

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING AND TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

**INVESTIGATION OF AUTOMATED TASK
LEARNING, DECOMPOSITION AND SCHEDULING**

By

Principal Investigator: David L. Livingston

and

Graduate Research Assistants: Gursel Serpen
Chandrashekar L. Masti

Final Report
For the period ended February 28, 1990

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NAG-1-962
Donald Soloway, Technical Monitor
ISD-Automation Technology Branch

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

July 1990

Table of Contents

| | |
|---|----|
| Introduction and Organization | 1 |
| 1.1 Introduction | 1 |
| 1.2 Organization | 2 |
| Graph Search | 3 |
| 2.1 Introduction | 3 |
| 2.2 Problem Definition | 4 |
| 2.3 The Boltzmann Machine Implementation | 5 |
| 2.4 Energy Function | 7 |
| 2.5 Example Application | 11 |
| 2.6 Conclusions | 14 |
| Finding Partitions | 18 |
| 3.1 Introduction | 18 |
| 3.2 Constraint Satisfaction Neural Networks | 19 |
| 3.3 Partitions and Decomposition | 20 |
| 3.4 Theoretical Development | 21 |
| 3.5 Derivation of the Energy Function | 21 |
| 3.6 Algebraic Basis of the Derivation | 23 |
| 3.7 The Transitivity Constraint | 23 |

| | |
|--|----|
| 3.8 The Substitution Property Constraint | 30 |
| 3.9 Ground State Characteristic of the Energy Function | 33 |
| 3.10 Interconnection Strengths and the Activation Rule | 34 |
| 3.11 Boltzmann Machine with Simulated Annealing | 34 |
| 3.12 Conclusions | 42 |
| 3.13 Future Research | 43 |
| | |
| Adaptive Constraint Satisfaction | 45 |
| 4.1 Introduction | 45 |
| 4.2 Hopfield Networks and the Adaptation Algorithm | 48 |
| 4.3 An Adaptive Constraint Satisfaction Network | 53 |
| 4.4 Simulation Results | 53 |
| 4.5 Conclusions | 79 |
| 4.6 Future Research | 81 |
| | |
| References | 83 |

INVESTIGATION OF AUTOMATED TASK SCHEDULING AND DECOMPOSITION

by

David L. Livingston¹

Gürsel Serpen²

Chandrashekar L. Masti³

Introduction and Organization

1.1 Introduction

This document reports on the details and results of research conducted in the application of neural networks to task planning and decomposition. Task planning and decomposition are operations that humans perform in a reasonably efficient manner. Without the use of good heuristics and usually much human interaction, automatic planners and decomposers generally do not perform well due to the intractable nature of the problems under consideration. The human-like performance of neural networks has shown promise for generating acceptable solutions to intractable problems such as planning and decomposition. This was the primary reasoning behind attempting the study reported herein.

¹ Assistant Professor, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529.

² Graduate Research Assistant, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529.

³ Graduate Research Assistant, Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529.

The basis for the work is the use of state machines to model tasks. State machine models provide a useful means for examining the structure of tasks since many formal techniques have been developed for their analysis and synthesis. It has been our approach to integrate the strong algebraic foundations of state machines with the heretofore trial-and-error approach to neural network synthesis.

1.2 Organization

The research we have performed can be broken down into three broad categories. The first category reported in section 2 examines the use of a type of neural networks, called constraint satisfaction networks, to plan a task by finding the transfer sequence of a state machine representing the task. The necessary background, theory and an example are presented in this section.

The second category, section 3, deals with using constraint satisfaction networks to find structures called s.p. partitions which are essential in the state machine decomposition process. As in section 2, background, theory and an example are included.

The final category, section 4, deals with a technique that was developed to overcome some of the shortcomings of constraint satisfaction networks used in the search for solutions of the problems studied in the previous two sections. We call this new technique an adaptive constraint satisfaction network and report on some of the primary results achieved at this point.

Graph Search

2.1 Introduction

The main action of task planners is the searching of a very large state space in the presence of constraints. Conducting this search in a serial fashion may not meet the time restrictions of some real-time tasks indicating a need for parallel methods. Neural networks are examples of implementations of algorithms that can perform the constrained state space search; i.e., solve constraint satisfaction problems in a parallel and distributed manner [1].

Two well-known examples of neural networks which perform constraint satisfaction searches are the Hopfield network [2], [3], [4], [5] which is a deterministic method and the Boltzmann machine [6], [7] which performs searches using stochastic techniques. Both paradigms effectively search for a local minimum of a performance function which is realized in the network by the interconnection topology.

In the case of a Hopfield network, once the constraints and the associated parameters have been defined, the local minimum the network settles into is solely determined by its initial conditions and the order the neurons are updated. Hopfield networks require an asynchronous update rule to prevent the network from getting trapped in limit cycles. In practice, asynchronous update is achieved by choosing the neurons for update in random order.

In the case of the Boltzmann machine, each local minimum can be visited with a certain probability irrespective of network initial conditions given that a sufficiently long annealing schedule is employed. The tradeoff for the insensitivity of the Boltzmann machine to its initial conditions is thus the sequential character of the algorithm introduced by the annealing schedule.

The types of task planning problems that are considered in this study are purely constraint satisfaction problems; that is, there are no costs involved. In these types of problems, referred to as syntactic constraint satisfaction problems, local minima of the performance function generally do not correspond to solutions as in the case of optimization problems. Hence it is necessary to find the global minimum of the performance function to find a solution which does not violate any of the constraints. Since the Boltzmann machine convergence process is independent of network initial conditions, it has a better chance of ending up in the global minimum as compared to the Hopfield network. Therefore the Boltzmann machine is preferable over the Hopfield network for syntactic constraint satisfaction problems if the time degradation introduced by the annealing schedule can be tolerated.

2.2 Problem Definition

Our purpose for the research reported herein is to demonstrate the use of constraint satisfaction networks to perform the search for the shortest, viable path between defined initial and final states in a state space. The resulting path represents a plan for executing the task over which the state space is defined. In our analogy, we are searching for the transfer sequence of a state machine for a given initial and final state pair.

A directed graph (digraph) is a functional definition for a state machine [8] and is used as an abstract model for the transfer sequence search problem. Vertices of the graph represent states and directed edges stand for transitions between associated states. By using a proper representation, a constraint satisfaction network can be constructed such that the final state the network settles into represents the shortest path through the digraph.

The shortest path between two vertices of a given directed graph can be defined as a subgraph which meets the following constraints:

- 1) the subgraph representing a path is both irreflexive and asymmetric,
- 2) each vertex except the source and target vertices must have in-degrees and out-degrees of exactly 1,
- 3) the source vertex has in-degree of 0 and out-degree of 1,
- 4) the target vertex has in-degree of 1 and out-degree of 0, and
- 5) the length of the path is equal to that power of the adjacency matrix which has the first nonzero entry in the row and column locations defined by the source and target vertices respectively.

2.3 The Boltzmann Machine Implementation

From a topological viewpoint a Boltzmann machine can be visualized as a symmetric and irreflexive directed graph where graph nodes and weighted edges represent the computation nodes (neurons) and the interconnections respectively. Each neuron output is binary valued. The activation function is defined as

$$p_i(s_i=1) = \frac{1}{1 + e^{-\frac{net_i}{T}}}$$

where p_i is the probability that s_i , the activation of neuron i , is equal to 1, T is a time-varying computational parameter analogous to temperature and net_i is the input sum to unit i . The term net_i for a typical second-order machine is defined by

$$net_i = \sum_j w_{ij} s_j + b_i,$$

where w_{ij} is the connection weight between neurons s_i and s_j , and b_i is a bias for s_i .

Neurons in a Boltzmann machine are updated asynchronously. As previously stated, the parameter T is analogous to temperature. A Boltzmann machine is able to escape local minima through the use of "thermally" induced noise. The temperature parameter is assigned a large initial value and is decreased at discrete time steps until it reaches a predetermined minimum value. This process effectively starts the machine in a very noisy mode and gradually reduces the noise in a manner analogous to annealing. A sufficient condition for converging to a global minimum with probability approaching one is to use an annealing schedule of the form

$$T(k) = \frac{T_0}{\log(1 + k)},$$

where T_0 is a sufficiently high initial temperature and the discrete time-step k approaches infinity [9]. Since a practical application requires a finite convergence time, an upper bound for the probability of converging to the global minimum is established by the time-limitations of the problem under consideration. This upper bound is necessarily less than one.

The probability of being in the state S_n , where a state is defined to be the vector of the activation values of all neurons in the network, is given by

$$P(S_n) = \frac{e^{-\frac{E(S_n)}{T}}}{\sum_i e^{-\frac{E(S_i)}{T}}},$$

where $E(S_n)$ is the energy of the network associated with the state S_n and index i implies a sum over 2^N states of an N -neuron network [6]. Thus the network will tend to settle into a state corresponding to a low value of the energy function with high probability. If the energy function is defined such that the minimum energy values correspond to the states which meet the constraints of the problem, then the Boltzmann machine will seek out and settle into those states with high probability.

The network topology employed to search for the shortest path in a digraph is an $N \times N$ array of neurons representing the adjacency matrix of a given directed graph. Each neuron in the network stands for an entry of the adjacency matrix and thus for an edge of the directed graph.

2.4 Energy Function

The general form of the quadratic performance function a Boltzmann machine minimizes is

$$E(S) = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j + \sum_i b_i s_i,$$

where w_{ij} is the weight between neurons s_i and s_j and b_i is a bias. The weight can be defined by

$$w_{ij} = \sum_n K_n \delta_{nij},$$

where $K_n \in R^+$ if the hypotheses both nodes represent for constraint n are mutually supporting and $K_n \in R^-$ if they are mutually conflicting. The term δ_{nij} is equal to 1 if the two hypotheses represented by s_i and s_j are related under constraint n and is 0 otherwise.

Given the graph-theoretic constraints a path specification has to satisfy, the next step is to define the corresponding topological constraints of the path search problem for an adjacency matrix topology of the neural network.

An irreflexive graph has all diagonal entries of its adjacency matrix equal to zero, which equivalently translates into clamping all neuron outputs along the main diagonal of the Boltzmann machine to 0.

Graph asymmetry (constraint #1) requires that only one of the two entries located at symmetric positions with respect to the main diagonal of the adjacency matrix be equal to 1. Hence $\delta_{1ij} = 1$ and $K_1 \in R^-$ if and only if the row index of node s_i equals the column index of node s_j and the column index of node s_i equals the row index of node s_j ; otherwise $\delta_{1ij} = 0$. Figure 2.1 illustrates the implementation of this constraint.

A digraph node with an in-degree of 1 implies the existence of a single 1 in the associated column of the adjacency matrix (constraint #2). The term $\delta_{2ij} = 1$ and $K_2 \in R^-$ if and only if the column index of node s_i equals the column index of node s_j ; otherwise $\delta_{2ij} = 0$.

Similarly, a digraph node with an out-degree of 1 requires only a single 1 to exist in the associated row of the adjacency matrix (constraint #3). The term $\delta_{3ij} = 1$ and $K_3 \in R^-$ if and only if the row index of node s_i equals the row index of node s_j ; otherwise $\delta_{3ij} = 0$. Constraints #2 and #3 are illustrated in figure 2.2.

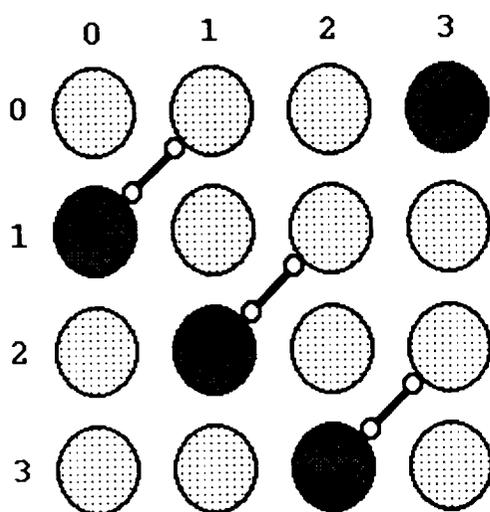


Figure 2.1. Constraint #1.

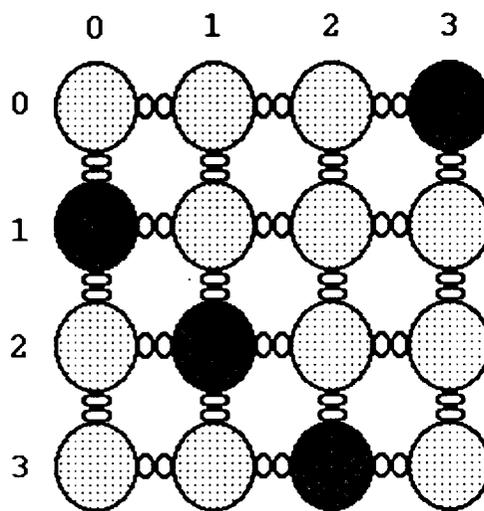


Figure 2.2. Constraints #2 and #3.

A path specification imposes the condition that any node that is not a source or target and is included in the path must have an in-degree and out-degree of 1 (constraint #4). In terms of the adjacency matrix, if there exists a 1 in a particular row/column, then there must exist a 1 in the column/row that has the same node label as the corresponding row/column. The term $\delta_{4ij} = 1$ and $K_4 \in R^+$ if and only if the row index of node s_i equals the column index of node s_j or the column index of node s_i equals the row index of node s_j ; otherwise $\delta_{4ij} = 0$. This constraint results in an excitatory connection as shown in figure 2.3.

2.5 Example Application

In order to test the proposed network, a digraph with a specially defined adjacency matrix was employed. It was assumed that the adjacency matrix of the digraph had all its lower triangular entries and the entries above the main diagonal equal to 1; e.g., let a_{ij} , the entry in the i -th row and j -th column of a matrix, belong to \mathbf{A} , the adjacency matrix of the digraph, then $a_{ij} = 1$ if and only if $i \leq j + 1$. The adjacency matrix for an example digraph with 10 vertices is shown in Table 2.1. Note that the entries which are equal to 0 are equivalently clamped to 0 in the Boltzmann machine network.

The digraph specification yields a path of length $N-1$ between vertices V_0 and V_{N-1} . Thus this is the largest possible path for an N -vertex digraph. Another feature of this digraph is that there are circuits of all possible lengths implying a difficult state space to search since all those circuits will be mapped to local minima of the quadratic performance function.

Since it is known that a path is irreflexive, we can clamp the neurons which are located along the main diagonal to 0 given that they represent the hypothesis; the path has a self-loop

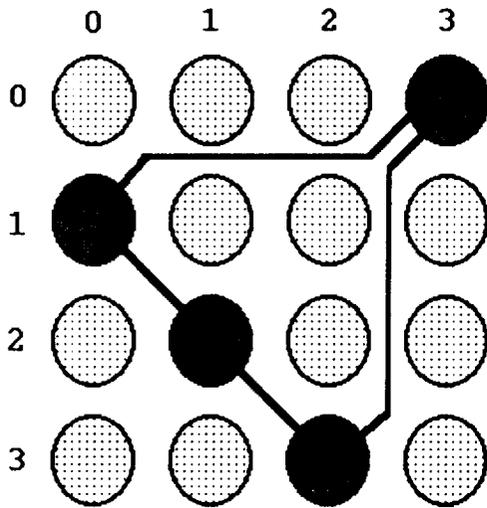


Figure 2.3. Constraint #4.

| | V_0 | V_1 | V_2 | V_3 | V_4 | V_5 | V_6 | V_7 | V_8 | V_9 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| V_0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_2 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_3 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| V_4 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| V_5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| V_6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| V_7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| V_8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| V_9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 2.1. The adjacency matrix of a specially defined digraph.

for vertex V_i . In order to map the in-degree of 0 for a source vertex, the neurons belonging to the column labelled by that vertex are clamped to 0 and similarly, the neurons of the row labelled by the target vertex are clamped to 0 to realize the requirement that the out-degree of the target vertex is equal to 0.

Assume that we are looking for a path starting with vertex V_0 and ending with vertex V_9 . Clearly, there exists only one path which includes the adjacency matrix entries above the main diagonal as depicted by table 2.2. This path represents the solution to the planning problem.

The neuron outputs which take part in computations were randomly initialized. The initial starting temperature was selected such that at least 80% of the neuron output probabilities belonged to the interval of reals given by $[0.4, 0.6]$. The set of parameters listed in table 2.3 were determined by trial and error and resulted in convergence to the state vector which represented the solution illustrated in table 2.2 in over 90% of all trial runs.

2.6 Conclusions

We have demonstrated the use of a second-order Boltzmann machine for the shortest path search in a directed graph. One important difficulty of employing neural networks to solve constraint satisfaction problems is the lack of a proper methodology to map a given problem into the network domain. It seems that discrete mathematics may provide a rich source for abstract tools to help with the mapping problem.

