Syracuse University

SURFACE

Electrical Engineering and Computer Science - Technical Reports

College of Engineering and Computer Science

1-1996

Unsupervised Algorithms for Learning Emergent Spatio-Temporal Correlations

Chaitanya Tumuluri
Syracuse University

Follow this and additional works at: https://surface.syr.edu/eecs_techreports

 Part of the Computer Sciences Commons

Recommended Citation

Tumuluri, Chaitanya, "Unsupervised Algorithms for Learning Emergent Spatio-Temporal Correlations" (1996). *Electrical Engineering and Computer Science - Technical Reports*. 145.
https://surface.syr.edu/eecs_techreports/145

This Report is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science - Technical Reports by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

SU-CIS-96-1

Unsupervised Algorithms for Learning Emergent Spatio-Temporal Correlations

C. Tumuluri, C. Mohan, A. Choudhary

January 1996

School of Computer and Information Science
Syracuse University
Suite 2-120, Center for Science and Technology
Syracuse, New York 13244-4100

Unsupervised Algorithms for Learning Emergent Spatio-Temporal Correlations *

Chaitanya Tumuluri	Chilukuri K. Mohan	Alok N. Choudhary
tumuluri@cat.syr.edu	*ckmohan@syr.edu*	*choudhar@cat.syr.edu*
Phone: (315) 443-1626	*Phone: (315) 443-2322*	*Phone: (315) 443-4280*
Fax: (315) 443-2583	*Fax: (315) 443-1122*	*Fax: (315) 443-2583*
121 Link Hall, ECE Dept.	2-120 CST, School of CIS	121 Link Hall, ECE Dept.

Syracuse University, Syracuse NY 13244

January 9, 1996

Abstract

Many applications require the extraction of spatiotemporal correlations among dynamically emergent features of non-stationary distributions. In such applications it is not possible to obtain an *a priori* analytical characterization of the emergent distribution. This paper extends the Growing Cell Structures (GCS) network and presents two novel (GIST and GEST) networks, which combine unsupervised feature-extraction and Hebbian learning, for tracking such emergent correlations. The networks were successfully tested on the challenging Data Mapping problem, using an execution driven simulation of their implementation in hardware. The results of the simulations show the successful use of the GIST and GEST networks for extracting spatiotemporal correlation information among emergent features of previously unknown distributions and, indicate the feasibility of hardware implementation for online use. Of the two networks, the GEST network evinced better performance in terms of the network map stability, feature/correlation tracking ability and network sizes evolved.

*This work was supported in part by an NSF Young Investigator Award CCR-9357840.

1 Introduction

Several important and difficult problems require tracking emergent spatio-temporal correlations. This involves tracking correlations among useful features of time-varying data distributions, where it may not be possible to obtain *a priori* analytical characterization of the processes generating the data. Hence, tools used to model such distributions need to function without assumptions about the underlying distributions. **A**rtificial **N**eural **N**etworks (ANNs) provide such tools. In this paper, we present two ANN models for **(a)** modeling emergent distributions and **(b)** tracking emergent correlations among distribution features.

Time-varying distributions arise in many stochastic physical processes such as tracking co-ordinated multi-target movements, Brownian motion and dynamics of insect swarms [1]. Usually, the aggregate temporal behavior of similar groups of points in the signal space (perhaps modes of the distribution) is of interest, rather than the behavioral evolution of individual points. Thus, a system to track these distributions must extract such modes or aggregates of similar data points in the signal space online. Unsupervised clustering algorithms (including ANNs) can be used for modeling such time-varying aggregates. Temporal behavior of the aggregates can be characterized in various application-specific ways. In this paper, we use a widely applicable measure, obtained in terms of temporal correlations in the evolution of the aggregates in the **S**ignal **V**ector **S**pace (SVS).

A prime illustration of emergent spatiotemporal dynamics is seen in the Data Mapping problem [2, 3]. This problem requires the mapping of disjoint sets of data to processors in a multiprocessor system, such that off-processor data accesses are minimized [3]. The mapping or migration of data requires online modeling of the distribution of references to data and tracking temporal correlations in the references. Data Mapping is known to be an NP-complete problem since it can be viewed as a graph partitioning problem [4] or as resource allocation problem [5] and hence one can only hope to obtain good 'suboptimal' solutions. ANNs have been used primarily as a *decision making* tool in their application to Data Mapping, for optimizing the allocation of computations on data to processors, in minimizing an objective function. In contrast, this paper develops and uses ANNs to serve as *online modeling* tools for capturing emergent data correlations among concurrent computations. The emergent correlation information is also described in terms of *data locality* or *computational locality*.

An unsupervised algorithm is needed for modeling the emergent (and previously unknown) distributions of data-aggregates in processor references. Among unsupervised ANNs, Fritzke's **G**rowing **C**ell **S**tructures (GCS) network in [6, 7] is more promising for such applications, primarily because of its non-stochastic learning dynamics and its adaptive architectural dynamics (network growth and shrinkage). The superior

performance of the GCS network [7] over other unsupervised ANNs such as Kohonen's SOM [8] can be traced to these two features. Further details are provided in Section 2. However, the GCS algorithm contains no provisions for tracking the emergent temporal correlations in the evolution of data-clusters. This paper presents two new networks, viz. the **G**CS **I**nstantaneous **S**patio-**T**emporal (GIST) network and the **G**CS **E**pochal **S**patio-**T**emporal (GEST) network, as well as their learning algorithms, and demonstrates their success in learning the spatiotemporal dynamics on the challenging Data Mapping problem. The GEST network combines the Hebbian learning algorithm with the self-organized learning methodology. The GIST and GEST networks are collectively referred to as the GST network in the remainder of the presentation.

The rest of the paper is organized as follows. Section 2 provides a survey of the background for the current efforts. Section 3 provides details of the GST network architecture. Subsequently, Section 4 describes the spatial mapping dynamics of the GST network. Section 5 then formulates the instantaneous and epochal methods for capturing the temporal correlations in emergent features within the GIST and GEST networks. The data mapping problem is defined in Section 6, and the use of the GST network in solving this problem is detailed in Section 7. Sections 8 and 9 describe the data mapping problem in relation to three representative application programs and the GST network hardware implementation model assumed. The simulation results using the GST network in these three cases are presented in Section 10. Finally, Section 11 presents the conclusions derived from the experiments.

