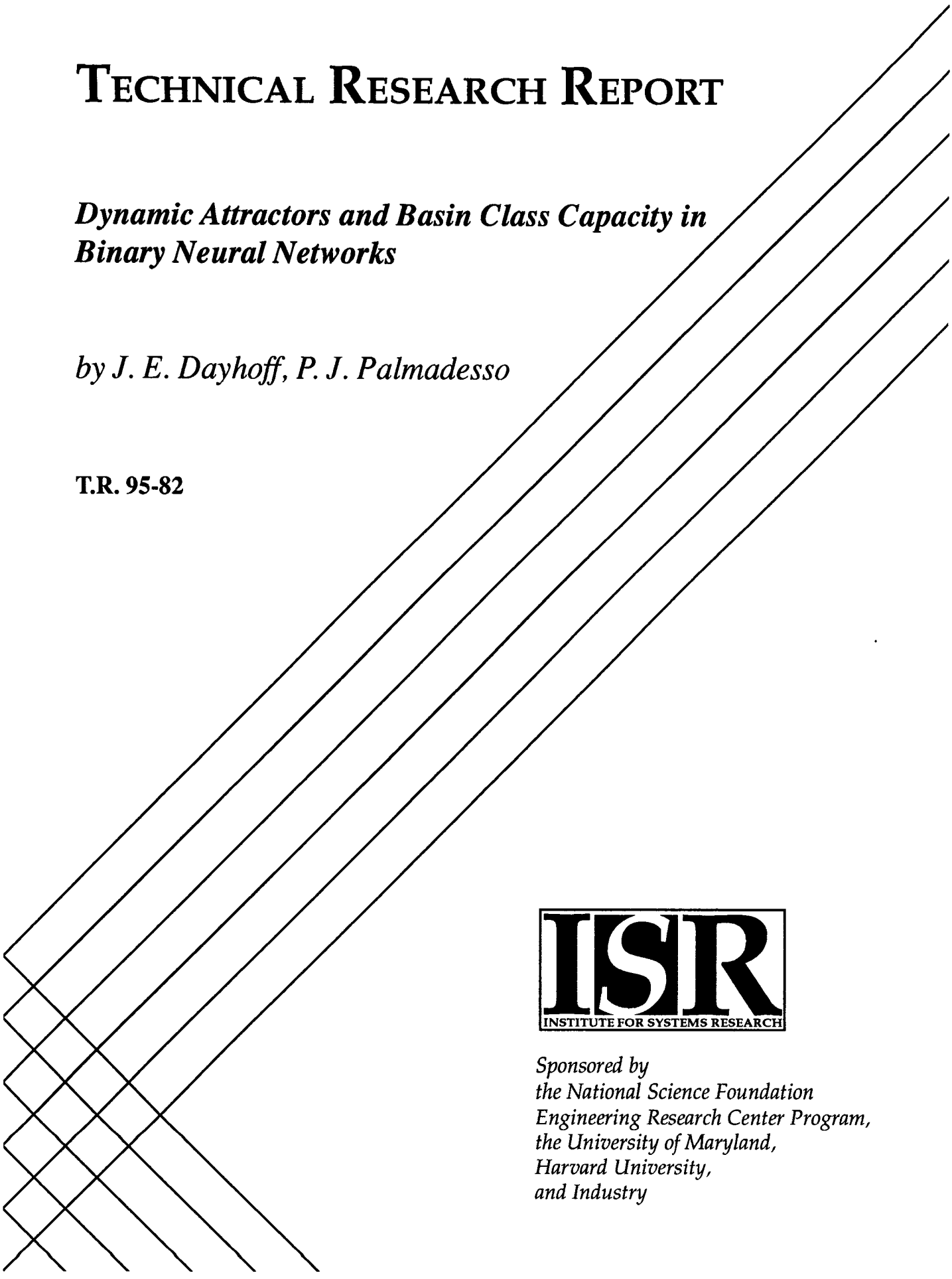
TECHNICAL RESEARCH REPORT

Dynamic Attractors and Basin Class Capacity in Binary Neural Networks

by J. E. Dayhoff, P. J. Palmadesso

T.R. 95-82



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

DYNAMIC ATTRACTORS AND BASIN CLASS CAPACITY IN BINARY NEURAL NETWORKS

Judith E. Dayhoff
Institute for Systems Research
University of Maryland
College Park, MD 20742

Peter J. Palmadesso
Plasma Physics Division
Naval Research Laboratory
Washington, D.C. 20375

December 21, 1994

Abstract

The wide repertoire of attractors and basins of attraction that appear in dynamic neural networks not only serve as models of brain activity patterns but create possibilities for new computational paradigms that use attractors and their basins. To develop such computational paradigms, it is first critical to assess neural network capacity for attractors and for differing basins of attraction, depending on the number of neurons and the weights. In this paper we analyze the attractors and basins of attraction for recurrent, fully-connected single layer binary networks. We utilize the network transition graph - a graph that shows all transitions from one state to another for a given neural network - to show all oscillations and fixed-point attractors, along with the basins of attraction. Conditions are shown whereby pairs of transitions are possible from the same neural network. We derive a lower bound for the number of transition graphs possible, 2^{n^2-n}, for an n-neuron network. Simulation results show a wide variety of transition graphs and basins of attraction and sometimes networks have more attractors than neurons. We count thousands of basin classes - networks with differing basins of attraction - in networks with as few as five neurons. Dynamic networks show promise for overcoming the limitations of static neural networks, by use of dynamic attractors and their basins. We show that dynamic networks have high capacity for basin classes, can have more attractors than neurons, and have more stable basin boundaries than in the Hopfield associative memory.

1 Introduction

Dynamic neural networks have sustained activity patterns that arise from the network's dynamic attractors - finite state oscillations, limit cycles, and chaotic attractors. These dynamic attractors show potential to ultimately be used for computational purposes. In the Hopfield Network, fixed-point attractors were used to represent memory patterns, and when the network attained one of these fixed points (stable states), the network's activation state was read off as the "memory" reported by the network. In dynamic networks, attractors are not limited to fixed point stable states, but can be simple oscillations, limit cycles, or chaotic behavior, as well as fixed points. These dynamic attractors may be used eventually for applications in associative memory, pattern classification, pattern completion, signal generation, and other domains.

We aim to eventually use and exploit the dynamic properties of the network's attractors, and the possible increased flexibility that dynamic attractors are expected to offer. Towards this end, we must first explore what are the attractors and basins of attraction of dynamic networks under various circumstances. It will be key to know how many different sets of attractors and basins are possible from neural networks of a given size and configuration, and to discover what degree of control can be exerted over basin shape. Ultimately, one could set up applications paradigms whereby the attractor that the network enters provides a pattern class, a memory, or other information bearing on the application.

In this research we have taken a different approach compared to Hopfield's network (Hopfield, 1982). In this paper, we relax the constraint that attractors be fixed points (stable states) and include architectures that readily organize dynamic non-fixed-point attractors. Thus we welcome oscillatory states as attractors. We allow asymmetric reciprocal weights ($w_{ji} \neq w_{ji}$), removing the Hopfield constraint for reciprocal equality. We allow synchronous updating of neurons as opposed to asynchronous updating, used

in convergence proofs to attain stable states. In the Hopfield Net, where usable attractors are only fixed points, there are at most about $0.15n$ reliable memories for n neurons (McEliece et al, 1987). Because large numbers of dynamic attractors are possible, exploiting dynamic neural networks shows promise for increased capacity. In a dynamic network the number of attractors can be greater than the number of neurons. Furthermore, increased numbers of basin classes - networks with differing basin boundaries - are expected with dynamic attractors.

Dynamic networks include fully and sparsely connected configurations as well as binary and continuous-valued units. We chose to analyze binary networks to simplify a problem that could otherwise be mathematically intractable. We evaluate how many attractors are available in each neural network, and address the characteristics of these attractors and the boundaries of their basins of attraction. We count many different possibilities for transition graphs and for basins of attraction.

We examine whether a high capacity for attractors results, and whether there are more possibilities for the placement of the boundaries of the basins of attraction. Since we ultimately would like to set or adapt these boundaries, here we address the question of how many different possibilities for boundaries exist. We have found that enormous numbers of attractors and basin boundaries are possible, even with small networks. Thus there is considerable freedom to design basins to order, e.g. to have "designer basins".

We introduce the concept of "basin class capacity" - the capacity of a set of neural networks to exhibit a variety of different basin classes. For example, the sets of neural networks considered here are binary networks with the same number of neurons (n) and fully connected configurations without self-loops. The number of different basin classes that can be generated from such a set would be its basin class capacity. A high basin class capacity opens the possibility for a set of neural network paradigms to be developed that train basin boundaries to desired specifications.

Since a basin of attraction is the set of states that eventually lead to the same attractor, it is important for computational purposes to be able to control or adapt the boundaries of basins of attraction. In Hopfield memory calculations, there is no control over the shape or boundaries of the basins of attraction. Furthermore, the random updating procedure leads to unstable boundary locations. Computation for weights assigns the appropriate "memory" state as a fixed point attractor, but does not adjust its basin boundaries. The lack of adjustment of boundaries is a glaring limitation.

For many prospective applications, adjustment of basin boundaries could be more important than the actual attractor in the basin. The basin boundaries determine which attractor the network goes into, and the attractor would represent the answer, result, or memory recalled from the neural network's computation. Thus, the exact nature of the attractor (fixed or oscillating) and its location (particular state(s) involved) can be less important than the basin boundaries.

We aim to eventually have paradigms that allow adjustment and training of attractor basin boundaries in dynamic networks. The first step towards this aim, however, is to explore how many sets of basins and basin boundaries are possible in a dynamic neural network. To this end, we have chosen a particular type of dynamic neural network and researched how to count the number of different basin boundaries possible with n neurons. The networks analyzed here are the single layer fully connected binary networks.

The individual states of these neural networks are binary vectors. Transitions from one state to another are ordered pairs of binary vectors. We define the transition graph, which has as nodes all possible states of the neural network, and as edges, transitions between states. We show how to construct a network transition graph (NT-Graph) for a neural network given a set of Neuron Transition Tables (NT Tables), which show the transitions of an individual neuron. For a network of n neurons, n such tables are needed. Each NT graph shows all the attractors and basins of attraction for a set of neural networks.

We have computed a lower bound on the number of NT graphs possible for networks of n neurons. First a lower bound on the number of NT Tables was found to be 2^{n-1}. Since n NT Tables are combined to specify an NT Graph, the lower bound on the number of NT Tables allows us to compute a lower bound on the number of Network Transition Graphs (NT Graphs). The lower bound on NT Graphs is $2^{n(n-1)}$. For a two-neuron network there are four NT Graphs and, for a three-neuron network there are 64 such graphs. For a 4-neuron network there are thousands of possibilities, and for $n = 19$ or more neurons there are over 10^{100} possibilities, greater than an estimate for the number of particles in the universe.

We have simulated large numbers of neural networks of varying sizes with random weights, and found their attractors and basin classes. Sometimes there were more attractors than neurons, especially when reciprocal weights were symmetric ($w_{ij} = w_{ji}$). The number of different basin classes was counted. Whereas fifteen basin classes were found for networks with three neurons, thousands were found for larger nets. Cases of asymmetric wieghts and symmetric weights were compared. Both showed large counts. Since asymmetric weights include the cases of symmetric weights, asymmetric weights lead to larger numbers of NT Graphs and basin classes. However in the simulation experiments sometimes more classes were observed for symmetric weights because of the more rapid sampling of the space. Symmetric weights caused more attractors and smaller basins on average. Asynchronous (random) updating was also simulated, but this paradigm leads to unstable basin boundaries unless a pre-specified ordering of neurons is used. However, even with this ordering, random weights led to only one basin and only one basin class.

In the next section we describe the structure of the binary neural networks considered in this paper. In Section 3, we give definitions, and explain transitions, transition graphs, transition tables, and basin classes. In Section 4 we give relationships about the compatability of multiple transitions in a neural network. We also prove the lower bound on the number of transition graphs. Section 5 gives calculations of the lower bound counts of transition graphs, and describes simulation results, with counts on numbers of attractors

and basin classes for neural networks of varying sizes. We conclude with a discussion on how to eventually exploit dynamic attractors and their basins for computational purposes, and a consideration of related research in this area.

2 Neural Network Structure

We consider the set of binary neural networks with a single layer that is fully interconnected. No self-loops are included. Each unit has activation level a_i, and weight w_{ji} is the weight on the connection to unit j from unit i. At each iteration, each unit computes its incoming sum as follows:

$$S_j(t+1) = \Sigma_{i=1}^{n} a_i(t) w_{ji} \tag{1}$$

where n = number of processing units ("neurons") and w_{ii}=0. Incoming sums are thresholded to yield binary activation values, as follows:

$$a_j(t+1) \quad = \quad sgn(S_j(t+1)) \tag{2}$$

$$sgn(x) = \begin{cases} +1 & if \;\; x \geq 0 \\ \\ -1 & if \;\; x < 0 \end{cases} \tag{3}$$

States of the network consist of n-tuples $(a_1, a_2, ..., a_n)$ with +1/-1 entries. Thus a state may be denoted as an n-tuple of +/- characters such as $(++---++-)$. There are 2^n possible states for a network of n neurons, and $n^2 - n$ weights. Figure 1 shows fully connected networks with 2, 3, and 4 neurons.

Simultaneous updating has been used in this study. Thus the whole network is updated at once based on its previous state. This means that for each neuron, (1) is computed for $S_j(t+1)$ based on activations $a_j(t)$ at time t.

3 TRANSITION GRAPHS AND DEFINITIONS

Def. A transition consists of an ordered pair of binary n-tuples, denoted $(a_1, a_2, ..., a_n) \rightarrow (b_1, b_2, ..., b_n)$. For example, $(+ + -) \rightarrow (- + -)$ is a transition for $n = 3$.

A neural network performs a transition $\mathbf{a} \rightarrow \mathbf{b}$ if a network that starts in state $\mathbf{a}$, changes to state $\mathbf{b}$ after each processing unit is updated by (1) and (2).