Another issue is the need for a heuristic approach combined with a trial and error search for the determination of the correct set of values for the gain parameters of the energy function. Chapter 4 details the incorporation of an adaptive component to the constraint satisfaction search

algorithm so that the neural network can learn while searching. This eliminates the need for the trial and error search that is required to determine the values of the gain parameters.

| | V_0 | V_1 | V_2 | V_3 | V_4 | V_5 | V_6 | V_7 | V_8 | V_9 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| V_0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_3 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| V_4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| V_5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| V_6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| V_7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| V_8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| V_9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 2.2. The path between V_0 and V_9 .

| Parameter | Gain |
|-----------|------|
| K_1 | -1 |
| K_2 | -2 |
| K_3 | -2 |
| K_4 | 1 |
| I | 1 |

Table 2.3. Parameter values.

Finding Partitions

3.1 Introduction

A significant portion of the effort in generating a decomposition of a task consists of the combinatorially explosive problem of finding the elements of the general structure from which the decompositions are obtained. The problem of decomposition has been approached from an algebraic perspective that essentially relies on partition algebra. Hartmanis and Stearns [10] pioneered the use of partition algebra for analyzing the structure of sequential machines. Partitions generated over state machines determine the possibility of decompositions and provide an understanding of their structure. Currently existing techniques for obtaining machine decompositions are based on performing an exhaustive sequential search into the machine's transition characteristics governed by a given input environment. This process is known to become intractable with growth in problem size and consequentially imposes intemperate demands on computation resources.

The recent resurgence of artificial neural nets has drawn keen interest from groups of researchers who have observed that these models demonstrate capabilities in addressing computationally "hard" problems including the travelling salesman problem [2], the graph partitioning problem [11] and the N-queens problem [12]. This new development offers a clear motivation for furthering a serious exploration into the potential of a neural approach to addressing the NP-hard machine decomposition problem.

3.2 Constraint Satisfaction Neural Networks

Artificial neural net models are also known as parallel distributed processing (PDP) or connectionist models [1]. The fundamental assumption for these models is that the information processing takes place through interactions between a large number of simple processing elements called units or neurons, where each can send either excitatory or inhibitory signals to other units in the system. In the application of neural nets to constraint satisfaction, the individual neurons are themselves used to represent hypotheses. The activations of the neurons are analogous to the validity associated with the different possible hypotheses. The constraints known to exist between the different hypotheses are represented by weighted interconnections between the neurons. When neural net models are used in this fashion they demonstrate the capability for performing optimization.

The computation of solutions to constraint satisfaction problems by connectionist networks is performed by an iterative relaxation search which starts with a randomly chosen initial state. This state can be interpreted as a proposed solution which the network progressively improves by reducing a well defined "objective" or "energy" function. The eventual low-energy (minimal) states to which the network converges represent the required "good" or valid solutions. The energy function defined for the network measures the extent to which the current interpretation violates the stipulated constraints. Each possible state of activity of the network has an associated energy. The activation rule used for updating activity levels of the neurons is so chosen that this global network energy shows a general decline with every iteration.

3.3 Partitions and Decomposition

A partition π on a set S is a division of S into disjoint subsets $S_k \ni S_i \cap S_j = \emptyset, \forall i \neq j$ and $S = \cup S_k$. Each S_k is called a block of π . It is usual to represent a partition by writing overlines atop the disjoint subsets and separating them by semicolons so that the subsets appear as blocks of set members. An example of a partition is:

$$\pi = \{ \overline{0, 2, 4}; \overline{1, 3, 5} \} .$$

The theory of finite state machine structure is based on laws derived from partition algebra. Mathematically defined, a finite state machine M is characterized by a three-tuple $M = \langle S, I; \{\delta\} \rangle$, where S is the state set of the machine M , I is its input set and $\{\delta\}$ is a set of state transition functions (or "next-state" mappings). State machines may be decomposed as determined by certain special properties exhibited by the partitions generated over their state spaces, the most significant among them being the substitution property. With respect to a state machine characterized by a three-tuple as defined above, we present a formal definition of the property of substitution as satisfied by a partition over the state set of a finite state machine.

A partition on a state set S of a state machine M is said to satisfy the substitution property (s.p.) and is denoted by π if, $\forall s_i, s_j \in S$ which are in the same block of π and any input $i_k \in I$, the states $\delta(i_k, s_i)$ and $\delta(i_k, s_j)$ are also in a common block of π .

For any n -state set $S = \{s_1, s_2, \dots, s_n\}$ of a machine M , there always exist two trivial s.p. partitions denoted by $\pi(0)$ and $\pi(I)$, where

$$\pi(0) = \{ \overline{s_1}; \overline{s_2}; \overline{s_3}; \dots; \overline{s_n} \} \text{ and } \pi(I) = \{ \overline{s_1, s_2, s_3, \dots, s_n} \} .$$

Decomposition theory guarantees that if the product of any two s.p. partitions for a state machine \mathbf{M} equals the trivial s.p. partition $\pi(0)$, then \mathbf{M} can be directly decomposed into two independent machines \mathbf{M}_1 and \mathbf{M}_2 operating in parallel.

3.4 Theoretical Development

In developing a neural approach to any problem, the issue of a proper representation merits primary attention. Partitions have a one-to-one correspondence with relations. We therefore use relation matrices to serve as the representation for partitions proposed by the network. The network is organized into a system of $N \times N$ neurons for a problem space of state dimension N . This means that a neuron in the "on" state at row- i , column- j is suggesting an equivalence between states i and j of the state machine subject to the incorporated constraints.

3.5 Derivation of the Energy Function

After addressing the issue of representation, the next step is to establish the interconnections between the neurons of the network. This is done by deriving the network energy function and then establishing a one-on-one correspondence of terms with those in the classical expression for the energy function of appropriate order. The derivation of the network energy function and a determination of its order are based on the identification of the constraint terms involved for the problem.

Partitions satisfying the substitution property define uniquely corresponding congruence relations. By definition, congruence relations are implicitly equivalence relations with the added requirement that they imply image equivalence whenever states are established as equivalent.

Thus, s.p. partitions must possess the equivalence relation properties: reflexivity, symmetry and transitivity as well as the (implication) properties of image equivalence.

The reflexive and symmetric relations can be implicitly encoded in the $N \times N$ representation scheme, by employing the following method:

1. Diagonal neurons in the $N \times N$ neuronal grid are clamped to remain in the "ON" state. Since this is representative of the equivalence of a state with itself, the reflexive relation is satisfied before the network starts the computation process.
2. Symmetry can be incorporated by ensuring that the states of off-diagonal neurons in the in the upper triangular portion of the $N \times N$ grid maintain exact correspondence with the neuron states in the lower triangular portion.

With the above method the search space for solution points in given problem domains becomes noticeably reduced since the two constraint terms which would have been necessary to enforce the reflexive and symmetry properties have been obviated.

The remaining properties are therefore transitivity and the substitution property. Thus, the network energy function may be interpreted to consist of essentially two parts. The first part encodes the transitivity constraint. The second part encodes the state-image congruence constraint, i.e., the substitution property.

The second part of the network energy function must essentially map the complete state transition behavior of the state machine under its stimulus (input) set to the network dynamics. Thus, if we denote the energy function of the network by E , we may write

$$E = k_1 E_1 + k_2 E_2,$$

where E_1 is the energy term due to the transitivity constraint and E_2 is the energy term due to the

state-image congruence (s.p. property or next state function) constraint. The parameters k_1 and k_2 are known as gain terms for the respective constraints. They determine the relative importance or weight that is assigned to each constraint.

3.6 Algebraic Basis of the Derivation

The neurons in the $N \times N$ network have only one of two possible states to which they can eventually converge: the "ON" state or the "OFF" state. This means that the ultimately stable solution states of the neural net are binary. We therefore use techniques from Boolean algebra to derive the required functional dependencies of the constraint terms on the neuron activation states, which we may regard as binary variables for our purpose of derivation. This is the basis that provides for a systematic and algebraic method of derivation for the network energy function.

3.7 The Transitivity Constraint

A relation R is transitive iff $a R b$ and $b R c \Rightarrow a R c$, $\forall a, b, c \in (\text{state}) \text{ set } S$ over which R is defined. In other words, this means that if state "a" \equiv state "b" and state "b" \equiv state "c", then state "a" must also be \equiv to state "c". Therefore, the third neuron responsible for representing equivalence between the states "a" and "c" in the $N \times N$ topological representation scheme for the network must be constrained to be "ON" whenever the pair of neurons representing equivalence between states "ab" and "bc" are in their "ON" states. This is a third-order functional dependence of the transitivity constraint term on the activation states of neurons in the network.

The above discussion leads to the conclusion that the transitivity constraint cannot be enforced into the network by a function that is of the more conventionally used quadratic order. We thus state a theorem which mathematically establishes that the transitivity constraint must be a third-order function of the neuron activation states.

Theorem

Third-order interconnections are required for an $N \times N$ topological neural net to verify the relation of transitivity in proposed partitions.

Proof

Defining Boolean matrix multiplications over n -th order square matrices A and B by

$$c_{ij} = \bigvee_{k=1}^n a_{ik} \wedge b_{kj} , \quad (1)$$

where the \vee and \wedge represent bit-ORing and bit-ANDing operations respectively, we see that the matrix C = AB contains a "1" in the row-column position indexed ij whenever a "1" exists in row- i , column- k in matrix A and row- k , column- j in matrix B $\forall k \in [1, n]$. For a relation that is reflexive and symmetric, its relation matrix M_R will encode transitivity in R [13] iff:

$$M^2 = M \times M = M_R . \quad (2)$$

The matrix M_R is a Boolean matrix because its elements are only 1's or 0's. Thus, using equation (1)

$$\tilde{m}_{ij} \in M^2 = M \times M \quad (3)$$

can be written

$$\tilde{m}_{ij} = \bigvee_{k=1}^n m_{ik} \wedge m_{kj} . \quad (4)$$

Equation (2) essentially means that $\forall k \in [1, n]$ with $m_{ij} \in M_R$,

$$\begin{aligned} \tilde{m}_{ij} \oplus m_{ij} &\doteq 0 \\ \Rightarrow (\tilde{m}_{ij} \wedge m'_{ij}) \vee (\tilde{m}'_{ij} \wedge m_{ij}) &\doteq 0 . \end{aligned} \quad (5)$$

Since the variables involved are binary, we have the following three results that can map Boolean logic operations $\forall x, y \in \{0, 1\}$, into the domain of integer arithmetic operations: The logical complement of a Boolean (binary) variable is

$$x' = (1 - x) . \quad (i)$$

The expression

$$x \wedge y = x \times y = xy \quad (ii)$$

equates the logical "AND" operation to the operation of multiplication in the integer-number domain. Finally, the equivalent expression for the logical "OR" operation is derived using the above two results (i) and (ii) with the second theorem of De-Morgan:

$$(x' \wedge y')' = (x \vee y) ,$$

yielding

$$\begin{aligned}
x \vee y &= (x' \wedge y')' \\
&= ((1 - x) \wedge (1 - y))' \\
&= ((1 - x) \times (1 - y))' \\
&= (1 - y - x + x \times y)' \\
&= (1 - 1 + x + y - xy) \\
&= (x + y - xy) .
\end{aligned} \tag{iii}$$

Thus, equation (5) may be rewritten as

$$\tilde{m}_{ij} (1 - m_{ij}) \vee (1 - \tilde{m}_{ij}) m_{ij} \doteq 0 \tag{6}$$

which when simplified using the results (i), (ii) and (iii) transforms to

$$\begin{aligned}
(\tilde{m}_{ij} - \tilde{m}_{ij} m_{ij}) \vee (m_{ij} - \tilde{m}_{ij} m_{ij}) &\doteq 0 \\
\rightarrow m_{ij} + \tilde{m}_{ij} - 2 m_{ij} \tilde{m}_{ij} &\doteq 0 .
\end{aligned} \tag{7}$$

Thus, based on all the above results developed, equation (4) may be written in the form

$$\tilde{m}_{ij} = \sum_{k=1}^n m_{ik} m_{kj} \tag{8}$$

Upon substituting in equation (7), we finally obtain

$$m_{ij} + \sum_{k=1}^n m_{ik} m_{kj} - 2 \sum_{k=1}^n m_{ij} m_{ik} m_{kj} = 0 .$$

This is an equation of third order in $m \in M$. ■

We now derive the transitivity constraint term using the result established by the above theorem and the laws of Boolean algebra. Since the activation states of the neurons in the system are binary variables, we enumerate all possible combinations of activation values considering triplets of neurons (V_{ij}, V_{jk}, V_{ik}) in the form of a classical truth table. This is shown in table 3.1.

The transitivity constraint term (E_T) assigns a penalty of (positive) unity to the network energy function (E) in situations where the required neuron combinations violate the definition of transitivity. Conformation to the definition of the relation of transitivity obviates the penalty.

To derive the functional dependency of the constraint term E_T on the neuron triple (V_{ij} , V_{jk} , V_{ik}), we draw the Karnaugh-map as shown in figure 3.1.

Using the rules of Boolean algebra to obtain the classical sum-of-products form, for the constraint E_T , we may write $\forall i, j, k$:

$$\tilde{E}_1 = (V_{ij} \wedge V_{jk} \wedge V'_{ik}) \vee (V_{ij} \wedge V'_{jk} \wedge V_{ik}) \vee (V'_{ij} \wedge V_{jk} \wedge V_{ik}) ,$$

which upon using the results

$$\begin{aligned} x' &= (1 - x) , \\ x \wedge y &= xy , \\ x \vee y &= (x + y - xy) , \end{aligned}$$

may be written in the form

$$\tilde{E}_1 = V_{ij}V_{jk} + V_{jk}V_{ik} + V_{ij}V_{ik} - 3V_{ij}V_{jk}V_{ik} .$$

Since the representation used is the $N \times N$ network topology, the indices i, j, k span only the upper triangular portion of the square grid of $N \times N$ neurons in the system (reference index starting at 0). Thus, the last equation may be compacted to the following overall form:

$$E_1 = \sum_{i=0}^{(n-3)} \sum_{j=(i+1)}^{(n-2)} \sum_{k=(j+1)}^{(n-1)} \tilde{E}_1 .$$

| Neuron V_{ij} | Neuron V_{jk} | Neuron V_{ik} | Transitivity Law: Violated ? | Constraint E_1 value (1 or 0) |
|-----------------|-----------------|-----------------|---------------------------------|------------------------------------|
| 0 | 0 | 0 | No | 0 |
| 0 | 0 | 1 | No | 0 |
| 0 | 1 | 0 | No | 0 |
| 0 | 1 | 1 | Yes | 1 |
| 1 | 0 | 0 | No | 0 |
| 1 | 0 | 1 | Yes | 1 |
| 1 | 1 | 0 | Yes | 1 |
| 1 | 1 | 1 | No | 0 |

Table 3.1. Transitivity function truth table $\forall V_{ij}, V_{jk}, V_{ik} \in N \times N$ neuron grid.

| | | V_{ij} | | V_{jk} | |
|----------|---|----------|----|----------|----|
| | | 00 | 01 | 11 | 10 |
| V_{ik} | 0 | 0 | 0 | 1 | 0 |
| | 1 | 0 | 1 | 0 | 1 |

Figure 3.1. Karnaugh map of transitivity truth values.

The equation for the transitivity constraint term E_T is of third order as required. We note that the equation sums only for indices spanning the upper triangular portion of the $N \times N$ topology of neurons. This is useful for achieving a reduction in computation overheads. The equation ensures that E_T generates a +1 contribution to the network energy \forall triples of neurons (V_{ij}, V_{jk}, V_{ik}) in the upper triangular portion of the $N \times N$ grid that violate transitivity. E_T evaluates to a value of zero energy for every triple that does not violate transitivity.

3.8 The Substitution Property Constraint

This constraint is responsible for motivating the network towards finding partitions satisfying the substitution property. To enforce this property into the generated partitions, the family of next-state functions determining the state transition behavior of the state machine must be incorporated into the constraint term. The state-transition function is defined for each present state by a pre-specified set of inputs. As a result, the constraint mapping this information into the neural net will need to scan the input set and determine the extent to which current solutions proposed by the net violate the requirement of preserving state-image congruence.

We interpret the neuron activation states as binary variables again, and generate a truth table enumerating all combinations for pairs of neurons (V_{ij} and its image neuron for a specific input "k" denoted $V_{\delta-k(i)\delta-k(j)}$). The subscript " $\delta-k(i)$ " denotes the next-state determined by input "k", for the current (present) state "i".

A truth table generated by considering pairs of neurons in general terms is shown in table 3.2. The s.p. constraint term (E_2), assigns a penalty of negative unity to the network energy for all cases where the required neuron combinations violate the definition of the substitution property. Conformation to the property of substitution obviates the penalty.

The entries in the truth table for constraint term E_2 show that its functional dependency on the states of the neurons V_{ij} and $V_{\delta-k(i)\delta-k(j)}$ has the same form as the classical "0110" combination corresponding to a two-variable logical "EXCLUSIVE-OR (XOR)" function. Thus, the sum-of-products form for the E_2 term $\forall i,j,k$ may be written from the truth values and the previously employed results as

$$\begin{aligned}\tilde{E}_2 &= V_{ij} V_{\delta-k(i)\delta-k(j)} \vee V'_{ij} V'_{\delta-k(i)\delta-k(j)} \\ &= V_{ij} + V_{\delta-k(i)\delta-k(j)} - 2 V_{ij} V_{\delta-k(i)\delta-k(j)} .\end{aligned}$$

As in the case of the transitivity constraint, since the indices span only the upper triangular half of the $N \times N$ neuron grid, the above equation may be compacted to the concise form:

$$E_2 = \sum_{i=0}^{n-2} \sum_{j=\{i+1\}}^{n-1} \sum_{\substack{\delta-k_{k_{\max}}(i)\delta-k_{k_{\max}}(j) \\ \delta-k_1(i)\delta-k_1(j)}} \tilde{E}_2 .$$

The above equation is quadratic; it is computationally efficient to the extent of offering the benefit of scanning only a total of $[N(N-1)]+2$ neurons from the $N \times N$ grid.