2 Background

Data mapping and similar applications in other fields, such as pattern recognition and adaptive control, are commonly concerned with the decision-making problem of allocating/partitioning/classifying incoming signals or data into one of many decision classes while minimizing an objective or cost function [4, 9]. Usually, decision-making systems have a preprocessing stage for performing feature-selection (and consequent dimensional selection) using incoming signals to ease the decision-making task. This study, by contrast, is focussed on the use of ANNs for modeling (i.e. feature extraction/selection) rather than decision-making. The most common technique in the literature performs feature-selection by minimization of an entropy measure derived from *a priori* probability distributions of the fixed decision-classes [9].

The Karhunen-Loève expansion offers an alternative strategy in situations where the fixed decision class distributions are not known *a priori* [10]. Another class of techniques involve functional approximation of so called "feature-functions" using stochastic methods or kernel-approximation techniques [9]. An interesting variant of such stochastic methods is the **S**elf **O**rganizing **M**ap ANN (SOM) [8], which performs *automatic*

feature-selection and dimensional-selection while formulating spatial maps of the unknown distributions. There is no strict statistical analog for the SOM network since it maps from a discrete or continuous high-dimensional space to a discrete low-dimensional space of feature-clusters [11].

However, all the aforementioned techniques for feature-selection implicitly assume stationarity of the underlying decision class distributions. In the class of applications being considered in this study, there is no *a priori* knowledge of the decision class distributions or their stationarity properties. Hence, the feature-selection technique must be able to *adaptively* remap nonstationary distributions online. For such applications, it is most natural to use non-parametric artificial neural networks whose architecture is adaptable, such as the GCS network [6], for performing *automatic* feature-selection and for mapping non-stationary distributions.

The GCS network has two advantages over the Kohonen Map. Firstly, its architectural adaptivity enables it to remap temporally-varying unknown distributions quickly. There is a fair amount of research into network-architecture construction using various approaches such as the analytic approach in the Cascade Correlation algorithm [12], stochastic optimization approach for pruning [13], genetic algorithms for evolutionary optimization in network construction [14] as well as the GCS method [6]. The simplicity of the GCS algorithm furthers its appeal among all these algorithms and is the second advantage of the GCS algorithm. The simpler non-stochastic GCS algorithm uses fixed network-parameter values, eliminating the need for a network-parameter cooling schedule during stochastic approximation. The time-varying nature of the distributions being modeled complicates the formulation of cooling schedules for approaches based on stochastic approximation. Other performance comparisons are provided by Fritzke in [7].

The use of ANNs for temporal processing can be classified into two categories. The first category uses variants of feedforward ANNs incorporating temporal filtering, feedback and error-correction learning for prediction tasks. The FIR Multilayer Perceptron [15] variant of the Backpropagation algorithm [16] and the Real Time Recurrent Network [17], for example, are commonly used in predicting values of system output-parameters based on the past history of such parameter values. The second category uses Hebbian learning and its variations for learning, recognition and recall of spatial patterns using temporal correlations among pattern-features which are determined *a priori*. An example application is character recognition using an instar-outstar network-mesh [18] or the Hopfield network mesh [19], where the neurons representing the image pixel-grid constitute the features of various characters. Other examples of such networks include the avalanche matched-filter and temporal associative memory among many others [20]. Usually, the network architecture constitutes an implicit and static determinant of the features in feature-space in such Hebbian-

learning networks.

The class of problems studied here produces temporally variable features (dynamically changing data-clusters) and requires the determination of temporal correlations in the evolution of these features. Hence, the architecturally adaptive GCS network is enhanced for capturing temporal correlations in the spatial feature-clusters as follows:

- The GCS spatial mapping dynamics is enhanced for remapping and tracking features in emergent non-stationary distributions and,
- Additional temporal connections is used to learn emergent correlations in the evolution of features using the Differential Hebbian learning dynamics [21, 22, 23].

There could be many applications for such a network combining the Hebbian learning schema with the unsupervised feature-mapping dynamics of architecturally adaptive ANNs. The data mapping problem solved in this study is one such application. Another example would be in learning production rule probabilities of stochastic grammars in syntactic pattern-recognition [24]. In such grammars, the emergent features would map to linguistic non-terminals and the temporal-correlations among features would translate to the production-rule probabilities of formulating strings using the non-terminals. Further exploration of this application is left for future work.

3 Spatio-Temporal Growing Cell Structures

This section describes the network architectures of the GIST and the GEST networks, and highlights the extensions made to the basic GCS architecture.

The STGCS network is a 2-dimensional network forming planar maps and the initial topology is a 2-dimensional triangle. Neuron addition and deletion during self-organization must always result in a network topology consisting of triangles as in the 2-d GCS network. Figure 1 illustrates the basic topology labelled **[A]**, the topology after neuron addition labelled **[B]** and the topology labelled [C] after neuron deletion. Note that the deletion of *N2* would leave *N1* connected only to *N5* and would not form a complete triangle. Since the topology must always consist of triangles, *N1* must consequently be deleted as well. Inputs to the network are simultaneously fed to all neurons via weighted input lines labelled W^I as shown in figure 2**[A]**.

