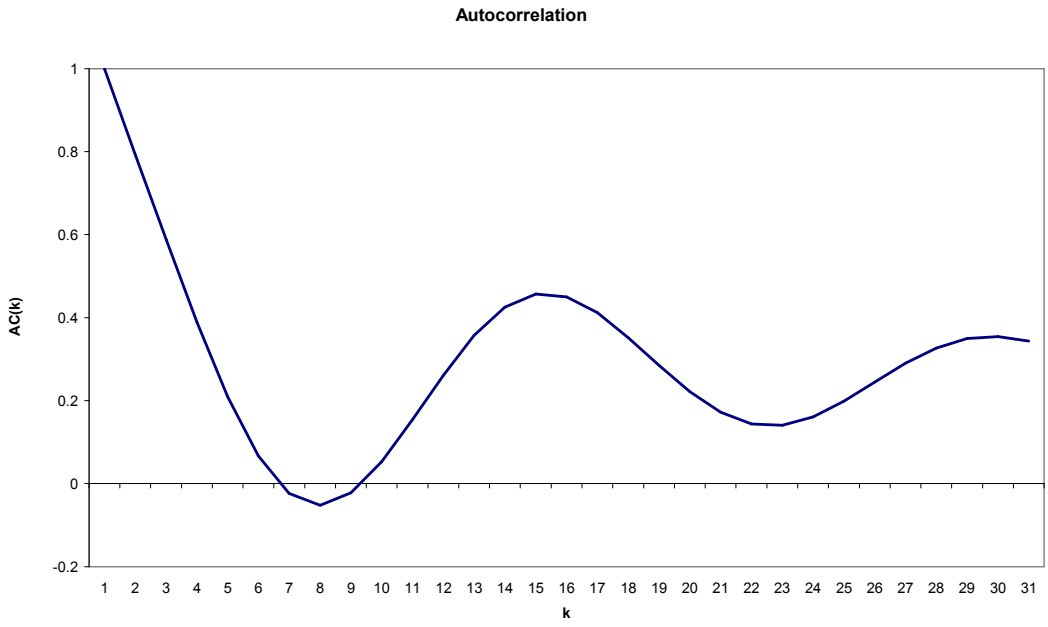
Some of the analytical parts have been done by hand and the pruning algorithm has not yet been implemented but these aspects are to be included in subsequent versions of this software. Due to the large degree of functionality provided in the software suite, an explanation and manual-like script has been placed in the appendix (including screen shots). Once again it is noted that the research aspects remains the focus of this project. The rest of the report will focus on the individual analytical components and experiments carried out.

Contribution analysis

Several networks exhibiting marked rhythm around their target period have been evolved. Most attention has been given to networks with $k = 2$ as these constitute the most interesting case in the synchronous class of networks. Nevertheless, other networks with $k = 3$ and 4 have been evolved also. The main aspect of the rest of this project is to determine the origin of rhythm and to verify whether or not general principles regarding rhythm in ARBNs can be obtained.



(a)



(b)

Fig. 3: Evolved network for n = 16 and k = 2; (a) graphical output, (b) autocorrelation.

The behaviour of the network is due to the interaction of the topological and Boolean layer subject to the update scheme. It is therefore of primary interest to investigate the network behaviour on the level of nodes and their Boolean functions, to determine if their individual behaviours are rhythmic themselves and if they operate upon different frequencies. The simplest approach is the utilisation of the pseudo-rhythmic measure applied to individual nodes. This is

done for every node and the resulting rhythmic functions can be compared in a graph. Most nodes do have very similar rhythms, marked around the target period. It is striking to see that almost all nodes are centred on y = 0. This is because a single flip implies inversed identity as the Hamming distance for single nodes can at most be 1. Stationary nodes, however, show flat lines close to y = 1. They score extremely high on the measure of similarity as they are a constant reflection of themselves.

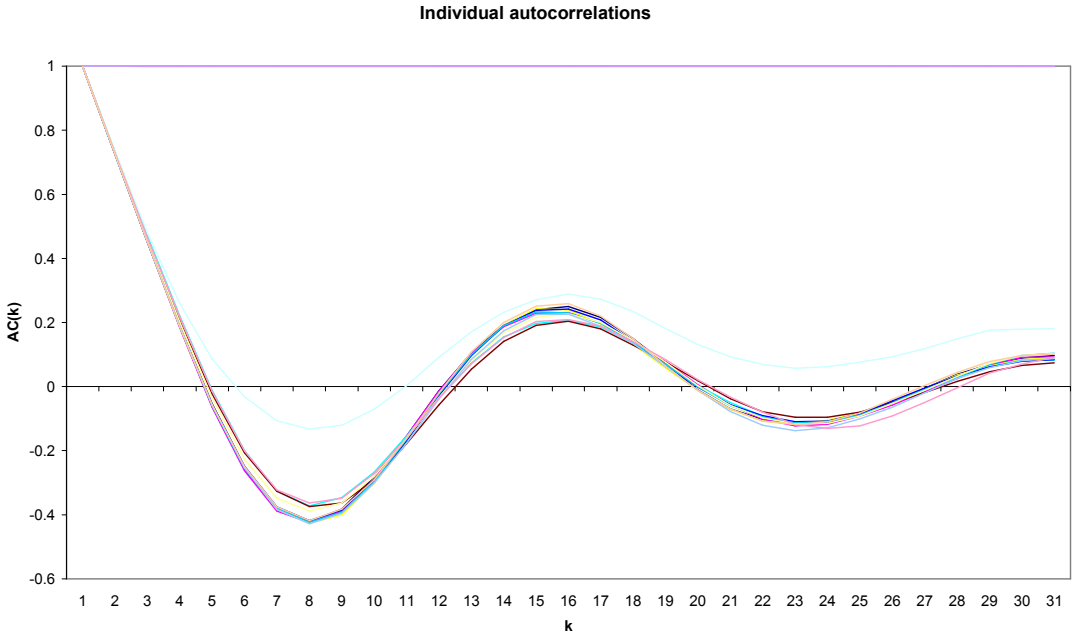


Fig. 4: Autocorrelations for individual nodes for n = 16 and k = 2.

As stationary nodes are not rhythmic themselves, it has been wondered how their omission may effect the overall behaviour of the network. Running the network with each stationary node lesioned in turn surprisingly revealed that most stationary nodes have a profound effect on the difference between the autocorrelation and the target function. Despite initial scepticism, stationary nodes seem to be of high importance to the workings of rhythmic networks. On closer inspection, however, another theory is more likely: stationary nodes score extremely high on the pseudo-rhythmic scale and shift the networks correlation, which is merely the summation of all the individual rhythms, along the y-axis and thus closer to the target correlation. Indeed, networks with stationary nodes necessarily have states which are more similar to each other. A network with a single active node, for example, only has two states with a Hamming distance of at most 1. In general, the Hamming distance is proportional to the number of active nodes.

This has led to the formulation of different target functions. The aim was to find out whether or not networks without stationary nodes may obtain higher degrees of rhythm. The different target functions were as follows:

- [-1,1]
- [-0.5,0.5]
- Cosine function
- Relative distance between peak and trough

It has also been tried to evolve rhythm for each node independently by adjusting the fitness evaluation accordingly. The results were mixed with the target function of [-1,1] not generating very promising results. The reason for this is probably the duration of the evolutionary process, requiring a longer execution times than a target function which is less extreme in turns of the Hamming distance covered. The function is probably more difficult to approximate as it incorporates both states of identity and inverse identity, the only cases that do not allow any

freedom for variations. The less extreme [-0.5,0.5] target function did work much better and showed networks approximating the target function rather well. Of course, the use of a target function which does not require identical states to re-occur every period p is questionable. A cosine curve has been implemented to see if a more natural target function would improve or accelerate the evolutionary process. The results were slightly but not significantly better than the previous results. The measure of relative distance defines the fitness of a network as the distance between its trough at period $p/2$ and peak at p. This function is assumed the least constraining one and has been the most successful producing good results. Trying to evolve rhythm for each node separately has produced only insignificant results. This could be due to the fact that the focus on individual nodes somewhat neglects the interaction between nodes and each node acts more or less in isolation. Nevertheless, this set of experiments showed that stationary nodes are by no means necessary as none of the above had any stationary nodes (except in one case which had a single stationary node). The effect of stationary nodes is the graph's location along the y-axis and networks with stationary nodes did in general approximate the score of 1 better that those networks without. Stationary nodes are the only reliable components in asynchronous networks as they are deterministic and thus help to stabilise the overall behaviour of the network. Kauffman (1969) also pointed out the large proportion of stationary nodes within cyclic attractors of synchronous networks.

SRBNs with a value of $k = 2$ either terminate in a point attractor or a cyclic attractor of normally short length. Evaluating the behaviour of the evolved networks when updated synchronously may uncover some intrinsic properties as the topological and Boolean layer are identical in both cases. This has been done for all networks and it is striking to see that the shapes of their autocorrelations are almost identical. The asynchronous autocorrelation is near-identical in all cases due to the target correlation and the genetic algorithm used. This need not hold true for the synchronous case, however, but most graphs have a perfectly triangular structure with amplitude little greater than 1.

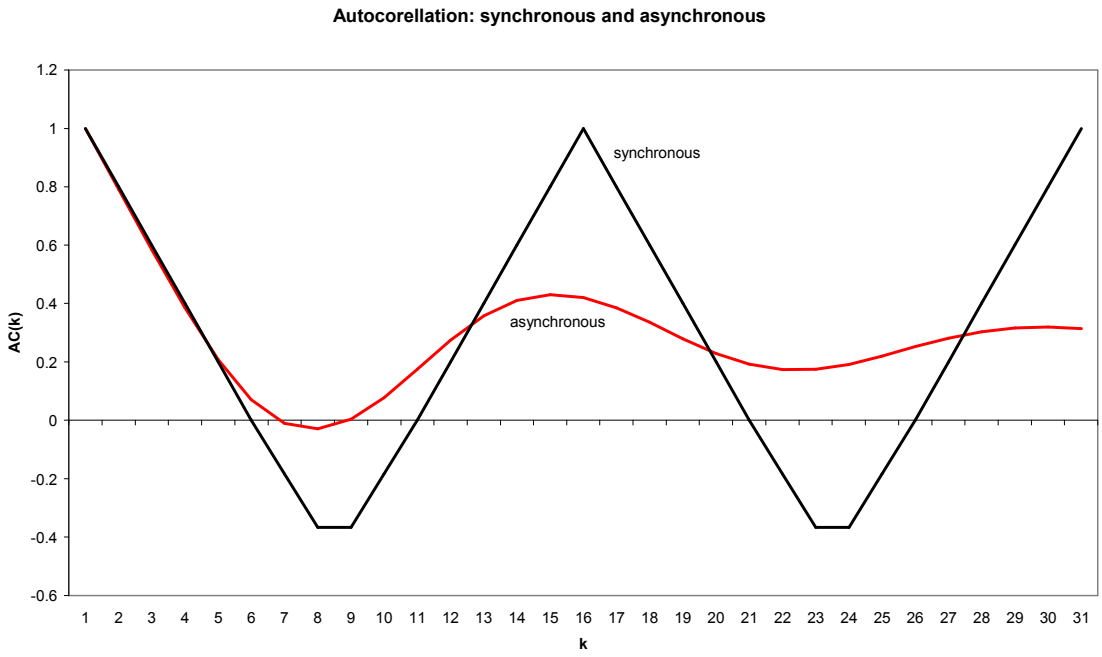


Fig. 5: Autocorrelations for network with n = 16 and k = 2 after synchronous and a synchronous updating.

This distinct feature is produced by a network that traverses the distance of period p in uniform steps covering an amplitude A of 1. In other words, the network moves from one state to the next with a (usually small) uniform number of changes. Some graphs exhibited slightly different shapes but it was found, that these networks too produced a triangular shape limited to

certain initial states. Interestingly, if the network was initially updated synchronously, the resulting pseudo-rhythmic function did not automatically resemble the triangular structure but if the network was updated asynchronously first the network ended up displaying the before mentioned form. Synchrony seems to result in the idealised behaviour of the network (i.e. without decay) given the lack of randomness or noise.