Combining the two constraint terms E_1 and E_2 derived so far into one expression, the overall equation for the network energy function may now be written as

$$E = k_1 E_1 + k_2 E_2 .$$

| Neuron V_U | Next-state Neuron $V_{\delta-k(n)\delta-k(j)}$ | Violation of Substitution Property ? | Value for E_2 |
|-----------------|---|---|--------------------|
| 0 | 0 | No | 0 |
| 0 | 1 | Yes | 1 |
| 1 | 0 | Yes | 1 |
| 1 | 1 | No | 0 |

Table 3.2. Substitution Property constraint term E_2 truth table.

The gain parameters k_1 and k_2 are set relatively equal to each other so that the constraint enforcing transitivity shares equal importance with the one influencing the substitution property. This is not only beneficial in eliminating "tuning" of the gains, but is also necessary due to the characteristic of s.p. partitions.

3.9 Ground State Characteristic of the Energy Function

The energy function (E) for the network has degenerate ground states. In other words, the solution points for the s.p. partition problem represent a value of zero to the network energy function. Neural net solutions to constraint satisfaction/optimization problems of the class of the TSP, graph partitioning problem and others, have always been compared for merit with solutions generated by classical techniques before accepting their "optimality". The degenerate ground-state characteristic of the network we have developed obviates the comparison exercise due to the fact that solution states are now uniquely identified by their associated zero-energy values. A further use of this characteristic is the easy detection of local minima since unlike global minima, these states will not have zero energies associated with them. Thus, the energy function more tangibly reflects the merit of suggested solutions by the network: If the partition energy is zero, the proposed solution by the network in its current state of activity is the required global solution, else the computation needs to be continued and the present interpretation of the net needs to be improved.

3.10 Interconnection Strengths and the Activation Rule

The classical form of the energy function (E) for a third-order network [14] is written as

$$E(\vec{V}) = -\frac{1}{3} \sum_{i \neq j \neq k} \sum_{j \neq k \neq i} \sum_{k \neq j \neq i} W^{(3)}_{ijk} V_i V_j V_k - \frac{1}{2} \sum_{i \neq j} \sum_{j \neq i} W^{(2)}_{ij} V_i V_j - \frac{1}{1} \sum_i W^{(1)}_i ,$$

where $W^{(3)}_{ijk}$ is the weight matrix storing the interconnection strengths between triples of neurons ($V_i V_j V_k$), $W^{(2)}_{ij}$ contains the strengths between pairs of neurons ($V_i V_j$) and $W^{(1)}_i$ represents the bias for each neuron.

Comparing this expression with the derived equation (E) for the network energy function and equating the coefficients of like terms, we obtain the entries to the three weight matrices.

The rule for activation of neurons in the net is based on the classical Hopfield-like constraint satisfaction neural net mechanism. Each neuron takes turns in evaluating the difference in the global energy of the network when it is in the "ON" state and when it is in the "OFF" state. The state of activity of the neuron that lowers the global energy of the network is then assumed by the neuron. This decision is made locally by each neuron and iterated enough number of times till further changes cease to affect the network globally. The network is thereupon understood to have performed the intended gradient descent in a $2^{N(N-1)/2}$ dimension landscape and effected an optimization of the energy function. The final, stable state of the neurons in the network then reflect the solution computed by the net.

3.11 Boltzmann Machine with Simulated Annealing

A third-order Boltzmann machine network employing simulated annealing was developed and simulated to solve for s.p. partitions. Amongst several algorithms that exist for incorporating

the technique of simulated annealing into parallel connectionist nets, the logarithmic rule derived by Geman and Geman [9] mathematically guarantees asymptotic convergence to global minima. Their rule is called classical simulated annealing (CSA). We used this technique in our simulations for Boltzmann machines. An example problem is now illustrated.

Example

A state space of dimension 18 is addressed in this example. A state machine with 18 states and three inputs was presented to the network. The state table for this machine is given in table 3.3. Figures 3.2 - 3.5 provide an illustration of the network as it generated a valid solution:

$$\pi = \{ \overline{0, 2, 4}; \overline{1, 9, 17}; \overline{3, 5}; \overline{6, 8, 10}; \overline{7, 15}; \overline{11}; \overline{12, 13, 14, 16} \}$$

Figure 3.2 shows a snap-shot of the 18×18 net in a random initial starting state. Two intermediate states of the net as it evolved ultimately to the final solution configuration are illustrated by figures 3.3 and 3.4. The valid global (required) solution state to which the network converged in four discrete time-steps is shown in figure 3.5. The performance of the net to fifteen independent runs was observed and is tabulated in table 3.4. The success rate of the net is impressive. A valid solution was always obtained - the fastest in as little as two time steps. Figure 3.6 is the legend for the various s.p. partitions listed in table 3.4. Classical simulated annealing was used to address the issue of local minima. Further relevant statistics are indicated in the snap-shot figures.

| Present state | Input I ₁ | Input I ₂ | Input I ₃ |
|---------------|----------------------|----------------------|----------------------|
| 0 | 0 | 2 | 4 |
| 1 | 1 | 9 | 17 |
| 2 | 0 | 2 | 4 |
| 3 | 3 | 3 | 3 |
| 4 | 0 | 2 | 4 |
| 5 | 5 | 5 | 5 |
| 6 | 6 | 8 | 10 |
| 7 | 7 | 7 | 7 |
| 8 | 6 | 8 | 10 |
| 9 | 1 | 9 | 17 |
| 10 | 6 | 8 | 10 |
| 11 | 11 | 11 | 11 |
| 12 | 12 | 14 | 16 |
| 13 | 13 | 13 | 13 |
| 14 | 12 | 14 | 16 |
| 15 | 15 | 15 | 15 |
| 16 | 12 | 14 | 16 |
| 17 | 1 | 9 | 17 |

Table 3.3. 18-state, 3-input state machine table.

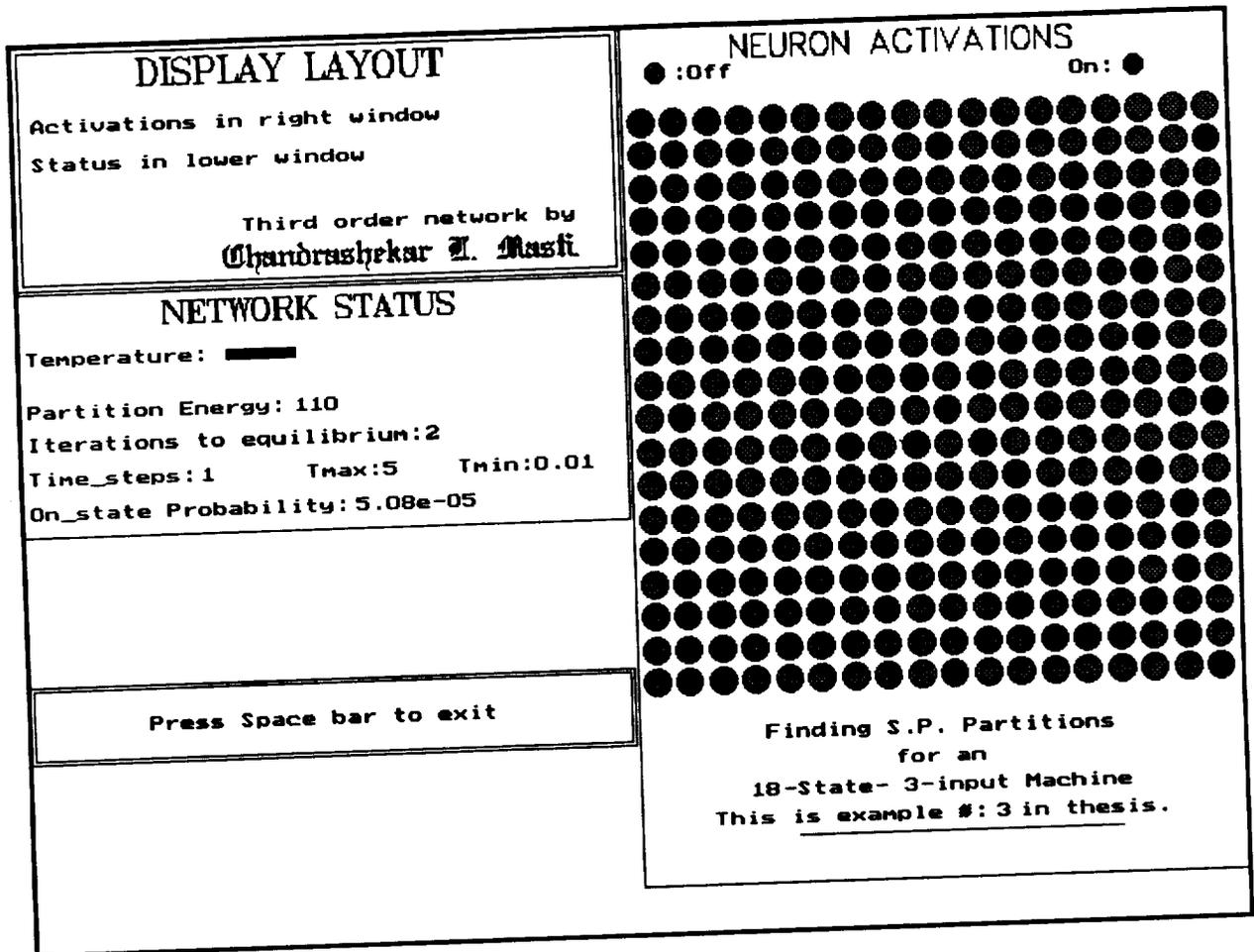


Figure 3.2. A random starting state of the 18 × 18 network.

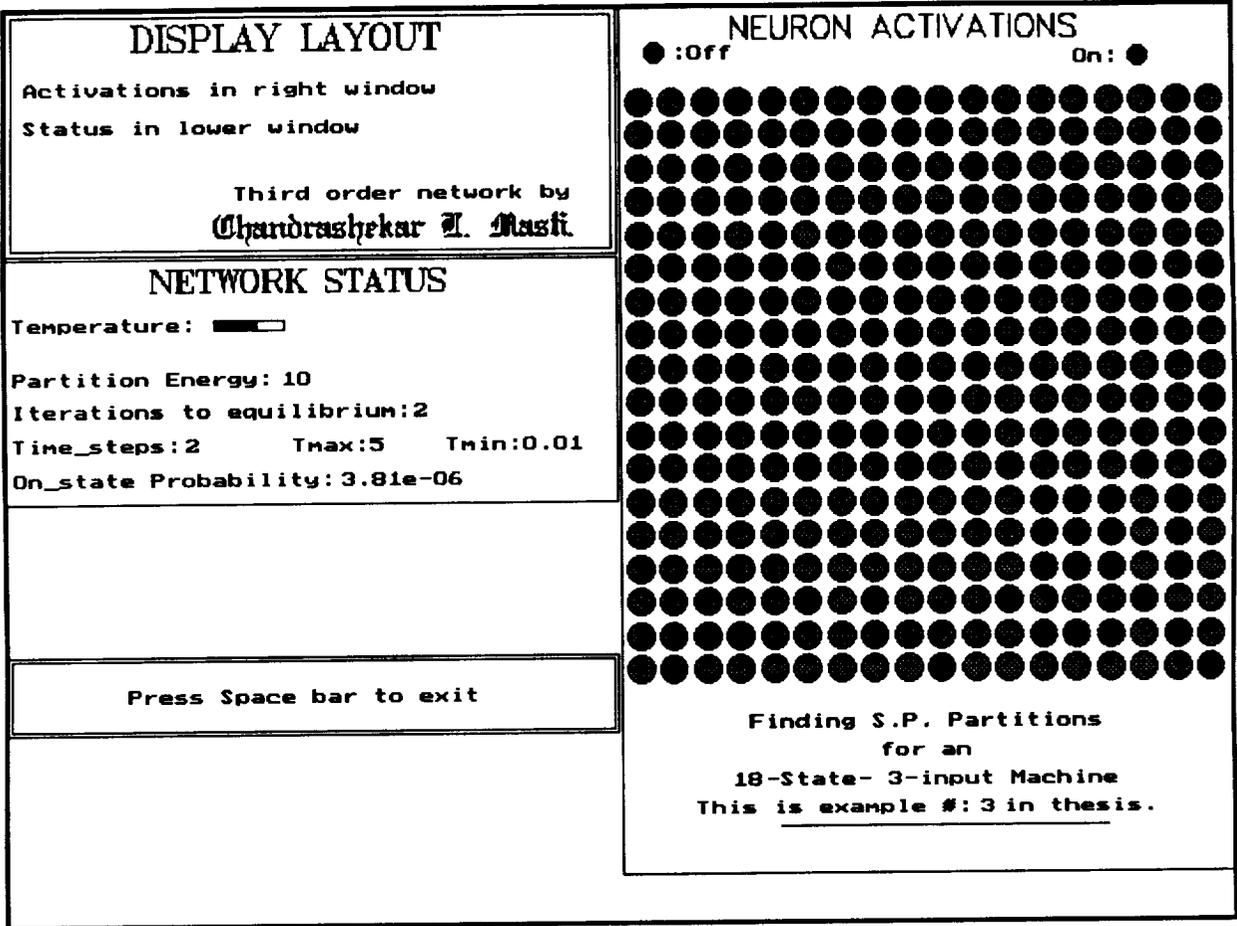


Figure 3.3. State of the net at the second time-step.

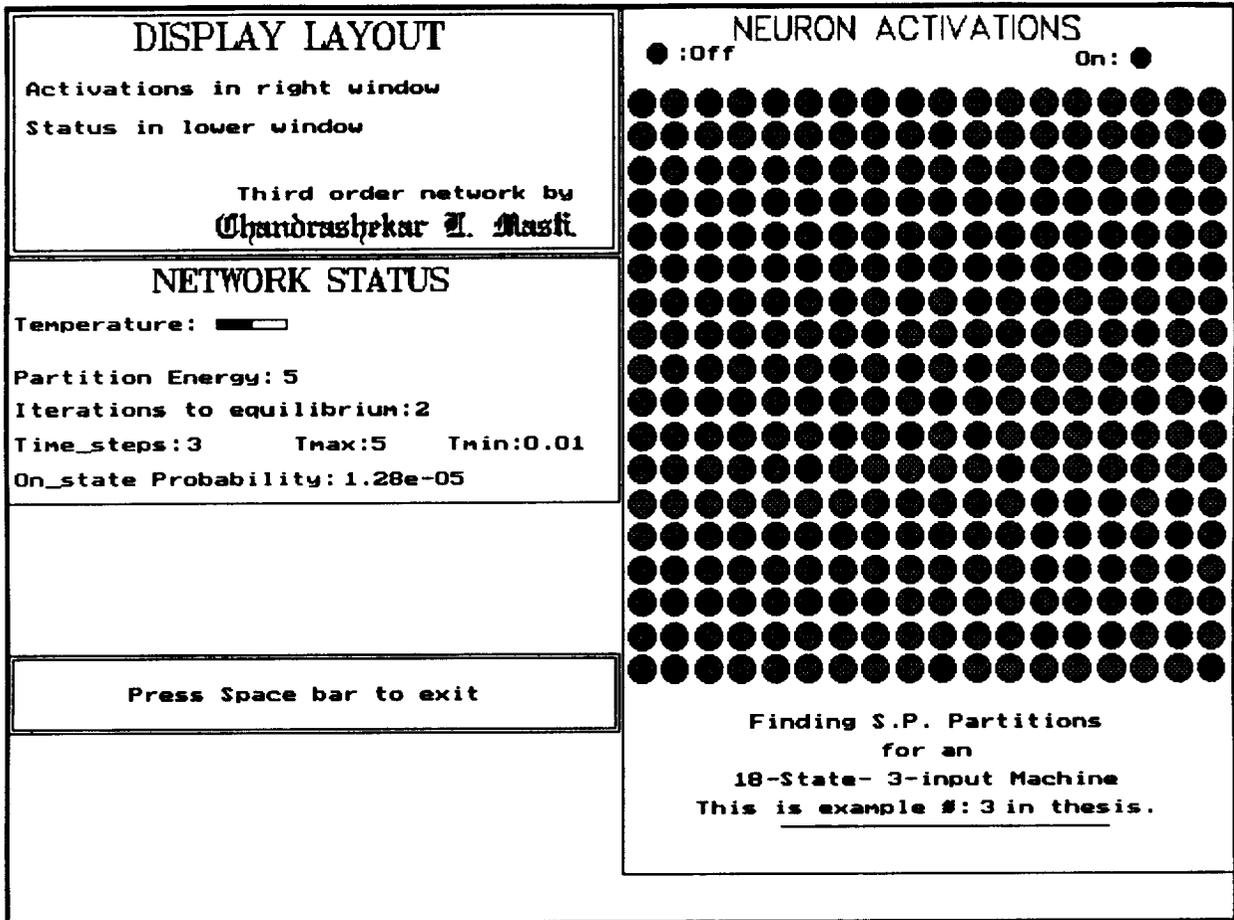


Figure 3.4. Time lapse snap-shot of the net.