The STGCS network, in addition, connects every neuron to all neurons by weighted, directed lateral temporal connections, as shown in Figure 2**[B]**. The lateral weight W_{ij}, indicates the connection from neuron j to neuron i and records raw measures of the emergent (pairwise) temporal correlations in the activations

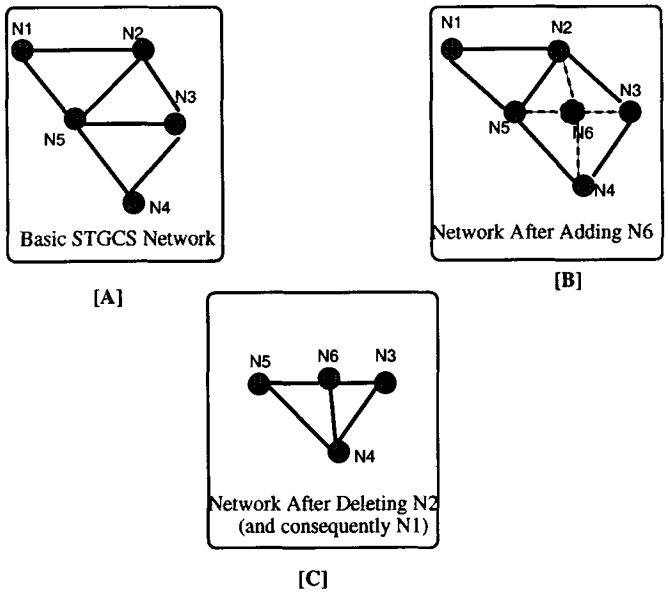

Figure 1: GIST and GEST Topology and Architectural Dynamics

of neuron i and neuron j ($W_{ij} \neq W_{ji}$). Thus each neuron has three kinds of connections associated with it, viz. the input connection, the topological connection and the lateral temporal connection as illustrated in Figure 2. A **S**ignal **R**eceptivity **M**eter (SRM) records the number of hits to each neuron over an epoch as in figure 2[**B**]. The SRM is an enhancement over the **S**ignal **F**requency **C**ounter (SFC) of the GCS network, which improves network stability during neuron deletions.

The next two sections describe the spatial and temporal dynamics of the GIST and the GEST networks. The spatial dynamics map the emergent data-aggregate or data-cluster distributions in the signal space and are common to both the GIST and GEST networks. The temporal dynamics track the correlations in the evolution of such clusters and differ for the two networks. In what follows the two networks will be jointly referred to as the GST network, unless one or the other is explicitly specified.

4 GST Spatial Mapping Dynamics

The GST spatial mapping dynamics position neurons at cluster centers in the input signal space. There are two modes of GST network operation: the training mode and the production mode. In the training mode, the network learns an approximation of the time-varying distribution offline, using training data. During the production mode, the approximate mapping is continually refined online using actual incoming data. Network performance parameters, evaluated at epochal checkpoints, help decide if the network needs offline retraining. The training-set size during a training phase determines the epoch length during the subsequent production mode of operation in the GIST network. In the GEST network, however, the production mode

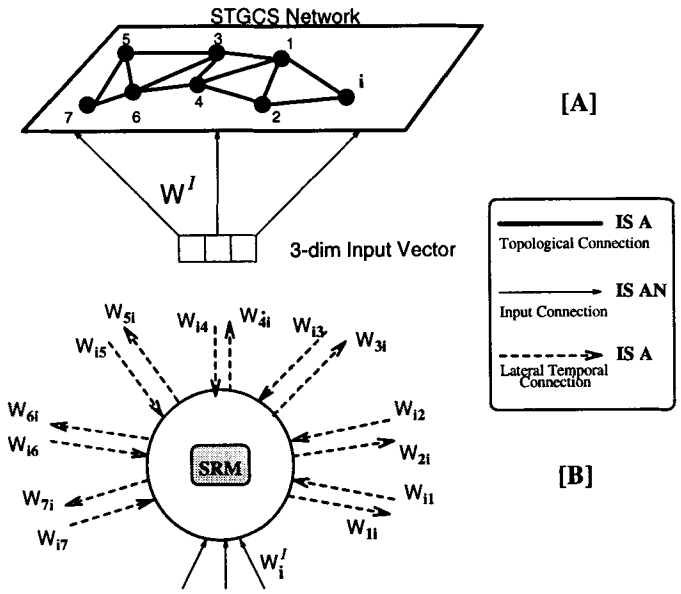

Figure 2: GIST and GEST Networks: Sub-figure **[A]** illustrates: *(a)* Inter-neuron topological connections within the network-plane, *(b)* Input connections to all neurons labelled W^I. Sub-figure **[B]** illustrates: *(a)* Input connections to neuron **i**, *(b)* Lateral connections of neuron **i** to all neurons in the network, *(c)* Neuron SRM.

epoch length is dynamically determined, as outlined in Section 7. Both the training and production modes of operation involve the following basic operations in the GST network.

1. Let A be the set consisting of all the neurons in the GST network. Let W^I to be the set of all incoming weight vectors $\vec{W}_i^I = (\vec{W}_{i1}^I, \ldots, \vec{W}_{iN}^I)$, $i \in A$.

2. Let $\vec{\nu}$ be drawn from the emergent temporal distribution of the N-dimensional **S**ignal **V**ector **S**pace SVS ($\vec{\nu} \in SVS$).

3. Determine, for a given input signal $\vec{\nu}$, the **Q**uantization **E**rror (QE) of each neuron based on which the winner neuron is chosen such that:

$$\| \vec{W}_{win}^I - \vec{\nu} \| = \min_{i \in A} \| \vec{W}_i^I - \vec{\nu} \|$$

where $\| \bullet \|$ indicates the Euclidean norm (QE) and $\vec{W}_{win}^I$ denotes the winner neuron's input weight vector.

4. The set, $Neigh$, of all neighbors of the winner neuron consists of all neurons directly connected to the winner neuron in the triangular topology of the GST network. Adapt the input weight vector $\vec{W}_{win}^I$ of

the winner and those of its neighbors ($\vec{W}_j^I$) according to:

$$\triangle \vec{W}_{win}^I = \eta_{win}(\vec{\nu} - \vec{W}_{win}^I) \quad \text{and} \quad \triangle \vec{W}_j^I = \eta_N(\vec{\nu} - \vec{W}_j^I), \quad j \in Neigh \tag{1}$$

η_{win} and η_N are adaptation **constants** (learning rates) for the winner and its neighbors, respectively[1].