To evaluate the effect of the number of changing nodes per state transition, a hypothetical network had been set up which simply consisted of a Boolean look-up table that recorded the node changed last. A node changes its state only if its preceding node had changed in the previous state transition. Consequently only one change per state transition is possible in the synchronous case producing a rhythm of period $2n$. The probability of disturbing the marked rhythm in the asynchronous case grows exponentially: Each node has the probability of $1/n$ being updated per state transition. As a time step is normally defined as n updates, this implies a node is updated once per state transition on average (the actual probabilities have been obtained from a network's execution and found to be in the range of 0.97 and 1.03). Therefore, if node a had been changed during the last state transition, node b is due to change next with a high probability. However, changing node b followed by c, which should be changed during the next state transition, is rather unlikely as the probability is $1/(2n)$. This unlikelihood grows proportional in n, also implying that large networks are more stable than smaller ones.

The hypothetical network shows above average pseudo-rhythm, producing waves of amplitude 1.2, oscillating between the values 0.51 and -0.51 for $n = 16$. Repeating this test with larger networks gave amplitudes up to 1.6, reaching 0.8 on the scale of similarity.

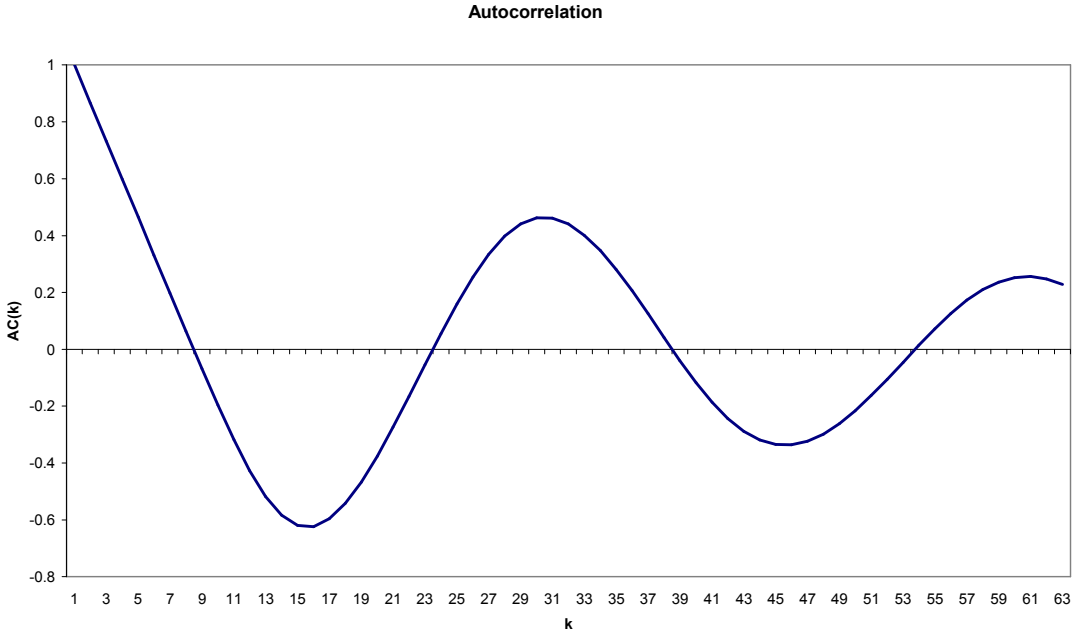


Fig. 6: Autocorrelation for hypothetical network with n = 16.

A Chain of Nodes

The low number of changing nodes is the result of a chain architecture in which each node is connected to its preceding node and itself, with wrapping around at the edges. Characteristics of this architecture can be found in the properties of the evolved networks.

The outputs of the evolved ARBNs only have a low number of nodes changing at each state transition. The number of changes are counted and averaged over a several time steps for several networks. These numbers are not uniform but have a low variance. This conforms well given the indeterminism of the update scheme and the fact that networks can only be an approximation to a perfect architecture. Table 5 lists some statistics obtained from networks evolved with a target function of [0,1].

active nodes	changes / transition	variance of changes
12	1.630	1.138
12	1.606	1.117
13	1.831	1.199
10	1.457	0.998
16	2.285	1.342

Table 5: Statistics for evolved networks using a target function of [0,1].

The next section will determine a couple of equations that describe the theoretical properties of a chain architecture. The applicability of these formulae is then tested on the real networks. It should be pointed out that these equations describe a deterministic system and as such provide either an upper or lower bound. The data from the asynchronous networks is therefore only expected to approximate these equations.

1.)

$$\frac{An}{2} = a,$$

where A is the amplitude of the pseudo-rhythmic graph, n is the number of nodes and a is the number of active nodes. The amplitude describes the lower bound on the active nodes: It is possible for a 16 node network with 16 active nodes to exhibit an amplitude of 1 if two sets of 8 nodes change in sequence back and forth. The lower bound is given as there needs to be at least 1 active node for an amplitude to exist and the size of the amplitude grows proportional to the number of active nodes. The maximum amplitude of 2 is only possible with 100% network activity. This also implies a tight relationship between the amplitude and period given a certain number of active nodes.

2.)

$$c = \frac{a}{p} \times 2,$$

where c is the average number of changing nodes per state transition, a the number of active nodes and p the period as previously.

Other equations may be used but most include the use of the amplitude which is a very uncertain component and given the maximum value of 2, the margin of error may be too great. The formulae are applied to a selection of data from 10 networks (n = 16, k = 2, period = n) which have been evolved using different target correlations.

active nodes	changes / transition	variance of changes	period	amplitude
12	1.630	1.138	15	0.468
12	1.606	1.117	16	0.475
13	1.831	1.199	15	0.509
10	1.457	0.998	14	0.456
16	2.285	1.342	16	0.575
16	2.175	1.280	15	0.769
15	1.987	1.173	16	0.554
12	1.596	1.073	16	0.449
12	1.551	1.161	17	0.602
13	1.765	1.139	16	0.487

(a)

$\frac{An}{2} - a = 0$	$\frac{a}{p} \times 2 - c = 0$
-8.3	-0.030
-8.2	-0.106
-8.9	-0.090
-6.3	-0.028
-11.4	-0.285
-9.8	-0.041
-10.6	-0.112
-8.4	-0.096
-7.2	-0.139
-9.1	-0.140

Mean	-8.8200	-0.1067
Variance	2.2973	0.0056
Std.	1.5157	0.0751

(b)

Table 6: (a) Data from 10 evolved networks updated asynchronously and (b) the results obtained from the equations.

Equation 2 seems to apply but equation 1 does not indicate the relationship holds. The reason for this could be the deterministic nature of the equation and the indeterminism of the update scheme. The data obtained from synchrony is similar except in the case of the amplitude which is generally higher. If the data obtained from synchronous updating is applied to the first equation the relation seems to hold.

active nodes	changes / transition	variance of changes	period	amplitude
12	1.600	0.640	16	1.360
12	1.498	0.561	17	1.453
13	1.624	0.701	17	1.296
10	1.332	0.443	16	1.166
16	2.372	0.966	17	1.719
16	2.135	0.580	16	1.815
15	1.873	0.328	17	1.687
12	1.499	0.625	17	1.296
12	1.497	0.685	17	1.484
13	1.732	0.488	16	1.233

(a)

$\frac{An}{2} - a = 0$
-1.12
-0.37
-2.63
-0.67
-2.24
-1.48
-1.50
-1.63
-0.12
-3.13

Mean	-1.4890
Variance	0.9458
Std.	0.9725

(b)

Table 7: (a) Data from 10 evolved networks updated synchronously and (b) the results obtained from the equation.

The data approximates the equation sufficiently considering that evolved networks are not clean architectures. The nature of this relationship is twofold as all networks have not just a similar topology but also identical pseudo-periodicity. To determine if the equations apply to a wider set of networks which are rhythmic, 5 synchronous networks are evolved using the same genetic algorithm and target function[1].

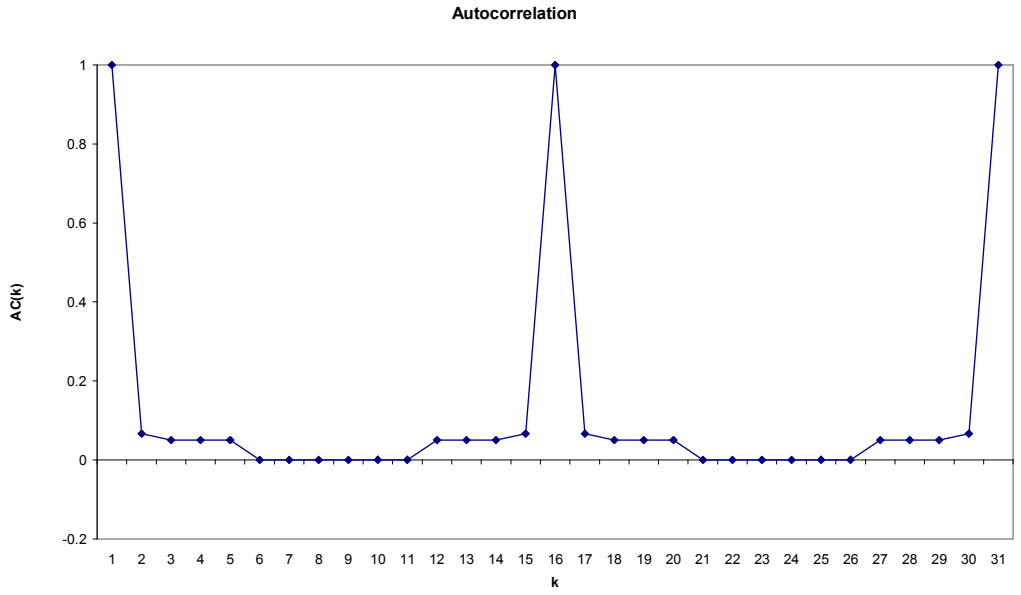


Fig. 7: Evolved synchronous network ($n = 16$, $k = 2$) with period n and amplitude 1.

[1] It is striking to note that mutation plays a much more important role than cross-over and large plateaus of neutrality seem to dictate the search space. A solution is normally found in less than 20 epochs, or the search continues for much longer, often without any increase of fitness over many epochs.

active nodes	changes / transition	variance of changes	period	amplitude
15	6.66	2.22	16	1
14	4.85	1.32	16	1
15	7.60	2.77	16	1
15	5.88	2.13	16	1
16	7.47	1.64	16	1

(a)

$\frac{An}{2} - a = 0$	$\frac{a}{p} \times 2 - c = 0$
-7	-4.785
-6	-3.100
-7	-5.725
-7	-4.005
-8	-5.470

Mean	-7	-4.6170
Variance	0.5000	1.1648
Std.	0.7071	1.0793

(b)

Table 8: (a) Data obtained from evolved synchronous networks (n = 16, k = 2, period = 1) and (b) the results obtained from the equations.

The results of both equations seem to validate previous results but a more extensive mathematical framework is needed. This is difficult to establish as the data components rely solely on the behavioural features of the network and can not make use of any structural information. This problem is also highlighted in the fact that there are many cases of synchronous networks which seemingly obey the equations but do not exhibit any marked rhythm in the case of asynchrony. The rest of this project will therefore focus on the structural properties. In particular, the pruning algorithm presented later can be used to prune a network and to uncover its functional architecture which is thought responsible for the behaviour of the network as a whole. This has been done for pseudo-rhythmic and de-correlated ARBNs and the SRBNs with period n and amplitude 1. The resulting topologies are very different, indicating the validity of the equations.

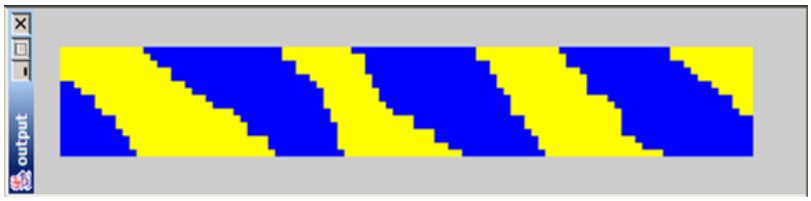
Strict cycles in asynchronous networks

The cases of synchronous networks whose attributes coincide with the relational properties of a chain architecture are thought to be the result of synchrony itself and not internal structure. A concept that will try to explain the role of local timekeeping describes the physical relationship between nodes in the form of coupling.

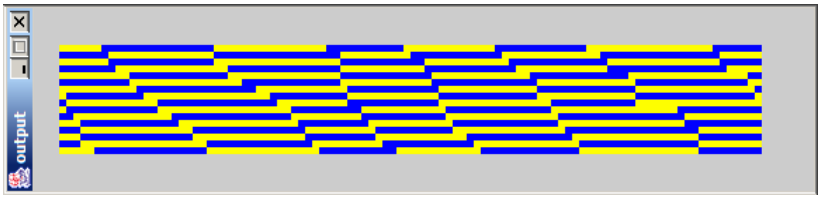
The nodes in a chain architecture are all coupled to their preceding/succeeding node, establishing a tight overall coupling with allows only one node to change per state transition; The network mentioned in Di Paolo's (2001) study also had a coupling between neighbouring nodes, but did allow for multiple nodes to change per state transition. Since there has been no feedback between the changing nodes, they may easily get out of sync, reducing the rhythmic performance.