Def. A Transition Graph is a graph whose nodes are all possible binary n-tuples, for some value of n. Any connections are allowed.

Def. A transition graph whose nodes are n-tuples is said to have order n.

Def. Global Transition Graph of order n. Nodes: all possible 2^n binary states. Connections: Fully connected, directed connections, including self-loops. The global transition graph for $n = 2$ appears in Figure 2. There is only one global transition graph for each order n.

Def. Two transitions between states of length n ($\mathbf{a} \rightarrow \mathbf{b}$ and $\mathbf{c} \rightarrow \mathbf{d}$) are compatible if there exists a matrix of weights $\mathbf{W}$ such that the neural network with weights $\mathbf{W}$ performs those two transitions. If not compatible, then the two transitions are incompatible.

Def. A set of transitions is compatible if there exists a weight matrix $\mathbf{W}$ such that the neural network with weight matrix $\mathbf{W}$ performs those transitions. Otherwise the set

is incompatible.

Def. Network Transition Graph (NT Graph) of order n. Nodes: all possible 2^n binary states. Connections: A set of transitions for which there exists a weight matrix **W** such that a neural network with weights W performs exactly those transitions in the network transition graph. There can be many network transition graphs for each order n. Figure 3 shows NT-Graphs for neural networks with $n = 3$ processing units.

A Network Transition Graph has the following properties. (1) The outdegree of each node is exactly 1. This is because for each state of a neural network, when updating (1) and (2) is applied, the network goes to a unique next state. (2) The indegree of each node is from the set $\{0, 1, 2, ..., n\}$. A state of the network may or may not result from another state.

An NT-Graph shows all fixed point attractors and oscillations in the corresponding neural networks, along with the basins of attraction. A fixed point attractor is a self-loop in the network transition graph. An oscillation is a closed circuit in the network transition graph. An m-state oscillation has m nodes in the closed circuit. Each basin of attraction is a detached subgraph of the network transition graph. Any node in a basin leads either to a fixed point attractor or an oscillator. All attractors that are not fixed points are finite state oscillations with 2 or more states. There can be more than one weight matrix **W** that implements a given network transition graph.

Def. A transition graph is complete if outdegree=1 for all nodes. All network transition graphs are complete.

Some sets of transitions are not compatible. Thus, some transition graphs are not NT-Graphs because there does not exist a network that performs the set of transitions in the graph. A set of compatible transitions does not have to be complete. An NT-Graph has a complete set of compatible transitions.

Def. A penultimate subset of {1,2,...,n} contains n-1 unique elements in ascending order (e.g., one element is removed). Let V be a penultimate subset of $1, 2, ..., n$. Then $\mathbf{I}_V$ is the vector that includes all positions from $\mathbf{I}$ that appear in V, taken in the order specified in V.

A basin is set of states that lead to the same attractor. The boundaries of the basins are the disallowed transitions resulting from the division of states into distinct, disjoint basins.

A basin class is a set of network transition graphs with the same basins. Two network transition graphs in the same basin class can have different attractors and different paths to those attractors within each basin.

The order of an attractor is 1 if a fixed point and m if an m-state oscillator.

Def. A Neuron Transition Table (NT-Table) of order n is a matrix $\mathbf{J}$ and a vector ζ such that the rows of $\mathbf{J}$ are all possible $n-1$-tuples of binary entries +1/-1, and ζ is a binary vector (+1/-1) with 2^{n-1} entries. The entries in ζ are subject to the following restriction: There must exist a set of $n-1$ values $\mathbf{x}$ such that

$$\mathbf{Jx} \quad O_{\zeta_j} \quad 0$$

where O_j is "$\geq$" if $\zeta_j = 1$ and is " $<$ " if $\zeta_j = -1$.

An NT-Table is used to specify the state of a neuron z as a function of the previous states of the other neurons. The vector $\mathbf{x}$ consists of the incoming weights to neuron z. For a network of n neurons, with n NT-tables one can construct an NT Graph. Figure 4 illustrates an NT-Table for n=4 neurons.

4 COMPATIBLE TRANSITIONS AND COUNTING OF TRANSITION GRAPHS

We next consider conditions needed for a pair or set of transitions to be compatible. First is Lemma L1, which covers the case of networks with n=3 processing units, then Lemma L2, which allows any number n of processing units, and covers incompatability criteria for pairs of transitions. Lemma L3 shows how to construct an NT Graph from n NT Tables, and computes a lower bound on the number of NT Graphs. In Section 5, we use this result to calculate lower bounds on counts of network transition graphs.

LEMMA L1.

Given a neural network with n=3 units and all incoming sums S_i non-zero.

Given two transitions, $(i_1, i_2, i_3) \rightarrow (j_1, j_2, j_3)$ and $(k_1, k_2, k_3) \rightarrow (l_1, l_2, l_3)$.

A. If $(i_x, i_y) = (k_x, k_y)$ and $j_z \neq l_z$ for any x, y, and z such that $\{x, y, z\} = \{1, 2, 3\}$ then the two transitions given are incompatible.

B. If $(i_x, i_y) =$ (-1)(k_x, k_y) and $j_z = l_z$ for any x, y, and z such that $\{x, y, z\} = \{1, 2, 3\}$ then the two transitions given are incompatible.

C. Conditions A and B are the only conditions under which two transitions are incompatible.

D. If there is a set of more than two transitions that are mutually incompatible, then there is at least one pair of those transitions that meets condition A or B.

PROOF OF L1.

Statement L1-A.

The set $\{x, y\}$ includes all the nodes in the neural network except for node z. If the initial state is (i_1, i_2, i_3), then, according to the first transition, the incoming sum, for node z is:

$$S_{z,\mathbf{i}} = i_x w_{zx} + i_y w_{zy}. \tag{4}$$

where $S_{z,\mathbf{i}}$ denotes incoming sum for node z given the network is in state $\mathbf{i}$.

When the initial state is (k_1, k_2, k_3), then the incoming sum for node z is:

$$S_{z,\mathbf{k}} = k_x w_{zx} + k_y w_{zy} \tag{5}$$

According to the Case A assumption, $(i_x, i_y) = (k_x, k_y)$, which implies that

$$S_{z,\mathbf{i}} = S_{z,\mathbf{k}} \tag{6}$$

After applying threshold equation (2) to $S_{z,\mathbf{i}}$ and $S_{z,\mathbf{k}}$, the same value must result (j_z and l_z, respectively), so $j_z = l_z$.

Therefore no set of weights (w_{zx}, w_{zy}) allows $j_z \neq l_z$, so the two transitions are incompatible.

Statement L1-B.

The set $\{x, y\}$ includes all the nodes in the network except for node z. If the initial state is (i_1, i_2, i_3), according to the first transition, then the incoming sum, by equation (1), for node z is:

$$S_{z,\mathbf{i}} = i_x w_{zx} + i_y w_{zy}. \tag{7}$$

When the initial state is (k_1, k_2, k_3), then the incoming sum for node z is:

$$S_{z,\mathbf{k}} = k_z w_{zx} + k_y w_{zy} \tag{8}$$

According to the Case B assumption, $(i_x i_y) = (\text{-}\ 1)(k_x, k_y)$, which implies that

$$S_{z,\mathbf{i}} = (-1) S_{z,\mathbf{k}} \tag{9}$$

Now assume that $S_{z,\mathbf{i}} \neq 0$, and apply the sgn operator to each side of (9). We get $j_z = (-1) l_z$. Then no set of weights $\{w_{zx}, w_{zy}\}$ allows $j_z = l_z$.

When $S_{z,\mathbf{i}} = 0 = S_{z,\mathbf{k}}$, then $j_z = l_z = 1$, but this special case is excluded in the assumption to this lemma.

Therefore when $j_z = l_z$, the two transitions are incompatible.

Statement L1-C.

Consider a situation where there is no subset $\{x, y\} \epsilon \{1, 2, 3\}$ such that $(i_x, i_y) = (k_x, k_y)$ or $(i_x, i_y) = (-1)(k_x, k_y)$.

Let O_m be an operator that is in the set of operators $\{\geq, <\}$, and let $m = \pm 1$. Set O_m as "$\geq$" if m=+1; set O_m as "$<$" if $m = -1$.

For each set $\{x, y\}$ that is a penultimate subset of {1,2,3}, apply the following argument:

The pair of inequalities that govern weights w_{zx} and w_{zy} are as follows:

$$i_x w_{zx} + i_y w_{zy} \quad O_{j_z} \quad 0 \tag{10}$$

and

$$k_x w_{zx} + k_y w_{zy} \quad O_{l_z} \quad 0 \tag{11}$$

If an operator above is $<$, then multiply the inequality by -1. All operators are then either "$\geq$" or "$>$".

A pair of inequalities result, both of the form:

$$\pm w_{zx} \quad + \quad \pm w_{zy} \quad >, \quad 0 \tag{12}$$

Where ">," means either ">" or "$\geq$". Each specifies a half-plane in the coordinate axes where w_{zx} and w_{zy} are the two axes, and Figure 5 shows the lines dividing these half-planes.

Case 1: Suppose the two inequalities differ by exactly one minus sign on the left. Then each specifies a half- plane whose edge is a different line intersecting the origin and at 45 degrees from the x-axis. Therefore the two half-planes have an overlapping area, and each point in the overlapping area specifies values for w_{zx}, w_{zy} that implement the pair of transitions.

Case 2: The two inequalities in (10) and (11) have the same signs on the left. But all operators are either ">" or "$\geq$", which implies a half-plane of solutions for w_{zx}, w_{zy}. Thus there is a half-plane of solutions for (w_{zx}, w_{zy}).

Case 3: The two inequalities in (10) and (11) differ by two minus signs on the left. Then either $(i_x, i_y) = (k_x, k_y)$ and $j_z \neq l_z$ (e.g., the signs on the left of (9) and (10) match but the operators don't), or $(i_x, i_y) = (-1)(k_x, k_y)$ and $j_z = l_z$ (e.g., the signs don't match but the operators do). These are cases A and B, in which the pair of given transitions are incompatible. But these cases were excluded from consideration in the first sentence to the proof of Statement C.

Statement L1-D.

All inequalities are of the form

$$\pm w_{zx} + \pm w_{zy} \quad O_* \quad 0 \tag{13}$$

where $O_* \epsilon \geq, >$.

Suppose a set of m transitions are not compatible, where $m > 2$. Each transition specifies an inequality of the form in (13), and each inequality specifies a half plane subtended by one of two lines through the origin, both lines at 45 degrees from the horizontal axis. (See Figure 5.) Since each of the subtending lines intersects at the origin, and they are perpendicular to each other, the only way to have the empty solution Φ is to specify two opposite sides of the same line, which means that a pair of the n transitions is incompatible. Thus D is true. QED

Lemma L1 supplies enough information to enumerate all NT-Graphs for $n = 3$. Since the system of inequalities for $n = 2$ is small, NT-Graphs of order 2 can be enumerated easily. In both cases we restricted our count to networks in which all incoming sums are non-zero, for convenience. Thus there are four NT-Graphs for $n = 2$ and 64 NT-Graphs for $n = 3$.

The next lemma addresses pairwise incompatabilities for cases where $n > 3$. Unfortunately, in these cases, higher order incompatabilities can arise, as we will describe after the prrof of Lemma L2.

LEMMA L2.

Given a neural network with n units, and all incoming sums S_J non-zero.

Given four network states, each binary vectors of length n: **I**, **J**, **K**, and **L**.

Given two transitions, $\mathbf{I} \rightarrow \mathbf{J}$ and $\mathbf{K} \rightarrow \mathbf{L}$.

Let V be a penultimate subset of $1, 2, ..., n$.

Let $\mathbf{I}_V$ be the vector that includes all positions from $\mathbf{I}$ that appear in V, taken in the order specified in V.

Let $Z = \{1, 2, ..., n\} - V$ and let z be the single element of Z.

Then:

A. If $I_V = J_V$ and $K_Z \neq L_Z$ for any penultimate subset V of length $n - 1$, then the two transitions are incompatible.

B. If $I_V =$ (-1) J_V and $K_Z = L_Z$ for any penultimate set V of length $n - 1$, then the two transitions are incompatible.

PROOF L2.

Statement L2-A.

The set V includes all the nodes in the network except for node $z \epsilon Z$. If the initial state is $\mathbf{I}$ (as in the first transition) then the incoming sum, by equation (1), for node z is:

$$S_{z,\mathbf{I}} = \Sigma_{x \epsilon V} i_x w_{zx} \tag{14}$$

where $S_{z,I}$ is incoming sum to unit z when the initial state is $\mathbf{I}$.

When the initial state is K then the incoming sum for node z is:

$$S_{z,\mathbf{K}} = \Sigma_{x\epsilon V} \ k_z w_{zx} \tag{15}$$

According to the Case A assumption, $\mathbf{I_V} = \mathbf{K_V}$, which implies that

$$S_{z,\mathbf{I}} = S_{z,\mathbf{K}} \tag{16}$$

But applying the threshold equation (2) to $S_{z,I}$ and $S_{z,K}$, we get the same result, thus $j_z = l_z$.

Therefore no set of non-zero weights $w_{z,i}$ allows $j_z \neq l_z$, so the two transitions are incompatible.

Statement L2-B.

The set V includes all the nodes in the network except for node $z\epsilon\mathbf{Z}$. If the initial state is **I**, as in the first transition, then the incoming sum, by equation (1), for node z is:

$$S_{z,\mathbf{I}} = \Sigma_{x\epsilon V} i_x w_{zx} \tag{17}$$

When the initial state is **K**, then the incoming sum for node z is:

$$S_{z,\mathbf{K}} = \Sigma_{x\epsilon V} k_z w_{zx} \tag{18}$$

According to the Case B assumption, $\mathbf{I} = (-1)\mathbf{K}$, which implies that

$$S_{z,\mathbf{I}} = (-1)S_{z,\mathbf{K}} \tag{19}$$

But applying the threshold equation (2) to $S_{z,\mathbf{I}}$ and $S_{z,\mathbf{K}}$, and assuming no non-zero incoming sums, we get $j_z = (-1)l_z$.