5. Let τ_i be the SRM value of the neuron $i \in A$. The SRM update rule for the winner and its neighbors is:

$$\triangle \tau_{win} = 1.0 \quad \text{and} \quad \triangle \tau_j = \frac{\| \vec{W}_{win}^I - \vec{\nu} \|}{\| \vec{W}_j^I - \vec{\nu} \|}, \quad j \in Neigh \tag{2}$$

6. All the τ values in the network are decremented by a fixed fraction α to prevent unbounded growth:

$$\triangle \tau_i = -(\alpha \times \tau_i) \tag{3}$$

7. All simulations in this study used the following parameter values, adapted from Fritzke's simulations in [6]: $\alpha = 0.01$, $\xi_B = 0.05$ and $\xi_N = 0.05 \times \xi_B$. Variations in the choice of these parameters do not affect the quality of the resulting maps unduly. Fritzke's parameter settings were determined empirically to produce a reasonably fast solution.

4.1 GST Performance Measures

The averaged neuronal QE of a neuron $i \in A$ is defined to be the statistical mean of its QEs over all inputs mapped to it (i.e., input signals for which i was the winner) over an epoch. Map quality is measured by the **A**veraged **M**ap **QE** (AME) which is the network average of all averaged neuronal QEs. In the training mode, the network is trained to map the offline exemplar SVS distribution until the AME decreases to a preset upper bound. The efficacy of individual neuronal mapping is estimated by the **D**eviation in the **AME** (DME). The DME is simply the standard deviation of the averaged neuronal QE distribution, where the distribution-mean is the AME value. The GST network is said to have converged on the training set during the training phase, when the network AME and DME satisfy their bounds under "positional equilibrium" (to be defined in Section 4.2).

Another performance criterion is the network sizes evolved using the GST algorithms. Finally, the network disregards incoming signals from the SVS during training mode. The sampling efficiency of the STGCS

[1] The update rule is $X_{new} = X_{old} + \triangle X$.

network measured by the ratio

$$\text{SampEff} = \frac{\text{No. of Inputs Processed}}{\text{Total No. of Inputs}}$$

indicates the network efficacy in mapping and tracking changes in the emergent SVS distributions online, without resorting to offline retraining.

4.2 GST Architectural Dynamics

The GST architectural dynamics relates to the structural adaptations in the network, i.e., the addition and deletion of neurons in the network as illustrated in subfigure **[A]** in Figure 1. In what follows, the original GCS addition and deletion dynamics are briefly described. Subsequently, extensions to the GCS dynamics incorporated in the GST networks are described. The GCS network positions neurons such that the preset network DME bound is achieved while satisfying the AME bound. The GCS dynamics accomplish this in two ways:

1. Periodic addition of a neuron, after every Γ_a input presentations. The new neuron is introduced as a topological neighbor of the neuron with the highest average QE in the GCS network.

2. Periodic deletion of *all* superfluous neurons, after every Γ_d presentations. Neurons become superfluous when their relative SFC values, $\tau_i / \sum_{j \in A}(\tau_j)$, are less than a preset threshold, where τ_j is the jth neuron's SFC value.

In equilibrium, for a given static input distribution and fixed network architecture, the input weight vector ($\vec{W}^I$) of each neuron oscillates about the centroid of the data samples mapped to it. In the simulations presented here, the GST network is considered to be in positional equilibrium if $\triangle AME$ over the current interval is less than 10% of $\triangle AME$ over the previous interval. Each interval refers to Γ_a or Γ_d presentations and equilibrium is usually achieved over several such intervals. In the GCS network, interval-lengths smaller than the time needed for achieving equilibrium produced unduly large networks. This problem was solved, in the GST network, by allowing neuron addition/deletion only under conditions of positional equilibrium.

The GCS deletion dynamics required the deletion of superfluous neurons as well as their neighbors possibly removing useful neurons in areas of high signal density in the SVS. The main reason for this was the SFC update rule which rewards only the winner neuron. This problem was solved in the GST network by replacing the SFC with the SRM. According to the SRM Update Rule (equation 2), the winner shares its win with its neighbors in proportion to their 'receptivity' to the input (quantified by the ratio of their QEs); hence

the name Signal Receptivity Meter. In doing so, the winner indirectly increases chances of its survival by sharing its "win" (increment in τ) with its neighbors which may be in areas of low signal density[2]. As a further precaution against undue map destruction and instability, the GST dynamics allow deletion of only **one** neuron (and resulting topologically dangling neighbours) every Γ_d presentations. This is in contrast with the GCS dynamics which require the deletion of *all* superfluous neurons and the resulting dangling neighbours.

4.3 GST Spatial Mapping Dynamics Summary

In summary, the basic GST dynamics differ from the GCS dynamics in three respects. The W^I adaptations continue during the production mode enabling refinement of training mode maps. The SRM values of both the winner and its neighbors are incremented. The SFCs are replaced by the SRMs in the GST network. Further, there are three differences between the training and production phases of operation in the GST network dynamics. The W_{ij} updates occur only in the production phase. Training phase W^I adaptations occur offline for a fixed set of training vectors while production phase W^I adaptations occur online in response to incoming data. Both neuron addition and deletion are allowed in the training mode while only neuron deletion occurs in the production mode.

These alterations to the GCS dynamics allow the GST network to quickly converge to stable maps of (and also track) non-stationary distributions *online*.

5 GST Temporal Mapping Dynamics

The dynamics described in the previous sections produce spatial maps of the underlying distributions in the SVS. This section details the production mode temporal dynamics involving: **(a)** lateral weight updates and **(b)** computation of temporal correlations in neuronal activations. A neuron i is said to be activated at time t, if it is the winner, denoted $Win(t)$, for the input at time t (i.e. $i = Win(t)$). The temporal correlation of a neuron's activations is a measure of the extent to which its activations are predicated upon the activations of some or all neurons in the network (including itself) over a prespecified interval of time. The instar lateral weights impinging on a neuron record the measure of that neuron's activational correlations over the prespecified correlation interval. Such temporal correlations in neuronal activations reflect correlations in the evolution of data aggregates or clusters in the SVS.