A chain architecture has been implemented, using n = 16 and k = 2. Each node is connected to its previous node and itself and all nodes but one have the same Boolean function which says that the node is identical to its preceding node. The only function that is different, the driving

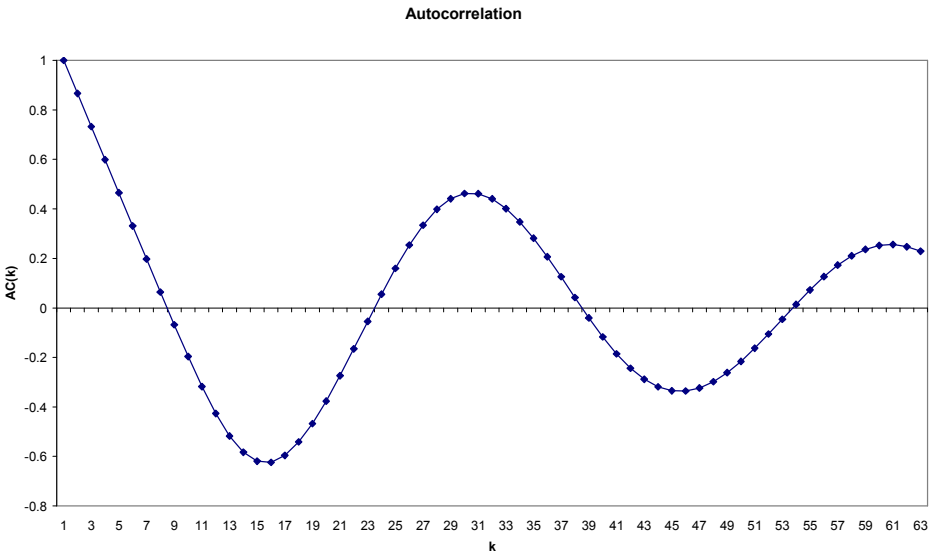
function, is the exact opposite. This network can be further reduced (using the pruning algorithm) to $k = 1$ which can be shown sufficient to produce maximum pseudo-rhythm.



(a)



(b)



(c)

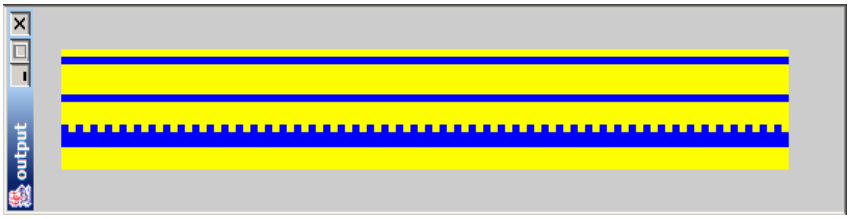
Fig. 8: Output of chain architecture asynchronous network with all nodes (a) follow (b) oppose each other and (c) the autocorrelation of this network.

There are two reasons for synchronous networks to exist that conform to the identified equations but are non-rhythmic given asynchrony:

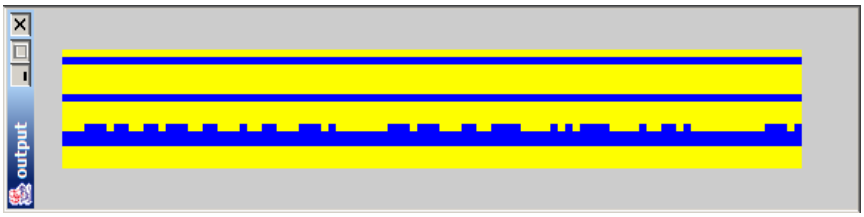
- The output is solely due to the synchrony and not the architecture of the network. Examples are de-correlated ARBNs which still exhibit rhythm given synchrony.

- The networks do produce rhythm but the discrete time steps for measuring state transitions is too insensitive to detect this rhythm.

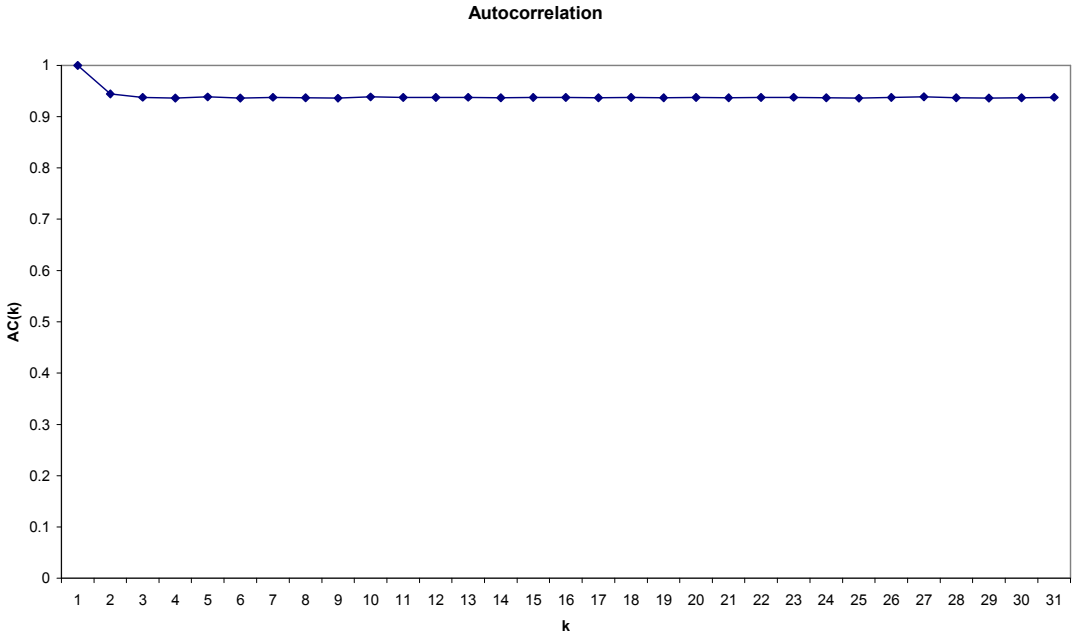
The simplest case is a synchronous network with one active node which changes at each state transition. This is only possible if the node is coupled to itself. Given the tight coupling, the network would be expected to exhibit rhythm in asynchronous updating but produces a flat line on the pseudo-rhythmic scale. The network's behaviour has to be cyclic as the network can only be in exactly two states which differ a Hamming distance of 1. Asynchrony allows remaining in the same state for several time steps due to 0 or multiple updates of the active node which leads to non-rhythmic behaviour over time. The reason for this behaviour is not, however, the non-existence of strict cycles but the measure of duration between states. The idea that ARBNs may have strict cyclic attractors does not imply the possibility of perfect rhythm as rhythm is defined as a behaviour over time. This network is an example of a much more general phenomenon as any perfect chain architecture exhibits strict cycles and in fact, any tightly coupled system can be expected to do the same. The discrete time steps that are chosen as a measure for the rhythmic activity, however, may be too coarse to detect subtle change. Di Paolo (2001) suggests the use of the external clock as a measuring and not driving device which lead to the formulation of pseudo-periodicity. The network's overall rhythmic behaviour is more the result of the operational relationship between states instead of the externally measured duration of states. This relaxation can be taken a step further by taking the clock out completely, or more specifically, by assuming the network and clock are one and the same thing. This, of course, raises the question how to detect new states and to verify a network is rhythmic. In the simple case above, a new state of the network is encountered whenever the value of a single node has been changed. An independent counter is used as timer and is incremented once every update. In doing so, the output of a network is strictly cyclic and a time line is obtained that indicates the degree of rhythm. This approach thus separates the aspect of rhythm into cyclic behaviour and time. The time line may then be compared to the time line that would correspond to fixed discrete time intervals (incremented by n updates) and it can be shown that they are very similar. An example is given in figure 9.



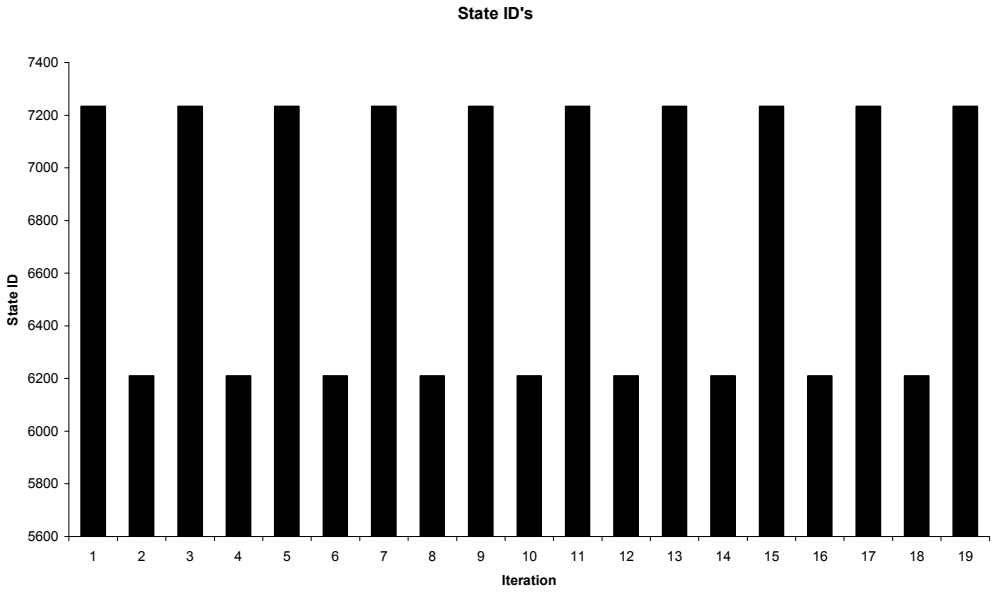
(a)



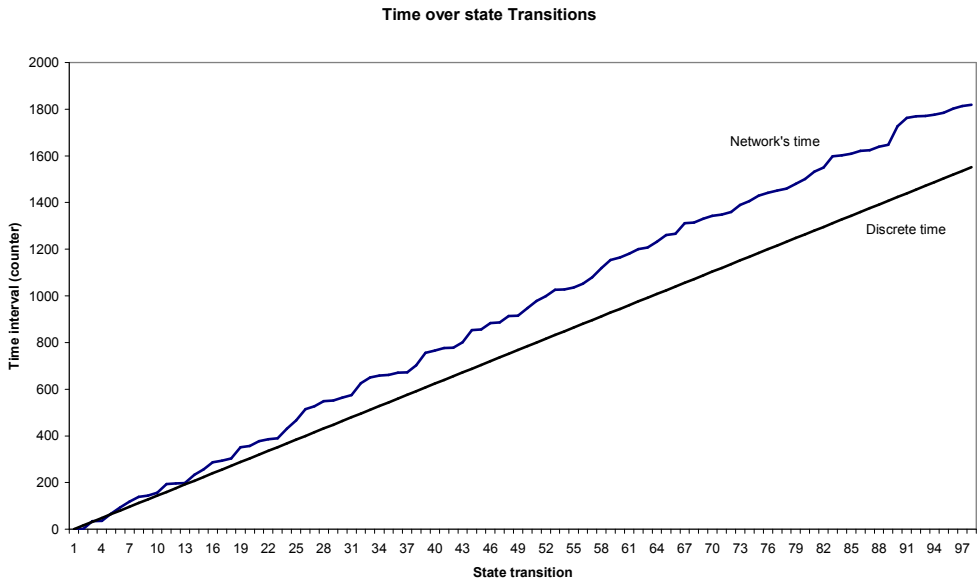
(b)


Autocorrelation
AC(k)
k
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

(c)


State ID's
State ID
Iteration
7400
7200
7000
6800
6600
6400
6200
6000
5800
5600
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

(d)