Therefore no set of weights $w_{z,i}(i\epsilon V)$ allows $j_z = l_z$, and the two transitions are incompatible. QED

Lemma L2 shows how incompatabilities arise between pairs of transitions, for networks with $n = 3$ neurons. Although Lemma L1 showed that pairwise incompatabilities are the only incompatabilities for $n = 3$, this result does not extend to $n > 3$ neurons. If $n > 3$, then it is possible for incompatabilities to arise among 3 (or more) transitions when every pair of those transitions taken alone is compatable.

As an example of such a 3-way incompatability, consider the case $n = 4$. Each transition then defines a half-space subtended by a plane that intersects the origin. Figure 6 shows a 3-way incompatability. Three planes appear in Figure 6. The drawing looks as though a book is placed vertically on a table, and opened a crack, with its binding still along the table. The front and back cover of the book are two of the planes (A and B) and the table the third plane (C). Half-space A is the side of the front cover facing the interior of the book, and half-space B is the side of the back cover of the book facing the interior of the book. Half-space C is the underside of the table. Any pair of half-spaces here have an intersecting volume, but all 3 half-spaces do not intersect anywhere. Their intersection is null. This situation could correspond to an incompatability of 3 conditions that implies no pairwise incompatability.

We must now consider the aims of this research. An important aim is to gain knowledge about how many NT-graphs are possible, and to demonstrate that large varietes of NT-graphs can occur. It is relatively simple to check for pairwise incompatabilities. If all

incompatabilities were pairwise for $n > 3$, then an algorithm could easily be constructed to count the number of NT-Graphs. However, since incompatabilities can arise that result from more than 2 conditions, such an algorithm becomes considerably more difficult to construct and more time-consuming to execute. To avoid such a combinatorial quagmire, we have instead derived a formula to calculate a lower bound on the number of NT-Graphs. The following lemma derives a formula for $L(n)$, a lower bound on the number of NT-Graphs for networks of n neurons. Knowledge of the lower bound will be sufficient to show that the number of NT Graphs is very large.

LEMMA L3

A. A network transition graph of order n can be constructed from n neuron transition tables (NT-Tables) by the following procedure.

Construct $m = 2^n$ transitions $K_i \rightarrow L_i (i = 1, 2, ..., m)$ from n NT Tables by the following steps.

Let $K_1, K_2, ..., K_m$ be the $m = 2^n$ binary states. To determine element z of state L_i, use the zth NT-Table. Let $K_{i(-z)}$ be the $(n-1)$-tuple taken from n-tuple K_i but with the z^{th} element of K_i removed. Let row j be the row that matches $K_{i(-z)}$ in the z^{th} NT-Table. Insert ζ_j from the z^{th} NT-Table as element z of state L_i. Fill in all elements of $L_i (i = 1, 2, ..., m)$ by this procedure to get the entire NT-Graph.

B. A lower bound on the number of neuron transition tables of order n is 2^{n-1}.

C. A lower bound on the number of network transition graphs of order n is $L(n) = 2^{(n-1)n}$.

PROOF / JUSTIFICATION

Statement L3-A.

Consider the z^{th} NT-Table to represent the transitions of neuron z, such that row J_i signifies the states of the other neurons that send connections to z. Given state J_i for the other neurons, the new state of z is then ζ_j. The entire construction is accomplised in this fashion.

Statement L3-B.

Let $\mathbf{J}$ be the matrix on the left of the NT-Table.

Order the rows of $\mathbf{J}$ so that the first $(n-1)$ rows are linearly independent. This step is possible because J has $(n-1)$ columns and its rows consist of all 2^{n-1} possible $(n-1)$-tuples of $-1, 1$. Let J_i be row i of J.

Given a vector $\mathbf{x}$ (which corresponds to the incoming weights to neuron z).

Then for each value of $i(i = 1, 2, ..., n-1)$, $\mathbf{J_i x}$ is either ≥ 0 or < 0.

Consider an arbitrary assignment of the operators $O_1, O_2, ..., O_{n-1}$ so that each operator is either "$\geq$" or "$<$".

Since $\mathbf{J}_1$,$\mathbf{J}_2$,...,$\mathbf{J}_{n-1}$ are independent, then you can find solutions for $\mathbf{x}$ for

$$\mathbf{J_i x} \quad O_i \quad 0 \quad i = (1, 2, ..., n-1) \tag{20}$$

To find the solution, replace "O_i 0" with "$= p_i$", and make $p_i \geq 0$ or < 0, appropriately.

Then the system $\mathbf{J}_i\mathbf{x} = p_i(i = 1, 2, ..., n-1)$ can be used to solve for the x's (by linear

algebra).

Now J_h $(h = n, n+1, ..., 2^{n-1})$ are the dependent vectors. Using the solved values for x, then $\mathbf{J}_h x$ is either ≥ 0 or < 0 for each h $(h = n, n+1, ..., 2^{n-1})$. Thus

$$\mathbf{J_i x} \qquad O_h \qquad 0 \quad h = (n, n+1, ..., 2^{n-1})$$

is true for some assignment of O_h as either $\geq$ or $<$.

Since the assignment of operators $O_1, O_2, ..., O_{n-1}$ was arbitrary and there were two possibilities for each operator, there are 2^{n-1} cases. Each case leads to solution for x.

Conclusion: There are 2^{n-1} unique cases here that each have a set of solutions for x. For each case, there may be multiple possibilities in the following sense. A different choice for the value of p_i could lead to different operators $O_h(h = n, n+1, ..., 2^{n-1})$. Thus each case could include multiple possibilities for NT-Tables. Thus $H(n) = 2^{n-1}$ is a lower bound.

Statement L3-C.

Each neuron in the network needs an NT-Table to specify what transitions it will undergo. There are n neurons, so the number of NT-Tables are multiplied together for the n neurons. Thus we have the number of NT-Tables raised to the number of neurons. So the number of NT Graphs has as lower bound $L(n) = 2^{(n-1)n}$.

5 Counts on Transition Graphs and their Basins

In this section we examine the issue of neural network capacity and flexibility regarding attractors and basins of attraction, considering attractors that are oscillators as well as fixed point attractors. We consider counts of NT Graphs, attractors, and basin classes, and address the basin class capacity of networks with varying sizes. We begin with computations of lower bounds for NT-Graphs, then show simulation results that were done to complement the theoretical calculations. We describe a series of simulations on counts of basin classes and attractors. We show differences between networks with symmetric versus asymmetric weights, and compare asynchronous updating rules used in the Hopfield network to the synchronously updated networks studied here.

Table I shows calculation of lower bounds on the number of NT Graphs. Column 1 is the number of neurons in the network (n), and Column 2 is the number of nodes in the NT Graph (2^n). Column 3 is H(n), the lower bound on the number of NT Tables (2^{n-1}). Column 4 is L(n), the lower bound on the number of NT graphs ($2^{(n-1)n}$). Formulas for Columns 3 and 4 are taken from Lemma L3, parts B and C, respectively.

Table 1 shows how large the number of NT-Graphs becomes, even for small neural networks. The number grows exponentially, but since n^2 appears in the exponent to 2 in $L(n)$, the number is faster growing than an exponential with n in the exponent. Even for a small number of neurons, there is a large number of NT-Graphs. Whereas $n = 2$ leads to at least 4 NT-Graphs, and $n = 3$ leads to at least 64, there are thousands for $n = 4$, and for $n = 5$, over a million. Each of the graphs counted has a unique set of transitions resulting from the neural network's weights. From the large counts we can conclude that there is a large space of behaviors among neural networks of the same size but with different weights. When there are 19 neurons, there are more NT-Graphs than 10^{100}, a number used to estimate how many particles are in the universe.

We are critically interested in attractors, their basins, and the boundaries of their basins of attraction. The NT-Graphs counted in the lower bound estimates of Table I show the numbers in the overall pool of network transition graphs, but do not show the properties of these graphs in terms of attractors and their basins. However, the attractors, their basins, and the boundaries of the basins are important properties to know as these are the properties needed for computational paradigms. Thus we need further analysis of the enormous numbers of NT-Graphs.

Simulations were done to generate NT Graphs and to examine their properties, specifically their attractors, basins, and basin classes. Nets were generated with random weights, with many networks having the same number of neurons. The NT Graph was calculated for each network. The attractors, the number of attractors, and the order of each attractor were found. The number of distinct basin classes found in the networks was also tracked.

Figure 7 shows the number of attractors found in sets of simulated networks with differing sizes, where weights were chosen at random between -1 and 1, and reciprocal asymmetry ($w_{ij} \neq w_{ji}$) was allowed. Four hundred networks of each size were simulated at random, with $n = 3$, 5, 10, and 12 neurons. For $n = 3$, Figure 7a histograms the number of attractors found. Between 1 and 4 attractors were found. Since the maximum number of attractors found was 4, the number of attractors can be greater than the number of neurons. Figure 7b histograms the number of attractors found with 5 neurons. Between 1 and 8 attractors were found, with a peak at 3. Again, the number of attractors can be greater than the number of neurons. Figure 7c shows attractors for networks with 10 neurons. Between 1 and 12 were found with a peak at 5 attractors. In Figure 7d, 12 neuron networks were simulated and histogrammed. A minimum of 1 attractor and maximum of 16 attractors were observed, the maximum of 16 having a 30 % margin above the number of neurons in the network. For larger n, there appears to be a larger number of attractors possible plus a larger range also.

Table 2 summarizes the numbers of attractors for networks of varying sizes. The size of the network is given in Column 1, and Column 2 shows the average number of attractors observed. The minimum and maximum numbers of attractors observed are in Columns 3 and 4. As before, sometimes the number of attractors is larger than the number of neurons. Networks with only 1 attractor are readily observed. The numbers of attractors are spread over a larger range for larger n. Column 5 shows the average number of fixed point attractors and column 6 gives the average number of oscillating attractors observed. On average, there are more oscillating attractors than fixed points, and the fixed point attractors are limited to about one per network, on average. This attests to the contribution of oscillating attractors in typical network dynamics. Restricting networks to only those with fixed point attractors would prevent the use of many of the networks simulated here.

Table 3 shows the same information for networks with symmetric weights only (e.g. $w_{ij} = w_{ji}$). The average number of attractors increases with the size of the network, but much higher counts were found compared with Table 2. For a network with symmetric weights having 10 neurons, up to 77 attractors were found whereas only 12 were found in the asymmetric nets simulated (Table 2). For 12 neurons, a maximum of 186 were observed in the symmetric networks whereas a maximum of only 16 was found in the asymmetric networks simulated. In both cases, the number of attractors sometimes exceeded the number of neurons, but the excess was far larger among the symmetric networks. Ironically, symmetric networks are a subset of asymmetric networks, as asymmetric networks are not restricted to having $w_{ij} = w_{ji}$ but can include equality or inequality of reciprocal weights. The results in the table, however, reflect the sample of networks found when 400 nets were generated at random, fully connected with weights uniformly distributed between -1 and 1. Among these simulations, symmetric nets tend to have larger numbers of attractors than asymmetric nets. The asymmetric nets do have larger size basins (on average) due to the smaller numbers of attractors.

Table 3 also addresses the issue of whether oscillating attractors contribute signifi-

cantly to network dynamics. Column 5 shows the average number of fixed point attractors, and column 6 shows the average number of oscillating attractors. For asymmetric weight networks covered in Table 2, the average number of fixed point attractors is about one, whereas the number of oscillating attractors grows with the network size. For the symmetric weight networks covered in Table 3, the average number of fixed points increases with network size, but the average number of oscillators increases even more. We can conclude again that the network dynamics depends significantly on the oscillating attractors.

The issue of fixed versus oscillating attractors arises in comparing the random networks here to networks used in Hopfield's associative memory. The restriction of reciprocal weight symmetry ($w_{ij} = w_{ji}$), used by Hopfield, limits oscillators to order 2 (i.e., two alternating states) in our simulated networks. Few networks simulated here have all fixed points and no oscillators, as in the Hopfield networks.

Each of the attractors has a basin of attraction, and the basins can be of even greater interest than the attractors themselves. A basin class is a set of network transition graphs with the same basins (and same basin boundaries). In other words, the set of states in each basin of attraction are the same in all networks in the same basin class. Two network transition graphs in the same basin class may have different attractors and different paths to those attractors within a basin. We can classify the nodes of the NT graph into different sets, one for each basin. Figure 8 shows NT-Graphs for 3-neuron networks with basin classes that match those from different networks used in Figure 3. The basin class of Figure 8a consists of one basin and is the same basin class found in Figure 3a although a different set of transitions occurs and a different attractor appears in the basin. The basin class of Figure 8b has 3 attractors and matches the class for the NT-Graph in Figure 3c.

There are 2^n nodes in a network transition graph of order n (n neurons). The number of ways of dividing k nodes into two distinct classes is 2^k. Thus there are over 2^{2^n} potential basin classes. Let $B(n)$ be the number of basin classes realized in the set of all

of the NT-graphs of order n. We next show simulation experiments that probe the size of $B(n)$. These results address how many of the basin classes are realizable under simulated conditions.

We defined the basin class capacity to be the number of basin classes possible from a given set of neural networks. The basin class capacity of a set of networks is a measure of the potential utility of the networks for dividing a pattern space into desired subspaces. Because of the large numbers of NT-Graphs shown in Table 1, the basin class capacity is expected to be high. However, more than one NT-Graph can have the same basin class. Thus we turn to simulations to show that large numbers of basin classes result from the many NT-Graphs. We need to assess how many different basin classes can be generated, and what are their properties.

Figure 9 shows the results of simulated basin class calculations. Networks were simulated with random weights, and for each network generated, its basin class was found. The basin class calculation consisted of assigning a basin number (ID) to each node of the NT-Graph. Two nodes were in the same basin if a path from one to the other was in the NT-Graph. Two NT-Graphs were in the same basin class whenever the basin ID matched for all nodes. Thus two NT-Graphs were in the same basin class whenever the basin boundaries matched but the attractors and paths to those attractors could be different within the same basin. Thus we can test the conjecture that large numbers of NT graphs lead to large numbers of basin classes.