Two algorithms are presented in this section, viz., the instantaneous method defining the GIST network

[2]This bears a resemblance to Richard Dawkins' explanation for apparently "altruistic" behavior in biological organisms (cf. The Selfish Gene)

and the epochal method defining the GEST network, to capture activational correlations via update rules for the lateral weights. Both methods are specified using a tuple (Λ, Φ) representing the *Correlation Interval* and the *Correlation Update Rule* respectively. In the specification, Φ specifies the lateral weight update rule at the end of all neuronal activations within the correlation interval $[t - \Lambda, t]$. At the end of a correlation interval, each neuron i computes its total probability of activation (P_i) with respect to **all** neurons using its lateral weights, as described in Section 5.1.

The total probability (P_i) of neuron i is defined to be the *Signal Utility* of the cluster of input signals in the SVS mapped to this neuron. The *Signal Utility* measures the usefulness or contribution of this cluster toward the overall temporal evolution of the emergent SVS distributions.

5.1 *Signal Utility* Determination

This section details the computation of the total probabilities, given the network W_{ij} values. The local probability of signals in the data-cluster mapped to neuron j in the SVS is measured by the ratio $\hat{p_j}$, where:

$$\hat{p_j} = \frac{\tau_j}{\sum_{k \in A} \tau_k} \tag{4}$$

The correlation of neuron i's activations with the activations of neuron j, normalized with respect to all correlations predicated upon j's activations is given by $P_{(i|j)}$ where:

$$P_{(i|j)} = \frac{W_{ij}}{\sum_{k \in A} W_{kj}} \tag{5}$$

$P_{(i|j)}$ is the conditional probability of the evolution of cluster i with respect to the evolution of cluster j in the current SVS. The conditional probability $P_{(i|j)}$ together with the *a priori* probability $\hat{p_j}$ yields the total probability P_i. From the theorem of total probability [25], we obtain the *Signal Utility* of cluster i to be:

$$P_i = \sum_j (P_{(i|j)} \times \hat{p}_j) \tag{6}$$

Substituting for $P_{(i|j)}$ produces an equation purely in terms of the GST network parameters:

$$P_i = \sum_{j \in A} \left(\frac{W_{ij} \times \hat{p_j}}{\sum_{k \in A} W_{kj}} \right)$$

where the total probability is computed from the neuronal SRM and lateral weight values of the network. Both the GIST and the GEST networks compute the *Signal Utility* using this formulation. The primary difference between the networks is in their lateral weight update rules which are the subject of Section 5.2 and Section 5.3. Subsequently, Section 5.4 compares the two networks and their temporal dynamics.

5.2 GIST Network: Instantaneous Method

The instantaneous method dynamics for lateral weight updates in the GIST network is specified as follows:

- $\Lambda = 1$. Thus, successive input signals at times $t-1$ and t defining the correlation interval, correlate the successor-neuron's ($i = Win(t),\ \ i \in A$) activation with the activation of the predecessor-neuron ($j = Win(t-1),\ \ j \in A$). In effect, the instantaneous method records the first-order pairwise correlations among neuronal activations.

- Given Λ, i and j as above, Φ defines the W_{ij} update to be $\triangle W_{ij}(t) = 1.0$.

The lateral weights are normalized at the end of every 100 epochs by dividing each W_{ij} by $\max_{i,j \in A}(W_{ij})$. All lateral weights then have magnitudes less than unity, and subsequent Φ-updates (increments of unity) erase previous correlation information. The normalization thus accomplishes two tasks:

- It prevents unbounded growth of the W_{ij} weights.

- Lacking any means for recording *decreases* in correlation, it serves to erase outdated correlation information and forces the network to relearn the correlations periodically.

Thus, *first-order correlations* among pairs of neurons are *continuously* quantified by the instantaneous frequency (recorded in the lateral weights) with which the predecessor-successor relationship is established between them. These raw frequencies are normalized into relative frequencies ($P_{(i|j)}$) by equation 5. The total probability (P_i) of each cluster i ($i \in A$), is the probability of its evolution over the correlation interval given its normalized first-order correlations captured in $P_{(i|j)}$. The total probability of a cluster's evolution can be seen as a measure of its utility in defining the emergent overall distribution in the SVS and is taken to be the cluster's *Signal Utility* in the GIST network.

5.3 GEST Network: Epochal Method

The epochal method is inspired by the Drive Reinforcement Theory (DRT) of differential Hebbian learning developed independently by Harry Klopf [22] and Bart Kosko [21]. In the DRT, inputs arriving at a neuron are termed the *pre-synaptic activations*, while the output after processing the inputs via lateral weights is termed the *post-synaptic activation*. The DRT modifies lateral weights to reflect correlations between

changes in the current post-synaptic activation and the *history of changes* in the pre-synaptic activations. A re-examination of equation 6 reveals it to be an activation equation where P_i represents the post-synaptic activation and the $P_{(i|j)}$ terms define a squashing function acting on the $\hat{p}_j$ pre-synaptic activations. In the epochal method, lateral weight updates capture correlations between changes in P_i (at the end of the current correlation interval) and the history of changes in $\hat{p}$ over several recent correlation intervals.

The epochal method temporal dynamics of the GEST network is specified below:

- The correlation interval Λ is defined as follows:

$$\Lambda = \frac{\sum_{k \in A} \tau_k}{\mid \bar{\tau} - \sigma_\tau \mid} \tag{7}$$

  where $\bar{\tau}$ represents the statistical mean of all the SRM values in the network and σ_τ represents the standard deviation of the SRM values. This value estimates the number of inputs required to enable each neuron to be the winner at least once in one correlation interval, based on the statistical distribution of hits to neurons estimated by the τ values. The variable correlation interval ($\Lambda \geq 1$) allow the lateral weight updates to adaptively capture higher-order pairwise correlations as well.