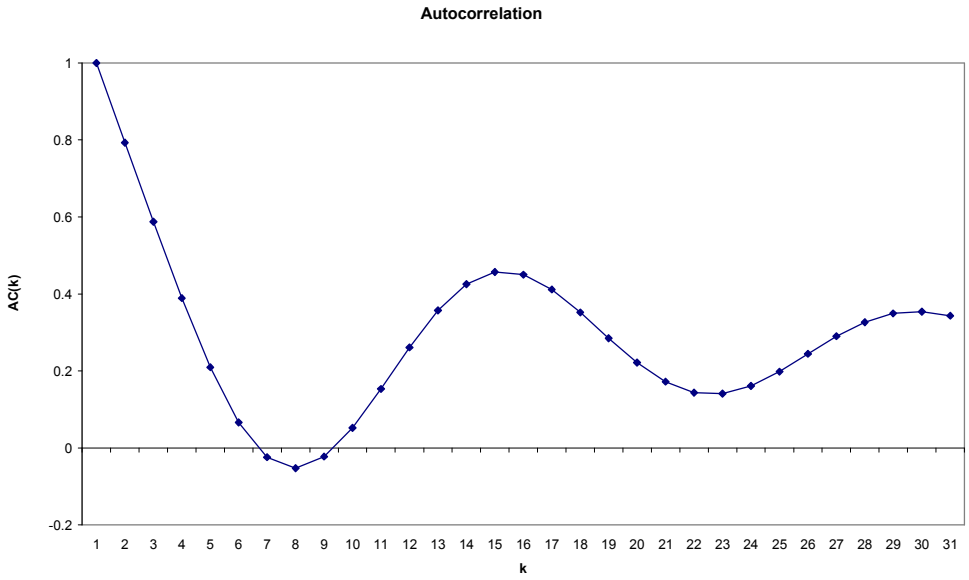


(e)

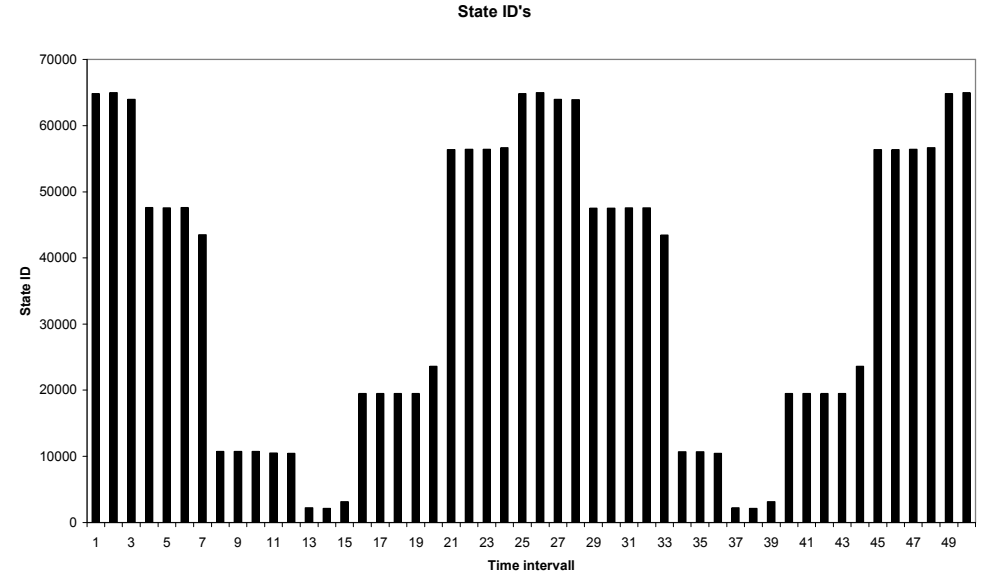
Fig. 9: The output of an almost stationary network with (a) synchronous updating and (b) asynchronous updating. (c) Autocorrelation for the asynchronous updating, (d) states traversed after each change and (d) timeline produced by the network.

The time line of the network (Fig. 9e) is compared to the time line obtained from discrete time intervals being *n* updates. The network's timeline diverges over subsequent state transitions, is, nevertheless, almost linear.

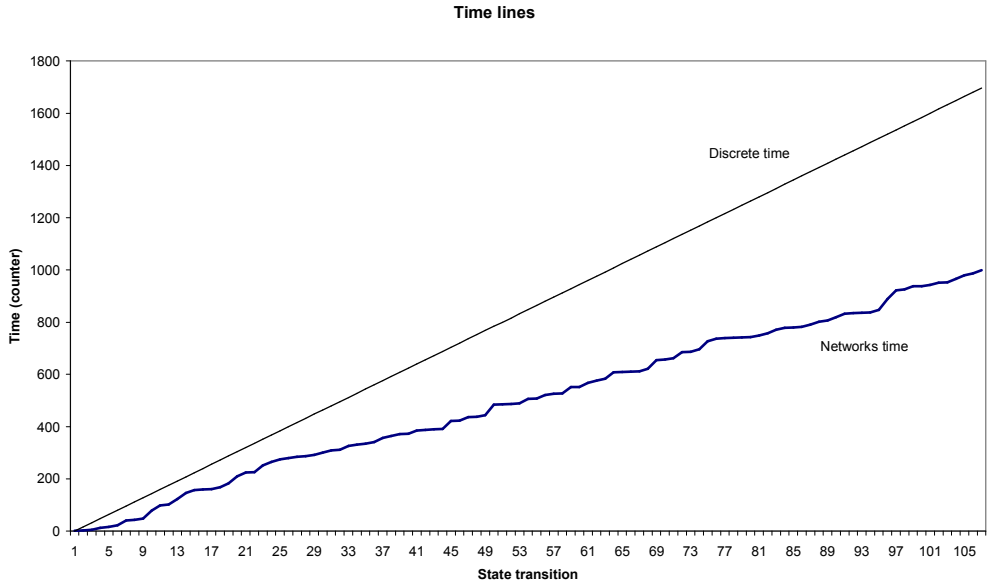
The same measure can be applied to larger chain-like networks which exhibit the same cyclic behaviour with a time line very closely related to the discrete one. This measure is easily applicable given knowledge about the number of changing nodes needed to encounter a new state. Normally this is not the case. Synchrony may be used to obtain the sequence of changing nodes but still leaves the problem at which point in the networks history this sequence starts. This is a problem of its own but it can be shown that a crude approximation suffices to illustrate the effect. Taking the average number of changing nodes as indicator works in the cases of evolved networks as the underlying architecture is known. This has been done for one of the evolved networks. The network's timeline diverges but is almost linear; the state sequence produced in almost perfectly cyclic.



(a)



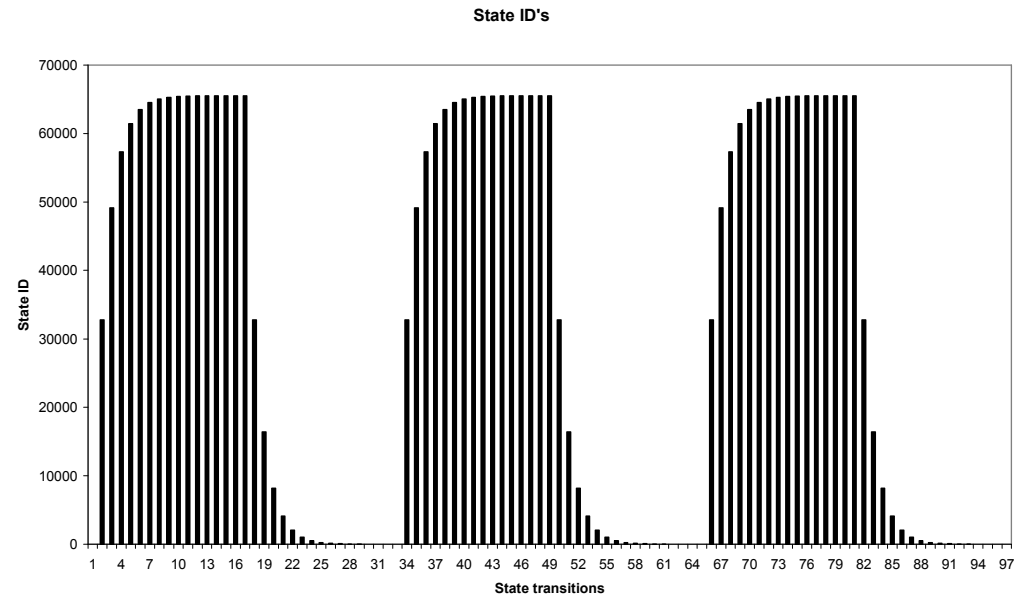
(b)



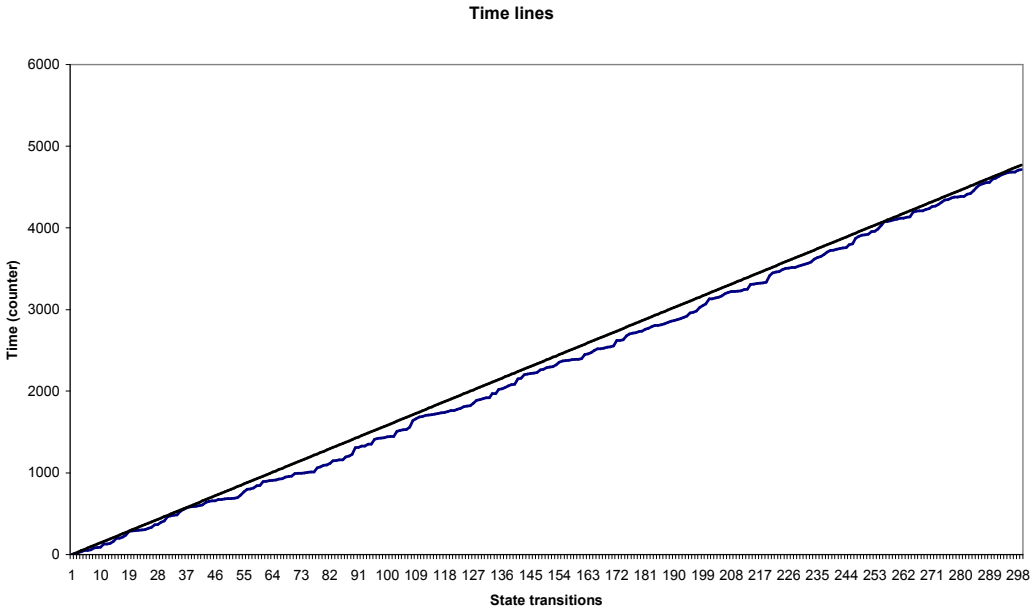
(c)

Fig. 10: Evolved pseudo-rhythmic network with n = 16 and k = 2. (a) Autocorrelation, (b) states traversed after a single update and (c) time line produced by the network.

The same technique may be applied to a perfect chain architecture which highlights the high degree of similarity of the time line produced.



(a)



(b)

Fig 11: Perfect chain architecture network. (a) States traversed each single node change and (b) the time line produced by the network.

To identify the exact architecture that is responsible for the emergence of rhythm, a simple pruning algorithm has been developed. This algorithm is subsequently used to uncover the functional architecture of several networks.

Boolean interaction

The following algorithm presents a simple technique to prune a network by deleting redundant links and nodes. It will be shown that the method is reliable and there is no reason to assume that a more complicated procedure is needed as natural systems would be likely to utilise the least costly method themselves (e.g. Ockams razor). A more complicated technique had been employed earlier, but with less success. It is included in the appendix for completeness.

Each Boolean function is classified as being determined, biased or neutral. If all the values are identical (i.e. all 1's or 0's) then the function is determined as its outcome is always known. If the function is exactly divided between the two values, then it is neutral as it does not have a natural tendency to be in one state or the other. If one value is dominant, the function is biased as it is statistically more likely to be in the state of the more frequent value. Examples are: 0000 (determined), 1010 (neutral), 1000 (biased). It will be shown that using the concept of natural tendencies suffices to approximate the networks determining structure (which we will call functional architecture) to a satisfactory degree.

1.) All the Boolean functions are examined and nodes whose function is either tautology or contradiction are labelled as 'stationary'. The inputs to these nodes can be pruned as they make no difference to the outcome of the node.

For each node in turn …

2.) If the Boolean function of the node in question co-insides with one of the columns of its truth table, the input node corresponding to that column is kept and all other inputs to this node may be pruned.

3.) The Boolean functions of the input nodes are examined: If all are neutral, none can be pruned. If any of the functions is biased, the effect of this bias is used to determine how the influences of all inputs are affected: If one node has a natural tendency towards the value 1, for example, all inputs are measured against the part of the Boolean function where the value of the identified input is 1 in the truth table. This is done for every node (i.e. if two nodes are biased, the focus is shifted towards the part in the Boolean function that corresponds to the bias expressed by both inputs). If one input turns out to be determining the output while none of the others do, all other inputs are pruned. If several inputs have identical influences but some have none, they may be pruned.

4.) Special attention has to be given if the same node serves as input multiple times. In these cases, only columns which have identical values need to be considered. This may lead to pruning despite all inputs being neutral.