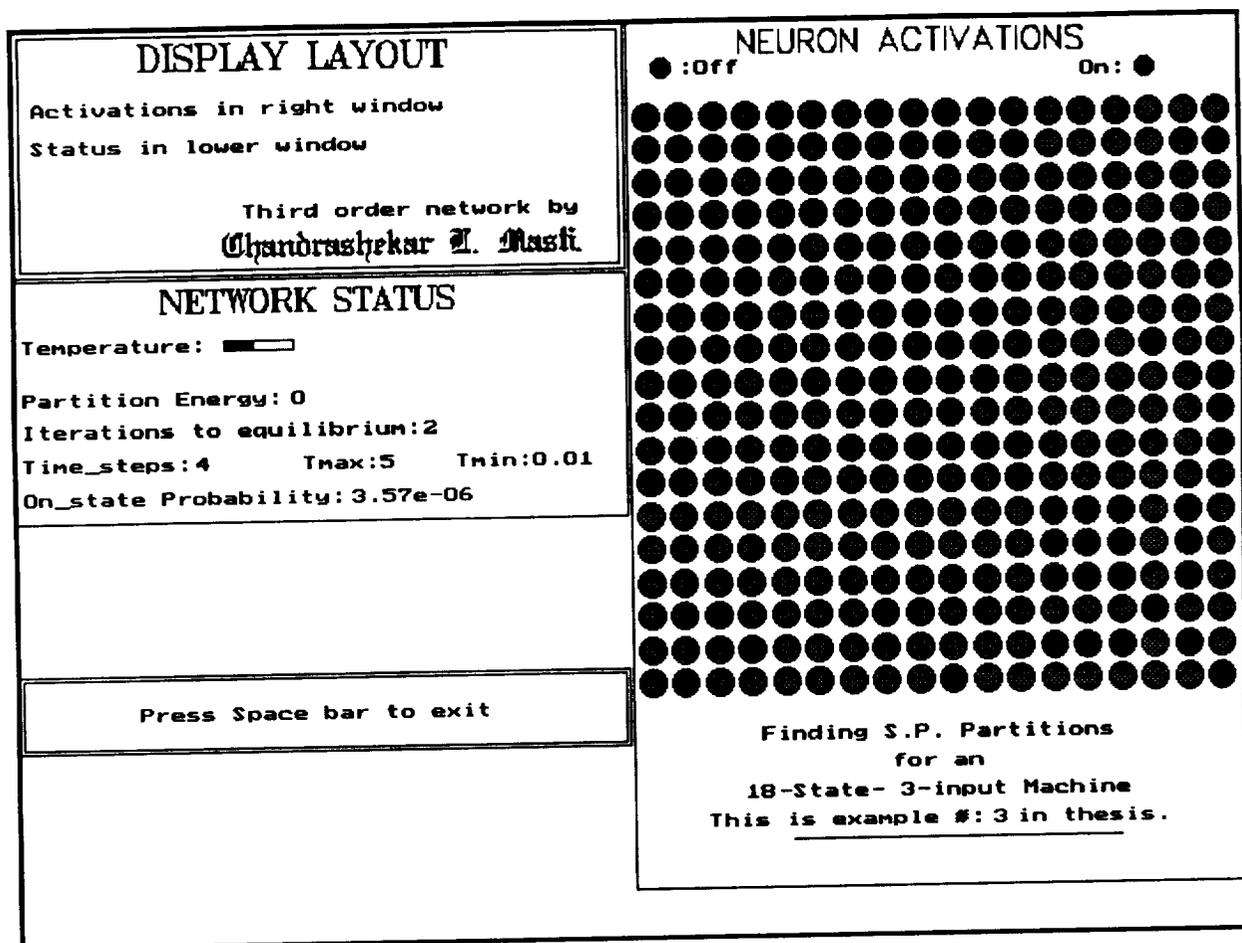


Figure 3.5. Final solution state of the 18 × 18 net.

| Trial number | S. p. partition | Time steps needed for solution |
|--------------|-----------------|--------------------------------|
| 1 | π_1 | 2 |
| 2 | π_2 | 2 |
| 3 | π_3 | 5 |
| 4 | π_4 | 2 |
| 5 | π_5 | 5 |
| 6 | π_6 | 3 |
| 7 | π_7 | 3 |
| 8 | π_8 | 2 |
| 9 | π_9 | 7 |
| 10 | π_{10} | 2 |
| 11 | π_{11} | 3 |
| 12 | π_{12} | 4 |
| 13 | π_{13} | 2 |
| 14 | π_{14} | 5 |
| 15 | π_{15} | 3 |

Table 3.4. Results of the network performance to 15 trial runs.

3.12 Conclusions

Present approaches to the machine decomposition problem have relied on a sequential and exhaustive search technique for generating partitions satisfying the substitution property. This is known to become intractable with growth in problem size and therefore imposes intemperate demands on computation resources. In this research we have developed an intrinsically parallel approach to addressing the intractability involved in the partition search problem by the use of an artificial neural network model that has potential for offering a viable alternative to existing sequential search methods.

The problem of decompositions has been cast into the framework of constraint satisfaction and a neural net model was developed to solve constraint satisfaction problems through optimization of a mathematically derived objective function over the problem space. The formulation strategies governing the derivation of objective functions for constraint satisfaction neural nets have been based so far on heuristics. This research presents a formalization for the method by exploiting useful theories from Boolean and relational algebras.

We have verified that the issue of decompositions belongs to a class of problems that are beyond the scope of solvability for second-order networks. A theorem has been stated and proved establishing that third-order correlations must be extracted by a neural net to generate substitution property partitions. A third-order deterministic network such as a hopfield net has been shown to fail in solving the s.p. partition search problem due to the lack of an adequate technique for escaping from any local minima of its associated objective function. Impressive results have been obtained by using a stochastic approach such as a third-order Boltzmann machine net with simulated annealing.

A spectrum of problem sizes was addressed using a third-order Boltzmann machine network. In many cases, solutions were obtained in under five discrete time-steps. Due to the excellent convergence properties of the higher order scheme involved in connection with the ground state characteristic of the energy function derived, it has been possible to introduce engineering ingenuities in identifying global solution states in arbitrary problem domains at unconventionally fast rates. The network performance has shown favorable scaling with the state dimension of the problem space. However, the network as simulated currently does show degradation in performance with respect to the input dimension. Intrinsic limitations behind the technique of a software simulation of an essentially parallel (neural) approach on a serial computing machine have precluded any substantial immediate improvements.

3.13 Future Research

Real world problems in decomposition can be expected to have large-sized state as well as input dimensions. As a result, a possible strategy for addressing the issue of scaling network performance in adequate proportion to the input dimension is to exploit the theory of intersection of lattices of sub-groupoids. A suggested technique is to develop a separate network for each dimension in the input set of the problem space being addressed and incorporate a parallel communication link (a constraint) between the individual nets as they descend toward their respective global minima [15]. The common intersection point of their individual global minima with respect to their convergent state of activity reflecting a valid, globally acceptable solution as determined overall by every input dimension would then represent the required solution. A little insight suggests that such an approach has its potential in essentially a hardware

implementation, since larger the communication bandwidth simulated, greater the number of iterative loops a single-CPU von Neumann-machine would be required to execute, rendering it an increasingly inadequate choice for the purpose. Our attempts at software simulations based on such a "decomposed approach to the decomposition problem" has provided corroborative results favoring our conviction toward emphasizing a future attempt on VLSI based hardware.

The majority of optimization/constraint satisfaction problems have been addressed till date using neural nets with the order of interconnections between the neurons in the network being no higher than quadratic. The potential of higher order neural nets has been known for some time, but hardware limitations so far have determined limitations to their use [16]. With the advent of high-speed, multi-level VLSI devices specially designed for parallel computation algorithms such as neural networks, we speculate that the foundations we have laid for a neural approach to the decomposition problem will one day obviate sequential methods altogether.

Adaptive Constraint Satisfaction

4.1 Introduction

The original constraint satisfaction network proposed by Hopfield [2], [3], [4], [5] has several drawbacks one of which is the network is guaranteed to converge only to local minima of the associated Liapunov function [17], which are reachable from the given initial position of the state vector. This property is useful if the Hopfield network is used as an associative memory [18], but is not a desirable feature for the case where it is used as a constraint satisfaction network simply because the local minima do not correspond to solutions in most applications.

To examine the main reason why a Hopfield network does not perform well for constraint satisfaction problems, we will analyze the process of solving a constraint satisfaction problem with a Hopfield network. The initial task is to find a suitable representation with the hypothesis each node represents and then define the weight matrix. Unfortunately, the definition of the weight matrix is complete only up to determining the actual magnitudes of the gain parameters. To observe this argument analytically, consider an element of the weight matrix, w_{ij} , the weight between nodes s_i and s_j . The weight can be constructed as

$$w_{ij} = \sum_{\alpha} K_{\alpha} \delta_{\alpha ij},$$

where $K_\alpha \in R^+$ if the hypotheses both nodes represent for constraint α are mutually supporting and $K_\alpha \in R^-$ if they are mutually conflicting. The term $\delta_{\alpha ij}$ is equal to 1 if the two hypotheses represented by nodes s_i and s_j are related under constraint α and is 0 otherwise. As one can easily see from the definition of the weight matrix entry w_{ij} , all variables but the actual magnitude of the gain parameter K_α can be determined.

Although there have been attempts in literature to determine the optimal set of gain parameter magnitudes for a given problem [19], [20], there is no convincing evidence that an acceptable formal method exists. The typical solution of this problem is to randomly initialize the gain parameters, observe the state vector to which the network converges and identify the constraints violated using that state vector. Once the unsatisfied constraints are observed, the magnitudes of the gain parameters which are associated with the violated constraints are increased. It is often the case that this process has to be repeated many times. There is no guarantee that a good set of gain parameters can be found such that the network will always converge to a global minimum of the performance function, regardless of the initial conditions. Another issue to note is that the increase in the gain parameter magnitudes is arbitrarily specified.

The goal of this research effort is to design a closed-loop system in which a classical Hopfield network performs the searching action and another functional block observes and evaluates the output of the network and then adapts the weights of the Hopfield network. The main function of the adaptive block is to redefine the weight matrix based on the information extracted from the state vector to which the Hopfield network converges. The redefinition of the

weight matrix should be performed in such a way that the search network is less likely to break the constraints which it violated previously when the network relaxes to a new state.

It is necessary that the adaptation block has the means to detect the local minima which are not solutions for the constraint satisfaction problem under discussion so that the adaptive block can decide to initiate the adaptation process. The trial and error procedure performed to search for a good set of gain parameters is eliminated by employing this closed loop system.

There are three important variables which determine the state vector to which a Hopfield network converges: the gain parameters which finalize the exact shape of the energy function in N-dimensional phase space, the initial state of the network node outputs and the random order the node outputs are chosen to be updated [21]. If a classical Hopfield network is started at a randomly chosen initial state, it will converge to one of the admissible set of local minima which is solely determined by the Liapunov function and the random order the nodes are updated.

If the converged state vector represents a local minimum of the performance function and hence at least one constraint is violated, the associated gain parameter magnitude is increased an arbitrary amount. Although this adjustment of the gain parameter follows the argument "if a constraint is violated then the weight magnitude representing the relative importance level of that constraint is not large enough, thus it needs to be increased," there is still no guarantee that the adjusted set of gain parameters will always cause the network to converge to a solution state vector regardless of the initial conditions.

On the other hand, the proposed adaptive search system starts with randomly specified initial state vector and set of gain parameters and then converges to the local minimum implied by the random update order. If the local minimum is not a solution then the algorithm adapts

the gain parameter magnitudes based on the information provided by the state vector representing the local minimum. The relaxation process of the network with the new set of gain parameters is restarted and this goes on until a minimum that represents a solution is found, thus effectively eliminating the dependency of classical Hopfield network on its initial conditions. It is also conjectured at this point of discussion that the random order the node outputs updated might still play a significant role for the number of iterations needed to find a solution.

4.2 Hopfield Networks and the Adaptation Algorithm

Given a vector of discrete neurons S , Hopfield [2] has derived an "energy" or Liapunov function for the discrete case:

$$E(S) = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j - \sum_i b_i s_i, \quad i \neq j,$$

where w_{ij} is the weight between neurons s_i and s_j , and b_i is a bias to neuron s_i . Assuming the following activation rule:

$$s_i = 1 \text{ if } \sum_j w_{ij} s_j + b_i > 0,$$

$$s_i = 0 \text{ otherwise}$$

the neurons will change states such that energy will always decrease or remain the same.

The main philosophy of the adaptation algorithm is to observe the state vector the Hopfield network converges to as implied by the initial conditions, node update order, and Liapunov function. An error signal is generated and is used to adapt the gain parameters thus restructuring the N-dimensional energy space. The adaptive algorithm is very similar to Widrow's adaptive linear element (adaline) in structure and indeed is a modified version of the

the adaline [22]. The convergence theorem for the adaline does not apply in the case of proposed adaptive algorithm because of the modification.

The functional definition of the adaptive algorithm is now presented. Let

$\mathbf{X}(k) = [x_0(k), \dots, x_{L-1}(k)]$ be a $1 \times L$ input vector,

$\mathbf{W}(k) = [w_0(k), \dots, w_{L-1}(k)]$ be a $1 \times L$ weight vector,

$d(k)$ be the desired output which is set to 0,

$\epsilon(k)$ be the error, and

$y(k)$ be the output at time step k .

The error used to adapt the weights is computed as follows:

$$y(k) = \mathbf{W}(k)^T \mathbf{X}(k)$$

and

$$\epsilon(k) = y(k) - d(k).$$

Given this error term, the next step is to define the adaptive element weight vector update rule:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \mu \epsilon(k) \mathbf{X}(k),$$

where μ is the learning rate parameter defined by

$$0 < \mu < \frac{1}{\sum_{\alpha} [x_{\alpha}(k)]^2}.$$

The elements of the input vector are the individual error terms associated with each constraint. Thus

$$x_{\alpha}(k) = E_{\alpha}(k) - E_{ref}$$

for inhibitive interaction and

$$x_{\alpha}(k) = E_{ref} - E_{\alpha}(k)$$

for excitatory interaction,

where $E_{\alpha}(k)$ takes on different functional forms depending on the type of the implementation and the constraint and E_{ref} is the value of $E_{\alpha}(k)$ when constraint α is satisfied. Figure 4.1 depicts the adaptation algorithm in block diagram form.

The gain parameter for each constraint is increased when the individual constraint is violated by the state vector representing the local minimum. The increase in magnitude is proportional to the magnitude of the total error generated multiplied with the learning rate. The learning rate parameter is inversely proportional to the trace of the input correlation matrix which is sum of the squares of the input vector entries. When the error is large, the learning rate parameter is small implying slow adaptation. Conversely, when the error is small, the algorithm adapts faster because the learning rate parameter is large.

If the type of the interaction for the constraint is inhibition which is implemented with a second order term, then

$$E_{\alpha}(k) = E_{ref} \text{ if } E_{\alpha}(k) \leq E_{ref}$$

$$E_{\alpha}(k) = \sum_i \sum_j \delta_{\alpha ij} s_i s_j \text{ otherwise.}$$

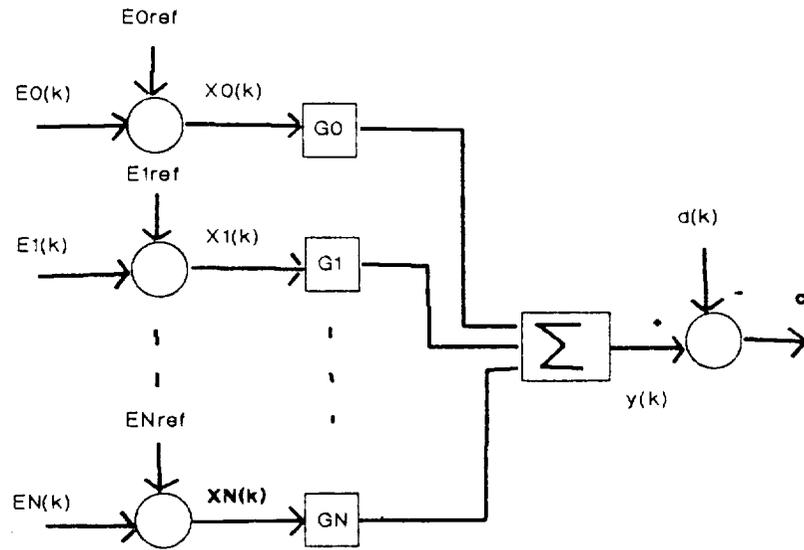


Figure 4.1. Adaptation algorithm block diagram.