- Given Λ, the lateral weight update rule (Φ) at the end of the correlation interval $[t - \Lambda, t]$, is defined as follows:

$$\Phi : \qquad \triangle W_{ij}(t) = \triangle P_i(t) \times \sum_{n=1}^{\kappa} (\mid W_{ij}(t - n\Lambda) \mid \times \triangle \hat{p}_j(t - n\Lambda))$$

  In all experiments presented here, the length of history chosen was $\kappa = 2$. The lateral weight update correlates changes in the post-synaptic activation ($\triangle P_i(t + \Lambda)$) at the end of the current correlation interval, with the history of changes in pre-synaptic activations over the past two correlation intervals. The history of pre-synaptic activations values and lateral weights values over κ is maintained using tapped delay lines on the neuronal SRMs and lateral weights. A positive correlation indicates that the (total probability of) evolution of data cluster i has an increased correlation with (the *a priori* probability of) the evolution of data cluster j and a negative correlation indicates otherwise.

Given the updated lateral weights that capture higher-order correlations in neuronal activations, the GEST network computes the *Signal Utility* as defined in Section 5.1. The use of higher-order correlations smoothes high-frequency variations in the correlations and extracts the long-term low-frequency correlations in neuronal activations. Thus, the GEST network's *Signal Utility* expresses an evolving cluster's long-term correlations with the evolution of other clusters in the SVS. For a more comprehensive discussion of the DRT

update rule, the interested reader is referred to [22] or [18].

5.4 GIST vs. GEST: Temporal Dynamics Comparison

The GIST network (instantaneous method) dynamics are simpler to understand and implement than the relatively sophisticated GEST network (epochal method) dynamics. One difference between the two dynamics is the Λ value from which they derive their names. The second difference lies in the network features being correlated. GIST network dynamics implicitly correlate first-order (successive) neuronal activations, whereas GEST dynamics implicitly correlate higher-order changes in probabilities of neuronal activations as well.

The GEST network has two distinct advantages over the GIST network. Firstly, the GEST dynamics use higher-order correlations in changes in P_i and $\hat{p}$ to smooth out high-frequency noise. Potentially, therefore, the GEST network *Signal Utility* values express low-frequency long-range variations. Secondly, the GIST dynamics do not explicitly decrement the lateral weight values (W_{ij}), and thus, large P_i values do not decay at times of low dependencies. The GEST dynamics correlate *changes* in (pre- and post-synaptic) activations and thus effect both increments and decrements in the lateral weights to automatically solve this problem.

Both the GIST and GEST dynamics assign the neuron total probability (P_i) as the *Signal Utility* value of all input signals mapped to neuron i. The P_i computation involves the generation of $P_{(Win|j)}$ and $\hat{p}_k$ values. The latter values are recomputed at every presentation, but extraction of $P_{(i|j)}$ differs in the two networks. In the GIST network, only $P_{(Win(t)|Win(t-1))}$ changes for each input signal requiring recomputations only at the winner neuron for deriving its new *Signal Utility*. In the GEST network there are no recomputations within a correlation interval, but a network-wide $P_{(i|j)}$ recomputation is required at the end of every correlation interval.

The weight update rule in the GIST dynamics is a simple increment, whereas the GEST weight update rule demands more bookkeeping (taps on neuronal $\hat{p}$ and W_{ij}) as well as increased computation. Thus, GEST dynamics are computationally tractable only if the network sizes are small (requiring fewer taps), and the Λ intervals are large enough to offset the periodic network wide $P_{(i|j)}$ recomputations.

6 GST Application: Problem Formulation

There has been a growing interest in the use of neural networks for solving diverse systems-related problems in modern computer systems. For example, at the architectural level, a Backpropagation network is evaluated in [26] for use as an alternative cache-replacement policy. At the operating system level, a Backpropagation network has been used for automatic allocation of computational resources in a heterogeneous

system to service incoming tasks in [27]. Similarly, comparator networks have been used in [28] for the same purpose. At the runtime-systems level, the PARTI system [2] uses the Hopfield-Tank network for suboptimal partitioning of data-dependancy graphs to achieve computational locality and load balance. All these efforts use neural networks as decision-making tools for optimization. It is generally known that there are faster *decision-making* tools which outperform ANNs and hence this study does not focus on this issue or use ANNs for optimization.

Rather, the GST network is used for *modeling* data-locality online in solving the data mapping problem. By online, it is meant that the GST networks operate concurrently with the ongoing application program execution. The following characterizing assumptions about computer systems and programming styles are made in formulating the problem:

- It is assumed that the computer system is a multiprocessor in which all processors execute the *same* set of instructions on *different* data elements (SPMD programming model).
- Memory in the system is distributed into geographically separated modules associated with processors in the system (Distributed Memory system).
- Processors are assumed to have direct access to data resident in remote-memory modules (Global Address Space model). The cost of access is higher for data in remote modules than local modules.
- Every data element is *owned* by one processor which performs computations on it ("Owner Computes" rule).
- Computation on a data element at every processor is abstracted into the sequence of memory accesses (or memory references) for other data elements and such references to other data elements (resident locally or remotely) during ongoing computation define *data-dependancies.*

Figure 3 illustrates the computational environment assumptions listed above. In data mapping, the data-dependency information is needed for the allocation of computations on data to processors. The problem of modeling temporally emergent dependencies (data locality) is solved by determining temporally emergent probabilities of *correlations* in references to data elements.

Memory is abstracted as consisting of indivisible and contiguous **A**tomic **M**emory **U**nits (AMUs). Data elements are assumed to be embedded within such AMUs and memory references generated during computations address such AMUs as illustrated in Figure 3. Each AMU has a *vector* of four descriptors associated with it. The fields in the vector are the *AMU Tag* for uniquely identifying each AMU, the *AMU Owner*