This concept can, of course, be formalised with the concept of information theory but unless the algorithm is to be implemented or used in conjunction with larger values of k, there is no need in doing so.

input 1	**input 2**	**truth table**	**prune**	**label**
0101 (node 4)	0011 (node 10)	00 0 01 0 10 1 11 1	input 2	same
1010 (node 2)	0000 (node 8)	**00 1** 01 0 **10 0** 11 1	input 2	opposite
0001 (node 9)	1100 (node 6)	**00 0** **01 1** 10 1 11 1	input 1	same
1001 (node 1)	0110 (node 13)	00 0 01 1 10 1 11 1	-	-

(a)

input 1	**input 2**	**input 3**	**truth table**	**prune**	**label**
10101010 (node 3)	00101111 (node 5)	00101111 (node 5)	**000 0** 001 0 010 0 **011 0** **100 1** 101 1 110 0 **111 1**	input 2 input 3	same
01011101 (node 1)	10101010 (node 3)	01001101 (node 4)	000 0 001 0 010 1 011 0 **100 1** **101 1** **110 1** **111 1**	input 2 input 3	same
10101010 (node 3)	10101010 (node 3)	10101010 (node 3)	**000 0** 001 1 010 0 011 0 100 0 101 1 110 0 **111 0**	input 1 input 2 input 3	(stationary)
01011101 (node 1)	00010111 (node 2)	00101111 (node 5)	000 0 001 0 010 0 011 1 100 0 **101 1** 110 1 **111 1**	input 2	-

(b)

Table 9: Examples of the application of the pruning algorithm for (a) k = 2 and (b) k = 3.

The method has been applied to several pseudo-rhythmic networks of size $n = 16$. The resultant graphs all had similar properties: A circular structure incorporating the majority of nodes with some links to single nodes and sometimes what seemed to be feedback loops. In the case of $k = 2$, it is evident that many links may be pruned leaving a structure similar to $k = 1$. Indeed, it can be shown that $k = 1$ suffices to construct rhythmic asynchronous networks. Evolving asynchronous networks with $k = 1$ has not shown very successful. The space of possible topologies is probably too constrained as the number of possible (and successful) approximations is reduced. Examples of functional architectures are given in figure 12.

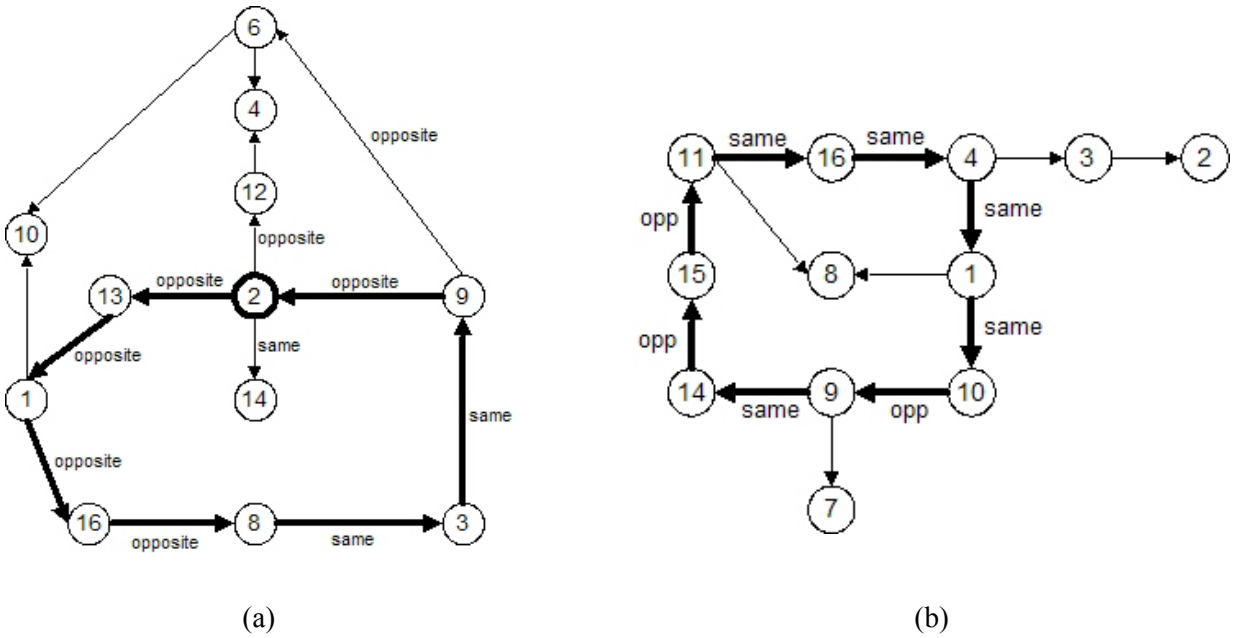


Fig. 12: The functional architectures of two distinct pseudo-rhythmic networks with n = 16 and k = 2 (stationary nodes omitted).

The labels indicate the relationship between neighbouring nodes. The output of the network is then analysed and it can indeed be shown that the relational features of the output are captured by the graphs. Some output parings of the nodes shown in graph (b) are listed for verification in table 10.

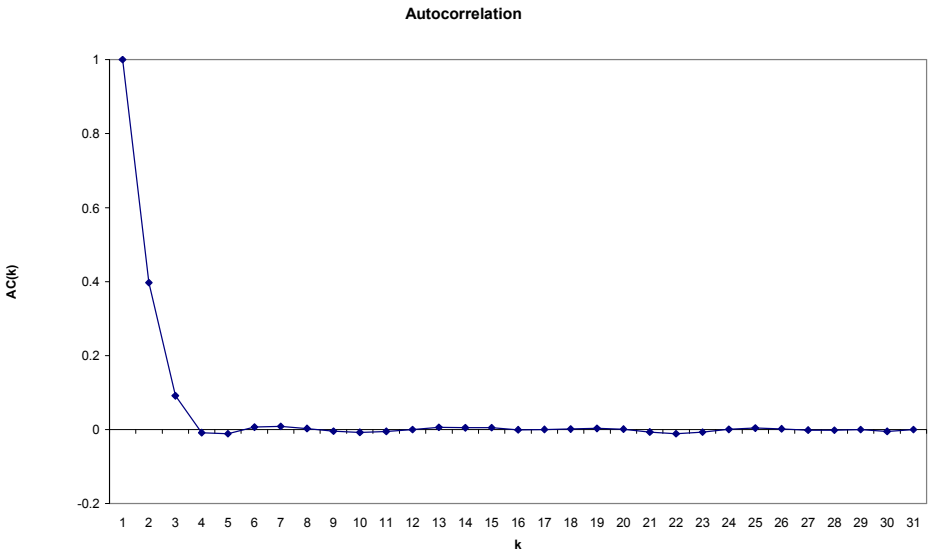
nodes	4 & 1 same		9 & 14 same		14 & 15 opposite		15 & 11 opposite	
	1	1	0	0	0	0	0	1
	1	1	0	0	0	1	1	0
	1	1	0	0	0	1	1	0
	1	1	0	0	0	1	1	0
	0	1	0	0	0	1	1	0
	0	1	0	0	0	1	1	0
	0	0	0	0	0	1	1	0
	0	0	0	0	0	1	1	0
	0	0	0	0	0	1	1	0
	1	0	0	0	0	1	1	0

Table 10: Output form network (b) to compare the relationship between nodes.

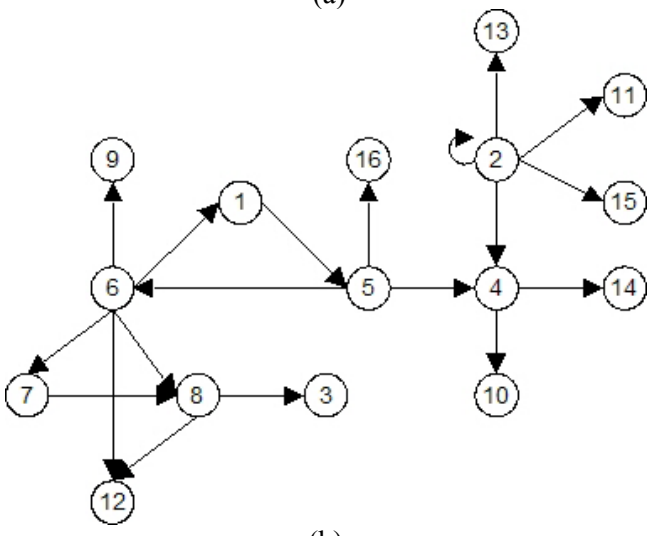
Additional information that can be used, especially in cases which seem unclear, is the probabilities of each node being in one state or another. Another example for the validity of the method can be illustrated with the first graph: node 2 seems to be of more importance than any

other node as it is connected to 4 other nodes. Indeed, the lesion analysis of node 2 results in the complete loss of rhythm whereas lesions applied to nodes 4, 10 and 14 had hardly any noticeable effect on the rhythm produced by the network. In the case of the second graph, a new network has been constructed using the graph as a guideline. The new network with $n = 16$ and $k = 1$ tried to implement the graph as accurately as possible (e.g. node 8 has 2 inputs but with k being 1, one of the input has to be omitted as k has to be uniform throughout). Despite the absence of a few links as indicated in the graph, which itself is a cut-down version of the original network, it has been shown that the output of the newly constructed network was very similar to original network's output.

Given this algorithm, it would be valuable to compare rhythmic networks with de-correlated ones to compare their functional architectures.



(a)



(b)

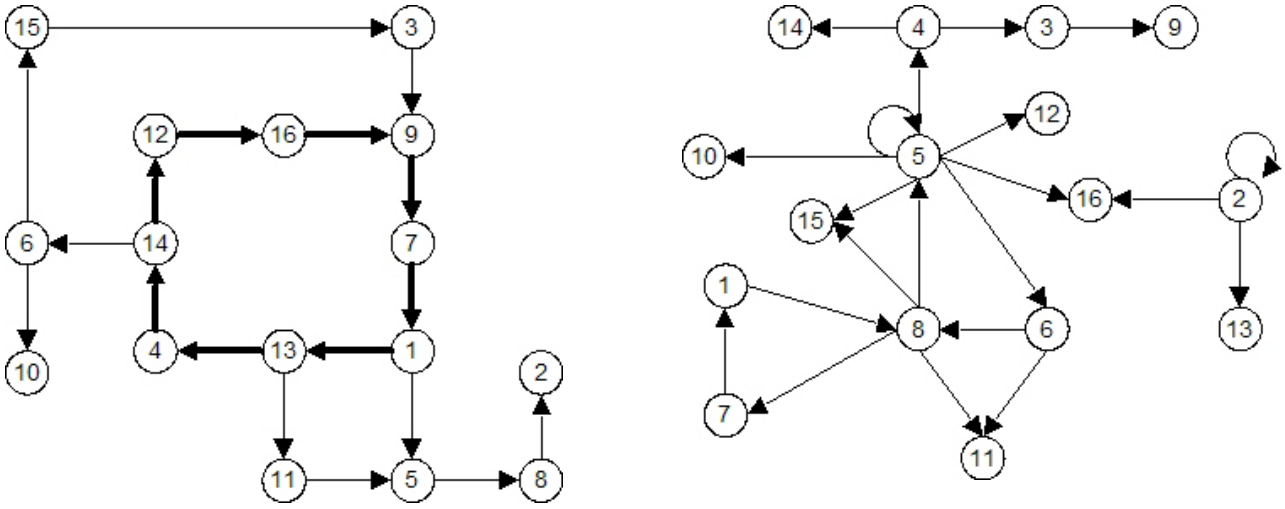
Fig. 13: De-correlated network with n = 16, k = 2. (a) Autocorrelation; (b) functional network.

De-correlated networks usually drain their inputs from one half of the nodes such that about 50% of nodes has an out degree of 0. The above architecture illustrates this and the lack of circles (except [1,5,6]) suggests the total lack of rhythmic behaviour. This gives another strong indication towards the origin of rhythm as pseudo-rhythmic networks seem widely distributed

(every node is identical in the chain architecture) whereas de-correlated networks are localised by focusing on a sub-group of nodes and often a single node has a very dominant out-degree. Also, the Boolean functions of a node seem to coincide with one of the columns of the truth table more frequently and allows decoupled units of nodes normally not found in rhythmic networks. Nodes are often connected in both ways, one being the input to the other and vice versa. Such local coupling is purely local and necessarily leads to chaotic long term behaviour.