For an excitation constraint the implementation associated with a second order term is

$$E_{\alpha}(k) = E_{ref} \text{ if } E_{\alpha}(k) \geq E_{ref},$$

$$E_{\alpha}(k) = \sum_i \sum_j \delta_{\alpha ij} s_i s_j \text{ otherwise.}$$

In the case of implementation of an excitation constraint with a first-order term

$$E_{\alpha}(k) = E_{ref} \text{ if } E_{\alpha}(k) \geq E_{ref},$$

$$E_{\alpha}(k) = \sum_i s_i \text{ otherwise.}$$

An inhibition constraint implementation with a first-order term employs the following:

$$E_{\alpha}(k) = E_{ref} \text{ if } E_{\alpha}(k) \leq E_{ref},$$

$$E_{\alpha}(k) = \sum_i s_i \text{ otherwise.}$$

The weight vector entries are the magnitudes of the gain parameters associated with each constraint:

$$W(k) = [G_1(k) \ G_2(k) \ \dots \ G_L(k)].$$

An argument which may serve to clarify the presentation of the adaptation algorithm follows.

Observation: The proposed adaptation algorithm as defined above increases the magnitude of the gain parameter when the constraint under discussion is violated, thus making it less likely for the network to violate the same constraint next time it converges.

Proof: Consider a state vector which violates constraint α , then $x_{\alpha}(k) > 0$. The update rule for the gain parameter magnitude associated with constraint α is given by

$$G_{\alpha}(k+1) = G_{\alpha}(k) + \mu \epsilon(k) x_{\alpha}(k).$$

We also have that $\epsilon(k) > 0$ and $\mu > 0$ thus

$$G_{\alpha}(k+1) \geq G_{\alpha}(k). \text{ Q.E.D.}$$

4.3 An Adaptive Constraint Satisfaction Network

Once the constraint satisfaction network is initialized and is allowed to relax to a state vector S at time instant k , two variables are passed to the adaptation block: state vector $S(k)$ and weight vector $W(k)$. Given these variable instances, the adaptation block will compute the error term. If the error is not zero the current state vector represents a local minimum assuming the solution state vector is a global minimum of the performance function with the error equal to zero. In this case, the adaptation block will update the gain parameters which are associated with the constraints violated.

Using this new set of gain parameters a new weight matrix for the constraint satisfaction network is computed and the current state vector is used as the network initial conditions. A new relaxation period is initiated and the cycle is repeated until a solution state vector is found. A block diagram description of the closed-loop system is shown in Figure 4.2. A flow-chart for the high-level functional definition of the closed-loop adaptive system is presented in Figure 4.3.

4.4 Simulation Results

The goal of this application is to demonstrate the use of adaptive constraint satisfaction networks to search for the shortest path in a state space and thus test the performance of the proposed algorithm. A directed graph is used as an abstract model for the state space search problem, where vertices of the graph represent the states and directed edges

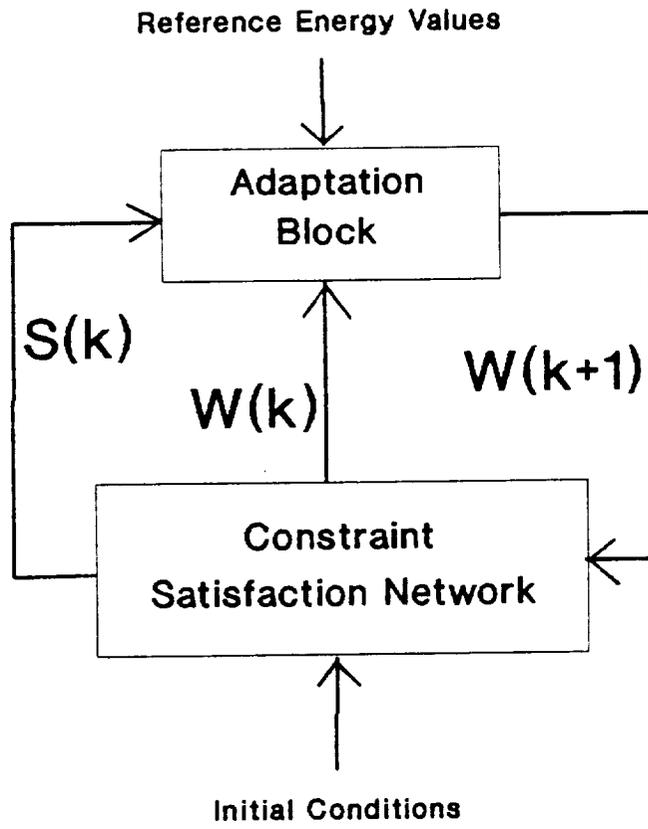


Figure 4.2. Closed loop system.

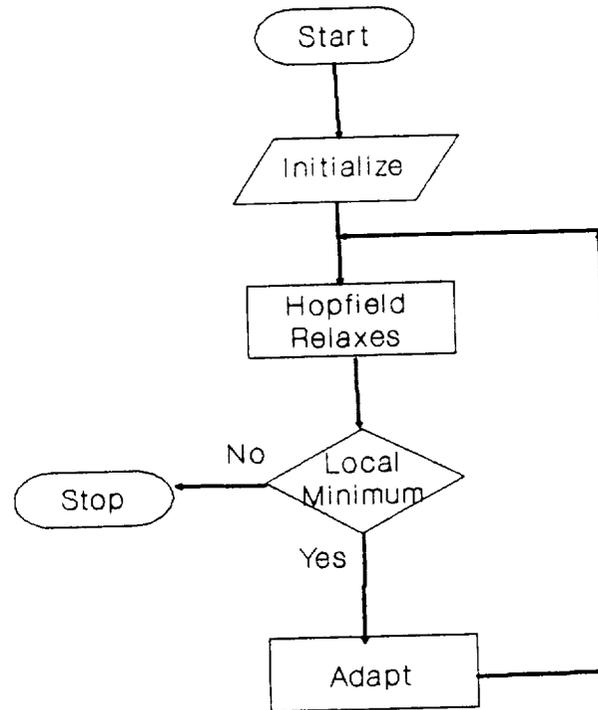


Figure 4.3. Flow-chart of adaptation algorithm.

stand for the transitions between associated states [8]. This problem effectively generates five constraints for the neural network search algorithm to satisfy and hence constitutes a considerable challenge.

The shortest path between two vertices of a given directed graph can be defined as a sub-graph which meets the following constraints:

- a. the sub-graph representing a path is both irreflexive and asymmetric,
- b. each vertex except the source and target vertices must have in-degrees and out-degrees of exactly 1,
- c. the source vertex has an in-degree of 0 and an out-degree of 1,
- d. the target vertex has an in-degree of 1 and an out-degree of 0, and
- e. the length of the path is equal to that power of the adjacency matrix which has the first non-zero entry in the row and column locations defined by the source and target vertices respectively.

Given the graph-theoretic constraints a path specification has to satisfy, the next step is to define the corresponding topological constraints of the path search problem for an adjacency matrix topology of the neural network. Since it is known that a path is irreflexive, we can clamp the neurons which are located along the main diagonal to 0 given that they represent the hypothesis: the path has self-loop for vertex V_i . In order to map the in-degree of 0 for a source vertex, the neurons belonging to the column labelled by that vertex are clamped to 0 and similarly, the neurons of the row labelled by the target vertex are clamped to 0 for realizing the constraint that the out-degree of the target vertex is equal to 0. Note that one also needs to implement the out-degree of 1 for the source vertex and in-degree of 1 for the target vertex, but

a special instance of the digraph search problem will be used such that there is only one path in the formulation of the problem which meets all other constraints, hence eliminating the implementation of the constraint.

Asymmetry of a graph (constraint #1) requires that only one of the two entries located at symmetric positions with respect to the main diagonal of the adjacency matrix be equal to 1. Hence $\delta_{1ij} = 1$ if and only if the row index of node s_i equals the column index of node s_j and the column index of node s_i equals the row index of node s_j ; otherwise $\delta_{1ij} = 0$. Since only one of two interacting nodes can be equal to 1, the type of interaction between the nodes is inhibition, $K_1 \in \mathbb{R}$. The reference value of the energy term associated with this constraint is 0.

A digraph node with an in-degree of 1 implies the existence of a single 1 in the associated column of the adjacency matrix (constraint #2). The term $\delta_{2ij} = 1$ if and only if the column index of node s_i equals the column index of node s_j ; otherwise $\delta_{2ij} = 0$. This constraint simply implements the "1 out of N nodes equal to 1" rule, thus the type of the interaction is inhibition with $K_2 \in \mathbb{R}$. The reference value of the energy term for this constraint is 0.

Similarly, a digraph node with an out-degree of 1 requires only a single 1 to exist in the associated row of the adjacency matrix (constraint #3). The term $\delta_{3ij} = 1$ if and only if the row index of node s_i equals the row index of node s_j ; otherwise $\delta_{3ij} = 0$. Again the type of the interaction is inhibition, $K_3 \in \mathbb{R}$, for the fact that only one of the N interaction nodes can be ON at a certain time. The reference value of the energy term for this constraint is 0.

A path specification imposes the condition that any node that is not a source or target and is included in the path must have an in-degree and out-degree of 1 (constraint #4). In terms of the adjacency matrix, if there exists a 1 in a particular row/column, then there must exist a 1 in the column/row that has the same node label as the corresponding row/column. The term $\delta_{4ij} = 1$ if and only if the row index of node s_i equals the column index of node s_j or the column index of node s_i equals the row index of node s_j ; otherwise $\delta_{4ij} = 0$. Whenever $\delta_{4ij} = 1$, the two nodes involved are supporting hypotheses; hence the interaction is excitatory with $K_4 \in \mathbb{R}^+$. The reference value of the energy term for this constraint is twice the shortest path length between source and target nodes minus 1.

In order to impose the constraint that any network solution must be the shortest path between source and target nodes, a bias for each neuron is employed. The reference value for the energy term is simply equal to the shortest path length computed from the powers of the adjacency matrix.

In order to test the proposed network, a digraph with a specially defined adjacency matrix was employed. It was assumed that the adjacency matrix of the digraph had all its lower triangular entries and the entries immediately above the main diagonal equal to 1; e.g., let a_{ij} (the entry in the i -th row and j -th column of a matrix) belong to \mathbf{A} (the adjacency matrix of the digraph), then $a_{ij} = 1$ if and only if $j \leq i + 1$.

The adjacency matrix for an example digraph with 10 vertices is shown in Table 4.1. Note that the entries which are equal to 0 are equivalently clamped to 0 in the network topology. This digraph specification yields a path of length $N-1$ between vertices V_0 and V_{N-1} . Thus this is the longest possible path for an N -vertex digraph. Another feature of this

| | V_0 | V_1 | V_2 | V_3 | V_4 | V_5 | V_6 | V_7 | V_8 | V_9 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| V_0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_2 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| V_3 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| V_4 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| V_5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| V_6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| V_7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| V_8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| V_9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 4.1. Adjacency matrix.

digraph is that there are circuits of all possible lengths implying a difficult state space to search since all those circuits may be mapped to local minima of the quadratic performance function.

Assume that we are looking for a path starting with vertex V_0 and ending with vertex V_9 . Clearly, there exists only one path which includes the adjacency matrix entries above the main diagonal. A number of experiments were done to test the performance of the proposed network. The measure of comparison is the number of iterations it takes the network to converge to the solution state vector.

Experiment 1

The goal of this experiment was to evaluate the performance of the proposed algorithm. A closed-loop version of the classical Hopfield network was developed to enable some form of comparison for the proposed algorithm since there is currently no closed-loop constraint satisfaction search algorithm available. A 10-node digraph was chosen as an instance of the problem. Two different versions of the proposed algorithm were employed to observe the effects of the random initialization of the state vector. The gain and state vectors for the classical Hopfield network were randomly initialized each time after the network converged to a local minimum and a new relaxation process was initiated with that set of parameters.

The frequency distribution of successful trials for the classical Hopfield network is shown in figure 4.4. It is clearly observable that most of the trials converged in the interval of 1 to 1300 iterations. Approximately 10% of the trials required somewhere between 1300 to 3800 iterations for convergence while a countable few needed 3800 to 6000 iterations.

In the case of the adaptive Hopfield network without state vector re-initialization after convergence to local minimum, almost all trials, 99%, converged in the interval 1 to 1100 with

significant percentage belonging to the 1 to 500 interval, 96%, figure 4.5.

The frequency distribution for the adaptive Hopfield network with state vector re-initialization has a lower peak and is more spread than the adaptive Hopfield without state vector re-initialization. Only 20% of all trials converged within 100 iterations for this algorithm. The percentage of trials which converged within 500 iterations is approximately 67 which is considerable lower than 96% realized by the adaptive Hopfield without state vector re-initialization.

One noticeable difference between the two frequency distributions of the classical and adaptive Hopfield algorithms is that the distribution is much more spread out for the classical Hopfield network. The distribution associated with the adaptive Hopfield network without state vector re-initialization clearly favors the interval 1 to 500 iterations with nearly 45% of the trials converging within 100 iterations.

There are only 4 trials which needed up to 1100 iterations to converge. This distribution has the most desirable properties for the experiment because the distribution is the least spread of three, located to the far left of others and skewed to the right. The fact that the distribution is the narrowest of three implies that a better estimate of the average iterations for convergence can be established.

The location of the distribution being to the left of the others is related to the claim that the mean value of the process is smallest and hence the mean number of iterations for

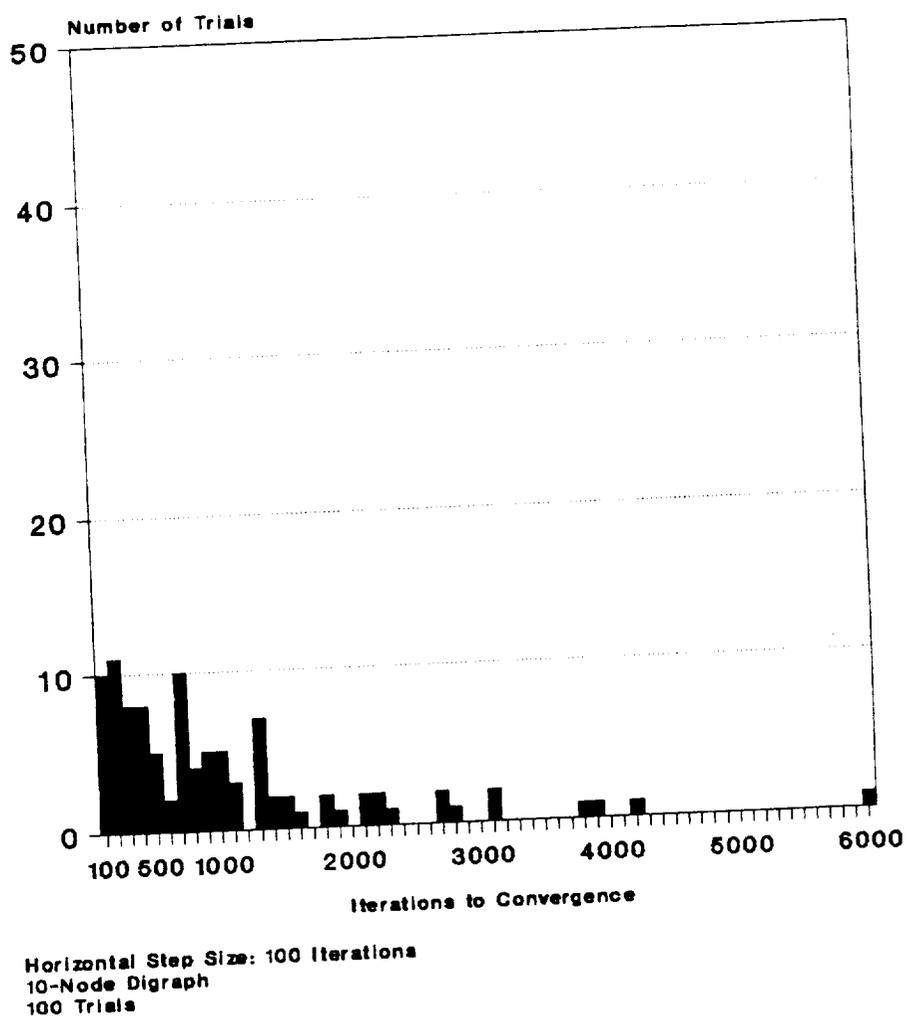
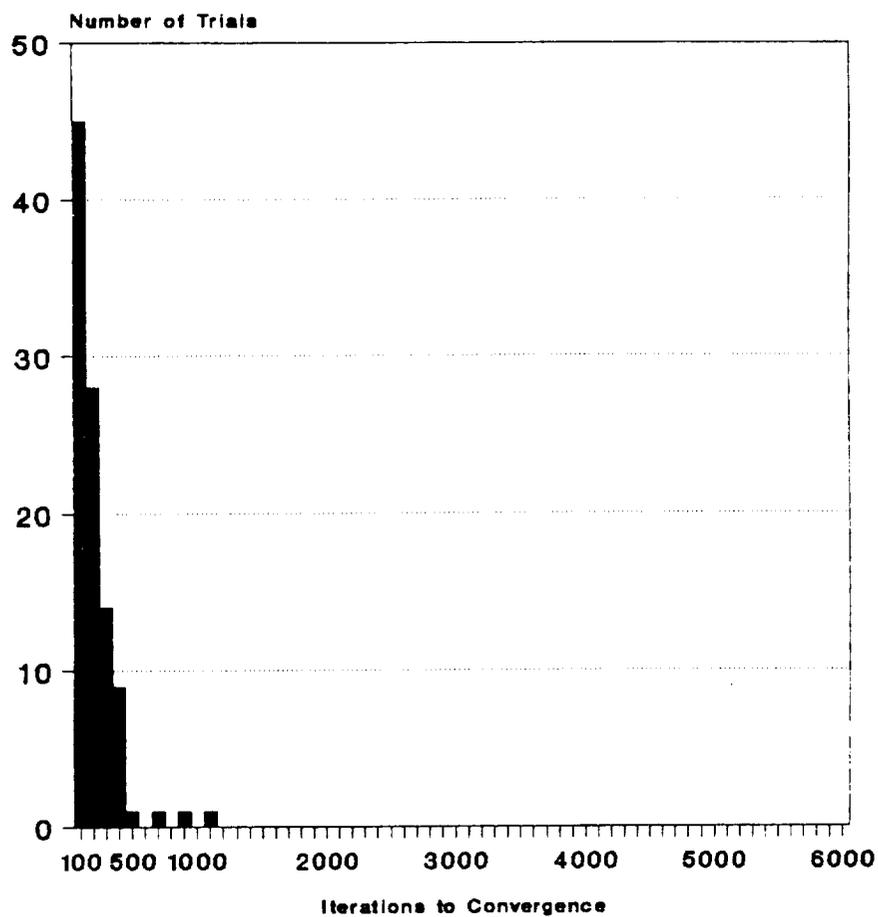


Figure 4.4. Classical Hopfield,
gain and state vector reinitialized.