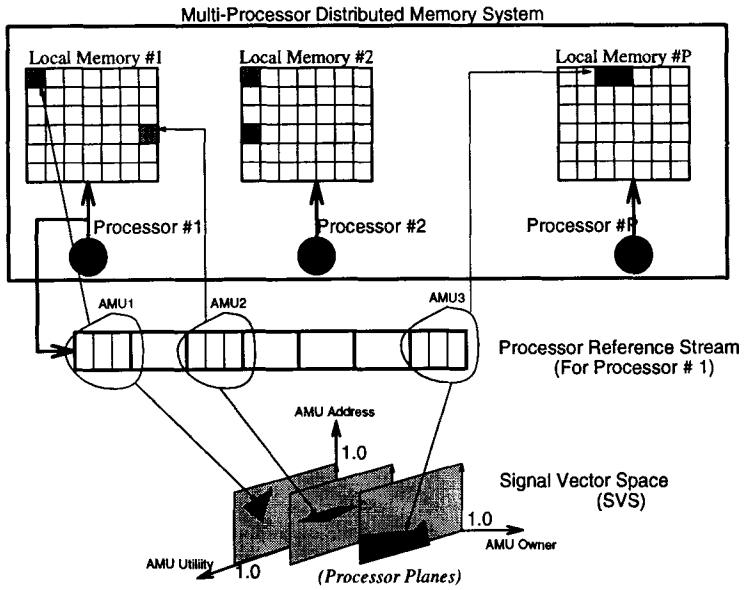

Figure 3: **Distributed Memory Multiprocessor system: AMU Processing Details**
A P processor-memory system is shown. Squares in local memory grids corresponds to AMUs. Reference stream of processor # 1 illustrates: (a) Local Reference to *AMU1* and *AMU2* and (b) Remote reference to *AMU3* (resident in Memory Module #P). The CMS at processor # 1 illustrates mapping of AMUs into Processor Planes in the CMS. GST network maps emergent AMU clusters and learns the correlation strength between them. Refer text for further details.

for tracking AMU ownership, the *AMU Utility* descriptor for recording the probability of local references to the AMU and finally, *AMU Work* for storing the computational workload of each AMU. The first three descriptors are used to capture the emergent locality. Hence, all AMUs reside in a three dimensional SVS as illustrated in Figure 3. Ongoing local computation induces changes in AMU descriptors producing non-stationary SVS distributions at each processor, consisting of planar clusters shown by the shaded spatial envelopes. The planes arise due to the use of fixed integer processor numbers to specify *AMU Owner* values. The SVS distribution is thus a spatial map of the current local working set of the processor [29].

Variations in *AMU Utility* are due to increases (decreases) in the frequency of references to various AMUs as they are inducted (expelled) from the local working-set. These values are used to decide where AMUs are best relocated in order to reduce the number of remote-memory references. The relocation reassigns ownership of the data, thus incrementally correcting computational load gradients (based on *AMU Work* values) to minimize computation time for solving the data mapping problem. Thus, the problem of modeling emergent data-dependencies consists of two components:

- Each processor must track the evolving subset of AMUs *needed* for the ongoing local computation.
- The measure of *need* computed should reflect the usefulness of the AMU at each processor for deciding

data *ownership*. The *Signal Utility* of each AMU, measuring its correlation with other AMUs referenced locally, satisfies this criterion.

7 GST Application: Solution Overview

A GST (either GIST or GEST) network is used at each processor, to model each local working-set by capturing spatial maps of AMU distributions in the local SVS from the processor reference streams. The *Signal Utility* value of the cluster to which each AMU is mapped is assigned as the *AMU Utility* value for the AMU. Offline training sets for networks are derived from past processor references and the current contents of local memory.

In this context, it is reasonable to say that $\hat{p_j}$ (*a priori* probability of neuron j activation) provides an *a priori* frequency-based probability measure, that AMU-cluster j contains AMUs of the current local working set. Also, $P_{(i|j)}$ (conditional probability of neuron i activation) can be viewed as the conditional probability that references to AMUs in cluster i are correlated with references to AMUs in cluster j. Thus, P_i is naturally interpreted as the probability of the presence of references belonging to cluster i in the reference stream. In other words, it connotes the **utility** of (or 'need' for) this cluster in the local working set. P_i, recomputed every Λ presentations, is assigned as the new *AMU Utility* value for all AMUs mapped to cluster i.

Further, under the constant decrementation in equation 3, each τ_i in the GST network can be assumed to express a time-averaged frequency of reference to AMUs belonging to AMU-cluster i. Thus, the sum of all network τ_i values is an approximation of the size of the local working set. This value (which is the numerator term in equation 7) is the size of the training set (and epoch-length in the subsequent production mode of operation) in the GEST network. Hence, variations in GEST Λ values provide a good indication of the variations in working set sizes. In the GIST network, however, the size of the training set is fixed at 512 exemplars in all experiments.

The network is required to distinguish between clusters resident on different processor planes in Figure 3 for decisions regarding data ownership. In the worst cases, these clusters may be distinguished only by their *AMU Owner* values. Thus the GST network mapping resolution should exceed $\frac{1}{TotalProc}$ to distinguish between AMUs owned by consecutively numbered processors. Also, the sample variance over an epoch indicates required mapping resolution. Hence the AME upper bound is set according to:

$$\text{AMEbound} = \min\left(\frac{1}{TotalProc}, \sigma_\nu\right), \quad \nu \in CMS$$

where σ_ν is the standard deviation of the AMU distribution in the local SVS. The results presented later

are for applications using 4, 8, 16, 24 and 32 processors.

The network is constrained to always maintain a DME value less than 50%. If the DME > 50% and the AME > AMEbound during the production mode, then the network is retrained. Neither the map AME nor DME are minimized as such, but are kept within bounds. The bound of 50% was determined empirically to produce relatively smaller network sizes (between 20 and 100 neurons) in the simulations. This study aims to provide an empirical proof-of-concept for using the GST network for capturing emergent locality. The quality of results obtained using a DME of 50% was sufficient for this purpose. Further tuning of parameters for optimizations is left for future studies.

The sampling efficiency of each local GST network is redefined to be the ratio:

$$\text{SampEff} = \frac{\text{No. of References Processed}}{\text{Total No. of References Produced}}$$

and is computed every 0.3 seconds of simulation clock time.

8 GST Application: Testbed

Three applications were used to test the GST network: **(1)** the Barnes-Hut algorithm for a Monte-Carlo N-body simulation of interacting galaxies [30, 31, 32], **(2)** the Unstructured Mesh kernel for an Eulerian solver [2, 33] and, **(3)** "WaTor", an ecological simulation of sharks and minnows in a toroidal ocean [34, 35]. The primary goal was to evaluate how well computational locality was maintained or achieved by the proposed system. The three applications chosen were representative of the range of locality flux normally encountered in parallel programs. The unstructured mesh simulation exemplifies applications with static locality characteristics, the N-body simulation typifies slowly changing localities for fixed dataset sizes and WaTor exhibits rapidly fluctuating localities among slowly varying dataset sizes.