Figure 9 shows the number of distinct basin classes observed versus the number of networks simulated. The first 2000 networks simulated are shown. Since the weights are all randomly generated numbers, we are sampling from the space of all binary networks with weights in the given range [-1:1]. Figure 9a shows networks with 3 neurons, Figure 9b shows networks with 4 neurons and Figure 9c shows networks with 5 neurons. The number of distinct basin classes initially rises quickly, but the rise slows until the graph appears to be approaching an asymptote. Not all basin classes that are possible will

appear, because with random weights some basin classes are unlikely, and so may not appear in the simulation. Higher numbers of neurons n take longer for the graph to flatten out, as a larger basin class capacity is being sampled. For $n = 3$, the flat region is found within the first 100 networks, as shown in Figure 9a. For $n = 4$ (Figure 9b), the slope is still steep at the 2000^{th} network simulated, but is less steep than the initial slope. In Figure 9c, $n = 5$, and the slope is very steep even at the 2000^{th} network. When there is a very large number of basin classes, the initial angle continues steeply for a long time.

The number of basin classes observed in the NT Graphs generated were counted, and Table 4 shows the results. Column 1 is the number of neurons, n, and Column 2 is the number of networks simulated. Column 3 is the number of basin classes observed, using asymmetric weights ($w_{ij} \neq w_{ji}$ allowed). Column 4 is the number of basin classes observed for symmetric weights ($w_{ij} = w_{ji}$). For $n = 2$, only a small number of classes was observed (only 3 for asymmetric weights). For $n = 3$, the number grows to 15 for asymmetric weights, and for $n = 4$, over a thousand were observed. Since $L(4) = 4,096$, the fraction of transition graphs that were observed as unique basin classes was 35 percent, a relatively high fraction.

For networks of up to four neurons, the number of basin classes observed with asymmetric weights was greater than the number observed with symmetric weights. This comparison reflects the fact that networks with asymmetric weights allowed includes, as a subset, the networks with symmetric weights. Thus the number of basin classes is higher for networks with asymmetric weights allowed. For $n = 2, 3$, and 4, the simulation experiment appears to have produced close to the total count of basin classes possible.

When $n \geq 5$, many thousands of basin classes were observed during the simulations. The first ten thousand networks simulated, in each case, resulted in 7,000 or more unique basin classes observed. The number of basin classes observed for this size network was larger when weights were restricted to symmetric values. The reason for this discrepancy is that the case of asymmetric weights had more repetitions in basin classes among the

first 10,000 networks simulated. There still must be more total basin classes among the networks with asymmetric weights allowed because the networks with symmetric weights are a subset. The repetitions in basin classes observed were not surprising because networks with asymmetric weights allowed tended to have fewer attractors with larger basins. Thus, rearranging the transitions within a basin was more likely.

An important issue in the development of neural networks has been that of asynchronous versus synchronous updating. Asynchronous updating means that neurons are updated one at a time, and the new state of the unit updated is then used when updating the next unit. In synchronous updating, the entire network of neurons is updated to new activation states $a_j(t)$ at the same time, based on inputs to each neuron from other neurons at time t-1 $(a_i(t-1))$. Asynchronous updating can be done by selecting the next neuron at random, or by selecting the neurons in a given order, with the same order used repeatedly.

When the next neuron is selected at random, the boundaries of the basins are unstable. The same starting point leads to different attractors at different times, depending on the sequence of neurons updated subsequently. In the case where neurons are updated in a given order repeatedly, the basins are stable, but changing the order of neurons changes their boundaries. Thus the boundaries are not stable with respect to the updating procedure.

The associative memory proposed by Hopfield had symmetric weights with asynchronous updates. Asynchronous updating appeared as part of the proof that the network would eventually reach a fixed state, and a fixed state was thought to be necessary for a computational paradigm. We argue here that fixed point attractors are not necessary for computational paradigms, and that new paradigms are made possible by the stable and flexible basin boundaries that appear in networks with synchronous updating.

In biological systems, neurons tend to fire at different times. With asynchronous up-

dating, neurons must change activity states at different times, also. However, in biological systems, neurons are not chosen by an externally generated random number to be the next to update; they update naturally as a result of incoming signals that cause their excitation level to surpass a threshold. The incoming signals are continuously received by each neuron. Synchronous updating divides time into time steps, and each neuron revises its incoming sum at each step. Neurons can still fire at different times. Thus the synchronous updating appears a better model of biological processes.

Ideally, a comparison between basin class capacities of synchronous and asynchronous networks would be useful, but such a comparison is not possible because of the unstable boundaries in asynchronous networks. When boundaries are unstable, one cannot compute a basin class because some states lead to different attractors at different times, depending upon the sequence of random numbers that determine the sequence of neurons updated. Thus we choose asynchronous ordered updating for a comparison with synchronous networks. A series of simulated random networks were generated and neurons were updated in sequential order. With this set of simulation experiments, we assessed the basin class capacities of asynchronous networks, and compare this capacity to that of synchronous networks. A series of networks were simulated.

Each state was used as initial position, and the target state of the next transition was computed to build the NT-Graph. The basin classes were then assessed and new basin classes were added to the inventory. With this approach, the number of basin classes found was always one for asynchronously updated networks with uniformly distributed random weights ($w\epsilon[-1, 1]$). In this one basin class, there was only one basin of attraction. This is not the total count possible for all weights, as the sample set was limited. Weights were uncorrelated in the simulated set. The Hopfield Net shows that other possibilities occur for particular weights that are correlated. We show here, though, that these cases are unlikely when random weights are used. Hence the multiple basin classes are not very accessible in the simulated networks.

In summary, a multitude of basin classes were demonstrated in asynchronous dynamic networks. High NT-Graph counts were found, and there is an increase in basin class counts that rises steeply with n, the number of neurons. Asynchronous networks give unstable basin boundaries with random updates, and, although the Hopfield paradigm for computing correlated weights yields multiple basins, a sampling of networks with random weights showed no variation in basin boundaries at all.

6 Discussion

This paper addresses the dynamic behavior of binary neural networks and the broad span of behaviors that can be observed, including oscillations as well as stable states. Theoretical results are combined with a simulation study aimed at determining the flexibility for binary neural networks to make different transitions, depending on their weights, and to develop attractors with differing basins. Section 2 defines the network structure with a standard transition rule for a binary net with synchronous updates, and a fully connected single layer. This connection topology allows for circuits (loops) in which a unit is connected to itself through other units. These networks are capable of self-sustained activity in the form of oscillations and can also evolve into stable states, depending on the weights and starting conditions.

Section 3 defined transition graphs, transition compatability, and neuron transition tables, used in the theoretical analysis later in this paper. Perhaps the most important definition in Section 3 was that of the basin class. Later results on counting basin classes showed the fundamental ability of these types of networks to divide binary n-tuples in a large number of ways, thus to have a large basin class capacity.

Section 4 shows some mathematical relationships that govern network transition graphs (NT-Graphs), which are graphs that show all transitions between all possible states of

binary neural networks. Some sets of transitions are mutually compatible (possible within the same network) but other sets cannot be implemented in a single network with a single weight matrix. A lower bound was found on the number of NT-Graphs (Lemma L3). This counts the number of sets of transitions that neural networks can implement, given any values for weights. Tremendously large numbers were computed for this lower bound, indicating a hugh span of behaviors for binary networks. Depending on the wieght matrix, the network may have one attractor with all states as its basin, or many attractors with basins of varying sizes. Attractors may be fixed points or oscillators.

Since the aim of this research is to understand binary dynamic neural networks so that we may ultimately apply them for computational purposes, it is important to assess the attractors, basins, and basin classes of these networks. In Section 5, networks were simulated at random and their dynamic characteristics were tracked. Large counts of basin classes were found, indicating a high basin class capacity for these networks. The attractors were found to sometimes outnumber the neurons, and for symmetric weights there could be 15 times as many attractors as neurons. These networks relied on oscillatory attractors more than fixed point attractors.

This study was limited to binary neural networks because the mathematics is more tractable for the binary case and the simulation counts are more practical to perform. An extension to this would address continuous-valued networks (Hopfield, 1984; Doyon et al, 1993; Dayhoff et al, 1994]. Continuous-valued networks are expected to show even greater flexibility and basin class capacity compared with their binary counterparts, as the state of each neuron can range between -1 and 1. The possibilities for their dynamic behavior is even larger than the results shown here for binary networks. Although it would be of interest to know the basin classes possible from all continuous-valued neural networks, the flexibility of the binary networks and their capacity to implement a large number of basin classes is sufficient to imply the potential of continuous-valued networks for applications paradigms.

This research provides groundwork for further study of dynamic neural networks and for eventual development of applications. For some applications, we will want to tailor the boundaries of the basins of attraction to desired specifications. Setting basin boundaries could be implemented by creating a set of disallowed transitions - transitions that, if allowed, would connect two regions that the investigator wants to have disconnected. The results in Section 4 form some of the groundwork to work towards such a goal analytically. The transitions in an NT-Graph can be translated into a series of inequalities, which can then be approached by existing methods for solving systems of inequalities (Walsh, 1985; Gahinet and Nemirovskii, 1993). Disallowed transitions also can be represented by a set of inequalities.

In the paradigm considered by Hopfield, associative memory is attained by using asynchronous updating, symmetric weights, and fixed point attractors. Inconsistent boundaries between basins of attraction occur with asynchronous updating, whereby the same initial state can lead to different attractors at different times. In the Hopfield associative memory, the fixed point attractors were set at the memory states; in other applications it will be important to divide the state space at the appropriate boundaries but the actual location of the attractor may not need to match a pre-set pattern. Thus additional degrees of freedom may be found by relaxing constraints on the position of the attractor and the nature of the attractor, to accept oscillating attractors as well as fixed points. Then there is more freedom to set basin boundaries as desired.

In this paper we have introduced the concept of basin class capacity. The basin class capacity represents the number of possible ways that a set of networks can divide a pattern space (e.g. the set of all states of a network). An adequate basin class capacity is needed for applications in which basin boundaries are set. In such an application, the initial state of a network would be a pattern to be classified, and the attractor in its basin would represent its pattern class. The network need only be updated until the attractor is found. We have shown that the capacity for attractors is high for dynamic networks, with attractors exceeding the numbers of neurons in some cases, and that binary dynamic

networks have high basin class capacity. Continuous-valued networks can be expected to have even higher capacity and flexibility.

Binary single-layer networks with closed circuits have been considered previously. Amari showed how pattern memories and pattern sequences were learned by self-organizing nets of threshold elements (Amari, 1972b). He showed that learning was most successful when the pattern sets and sequences consisted of orthogonal patterns. He also characterized random networks with analog neuron-like elements (Amari, 1972a). Amari and Maginu (1988) analyzed the non-equilibrium dynamical behaviors of an autocorrelation associative memory model, built from a recurrent binary network.

Hao and coworkers have shown a method for fixing convergence balls within basins of attraction in an asynchronous symmetric network, so that each basin is guaranteed to include a ball of a certain diameter centered on a fixed point attractor (Hao et al, 1994). Oscillating attractors were not considered in this work. Psaltis and coworkers considered a set of eigenvalues that allowed partial control over some aspects of the basins of attraction in a binary network with correlated weights and fixed point attractors (Venkatesh et al, 1990). The number of eigenvalues was limited, however, and each could only control a region covered by a basin. Wuensche has examined transitions and attractors in Boolean networks, which include as a subset the binary neural networks (Wuensch and Lesser, 1992).

Stability conditions for networks have been found under configurations of asymmetric weights (Matsuoka, 1992). Correlation associative memories basen on binary networks have been proposed by Nakano (1972), Anderson (1972), and Kohonen (1972). A variety of dynamic networks have been studied. Oscillations and chaos have been observed in networks built from a multitude of small oscillators, and some of these networks could perform computational tasks (Yao and Freeman, 1990). Networks with dynamic thresholds were found to have oscillations and temporal sequences of patterns (Horn and Uscher, 1989). Networks of neurons with hysteretic properties and self-connections were found

to have positive associative memory characteristics (Yanai and Sawada, 1990). Networks with continuous-valued neurons were found to exhibit multistate finite oscillations, limit cycles, and chaos as well as fixed point stable states (Sompolinsky et al, 1988; Doyon et al, 1993.).

Attractor networks have been proposed as models that potentially explain neural activity in the brain and nervous system. Amit has considered the relationship between binary recurrent networks and properties of mental processing (Amit, 1989). The associative memory model of Hopfield utilizes fixed point attractors as memories. Models of mental illness have been proposed with networks that have dynamic attractors and sometimes spurious attractors. Oscillations and repetitive motions are ubiquitous throughout the phylogenetic tree, from invertebrate swimming to primate walking. Gross (Gross and Tam, 1994; Tam and Gross, 1994) even observed self-organized oscillations of neurons growing in culture. Feature binding in the visual system - a mechanism that matches features with objects - has been proposed to occur via entrained oscillations (Grossberg and Somers, 1991).

Dynamic attractors can be understood as a collage of exemplars instead of a single memory, with each state in the oscillation representing a different instantiation of a pattern. Instead of an individual state representing grandmother, for instance, a set of states in the oscillator could represent grandmother from differing angles, with differing expressions. Thus the question of the neural code or the neural engram might be answered in terms of dynamic attractors. Reasoning with encoded information (e.g. objects represented by attractors) might then be accomplished via dynamic transients along the energy landscape.

Overall, the rationale for looking into the regime of dynamic networks is to seek more flexibility, increased computational capacity, more control over basin shapes, and a higher capacity for basin classes. A spectrum of new paradigms are expected to arise from this approach, in which dynamic attractors are utilized to classify patterns, to sort spatiotem-

poral signals, to identify memories, to generate trajectories and autonomous signals, and to model the brain and nervous system. We conclude that these goals are likely to be attained through the use of dynamic neural networks. Furthermore, biological systems show much evidence of having dynamic networks, which argues towards the development of further possibilities for new paradigms using dynamic networks and dynamic attractors.

7 References

S.-I. Amari, 1972a. Characteristics of random nets of analog neuron-like elements. IEEE Trans. SMC, 2 (5): 643-657.