It might be helpful to have some quantitive measure on the distribution of Boolean function to verify whether or not some input pairs occur more frequently than statistically likely. Such a measure can be easily obtained but would not serve its purpose as the Boolean functions of the actual node needs to be considered to the same degree as the inputs. In other words, if the Boolean function of a node is identical to one of the columns in the truth table, then it is already indicated that this input will be determined no matter what the configuration of inputs is. Therefore it should not be expected that certain groups of function occur more frequently than others in general; this is in accord with the analysis of several networks, not showing any statistical irregularities. A more sophisticated approach may be able to establish a mathematical relationship for the distribution of Boolean functions in pseudo-rhythmic networks.

Another valuable comparison can be made between the synchronous and asynchronous networks which both have a period of n and amplitude of 1. The topological difference is striking: The functional architecture of the synchronous network has significant similarities with the de-correlated architectures of the asynchronous case. The reason for this has not been concluded.



(a) (b)

Fig 14: Networks with period n and amplitude 1: (a) asynchronous network and (b) synchronous network

A plate of further graphs may be found in the appendix.

The emergence of rhythm

Evidence has been brought forward to suggest the use of local time keeping in its simplest form of the chain architecture. Further support includes:

- Pseudo-rhythmic networks have rapid transients. This could be due to the target function used in the GA which does favour networks of rapid transients. Nevertheless, as Harvey and Bossomaier (1997) pointed out, the asynchronous updating scheme produces the shortest transients. The short transients would correspond to the phase in which the network establishes the relational order among nodes. Once each node has been updated, it is in the correct relational state to its inputs.

- It has been shown (Di Paolo, 2001) that the basin of attraction covers most of the state space if a loose attractor exists. This could be explained by the fact that almost any initial state can be 'converted' to obey the relational properties imposed by the Boolean interaction. The relationships between nodes, as indicated by the output (the patterns produced are reflections of the relational attributes of the functional architecture) are always similar, independent of the initial state indicating the existence of a single attractor.

- The diversity of output patterns produced by the distinct networks given can easily be explained. As has been shown with the graphs of pruned networks, a single link may suffice to dictate the state of the connecting node, either identical or opposite. Given a network of size n, there are n effective links, each of which can take on one of the two labels. This allows for 65534 different output patterns (given $n = 16$) given the exact same topology but different Boolean functions. In addition, the evolved networks are all approximations which increases the number of possible topologies by magnitudes. On the other hand, the relatively small number of possible networks can be used to explain other findings made by Di Paolo (2001). Di Paolo investigated the widespread of rhythmic attractors among the class of non-stationary attractors and found that rhythmic attractors are much less frequent.

If the emergence of rhythm is purely topological, the evolved networks should be robust towards different sequences of updates. It can be shown that this is the case: Each time step (n-updates), some of the active nodes change their values. In the deterministic case, this leads to a sequence of changing nodes which repeats itself each period. Random asynchrony is non-deterministic and the order of nodes chosen for update differs each time step. If there are a active nodes and c changes per state transition (where $c \leq a$) then there are a possible a^c different combinations of update sequences. This is the upper bound as every active node is considered every time step which is normally not the case. If, for example, a network has 3 active nodes and typically 2 changes per state transition then there are $3^2 = 9$ different combinations. Assume nodes 1, 2 and 3: 11, 12, 13, 21, 22, 23, 31, 32, 33. The length of the sequence increases exponentially in a and c. Some sequences will be redundant once higher numbers of active and changing nodes are encountered as a sequence of odd length involving only one node is identical to a single update: 1 = 111 = 11111, etc.. The average number of changing nodes has been used to this measure as exploiting all possible sequences is clearly computational infeasible (E.g. $a = 12$ would theoretically imply 12^{12} different sequences).

The network is executed as usual but at each state transition within the cycle, all possible update sequences are applied to the same state and their individual outputs are collected. The measure has been applied to 10 pseudo-rhythmic networks and it has been found that at each state, the likelihood of diverging onto different states is close to 0. Almost all states throughout the cycle are very robust with some exceptions where a state may diverge onto several distinct successors. This measure provides strong evidence for the purely structural emergence of rhythm. The likelihood of divergence as indicated by the Hamming distance for the run of 5 networks is shown in table 11.

1	#	2	#	3	#	4	#	5	#
19517	**0**	17828	**1**	52797	**0**	28147	**1**	9656	**1**
19493	**0**	21925	**2**	20029	**0**	58739	**1**	1464	**1**
23589	**0**	21669	**1**	20025	**0**	37235	**2**	1336	**0**
56357	**1**	46245	**0**	20027	**0**	37171	**0**	4920	**0**
56421	**0**	46693	**0**	18043	**2**	37683	**0**	21304	**0**
64868	**0**	46691	**0**	18267	**1**	5043	**1**	21052	**1**
47520	**1**	46658	**0**	17243	**1**	7091	**0**	21044	**0**
10680	**0**	34370	**1**	17235	**1**	15283	**0**	53812	**0**
10425	**0**	50754	**0**	50134	**2**	27571	**1**	65205	**1**
2233	**0**	17540	**1**	50116	**2**	28659	**1**	61109	**1**
		54437	**1**	54220	**1**	28023	**0**	60597	**0**
		62693	**1**	52909	**1**	60791	**1**	44213	**0**
		46177	**1**	20153	**1**	50551	**0**	44469	**2**
		46689	**0**	20155	**1**	34167	**0**	9657	**2**
		46659	**0**	18259	**1**	38199	**1**		
				22355	**0**	37175	**0**		
				54099	**0**				
				54212	**1**				
				54148	**1**				
				52140	**1**				
				52908	**0**				

Table 11. The number of additional states that may lead to divergence of the loose attractor (the maximum Hamming distances has been 1 in all cases).

An example of this procedure is given in figure 15 which shows the states reached after all different update sequences. Only 2 states differ from the initial state, one of which is divergent.

Mean number of changing nodes: 1
Number of active nodes: 12
Initial state ID: 63780 (111101000010)

Outcome of all combinations:

111101000010
111101000010
111101000010
111101000010
111111000010
111101000010
111101100010
111101000010
111101000010
111101000010
111101000010
111101000010

SCORE: 1

Fig 15: Possible outcomes of pseudo-rhythmic network after all possible update sequences.

It should be noted that this is merely an approximation and results vary from trial to trial but the average number of divergent states is normally constant. In cases with more active nodes, it frequently happens that the same state may be reached by several update sequences. The relationship between the number of active and changing nodes does illustrates that a low

number of changing nodes only allows a low number of possible update sequences and therefore reduces the chance of diversion.

The fact that only one architecture has been identified so far that seems suitable for topological constraints robust enough to withstand the non-determinism of the update scheme, allows for the formulation of generic properties of pseudo-rhythmic networks. If each node has identical frequencies with change of a single node per state transition, only a period of $2n$ is possible. After each node has been updated once, the network reached its inversed identity and it takes another n changes, to re-encounter the initial state. Shorter periods may be obtained by changing a larger, but still uniform number of nodes at each time step. If, for example, two nodes are changed during each transition, the period is cut in half. This, however, degrades the degree of rhythm unless individual nodes are conditioned on at least two distinct nodes. Another way to achieve shorter periods is by using stationary nodes: if s nodes are stationary, the network is effectively reduced implying a cycle of n-s nodes. Longer periods are much more difficult to achieve: the sequence of changes is more complex, with some nodes changing multiple times over a short period of time whereas other nodes change during subsequent time steps. It should also be pointed out that the size of the network is important as larger networks are generally more robust given the lower probabilities of state diversion.

Other Studies

It is important to see how this framework copes with data obtained in other studies. There are only four studies known to the author that deal with random asynchronous networks, only two of which deal with pseudo-rhythm explicitly.

- Di Paolo (2000) evolved networks of size n = 16, 32, 64 with k = 2, 3, 4 and target periods $p = n/2, n, 2n$. Di Paolo reports that shorter or longer target periods have been attempted only with minor success. He used the target function [0, 1] which, given the identified structure, implies almost half the nodes being stationary while the active nodes change one at a time. For the sake of simplicity, let us assume that 50% of all nodes are stationary and exactly one node changes at a time. This is natural rhythm of n. The smaller period may be achieved by changing two nodes at a time, probably involving $k > 2$. The longer period of $2n$ is more difficult and probably involved higher values of k. In any case, it has not been possible to produce longer rhythms for the reasons given earlier. Shorter rhythms would be possible but not given the amplitude of 1 (by the target function).

- Another study from Adams (2002) includes a table (see table 12) which in part indicates the relationship between stationary nodes and fitness of the network (using the [0,1] target function, networks $n = 16$, $k = 3$). There is a correlation between the number of stationary nodes and the fitness of the network. It is estimated that little less than half the nodes are stationary in an optimum network, and the data indicates the tendency towards this.

Network	Fitness	# stationary nodes
1	65.0%	4
2	64.1%	5
3	63.6%	3
4	61.3%	3
5	60.8%	2

Table 12: A summary of a table given by Adams (2002) of evolved ARBNs with increasing fitness and number of stationary nodes.

- The fitness of an asynchronous pseudo-rhythmic network is proportional to its similarity with the synchronous updating scheme as indicated with the network's timeline. This complies well with the findings by Gershenson (2002) who concluded that the differences between network behaviours arise from the degree of determinism rather than the update

scheme itself. A carefully constructed architecture that locally keeps track of time is in itself deterministic. Therefore, the higher the degree of coupling given certain values of n and k, the networks behaviours will become more and more similar in both a determined and non-determined update scheme.

Rhythm and the degree of coupling

It has been shown that the natural period of a chain architecture is $2n$ if all nodes are active. Shorter periods are achieved by the use of stationary nodes or multiple changes per state transition. In any case, there is a relationship between period, amplitude and active nodes which should be applicable to all networks such that similar rhythms will exhibit similar graphs of rhythm. Longer periods are more difficult to obtain as they require more sophisticated coupling using higher values of k. As k increases, however, Boolean functions and tables become much more complex. While it is believed that k is proportional to the amount of coupling that may be achieved it is shown that this is exponentially unlikely to occur. Many different topologies are imaginable: consider a network with a large chain of nodes, a smaller chain of nodes and a sufficient amount of feedback links at appropriate locations. The smaller chain traverses normally and thereby dictating the progress made by the larger chain using the feedback links. Given variations in the size of these two chains, different rhythms could be obtained. High values of k could also lead to networks of short rhythms without the typical characteristics of the chain architecture. That these networks are unlikely to emerge can be demonstrated with a simple experiment which evaluates the space of non-stationary attractors for different values of k.

Di Paolo (2001) used a target correlation of [0] to evolve de-correlated asynchronous networks to get an idea about the distribution and quantity of pseudo-rhythmic attractors in the space of non-stationary attractors for ARBNs. Di Paolo showed that de-correlated networks could be evolved much faster, suggesting that they occupy most the attractor space. This measure can be used to examine the distribution of de-correlated attractors proportional to k as the size of the state of non-stationary attractors is fairly constant for any k (Harvey and Bossomaier, 1997; this project). If de-correlated attractors are easier to find as k increases would suggests that sophisticated rhythmic ARBNs are very rare. This experiments used identical settings for the GA throughout the trials, using the elite selection scheme as deemed most appropriate for this comparison. The trial is aborted as soon as a network reaches the fitness score of 0.92.

k = 2		**k = 3**		**k = 4**	
18	0.920	**20**	0.920	**6**	0.93
30	0.921	**6**	0.922	**11**	0.921
14	0.924	**24**	0.920	**3**	0.923
15	0.921	**10**	0.926	**4**	0.920
16	0.923	**13**	0.922	**10**	0.920

k = 5		**k = 6**	
10	0.921	**2**	0.922
4	0.921	**1**	0.920
4	0.921	**2**	0.920
3	0.924	**1**	0.926
6	0.920	**3**	0.925

Table 13: The number of epochs needed to evolve a de-correlated network with at least a score of 0.92 and the score reached.

Comparison:

k	**2**	**3**	**4**	**5**	**6**
mean	18.6	14.6	6.8	5.4	1.8

The data clearly shows that de-correlated attractors populate an area of growing proportions in the space of all non-stationary attractors and thus provide good support for the assumption made earlier. Also, it had been shown that the transients of point attractors is much longer for higher values of k indicating that any stable behaviour is more difficult to establish with an increasing number of connections. It is striking to see that the number of de-correlated attractors is seemingly higher for $k = 3$ than it is for $k = 2$; it had been shown earlier that $k = 3$ has the largest space of non-stationary attractors but it seems that $k = 3$ is the value most suitable for de-correlated and not pseudo-rhythmic networks. As had been shown, $k = 2$ suffices to produce pseudo-rhythmic behaviour and it is assumed that this is merely due to the simpler space of topologies.