Horizontal Step Size: 100 Iterations
10-Node Digraph, 100 Trials
Learning Rate Scaler: 0.1

Figure 4.5. Adaptive Hopfield,
state vector not reinitialized.

convergence is smallest. The right-skewed distribution is an indication of the fact that most trials actually converge in fewer iterations than the median value.

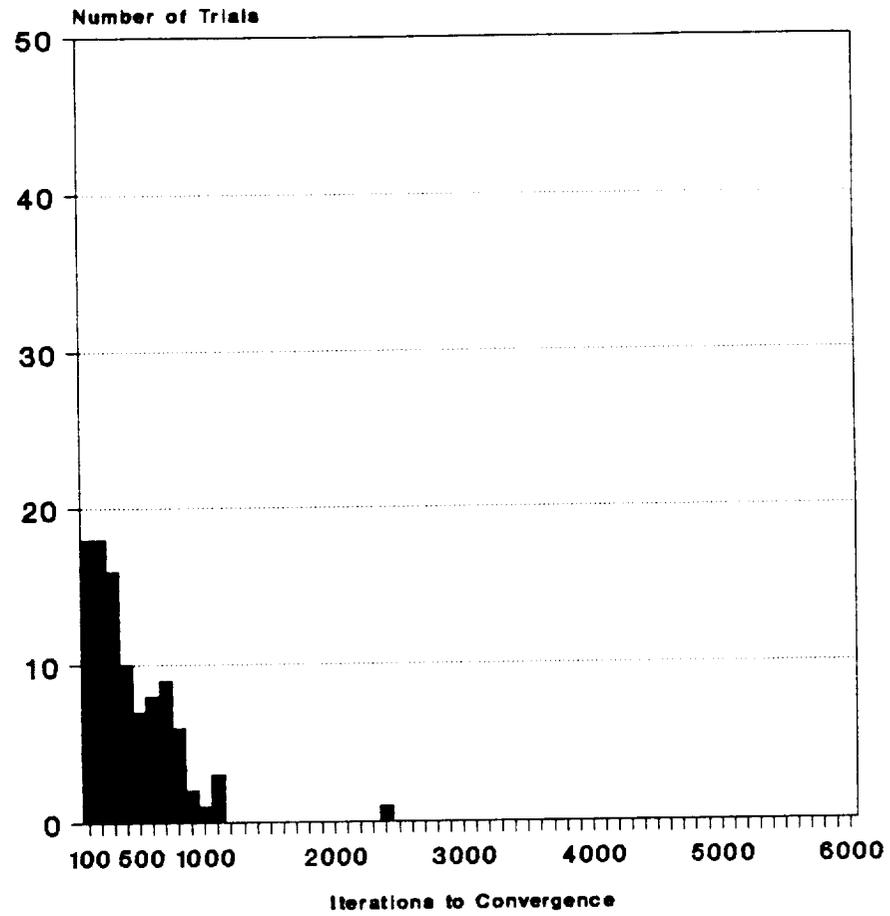
The second best algorithm is the adaptive Hopfield with state vector re-initialization which outperformed the classical Hopfield network. The difference in performance for the two adaptive Hopfield networks points to the importance of the state vector re-initialization aspect of the algorithms. Employing the current state vector as the initial condition for the next relaxation period improves the performance of the adaptive algorithm.

Experiment 2

This experiment was conducted to observe the effects of the learning rate on the frequency distribution of the successful trials. The learning rate parameter was varied from 0.1 to 0.9 in steps of 0.1 and 100 trial runs were attempted for each value of the learning rate parameter for a 10-node digraph. The results are shown in figures 4.6-14.

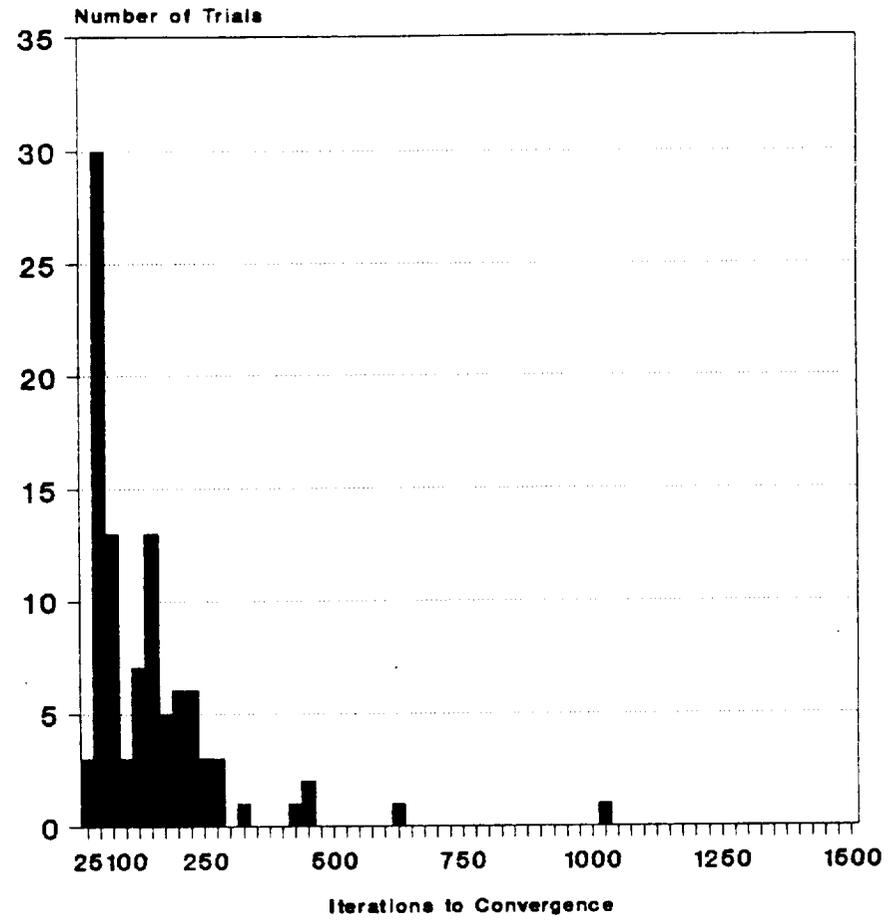
One observation is that the curve is the least spread for learning rate value in the interval of [0.4, 0.6] and spreads out as the value is varied toward both ends of the interval [0.1, 0.9]. Thus, the statistical mean estimate of the distributions is more accurate for learning rate values of 0.5 ± 0.1 .

The frequency distributions for learning rate values in the interval [0.3, 0.6] have higher peaks which are located to the left of the horizontal axis and hence more trials for those learning rate values converged within a small number of iterations. Indeed, 78% of 100 trials converged within 100 iterations for a learning rate of 0.5 and 99% of all trials converged within 275 iterations for the same learning rate.



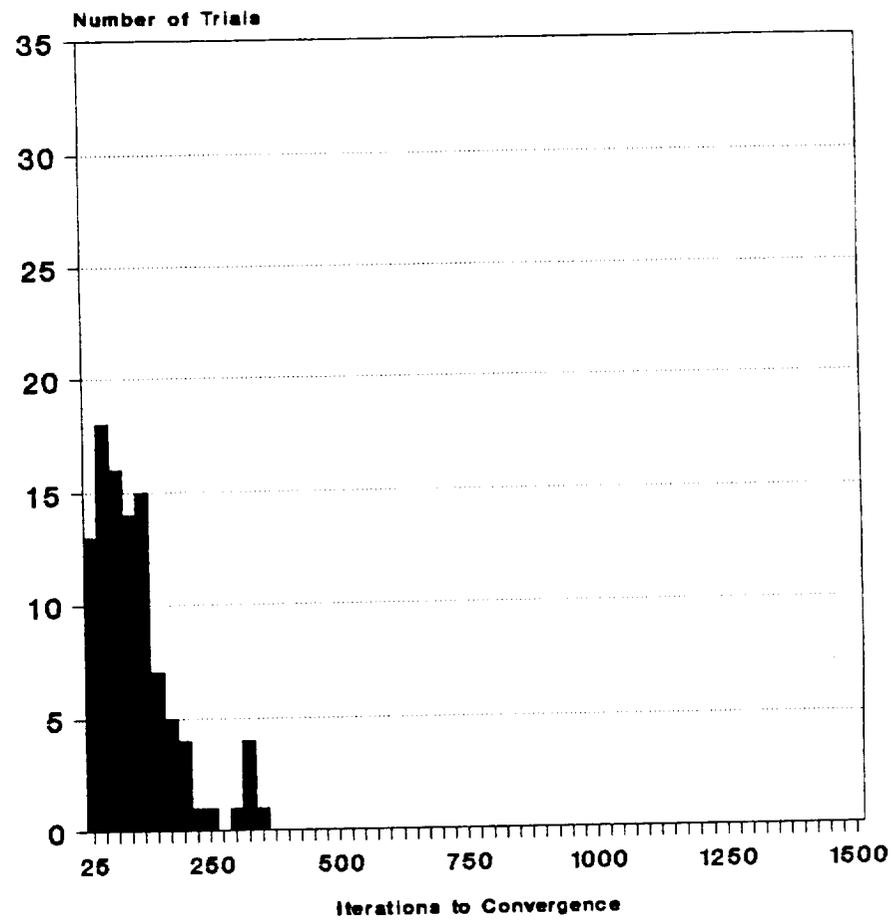
Horizontal Step Size: 100 iterations
10-Node Digraph, 100 Trials
Learning Rate Scalar: 0.1

Figure 4.6. Adaptive Hopfield,
state vector re-initialized.



Horizontal Step Size: 25 Iterations
 10-Node Digraph
 100 Trials

Figure 4.7. ACSN performance,
 learning rate scaler: 0.1.



Horizontal Step Size: 25 Iterations
10-Node Digraph
100 Trials

Figure 4.8. ACSN performance,
learning rate scaler: 0.2.

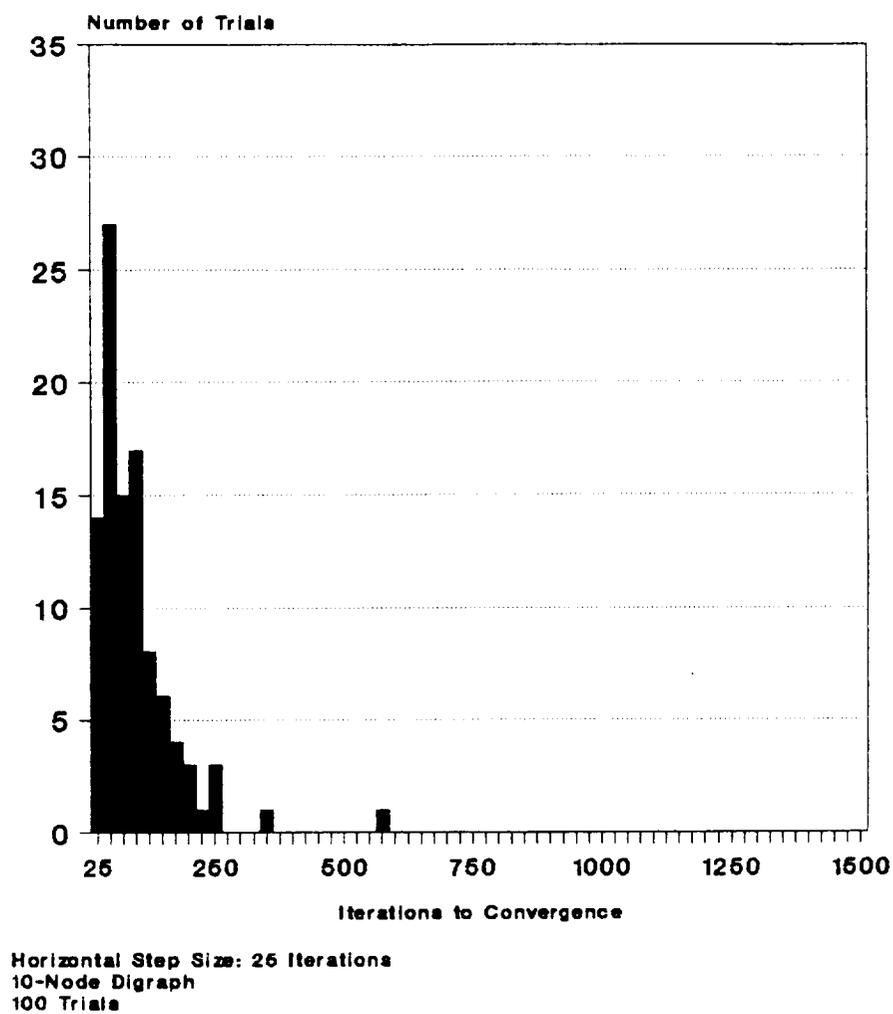
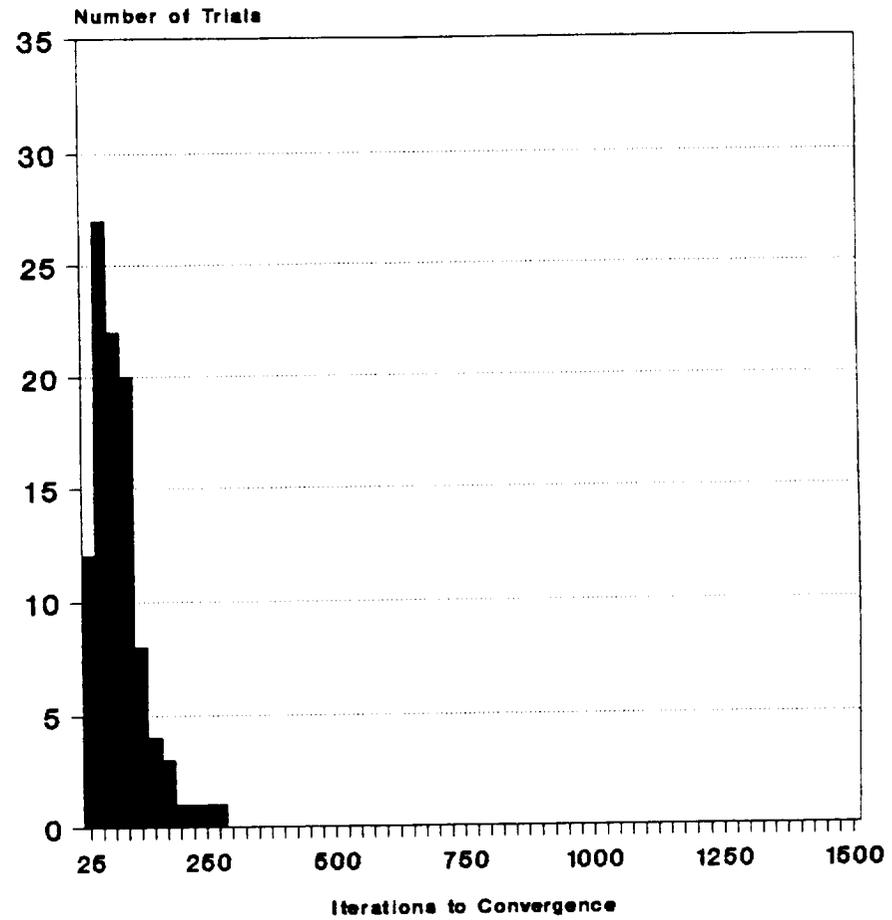
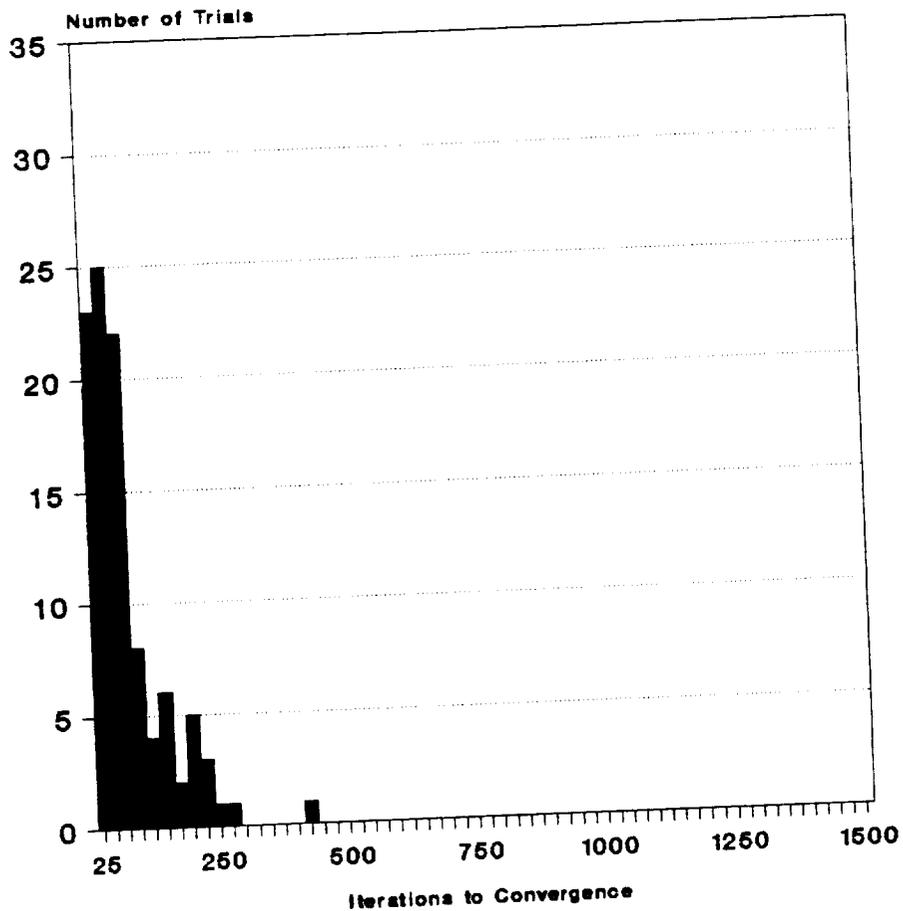