S.-I. Amari, 1972b. Learning patterns and pattern sequences by self-organizing nets of threshold elements. IEEE Trans. Computers, 21 (11): 1197-1206.

S.-I. Amari and K. Maginu (1988). Statistical neurodynamics of associative memory. Neural Networks 1: 63-73.

D. J. Amit (1989). Modeling Brain Function: The World of Attractor Neural Networks. New York: Cambridge University Press.

J. A. Anderson, 1972. A simple neural network generating interactive memory. Mathematical Biosciences 14: 197-220.

F. Chapeau-Blondeau and G. Chauvet, 1992. Stable, oscillatory, and chaotic regimes in the dynamics of small neural networks with delay. Neural Networks 5: 735-743.

J. E. Dayhoff, P. J. Palmadesso, and F. Richards, 1994. Developing multiple attractors in a recurrent neural network. Proceedings, World Congress on Neural Networks (WCNN-

94) IV: 710-715.

B. Doyon, B. Cessac, M. Quoy, and M. Samuelides, 1993. Control of the transition to chaos in neural networks with random connectivity. International Journal of Bifurcation and Chaos 3 (2): 279-291.

P. Gahinet and A. Nemirovskii, 1993. LMI Lab: A package for manipulating and solving LMI's. Natick, Massachusetts, USA: The MathWorks Inc.

G. W. Gross and D. C.Tam, 1994. Pre-conditioned correlation between neurons in cultured networks. WCNN-94 II: 786-791.

S. Grossberg and D. Somers (1991). Synchronized oscillations during cooperative feature linking in a cortical model of visual perception. Neural Networks 4: 453-466.

J. Hao, S. Tan, and J. Vandewalle, 1994. A new approach to the design of discrete Hopfield associative memories. Journal of Artificial Neural Networks 1 (2): 247-266.

T. Kohonen, 1977. Associative Memory. New York: Springer.

A. Dembo, 1989. On the capacity of associative memories with linear threshold functions. IEEE Trans. Infor. Theory 35 (4): 709-720.

J. J. Hopfield, 1982. Neural networks and physical systems with emergent collective computational abilities. Proc. National Academy of Sciences 79: 2554-58.

J. J. Hopfield, 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proc. National Academy of Sciences 81: 3088-92.

D. Horn and M. Usher, 1989. Neural networks with dynamical thresholds. Physical Review A 40 (2): 1036-1044.

T. Kohonen, 1972. Correlation matrix memories. IEEE Trans. Computing, C-21: 353-359.

R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. s. Venkatesh, 1987. The capacity of the Hopfield associative memory. IEEE Trans. Information Theory 33: 461-482.

K. Nakano, 1972. Association - a model of associative memory. IEEE Trans SMC 2: 381-388.

H. Sompolinsky, A. Crisanti, and H. J. Sommers (1988). Chaos in random neural networks. Phys. Rev. Let. 61 (3): 259-262.

D. C. Tam and G. W. Gross, 1994. Post-conditioned correlation between neurons in cultured networks. WCNN-94 II: 792-797.

S. S. Venkatesh, G. Pancha, D. Psaltis, and G. Sirat, 1990. Shaping attraction basins in neural networks. Neural Networks 3: 613-623.

G. R. Walsh, 1985. An Introduction to Linear Programming. New York: John Wiley and Sons, Inc.

A. Wuensche and M. Lesser, 1992. The Global Dynamics of Cellular Automata. Reading, Massachusetts: Addison-Wesley.

H. Yanai and Y. Sawada, 1990. Associative memory network composed of neurons with hysteretic property. Neural networks 3: 223-228.

Y. Yao and W. J. Freeman, 1990. Model of biological pattern recognition with spatially chaotic dynamics. Neural Networks 3: 153-170.

8 Figures

1. Single layer neuron networks that are fully connected with no self-loops. (a) 2-neuron network, (b) 3-neuron network, (c) 4-neuron network.

2. The global transition graph for $n = 2$.

3. Network transition graphs for two neural networks with $n = 3$ processing units. (a) NT-Graph with one attractor having a 4-state oscillator and one basin. The basin class is /01234567/, where nodes are numbered 0-7 and / delimits basins. (b) NT-Graph with four attractors consisting of two fixed points and two 3-state oscillators, and four basins, with basin class /0/124/356/7/. (c) NT-Graph with three attractors consisting of two fixed points and one 2-state oscillator, and three basins, with basin class /0347/15/26/.

4. An NT-Table for neuron z in a 4-neuron network. Let J_i be row i of matrix J. Then $J_i x \geq 0$ whenever $\zeta_i = -1$. This defines the transitions of unit z.

5. Two 45 degree lines with shaded areas defined by any two half-planes subtended by the two lines. (a)-(d) give the four possibilities, depending upon which half-planes are chosen.

6. A construction of three intersecting planes, each of which subtends a half-space. There are choices for the half-spaces in which there is no 3-way intersection but always a 2-way intersection.

7. Histogram of number of attractors found in simulated networks, where weights were chosen at random between -1 and 1. For each network size, 400 nets were simulated, and reciprocal weights were allowed to be unequal ($w_{ij} \neq w_{ji}$). (a) Number of neurons $n = 3$ yielded 1-4 attractors. (b) Number of neurons $n = 5$ yielded 1-8 attractors. (c) Number of neurons $n = 10$ yielded 1-12 attractors. (d) Number of neurons $n = 12$ yielded 1-16 attractors.

8. Two NT-Graphs in the same basin classes as graphs in Figure 3. (a) Same basin class as Figure 3a. (b) Same basin class as Figure 3b.

9. The number of basin classes found as a function of the number of networks simulated. Two thousand networks of each size ($n = 3, 4$, and 5) were simulated. Weights were randomly selected from -1 to 1, and reciprocal weights were allowed to be unequal ($w_{ij} \neq w_{ji}$). (a) For $n = 3$ there is initially a steep rise, then the graph flattens off with little increase in the number of basin classes found. (b) For $n = 4$, there is an initial steep rise that begins to become less steep by the 2000^{th} network. (c) For $n = 5$, the initial steepness continues throughout the first 2000 nets, and the flattening out must occur later.

9 Tables

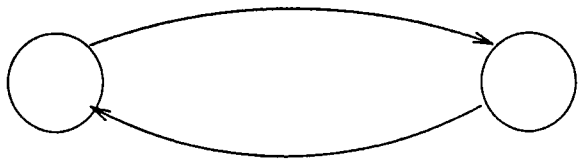
1. Synchronous updating, asymmetric reciprocal weights allowed. Column 1 is the number of neurons in the network, Column 2 is the number of nodes in the network transition graph, Column 3 is $H(n) = 2^{n-1}$, a lower bound on the number of NT-tables, and Column 4 is $L(n) = 2^{n(n-1)}$, the lower bound on the number of network transition graphs.

2. Synchronous updating, asymmetric reciprocal weights allowed. Column 1 gives the number of neurons in the network, Column 2 shows the average number of attractors

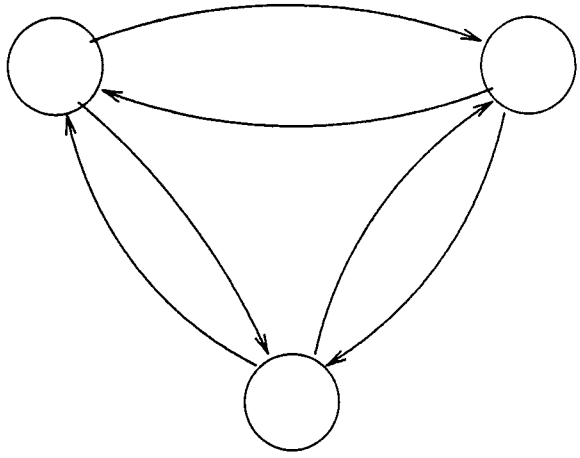
observed, Columns 3 and 4 show the minimum and maximum number of attractors observed, respectively. Column 5 shows the average number of fixed points, and Column 6 shows the average number of oscillators observed. Four hundred networks of each size were generated with random weights, fully connected with no self-loops.

3. Synchronous updating, symmetric reciprocal weights only. Column 1 gives the number of neurons in the network, Column 2 shows the average number of attractors observed, Columns 3 and 4 show the minimum and maximum number of attractors observed, respectively. Column 5 shows the average number of fixed points, and Column 6 shows the average number of oscillators observed. Four hundred networks of each size were generated with random weights, fully connected with no self-loops.

4. Basin classes were counted among simulated networks which were fully connected (no self-loops) with random weights. Column 1 shows the number of neurons in the network, Column 2 shows the number of networks simulated, Column 3 shows the number of basin classes observed when reciprocal weights were allowed to be asymmetric, and Column 4 shows the number of basin classes observed when reciprocal weights were restricted to symmetric values.

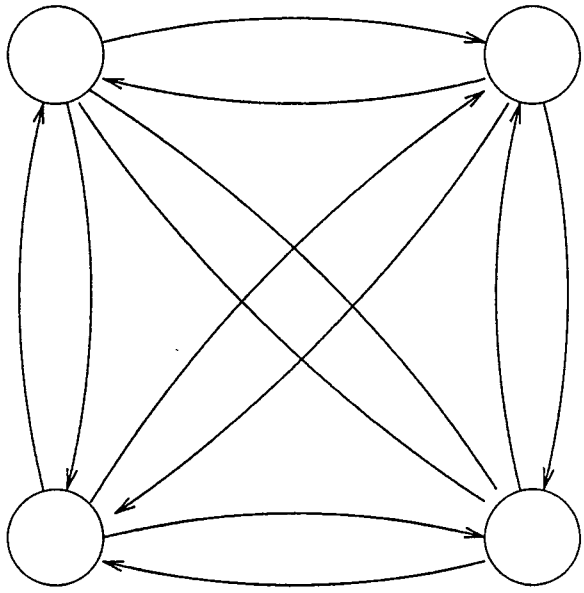


(a)



(b)

Figure 1



(c)

Figure 1

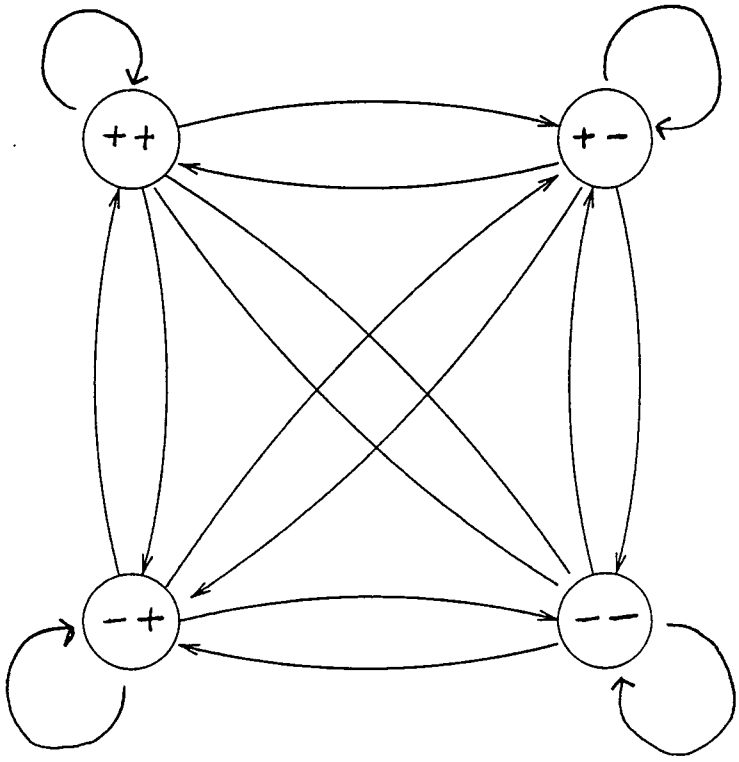


Figure 2

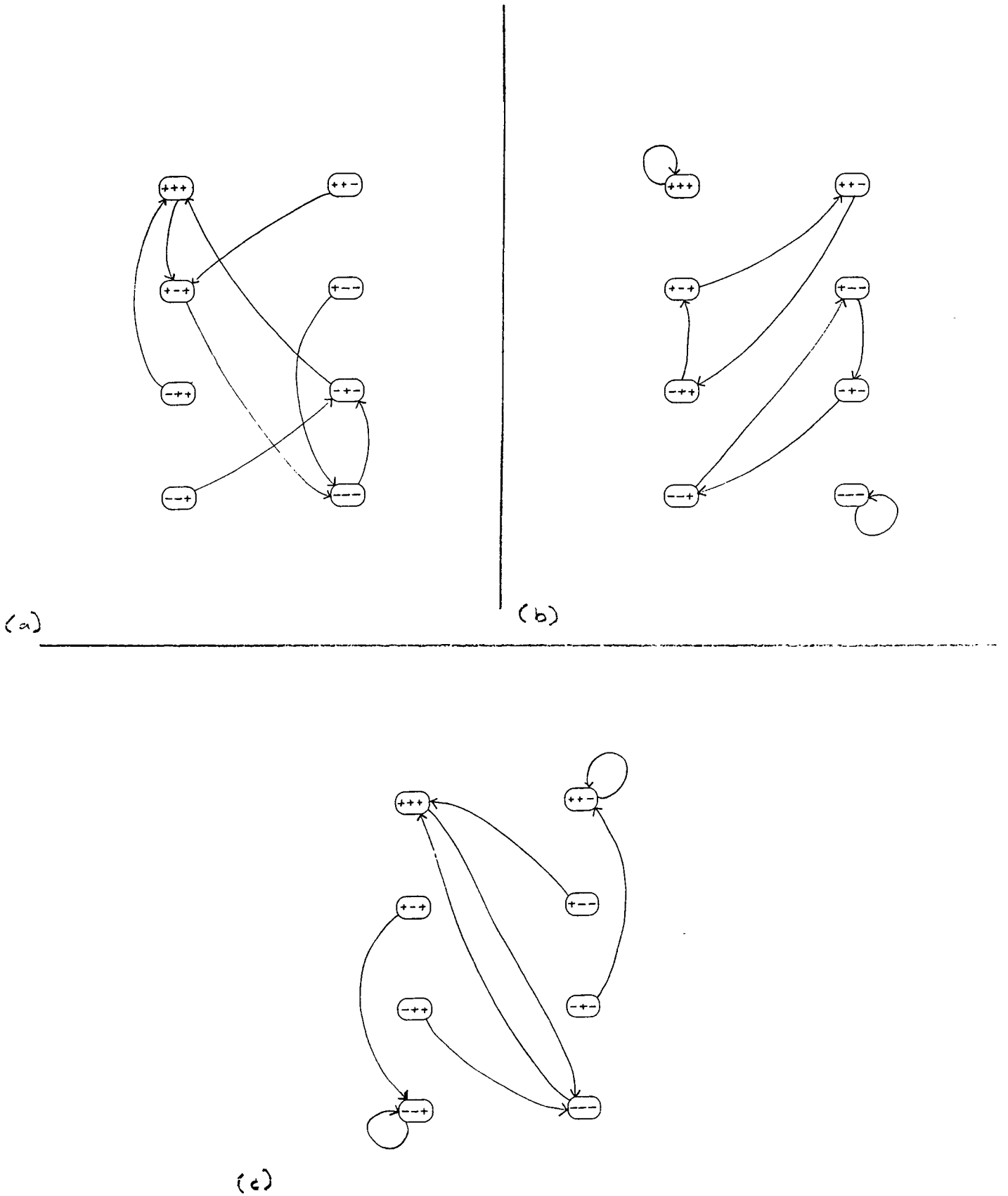
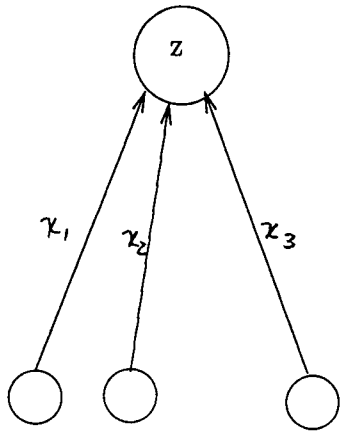