Biological plausibility

The fact that long cycles ($> 2n$) seem difficult to obtain is of little relevance if ARBNs were to be used as models of biological system as a cycle length of $2n$ is too great. As a matter of fact, it was the existence of small cyclic attractors has given reason to use Boolean networks as biological models in the first place. Kaufman (1969) showed that synchronous random Boolean networks with $k = 2$ can be used as models for genetic regulatory networks. First, the number of genes that makes up a cell in the human body is estimated 100 000 the root of which is approximately 300, more or less the number of cells in the human body. Also, the typical cycle length of $\sqrt{n}$ can be used to predict cell replication times. The size of the cycles is extremely small given the number of possible states the network may traverse through, especially considering networks of size more than 100 000 nodes. Kauffman did report on the high number of stationary nodes (approx. 70 %). This would not work in the case of asynchronous networks. It has been shown that a low k seems essential for pseudo-rhythmic networks to exist in the first case. A low k, however, is bound to period and amplitude requiring a larger amount of stationary nodes if small cycles are to be obtained. Also, having multiple nodes changing per state transition is not an option as anything more than two changing nodes would loose rhythm unless there are enough feedback loops. At the beginning it has been stated that one of Kauffman's reason to believe that genetic regulatory networks are random is the careful evolutionary selection required for topologically refined nets to emerge. If one believes that this evolutionary process is no more unlikely than a perfectly synchronous updating scheme, random asynchrony may be used as model for these networks even though it seems, they do not fit the data projected by Kauffman.

It is also biologically plausible that the emergence of rhythm is uniformly distributed among the nodes. Such a distribution normally ensures robustness towards outside perturbations but the chain architecture itself is very brittle (see figure 16) as the lesion of a single node has devastating effects. Therefore it is interesting to note that the evolutionary process has produced networks which only approximate the chain architecture to some degree. These networks are much more stable towards outside perturbation. A comparison to synchronous networks may be valuable also.

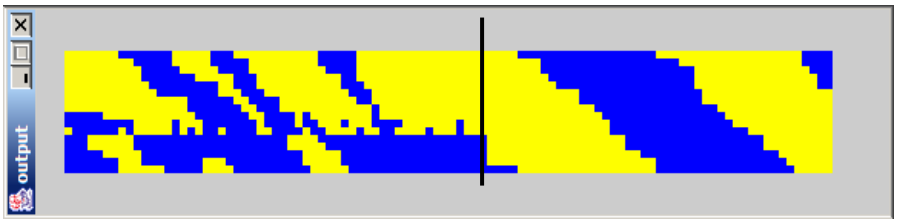


Fig. 16: A perfect chain architecture has a single node lesioned (marker), showing the transition from order to chaos (output goes from left to right).

Conclusion

This work suggests the origin of rhythm in ARBNs as the result of well defined topological relations between nodes. Ultimate proof can not be obtained from the small subset of networks tested and a larger quantitive analysis is needed. Only further verification and mathematical proof can determine the validity of the results obtained, but the data gathered in this project looks promising. Larger networks need to be tested and the relation between attractor-space and rhythm needs to be analysed. The most important results of this project are as follows:

- Rhythm seems to emerge from the topological organisation of the network whereas tight coupling between nodes establishes order.

- The chain architecture has been identified as the single most important structure so far given low values of k.

- An algorithm has been identified that can be used to prune the network and uncover the functional architecture of the network.

- The value of k dictates the amount of coupling in the network. High values are thought to allow the emergence of any rhythmic behaviour. Low values seem to allow any rhythm equal or below a period of $2n$. These networks have a functional relationship between their period, amplitude and stationary nodes.

- More versatile networks are unlikely to emerge given the constraints upon the value of k: the number of rhythmic networks decreases as k increases. Chaotic de-correlated attractors occupy almost the entire space of non-stationary attractors.

Future work includes automatic construction of asynchronous networks displaying any degree of rhythm. The software suite is modular and should be extended to cover all update schemes as identified by Gershenson (2002). This has not been done to date given time constraints but could reveal interesting properties about other classes of networks. As the analysis of complex dynamical system is often more successful on an observational basis, more sophisticated graphical tools would be a useful addition. Especially modes of real time interaction, such as lesioning, could prove useful in generating the overall picture needed to conclude about the usefulness of ARBNs as biological models. The biological relevance of ARBNs may be judged upon the results but is a matter of its own. One has to decide if a carefully selected topology is more plausible than a perfectly synchronised updating scheme.

References

Adams, J. (2002). On asynchronous random Boolean networks. Submitted as dissertation for MSc EASy, University of Sussex, Brighton, UK.

Aharonov, R., Segev, L., Meiilijson, E., Ruppin, E. (2003). Localisation of Function via lesion analysis. Neural computation, 15(4), 885-914.

Di Paolo, E. A., (2001). Rhythmic and non-rhythmic attractors in asynchronous random Boolean networks *BioSystems*, 59(3), 185 - 195.

Di Paolo, E. A., (2000). Searching for rhythms in asynchronous Boolean networks. Artificial Life VII: The Seventh International Conference on the Simulation and Synthesis of Living Systems, Reed College, Portland, Oregon, USA.

Gershenson, C. (2002). Classification of Random Boolean Networks. In Standish, R. K., M. A. Bedau, and H. A. Abbass (eds.) *Artificial Life VIII: Proceedings of the Eight International Conference on Artificial Life.* . pp. 1-8. Sydney, Australia. MIT Press.

Harvey, I., Bossomaier, T., (1997). Time out of Joint: attractors in asynchronous random Boolean networks. In: Husbands, P., Harvey, I. (eds.), Proceedings of the Fourth European Conference on Artificial Life. MIT Press, Cambridge, MA, pp. 67 – 75.

Kauffman, S. A. (1993). The origins of order. Oxford University Press.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. 22, 437-467.

Nowak, M. A., May, R. M. (1992). Evolutionary games and spatial chaos. Nature, 359:826-829.

Wuensche, A. (1994). The ghost in the machine: basins of attraction of random Boolean networks, in Artificial Life III, Langton (ed.), SFI Studies in the Sciences of Complexity, Proc. Vol. XVII, Addison-Wesley.

Appendix

Log

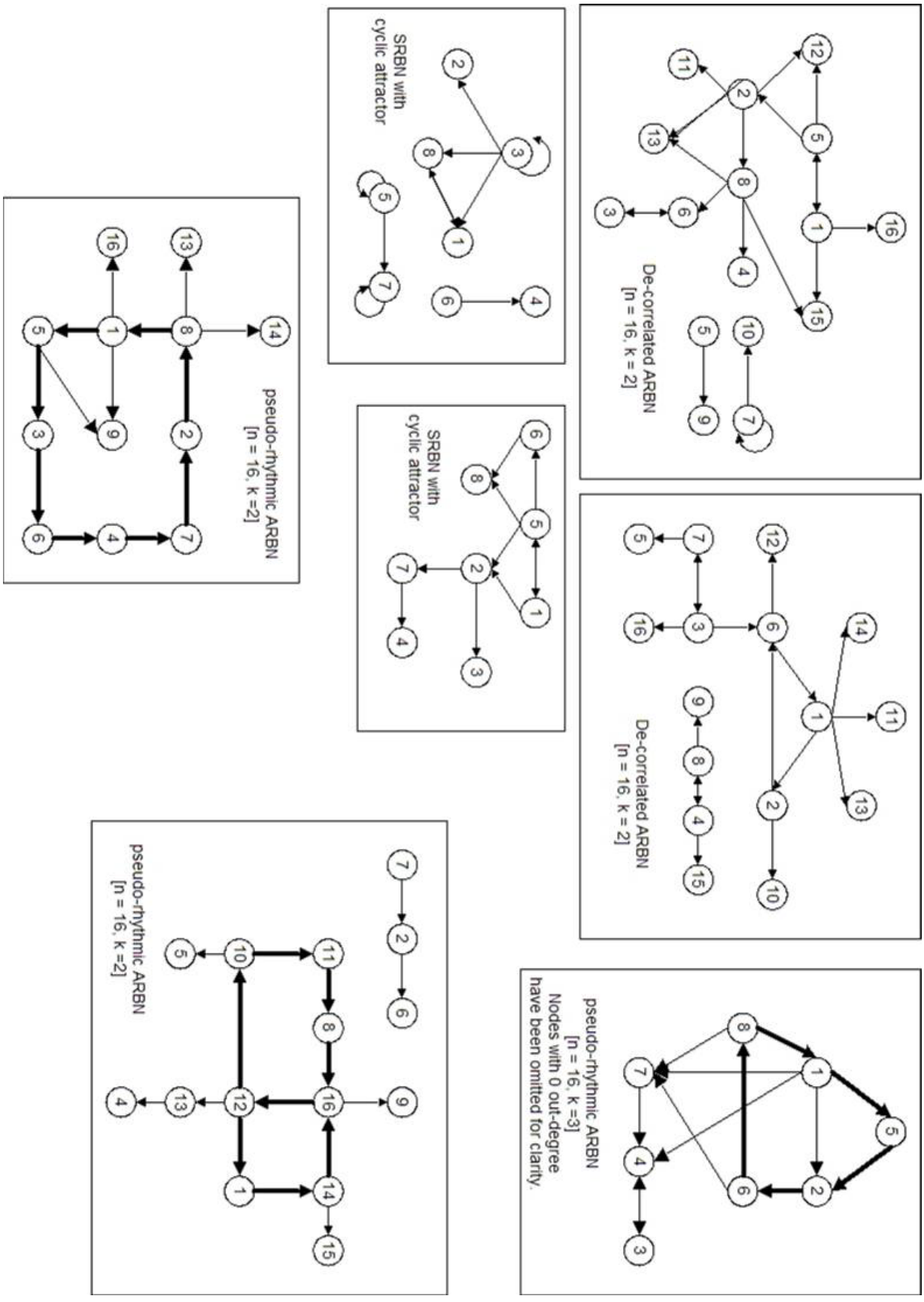
Autumn term

1	Initial discussion and clarifications. Also looking at other options.
2	First refinement of the task. Initial outline and clarification of some details.
3	Discussion of basic program which should be completed within a week. Basic program will simply simulate RBNs and ARBNs (3 different update methods in total)
4	Doing analysis on the data obtained from the program to verify correct workings.
5	Analysis of Initial data shows flaws in the number of attractor found in ARBNs. Task for next week is to debug the program and repeat analysis.
6	Program is working. Talk about implementation details. Proposal of lesion analysis; 2 papers by Ruppin et al are suggested and to be prepared for the next meeting.
7	Final decision on implementation detail: lesion analysis for ARBNs with evolved pseudo-periodicity. Estimate of first executable version beginning of spring term.

Summer term

1	Examining the data produced over the Winter break as the genetic algorithm has been fully implemented. The data seems to indicate that the program works.
2	Attempting a first approach to implement the lesion analysis after some discussion of the details as indicated in the paper by Aharonov et. al (2003).
3	Lesion analysis will be approximated using a simpler contribution analysis given the complexity of the mathematical framework required to properly implement a multi-lesion analysis.
4	Discussion of the contribution analysis and the incorporation of fitness. Results are not yet very good.
5	Changes are made to the contribution analysis. All techniques should be applied to a single network.
6	Hypothetical network is discussed. More focus on statistical analysis. Contribution analysis showed better results when compared to single lesion analysis.
7	Boolean influences are discussed.
9	Discussion of biological relevance and circadian cycles. Further applicability and work over the Easter break.

Functional architectures



Network contribution analysis

This contribution analysis was the first attempt to discover the origin of rhythm in ARBNs. It has proven successful only to a limited degree as comparisons with lesioned networks revealed. The approach used in the pruning algorithm has shown to be more effective despite the fact that no numerical measure is produced. This approach has been included for reasons of completeness:

Each network consists of nodes which directly and/or indirectly influence each other's output as long as there is a path between them. If a network consists of several isolated connected components these do not interact with each other. Connections imply direct influence, otherwise 2^{nd}, 3^{rd}, etc. degree of influence. The network may be visualised as a weighted directed graph, the weights being the direct influences calculated as shown below.

Step 1: Calculate the information content of each node's Boolean function

$\sum_{i=1}^{n} - P(v_i) \log_2 P(v_i)$, according to the occurrences of 1's and 0's in the Boolean function. It follows that the information content of each node falls in between the range [0..1] where 0 indicates the output always being the same, 1 being an equal (50/50) distribution.