Figure 4.9. ACSN performance,
learning rate scaler: 0.3.



Horizontal Step Size: 25 Iterations
10-Node Digraph
100 Trials

Figure 4.10. ACSN performance,
learning rate scaler: 0.4.



Horizontal Step Size: 25 Iterations
10-Node Digraph
100 Trials

Figure 4.11. ACSN performance,
learning rate scaler: 0.5.

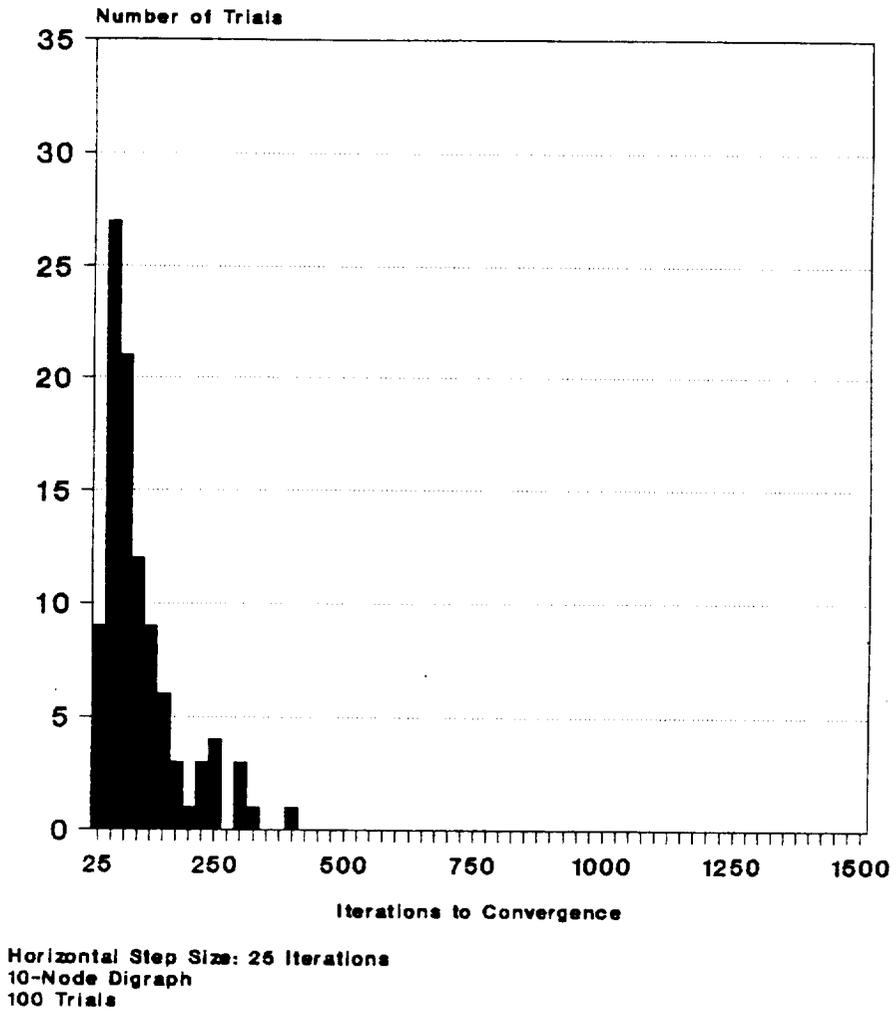


Figure 4.12. ACSN performance,
learning rate scaler: 0.6.

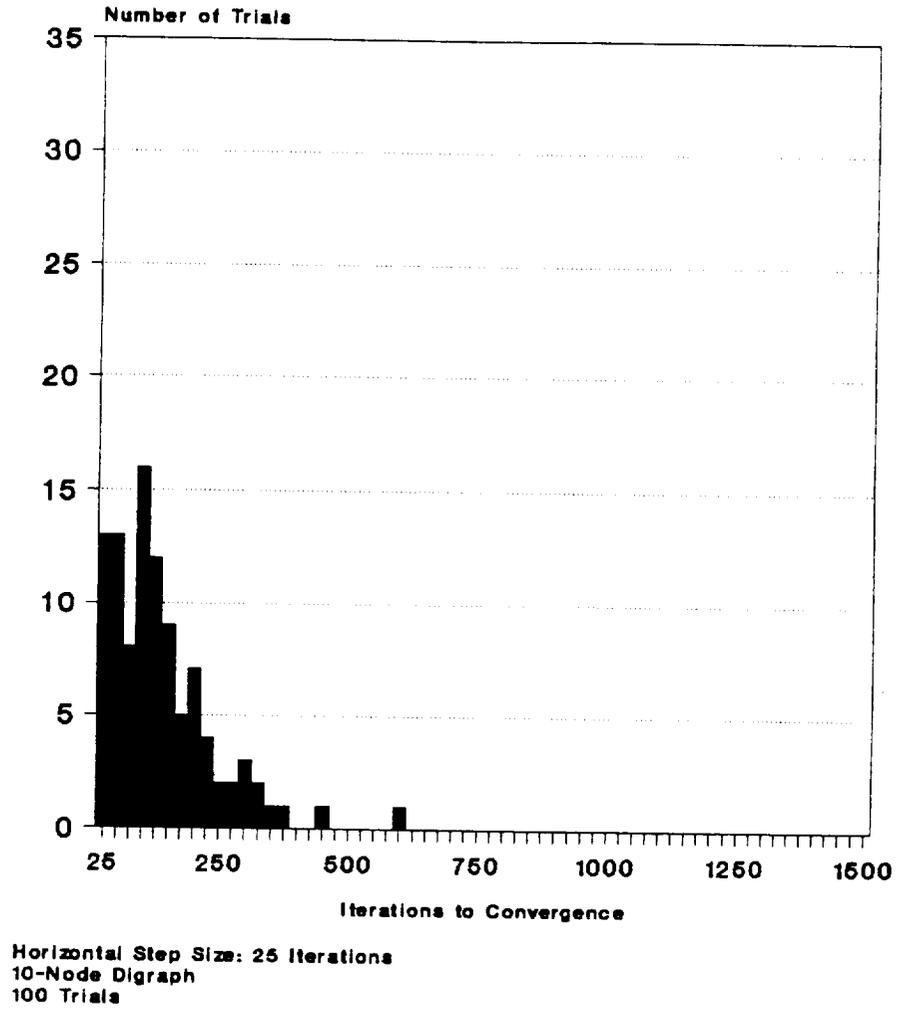


Figure 4.13. ACSN performance,
learning rate scaler: 0.7.

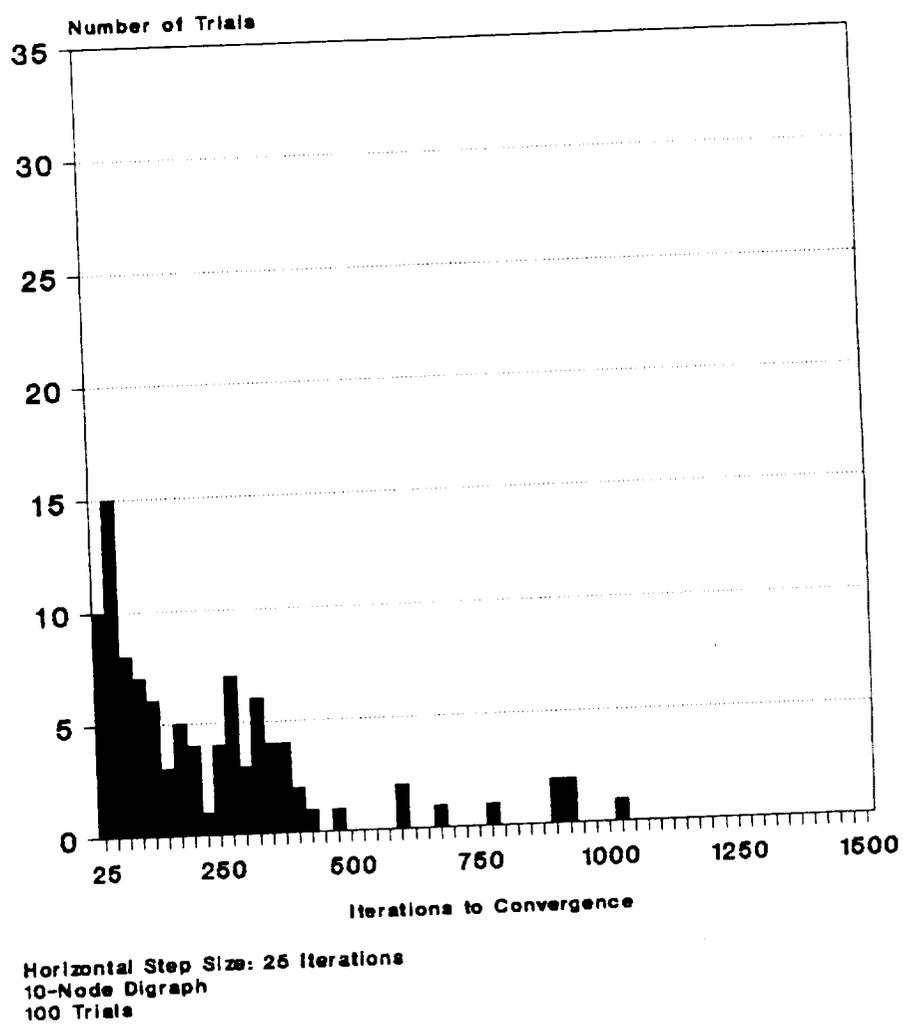


Figure 4.14. ACSN performance,
learning rate scaler: 0.8.

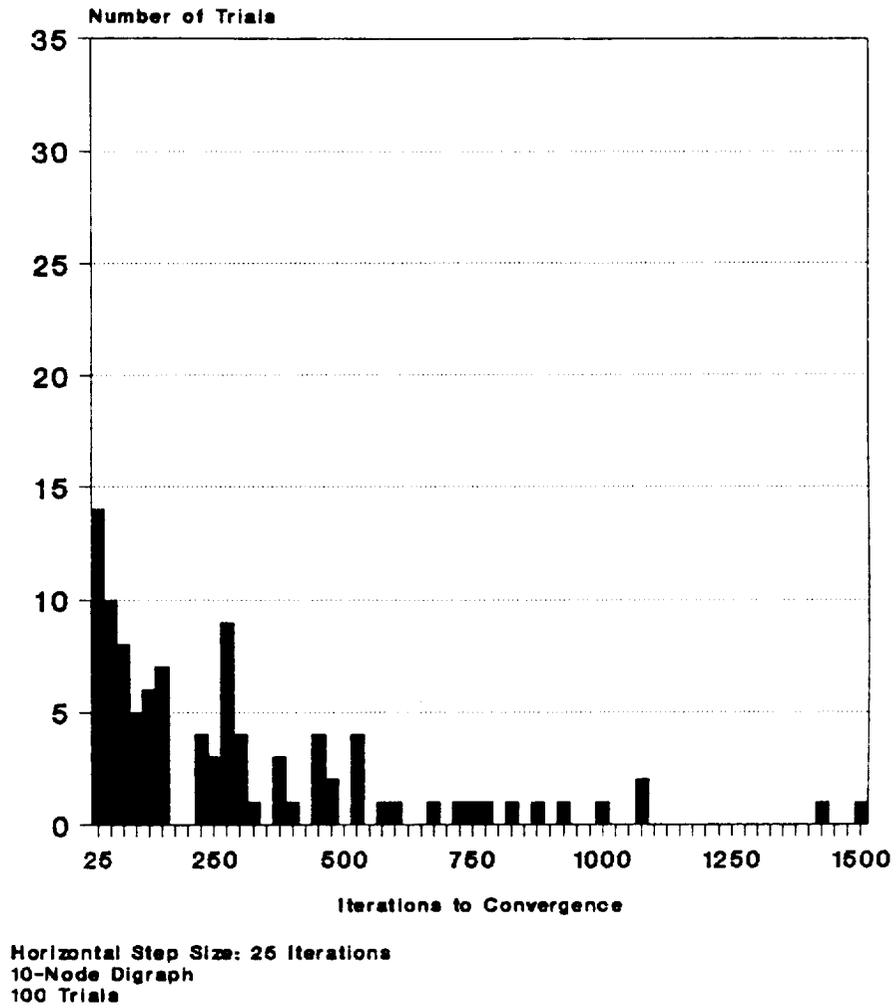


Figure 4.15. ACSN performance,
learning rate scaler: 0.9.

Note that for learning rate values of 0.2, 0.7, 0.8 and 0.9, the distributions have lower peaks and spread out considerably compared to the distributions in the interval [0.4, 0.6]. Although the distribution for a learning rate value of 0.1 has a high peak, the width of the peak is notably less than that of distributions associated with learning rates in the interval [0.4, 0.6].

The maximum number of iterations a trial took to converge was for the learning rate value of 0.9 and is approximately equal to 1500. Clearly, one can claim that the optimal value of learning rate parameter for the path search problem of 5 constraints for a 10-node digraph belongs to the interval [0.4, 0.6]. The optimality is defined based on the features of the frequency distribution of the successful trials. We will consider a learning rate value optimal if the associated distribution is located to the far left, right-skewed, not spread and as high as possible with maximum peak width.

Experiment 3

The goal of this test was to observe the effects of the varying network size on the frequency distribution of the successful trials. The learning rate was set to 0.5 for all network sizes tested. A total of 100 trials were run for each network size.

Networks of 5, 10, 15 and 20 nodes were considered initially but the simulations for the network of 20 nodes required more computational effort than currently available computing resources can provide. Hence, only a small number of trials were attempted for this network size. Full experiments was conducted on network sizes of 5, 10 and 15 with associated frequency distributions shown in figures 4.16-18. Each step size in those figures corresponds to 100 iterations.