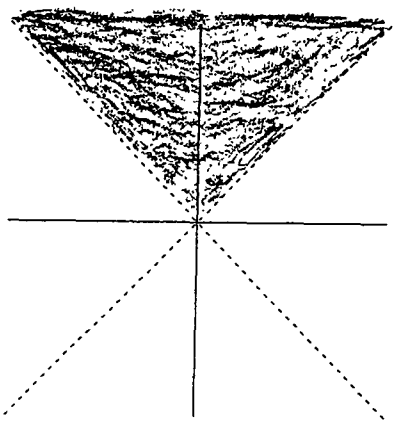
Figure 3



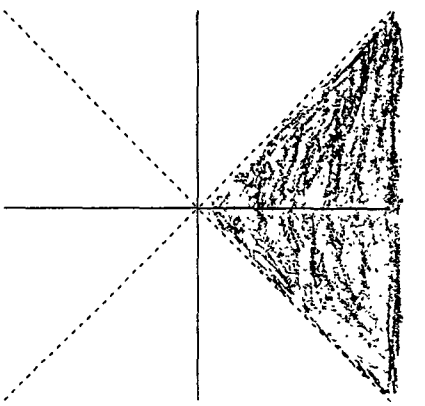
J			ζ
-1	-1	-1	1
-1	-1	1	1
-1	1	-1	1
-1	1	1	-1
1	-1	-1	-1
1	-1	1	1
1	1	-1	-1
1	1	1	-1

An NT-Table for neuron z in a 4-neuron network. Let J_i be row i of matrix J. Then $J_i x \geq 0$ whenever $\zeta_i = -1$. This defines the transitions of unit z.

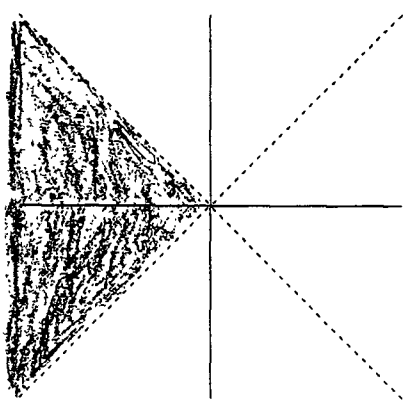
Figure 4



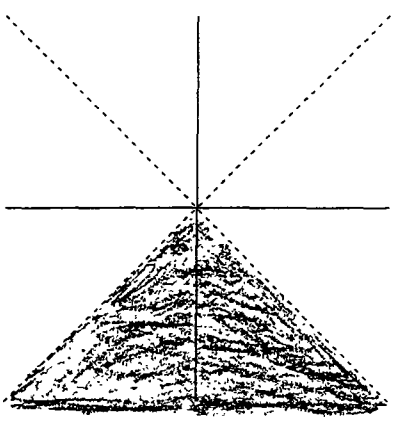
(a)



(b)



(c)



(d)

Figure 5

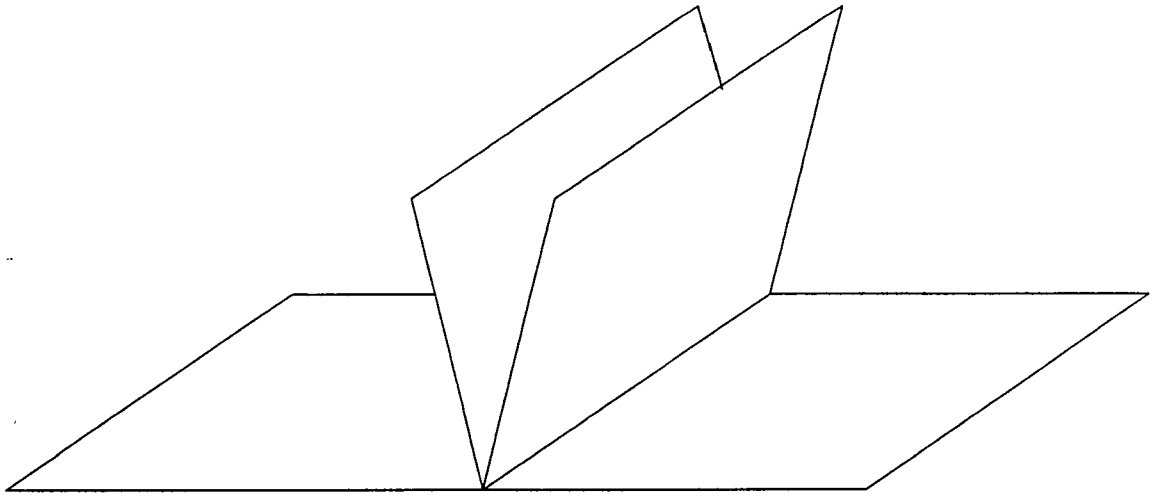


Figure 6

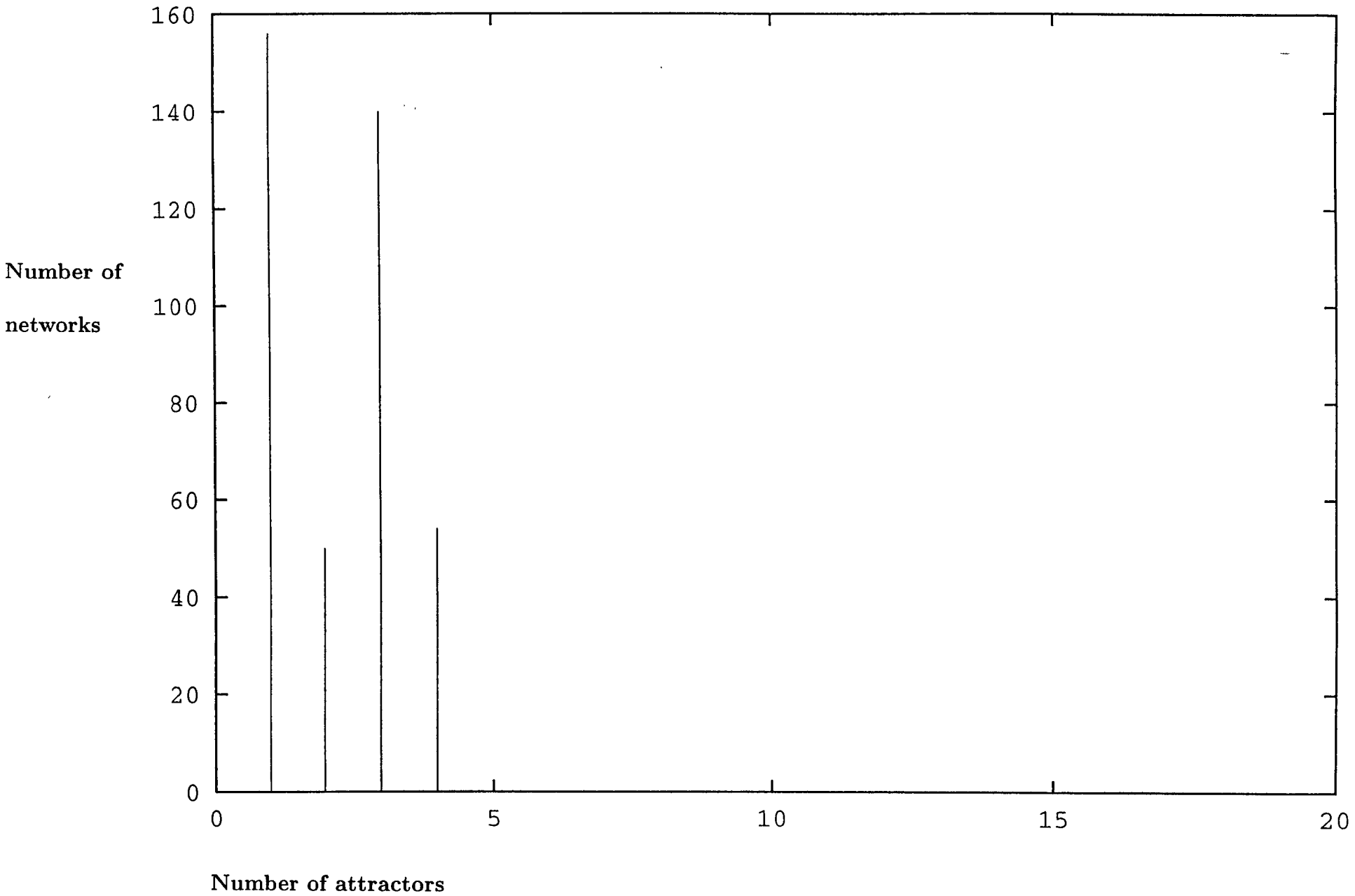

160
140
120
100
80
60
40
20
0
Number of networks
0
5
10
15
20
Number of attractors

Figure 7 (a)

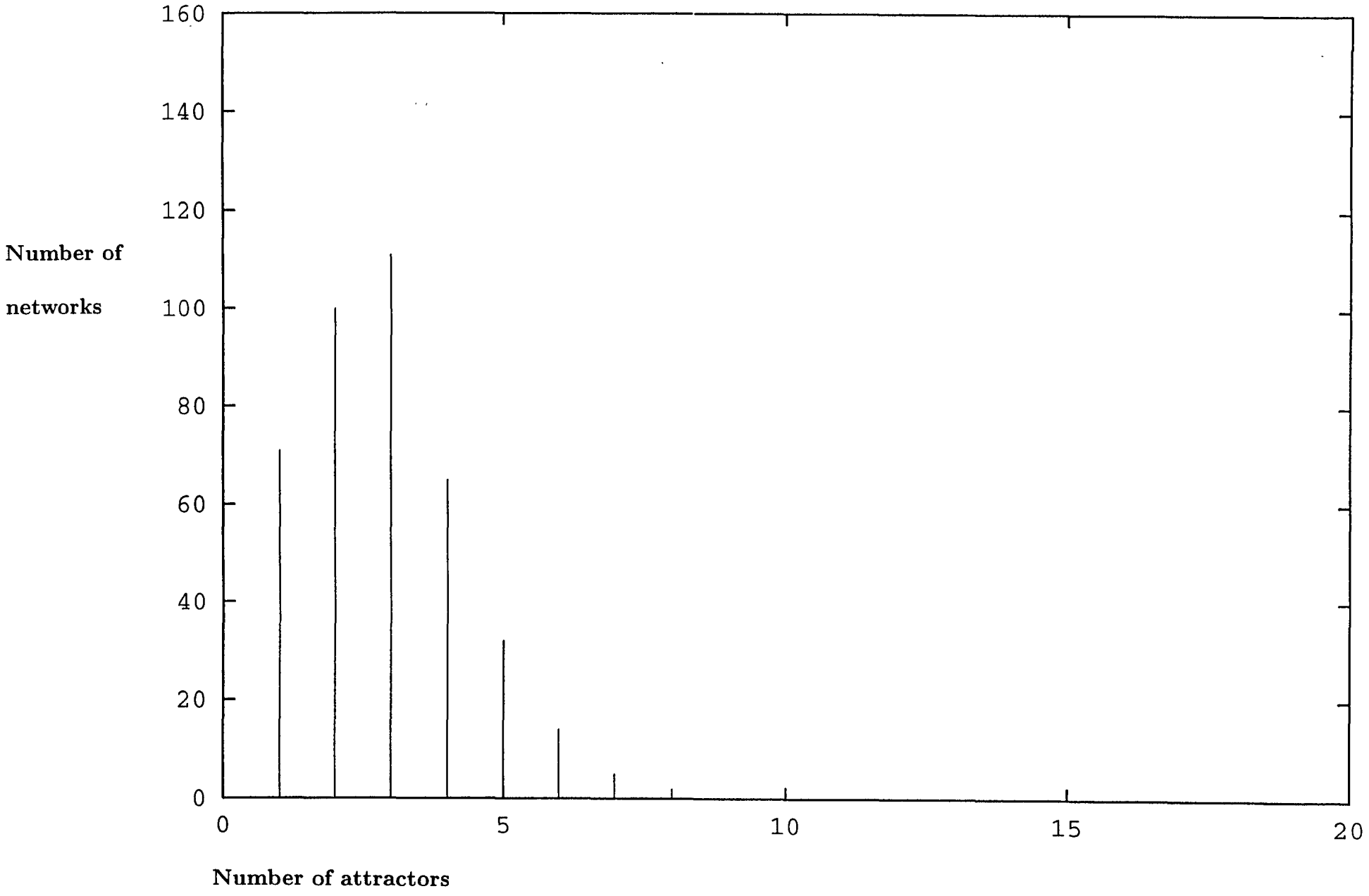

160
140
120
100
80
60
40
20
0
Number of networks
0
5
10
15
20
Number of attractors