Step 2: For each node calculate the direct influence on neighbouring nodes

This measure is simply the number of co-in-siding input/output pairings. For example, 6/8 cases the input from node a (to node b) and the output from node b are the same, then the influence is simply 6/8. Equal or opposite pairings count equal as they equally constitute towards a gain in the uncertainty of the nodes outcome. It therefore follows that the lowest influence is ½ and the actual influence is thus linearly scaled such that ½ = 0: 2*x-1. The influence therefore ranges between 0 and 1.

Step 3: Normalise and adjust influences

The influences on each node should add up to the information content of that node. If a node has information content of 1 then the influences of all incoming nodes should add up to 1. If, however, the information content of a node is below 1 then the influences add up to that amount. This implies that the influences are relative to each other and also that the remaining influence is coming from the node itself despite there not being a connection. If, for example, a node has a uniform output, then, there can't be any influence as it does not affect the node's output in any way. Thus the output is entirely determined by the node itself, expressed implicitly within the Boolean function.

Step 4: Adjust the contribution values to actual outputs of the network

This step introduces how the contributions scale up to the behaviour of the network itself. The behaviour of any network is merely the output it produces. The output of a network is analysed as follows: Whenever a node changes, all the other nodes are checked whether or not they will change the next time step. This is a notion of induced changes; the contrary inhibitory behaviour is not of interest as will be shown later. It therefore follows (for example) that if node 1 influences node 2 by 0.4, say, and node 1 also causes node 2 to change for 80 times, say, then the influence is 0.4x80=32. This is done for all nodes. This measure captures and incorporates the network's behaviour into the measure of analysis.

Step 5: Perform a modified and simplified form of transitive closure and adjust contribution values

The idea here is that if node 1 influences node 2 which in turn influences node 3 then there is an influence of 2nd degree of node 1 on node 3. That is also the reason that inhibitory behaviour can't be used to adjust the contribution values to the network behaviour: it logically follows that higher degree influences result in later time steps during the network's output and do not require the original node to change while its influence percolates through the network. Therefore, if node 1 changes but 3 doesn't and the other way around, that does not imply that node 1 actually has an inhibitory effect on 3 but could imply that node one's influence percolates through the intermediate connections. To get an estimation of these indirect effects a sort of transitive closure is performed. This could be arbitrarily deep (depending on how many 'layers' are implied by the network's topology). So if node 1 is linked to 2 which in turn is linked to 3 then the influence of 1 on two is multiplied by the influence of 2 on 3 (since all influences are <1 the value degrades as it should) which is then added to the existing influence (if any) of 1 on 3.

Step 6: Finalising the contribution values

Now these values are just added together and normalised to give a single contribution value for each node (whereby all contributions add up to one).

Classifications of RBNs (Gershenson, 2001)

Following the results by Harvey and Bossomaier (1997), Carlos Gershenson provides a classification of RBNs and outlines the differences resulting from the choice of update method. He points out that the concept of determinism and non-determinism causes greater differences in the behaviour of RBNs than the updating scheme itself. All Boolean networks have strong similarities (discrete time, space and values) and can be summarised under the term Discrete Dynamical Networks (DDN), a term introduced by Wuensche (1997). As only the update methods and implications of determinism are important to this project a brief summary is given which only outlines theses aspects:

Network	Updating scheme
Classical Random Boolean Networks (CRBNs)	synchronous, deterministic
Asynchronous Random Boolean Networks (ARBNs)	asynchronous, non-deterministic
Deterministic Asynchronous Random Boolean Networks (DARBNs)	asynchronous, deterministic
Generalised Asynchronous Random Boolean Networks (GARBNs)	semi-synchronous, non-deterministic
Deterministic Generalised Asynchronous Random Boolean Networks (DGARBNs)	semi-synchronous, deterministic

Java Code

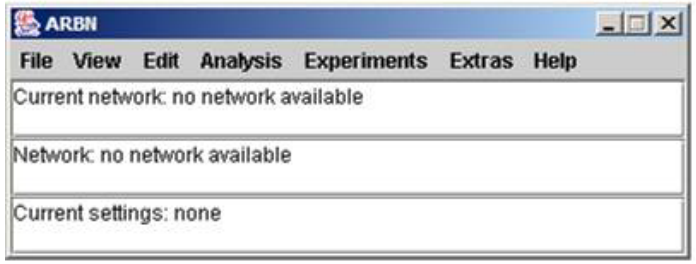
The software suite has been developed as a tool of analysis. This is an ongoing process as new insights require the addition of new tools. The structure is modular and extensible, but some code should be rewritten for better performance. Some user specific aspects are not fully functional but these only include elements such as disabling fields or displaying information. In some cases, results as mentioned in the report are output directly to the command window. Some sections are commented out in the code below and will be incorporated in full in future versions. Also, statistics generated can not yet be saved directly. The ongoing process of research and software extension is sometimes difficult to follow and a full implementation of a new module requires extensive testing afterwards, especially in the case of a software tool as such because minor errors easily result in significant errors. A full manual has yet to be written but there is no need for such unless the software is made publicly available which might the case for future versions. Once again it is stressed that the primary purpose of the software is for the purpose of analysis and not a tool for a wide group of users. The code for the file filter and PCU monitor are not included as they are taken from Sun's demonstration libraries. Further it is pointed out again that some aspects have been inspired by the software suite provided by Gershenson (2002) but the code has been independently written. To get an idea of how the program and especially the visualisation aids look, some screenshots are included over the next two pages followed by the annotated Java code.

Classes in alphabetical order:

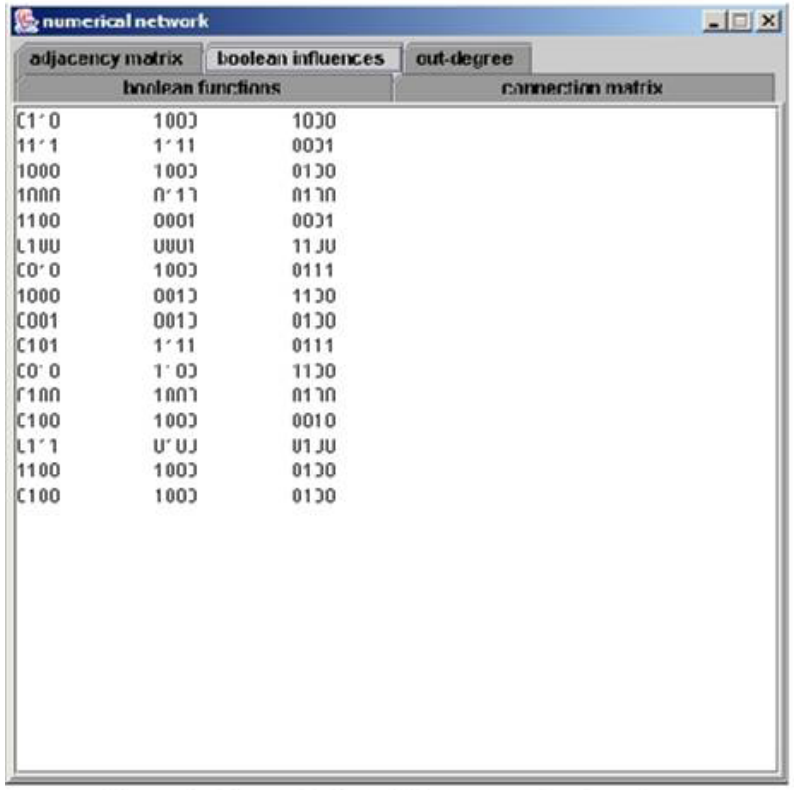
Analysis.java	GAInterface.java
AnalysisInterface.java	GaRun.java
Attractor.java	Help.java
CADisplay.java	InfoMsg.java
Clock.java	Net.java
Controller.java	NetworkInterface.java
Correlation.java	NetNumerical.java
EditNetInterface.java	NetworkOutputDisplay.java
ErrorMsg.java	PseudoRhythm.java
ExpActivity.java	RBNLabs.java
ExpPointCycle.java	RhythmInterface.java
ExpSyncAsync.java	Saver.java
ExpTransient.java	Score.java
Experiments.java	UpdateHistogram.java
ExperimentsInterface.java	Utils.java
GA.java	

Not included:

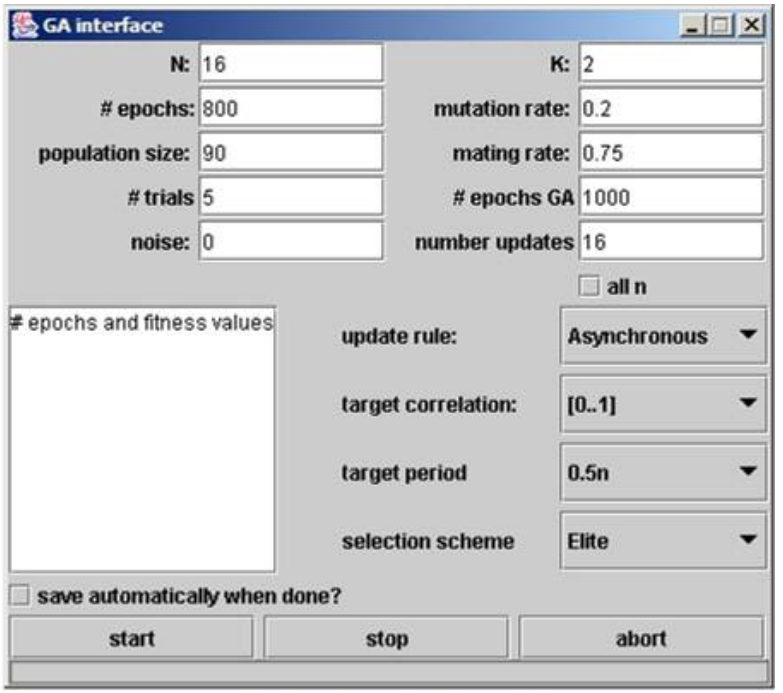
RBNFileFilter.java (borrowed from Sun's demonstration classes)
CPUMon.java (borrowed from Sun's demonstration classes)



Main window from which everything is reached

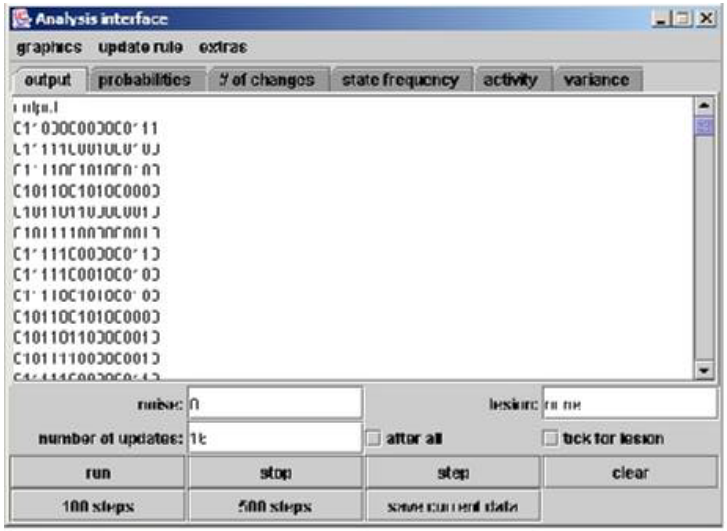


Numerical description of the current network



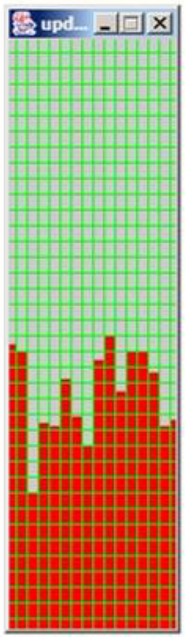
Genetic algorithm interface

Network interface



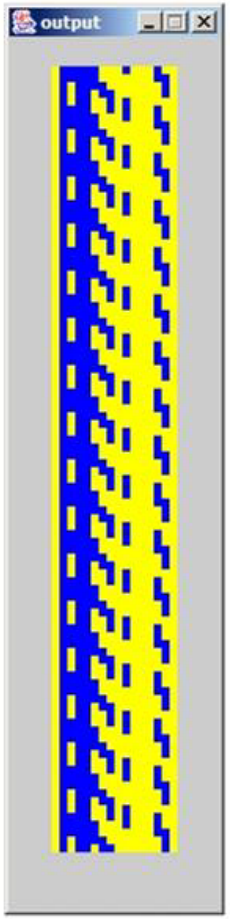
Analysis interface

Experiment interface



Update histogram

CA-like output



Network's visual output display