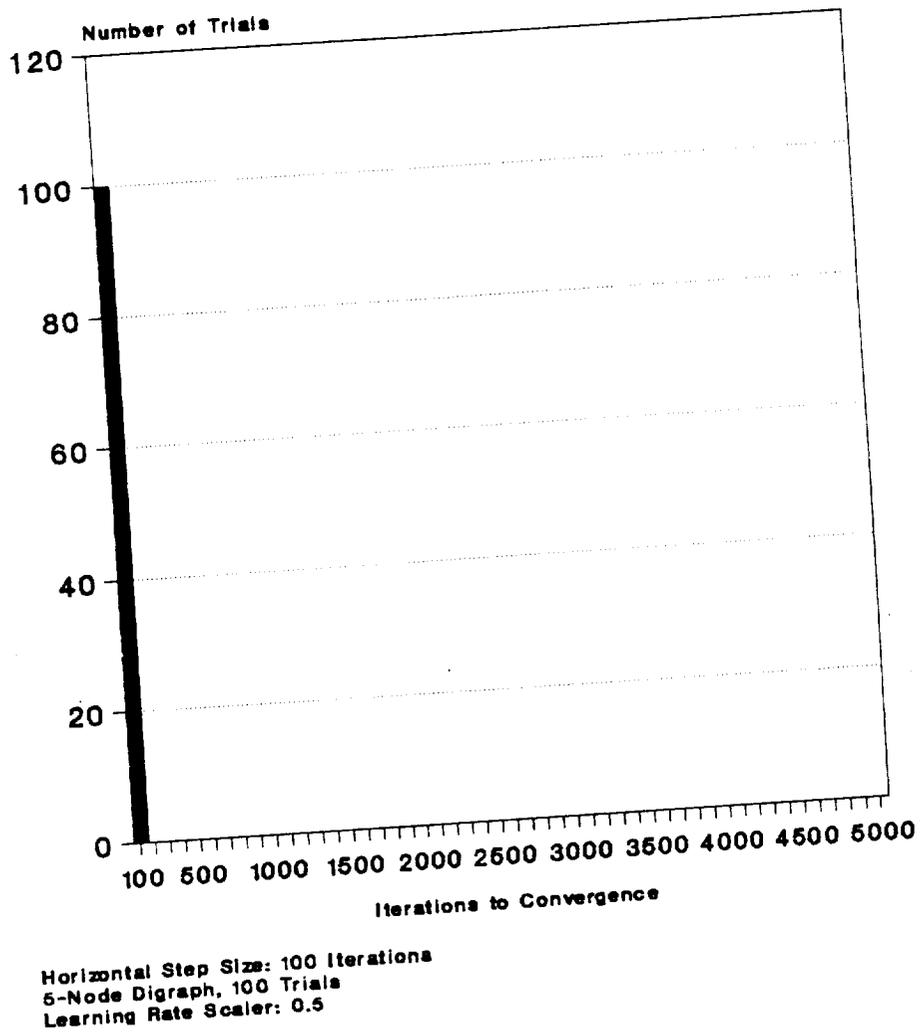
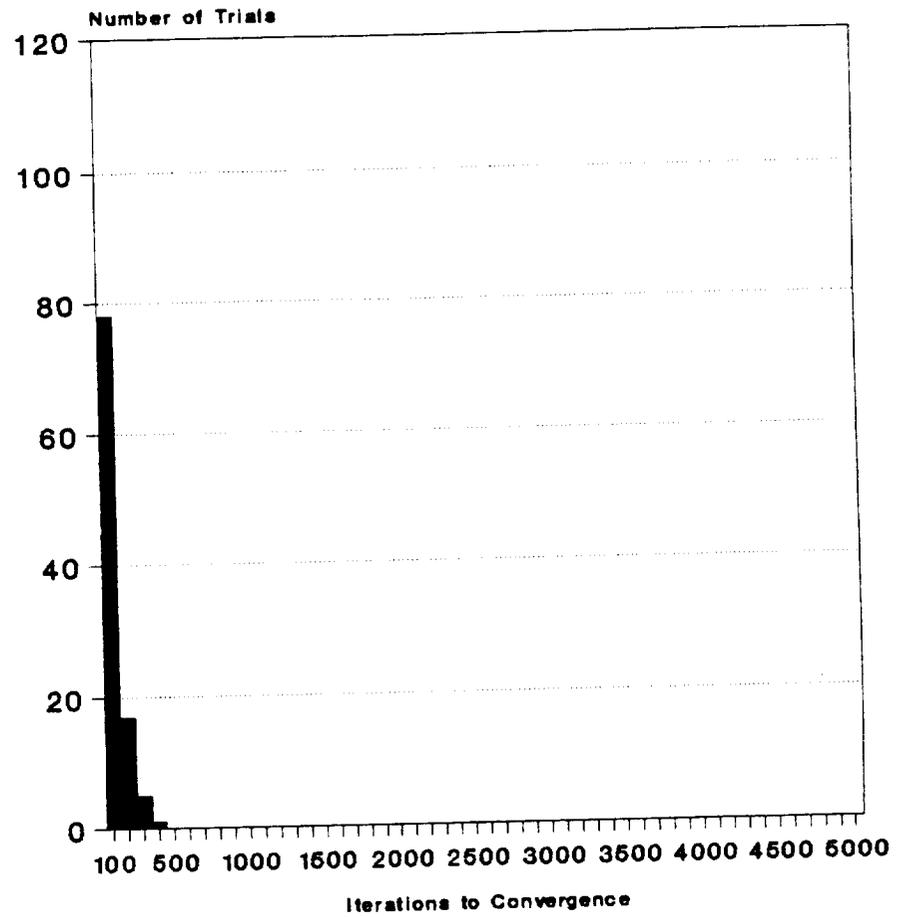


Figure 4.16. Adaptive Hopfield,
effect of network size 1.



Horizontal Step Size: 100 Iterations
10-Node Digraph, 100 Trials
Learning Rate Scaler: 0.5

Figure 4.17. Adaptive Hopfield,
effect of network size 2.

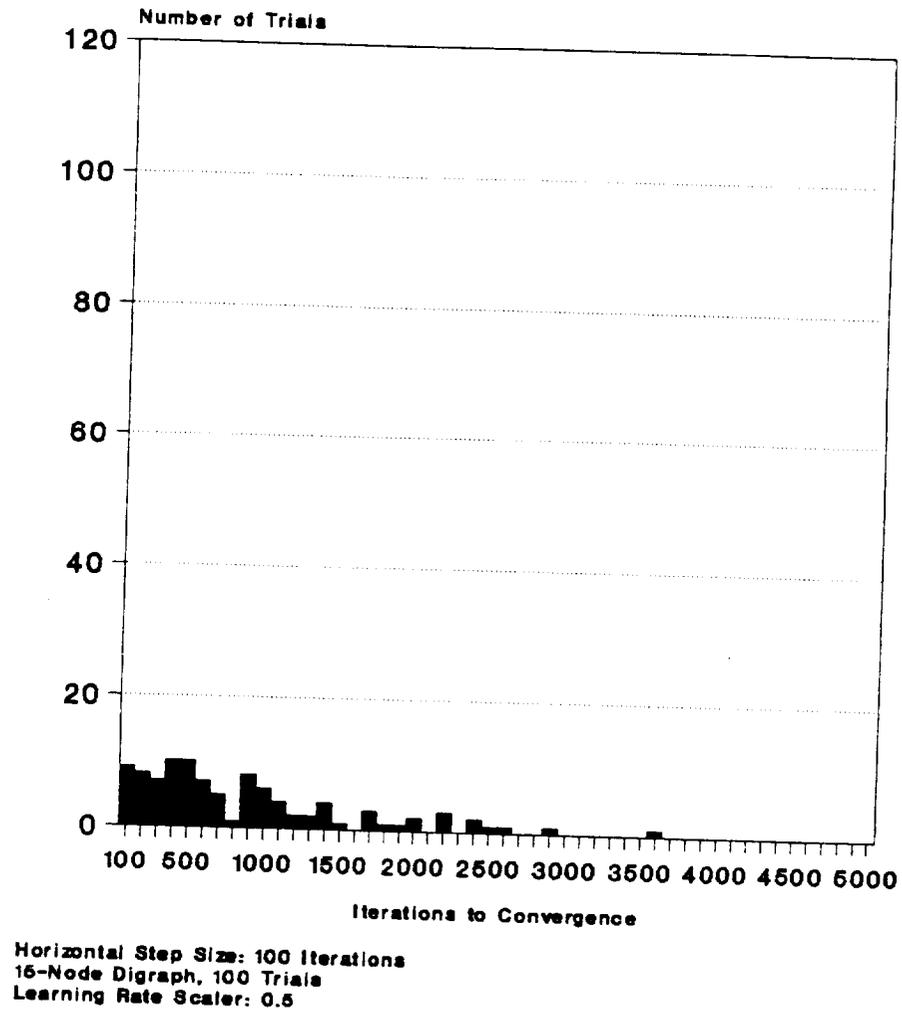


Figure 4.18. Adaptive Hopfield,
effect of network size 3.

In the case of 5 nodes, all trials converged within 100 iterations. It took up to 300 iterations for more than 90% of the 10-node networks to converge. The maximum number of iterations to convergence for a 10-node digraph was on the order of 500. Approximately, 80% of the trials converged to solution state within 1000 iterations for a 15-node digraph. Only a small percentage needed on the order of 2000 to 4000 iterations to converge.

An approximation to the function which shows variation of average number of iterations with respect to network size is shown in Figure 4.19. Although there is not enough data to infer the behavior of the function for network sizes above 15 nodes, if one follows the assumption that function behaves in a predictable manner in the region past 15 nodes an extension of the function in that region shows that average number of iterations for a 20-node digraph is more than approximately 2000. A couple of trials with a 20-node digraph indicates that number of iterations necessary for convergence is not less than that value, five trials resulted in 3806, 5711, 10001, 7156 and 5741 iterations with average equal to 6482.

4.5 Conclusions

The proposed adaptive constraint satisfaction network is the first closed-loop algorithm of its type. An earlier algorithm developed by Ackley [23] has similar features to the proposed algorithm but addresses a somewhat different type of constraint satisfaction/optimization problem. Two main differences are that the function to be optimized in Ackley's case is not known by the adaptation block and the adaptation signal provided by the same block is a scalar in the interval of $[-1, +1]$.

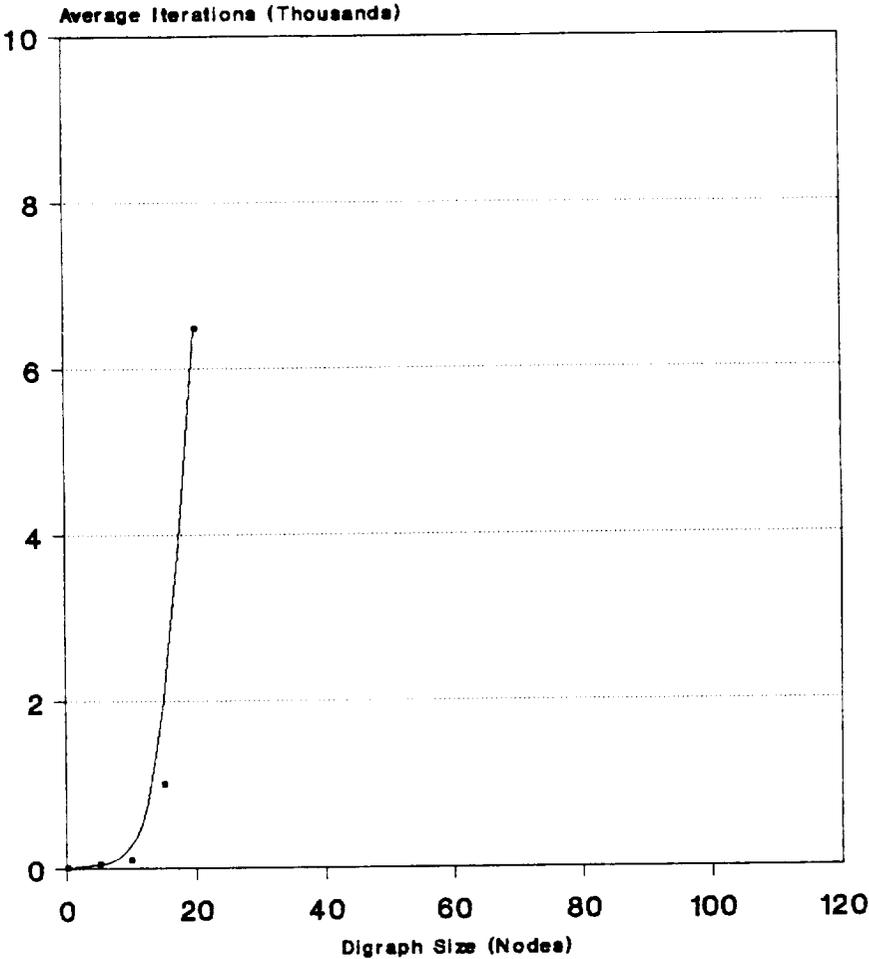


Figure 4.19. Adaptive Hopfield, network size vs. performance.

The results of experiment 1 show that the proposed network is very promising for moderate size problems. The performance of the algorithm for larger sized networks remains to be seen. The performance of the adaptive network compared to closed-loop classical Hopfield network with gain parameters and state vectors randomly initialized after each iteration is visibly superior as shown by the frequency distributions of the successful trials. The effects of the learning rate parameter on the algorithm performance is also noted such that there exists an optimal interval for learning rate values for which algorithm is most successful.

4.6 Future Research

A complete mathematical description of the adaptation algorithm and the closed-loop system will be attempted. This includes analysis of the effects of restructuring of the energy function in N-dimensional space by adapting the weights, effects of the initial state vector on convergence rate and the effects of random update order on the performance of the closed-loop search algorithm.

The adaptation algorithm will be analyzed from a viewpoint of a reinforcement learning algorithm. A reinforcement algorithm generally employs a one-bit piece of information compared to the proposed algorithm which utilizes more detailed data during the adaptation cycle, thus one would expect the proposed algorithm to converge much faster.

The closed-loop adaptive algorithm is located somewhere in the middle in the spectrum of learning algorithms. At one extreme is reinforcement learning and at the other extreme is backpropagation [1]. It is also of interest to identify some sort of quantity to use whose expected value or the value itself might be optimized by the adaptation algorithm.

The testing of the proposed algorithm with a complete set of problems needs to be done since the initial study only included the path search problem. The set of problems should be selected such that the performance of the algorithm with respect to the number of constraints, the size of the problem which implies the size of the network and the learning rate parameter are clearly observed.

References

- [1] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*. Cambridge, MA: The MIT Press, 1986.
- [2] J. J. Hopfield and D. W. Tank, "Neural Computations of Decisions in Optimization Problems," *Biological Cybernetics* 52, pp. 141-152, 1985.
- [3] J. J. Hopfield and D. W. Tank, "Computing with Neural Networks: A Model," *Science* 233, pp. 625-632, 1986.
- [4] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Properties," *Proc. Nat. Acad. Sci., USA* 79, pp. 2554-2558, 1982.
- [5] J. J. Hopfield, "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons," *Proc. Nat. Acad. Sci., USA*, vol. 81, pp. 3088-3092, 1984.
- [6] P. K. Mazaika, "A Mathematical Model of the Boltzmann Machine," *Proc. of ICNN*, vol. III, pp. 157-163, 1987.
- [7] D. H. Ackley, G. E. Hinton and T. J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, vol. 9, pp. 147-169, 1985.
- [8] J. P. Tremblay and R. Monahar, *Discrete Mathematical Structures with Applications to Computer Science*. New York: McGraw-Hill Inc., 1975.
- [9] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721-741, 1984.
- [10] Hartmanis, J. and Stearns, R. E., *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1966.
- [11] J. Ramanujam and P. Sadayappan, "Optimization by Neural Networks," *Proceedings of IJCNN-88*, vol. II, pp. 325-332, June 1988.
- [12] G. A. Tagliarini and E. W. Page, "Solving Constraint Satisfaction Problems with Neural Networks," *Proceedings of IEEE First International Conference on Neural Networks*, vol. III, pp. 741-747, June, 1987.
- [13] A. Ginzburg, *Algebraic Theory of Automata*. New York: ACM Monograph series, Academic Press, 1968.

- [14] T. J. Sejnowski, "Higher Order Boltzmann Machines," *AIP Conference Proceedings 151, Neural Networks for Computing*, Snowbird, Utah, 1986.
- [15] C. L. Masti and D. L. Livingston, "Neural Networks for Addressing the Decomposition Problem in Task Planning," *Proceedings of the International Joint Conference on Neural Networks, IJCNN-90-WASH-DC*, 1990.
- [16] Y. C. Lee, G. Doolen, H. H. Chen, G. Z. Sun, T. Maxwell, H. Y. Lee, and L. C. Giles, "Machine Learning a Higher Order Correlation Network," *Physica 22D*, pp. 276-306, North-Holland, Amsterdam, 1986.
- [17] M. W. Hirsch, "Convergence in Neural Nets," *Proc. IJCNN '88*, vol. II, pp. 115-125, 1988.
- [18] A. Krowitz, L. Rendell and B. Hohensee, "The State Space of Memory Clusters in the Hopfield Network," *Proc. IJCNN '89*, vol. III, pp. 339-346, 1989.
- [19] S. U. Hegde, J. L. Sweet and W. B. Levy, "Determination of Parameters in a Hopfield/Tank Computational Network," *Proc. IJCNN '88*, vol. II, pp. 291-298, 1988.
- [20] G. W. Davis and A. Ansari, "Sensitivity Analysis of Hopfield Neural Net," *Proc. IJCNN '89*, vol. III, pp. 325-328, 1989.
- [21] J. Bruck and J. W. Goodman, "A Generalized Convergence Theorem for Neural Networks and Its Applications in Combinatorial Optimization," *Proc. IJCNN '89*, vol. III, pp. 649-656, 1989.
- [22] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1985.
- [23] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. Boston: Kluwer Academic Publishers, 1987.