Figure 7 (b)

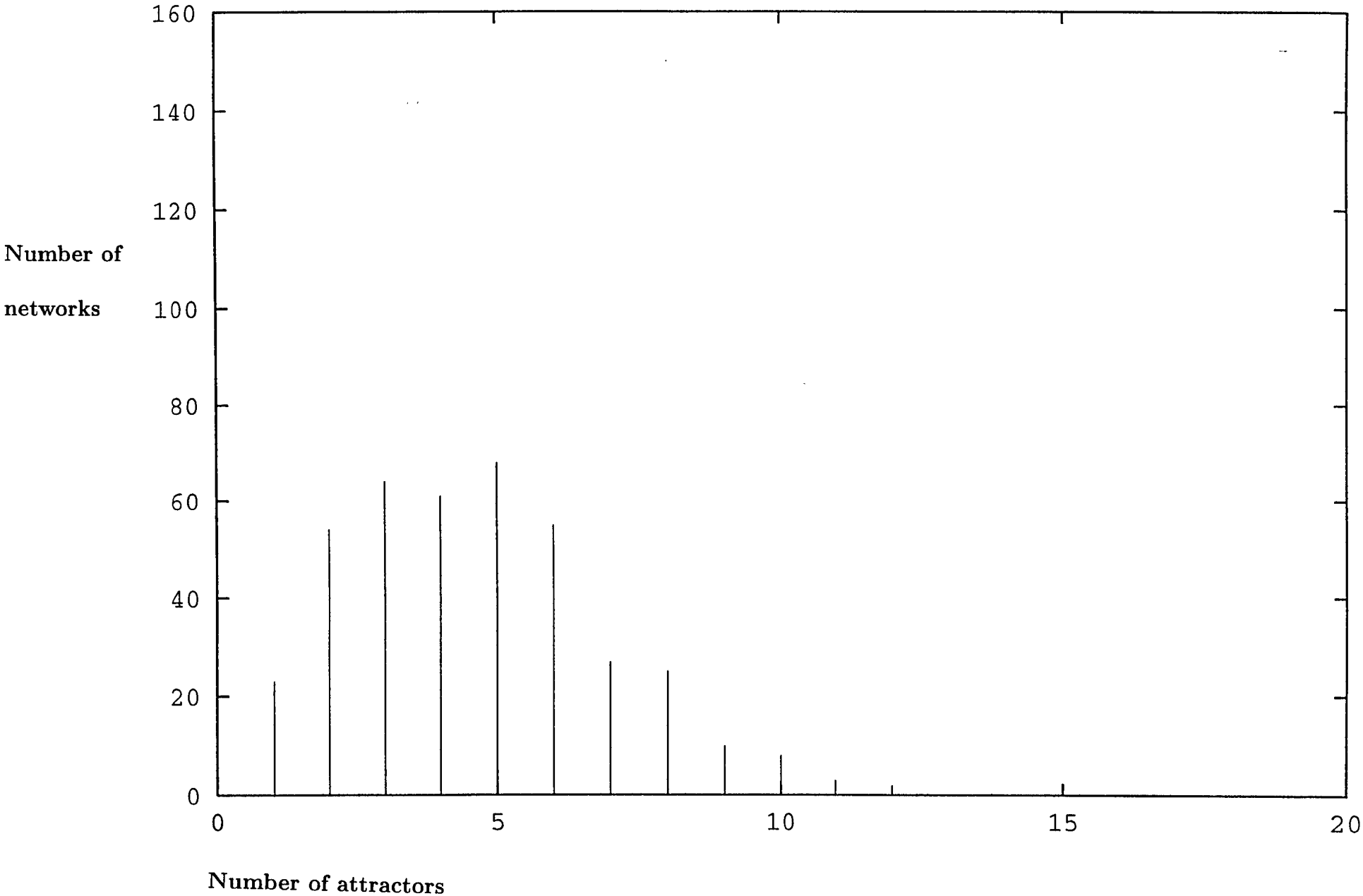

160
140
120
100
80
60
40
20
0
Number of networks
0
5
10
15
20
Number of attractors

Figure 7 (c)

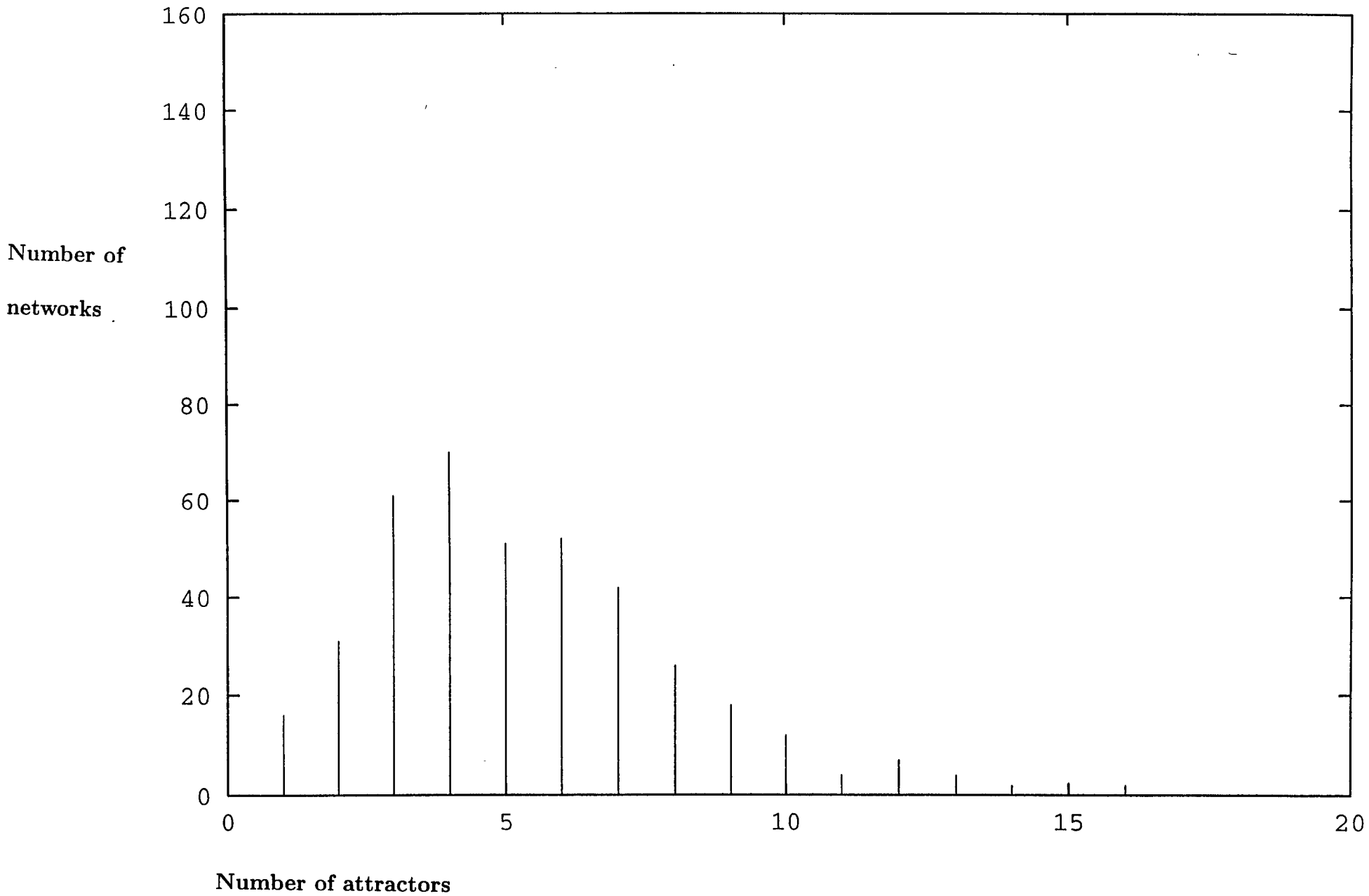

160
140
120
100
80
60
40
20
0
Number of networks
0
5
10
15
20
Number of attractors

Figure 7 (d)

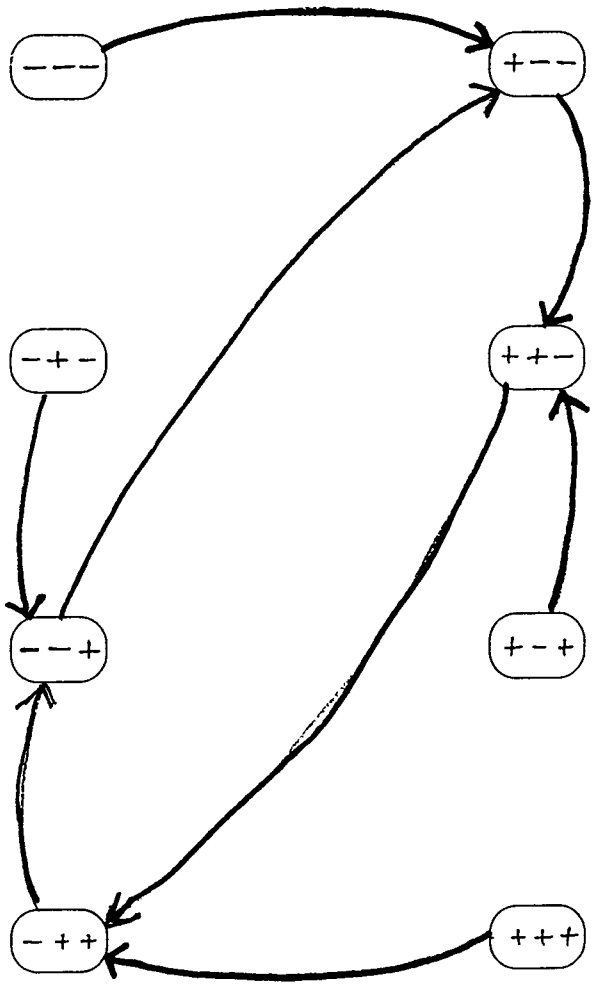


Figure 8(a)

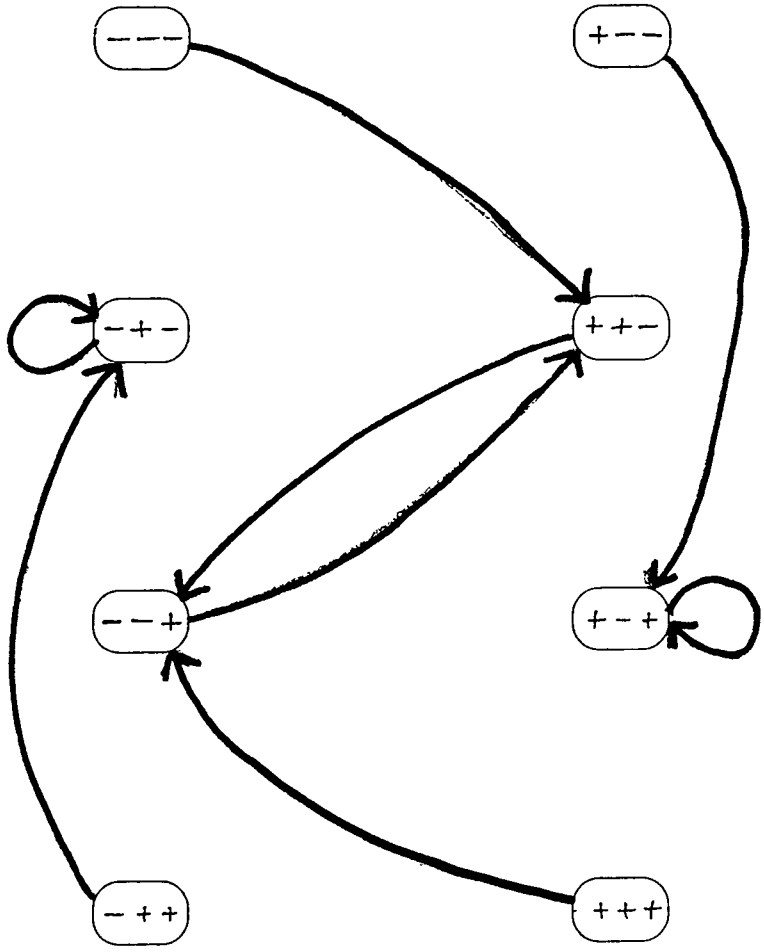


Figure 8(b)

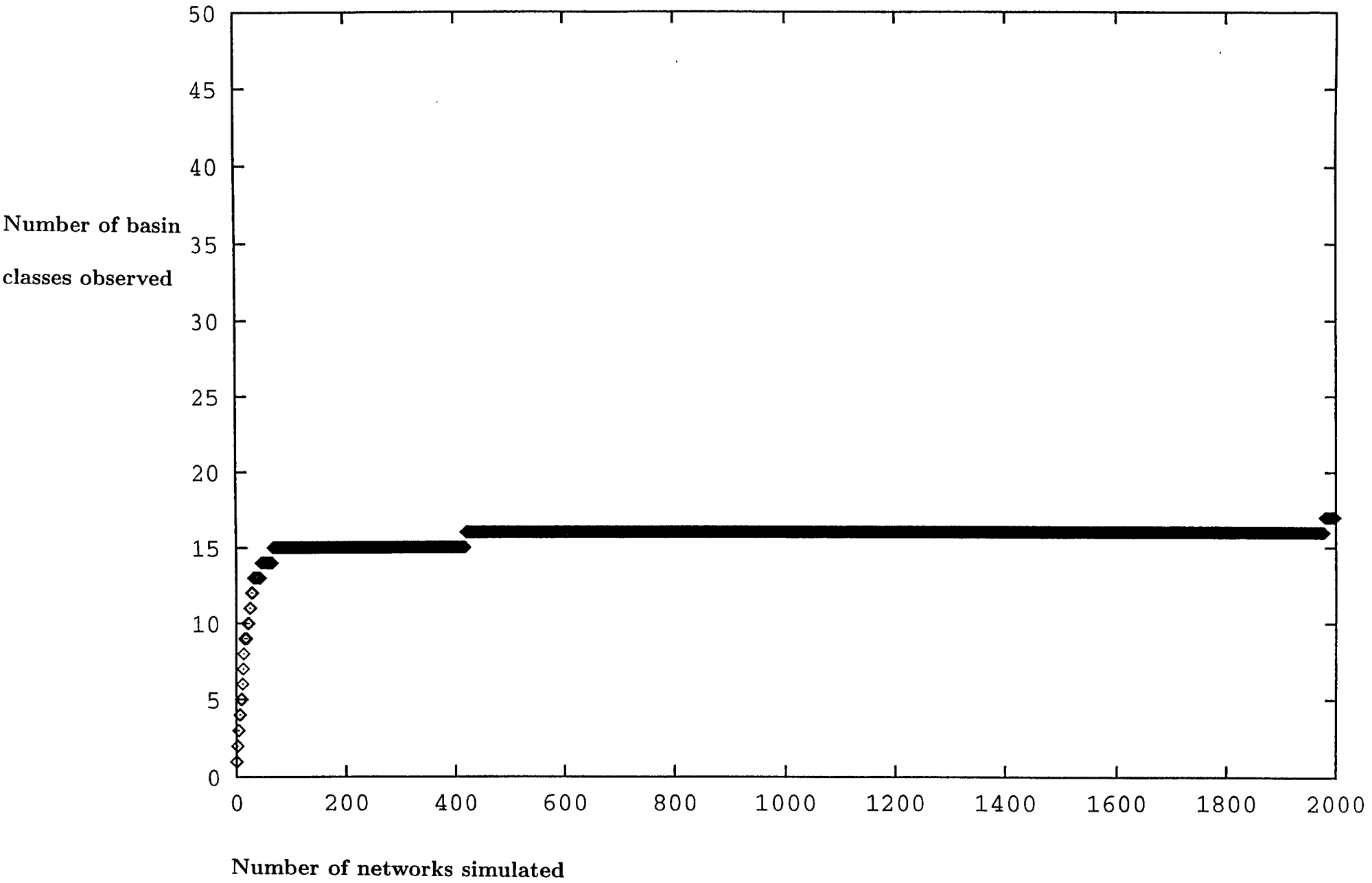

Number of basin classes observed
50
45
40
35
30
25
20
15
10
5
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Number of networks simulated

Figure 9(a)

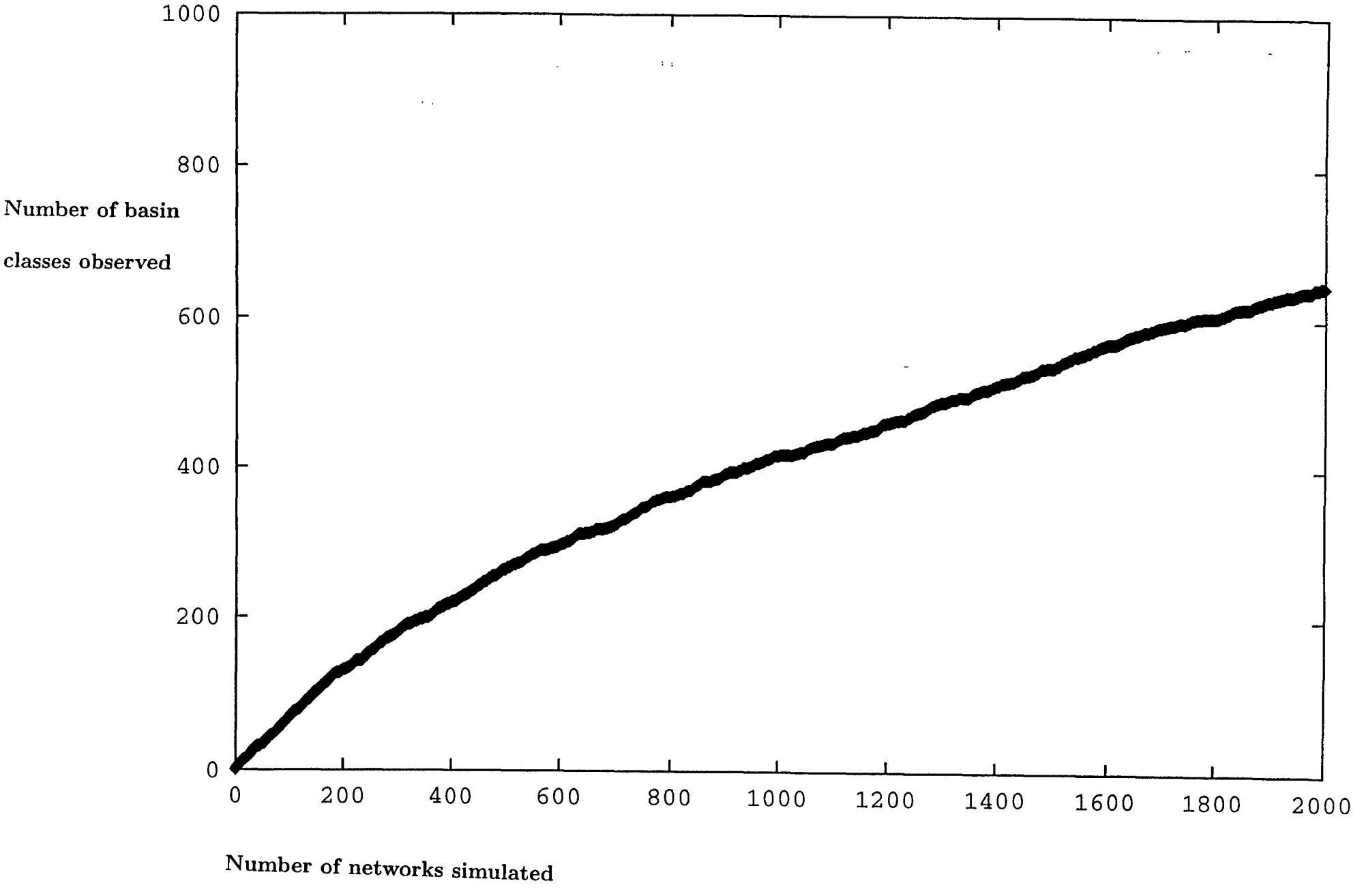

1000
800
600
400
200
0
Number of basin
classes observed
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Number of networks simulated

Figure 9 (b)

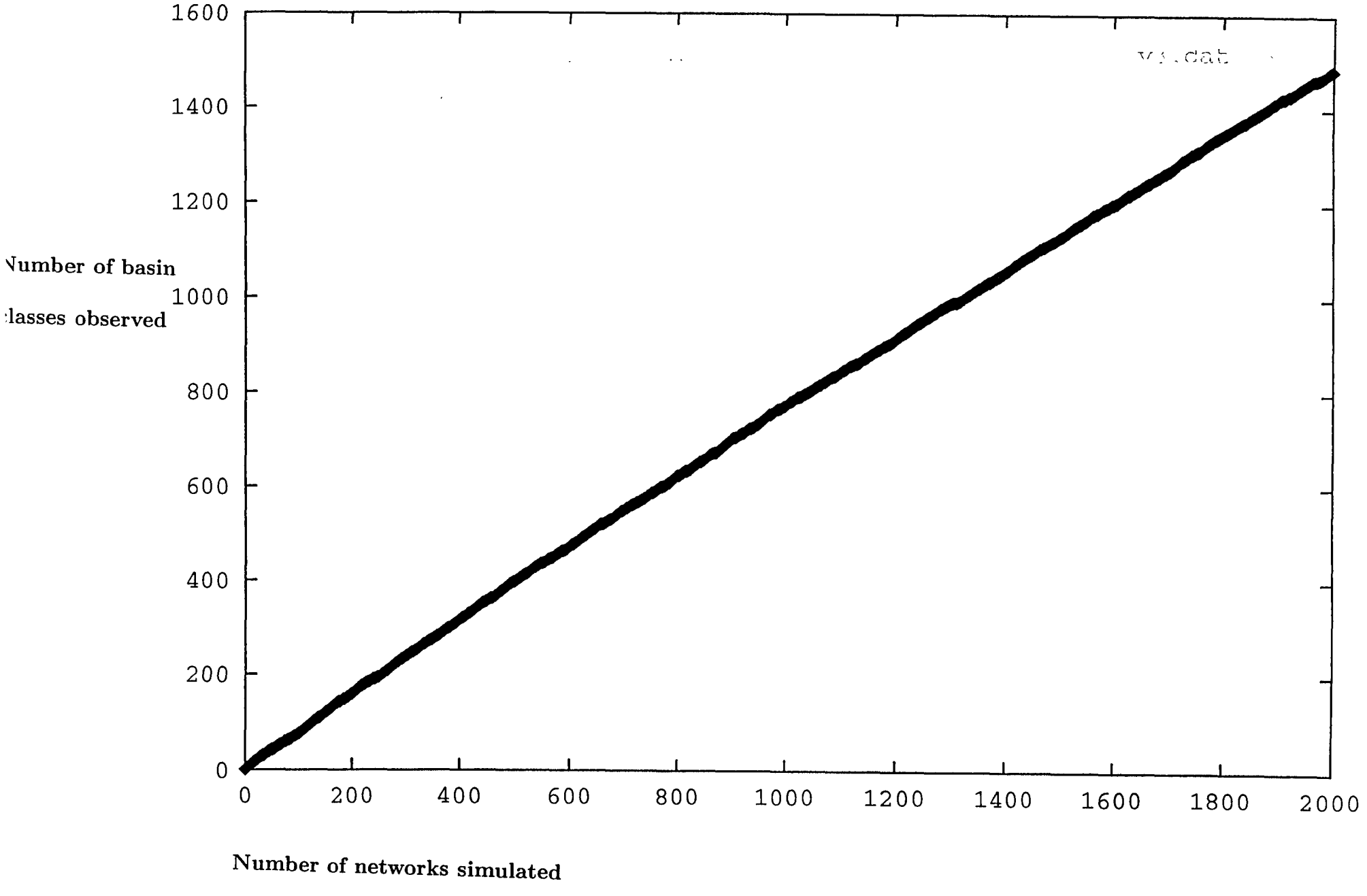

1600
1400
1200
1000
800
600
400
200
0
Number of basin classes observed
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Number of networks simulated

Figure 9(c)

neurons	2^n	$H(n)$	$L(n)$
2	4	2	4
3	8	4	64
4	16	8	4,096
5	32	16	$> 10^6$
6	64	32	$> 10^9$
10	1024	512	$> 10^{18}$
19	524,288	262,144	$> 10^{100}$

Table 1: Synchronous updating, asymmetric reciprocal weights allowed. Column 1 is the number of neurons in the network, Column 2 is the number of nodes in the network transition graph, Column 3 is $H(n) = 2^{n-1}$, a lower bound on the number of NT-tables, and Column 4 is $L(n) = 2^{n(n-1)}$, the lower bound on the number of network transition graphs.

neurons	ave	min	max	f.p.'s	oscill
2	2.03	1	3	1.03	1.00
3	2.23	1	4	0.97	1.26
4	2.56	1	10	1.10	1.46
5	2.90	1	8	1.06	1.84
6	3.20	1	9	1.01	2.19
7	3.43	1	10	0.97	2.46
8	3.81	1	16	1.02	2.79
9	4.22	1	19	1.12	3.10
10	4.63	1	12	0.94	3.69
11	4.94	1	15	0.97	3.97
12	5.41	1	16	1.05	4.36

Table 2: Synchronous updating, asymmetric reciprocal weights allowed. Column 1 gives the number of neurons in the network, Column 2 shows the average number of attractors observed, Columns 3 and 4 show the minimum and maximum number of attractors observed, respectively. Column 5 shows the average number of fixed points, and Column 6 shows the average number of oscillators observed. Four hundred networks of each size were generated with random weights, fully connected with no self-loops.

neurons	ave	min	max	f.p.'s	oscill
2	3.00	3	3	2.00	1.00
3	3.00	3	3	2.00	1.00
4	4.41	3	10	2.38	2.03
5	6.43	3	14	3.00	3.43
6	8.79	3	21	3.48	5.31
7	13.03	3	29	4.33	8.70
8	18.61	3	46	5.35	13.26
9	26.85	5	80	6.14	20.71
10	38.67	7	77	7.61	31.06
11	55.63	14	134	9.49	46.14
12	82.97	21	186	11.68	71.29

Table 3: Synchronous updating, symmetric reciprocal weights Column 1 gives the number of neurons in the network, Column 2 shows the average number of attractors observed, Columns 3 and 4 show the minimum and maximum number of attractors observed, respectively. Column 5 shows the average number of fixed points, and Column 6 shows the average number of oscillators observed. Four hundred networks of each size were generated with random weights, fully connected with no self-loops.

neurons	nets	asym	sym
2	100	3	2
3	100	15	6
4	10,000	1,418	300
5	10,000	7,030	9,355
6	10,000	8,022	10,000

Table 4: Basin classes were counted among simulated networks which were fully connected (no self-loops) with random weights. Column 1 shows the number of neurons in the network, Column 2 shows the number of networks simulated, Column 3 shows the number of basin classes observed when reciprocal weights were allowed to be asymmetric, and Column 4 shows the number of basin classes observed when reciprocal weights were restricted to symmetric values.

Acknowledgements

The first author was supported by the Naval Research Laboratory (N00014-90K-2010) and the National Science Foundation (CDR-88-03012 and BIR9309169). The second author acknowledges support from the Office of Naval Research.