The dataset for each of the 4, 8, 16 and 24 processor runs of the Unstructured Mesh application consisted of 2800 vertices and 17,377 edges, and was executed for 50 iterations. The 8 processor N-body simulation run involved 8192 stars, while the 16, 24 and 32 processor runs involved 16384 stars. The simulation was run for 10 iterations. Finally, the WaTor simulation parameters included a 200×75 ocean grid with 5000 sharks and 1563 minnows. The minnow and shark breeding ages were 3 and 10 iterations respectively and the shark starvation age was set at 3 iterations. The simulation was run for 10 iterations unless otherwise indicated.

GST Activities	Co-Processor Execution Speed
Training Vector Generation (T_{TVect})	7 KSR1 cycles
Snoop Vector Generation (T_{SVect})	62 KSR1 cycles
Input Vector Quantization (T_{Quant})	10 KSR1 cycles
Lateral Weights Update (T_{IM})	34 KSR1 cycles
Lateral Weights Update (T_{EM})	$(n^2 + 30n)$ KSR1 cycles
Neuron Addition (T_{Add})	100 KSR1 cycles
Neuron Deletion (T_{Del})	200 KSR1 cycles

Table 1: GST Simulation Timing Parameters. n in the T_{EM} timing refers to the network size.

9 GST Application: Simulation Model

We postulate the use of an on-chip GST coprocessor unit per processor in the multiprocessor system as illustrated in Figure 4. The coprocessor is a functional unit executing concurrently with the main processor, which could be implemented using synaptic parallelism with $\Sigma\Pi\Sigma$ pipelined functional units as in [36]. The timings for the main GST activities is given in Table 1, and have been derived from simulations on the KSR1 and results in [37, 38, 36].

An execution driven simulation was implemented on the KSR1 [38] to simulate concurrent execution of the GST coprocessor and the main application. A GST thread was spawned for each application thread and run concurrently on a dedicated processor. Each application thread's AMU references were buffered along with the inter-reference times and communicated to the corresponding GST thread, which regulated its actions as per Table 1 and the inter-reference times. GST threads periodically check load-levels of their application threads and the remote AMUs referenced in the recent past captured in the GST neuron memory. The load-levels are compared locally and at the current owner of each AMU. If the local load is less and the local *AMU Utility* is greater, the AMU ownership is acquired by the local thread and its *AMU Owner* descriptor updated. Thus, AMU transfers try to balance *emergent* load and locality gradients in the system via a task *PULL* mechanism.

Finally, each WaTor application thread's task partition consists of minnows distributed in the toroidal ocean mesh. The minnow movement and survival dynamics loop (minnow loop) requires near-neighbor interactions. Remote references would be minimized if each thread's partition consisted of neighboring minnows, thus maximizing local references to yield good locality characteristics. Hence, a measure of execution locality achieved is the average Euclidean distance (in the ocean) separating minnows within each task partition.

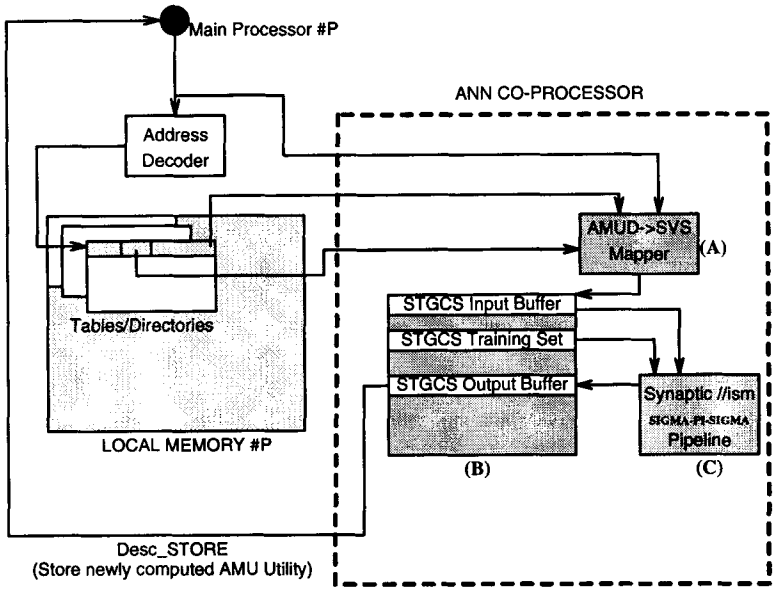

Figure 4: Architectural Support: Coprocessor Overview, **(A)** = AMU descriptor-SVS Mapping Logic, **(B)** = CoProcessor Buffers, **(C)** = GST Processor Unit

10 Simulation Results

The primary goal of the simulations was to evaluate how well the proposed system was able to maintain computational locality and achieve load balance in the face of an initial skewed load distribution. First, the performance of the GST network is described in Section 10.1. Subsequently, Section 10.2 presents the execution performances for only the WaTor application due to space limitations.

10.1 GST Network Performance

In this section, we examine the network sizes evolved and the sampling efficiency of the GST networks in the three applications.

10.1.1 GST Network Sizes

GST networks of widely varying sizes were evolved, underscoring the non-stationarity of the SVS distributions and the utility of an architecturally adaptive ANN. For example, in the GIST network runs of the WaTor simulation, the network sizes varied from 120 neurons (thread 15 in the 24 processor run) to 45 neurons (thread 2 in the 4 processor run), as seen in Figure 5. The network sizes in the GEST network runs of the WaTor simulation (shown in Figure 6) were smaller, ranging between 5 and 20 neurons.

The reason for larger GIST network sizes is the large fixed lengths of the offline training epochs mentioned in Section 7. The large training set sizes induced large GIST networks which tracked the state of the entire local memory including the local working set. However, the working set size estimates used as the offline epoch sizes induced GEST networks which tracked only the emergent working set. The variations in the

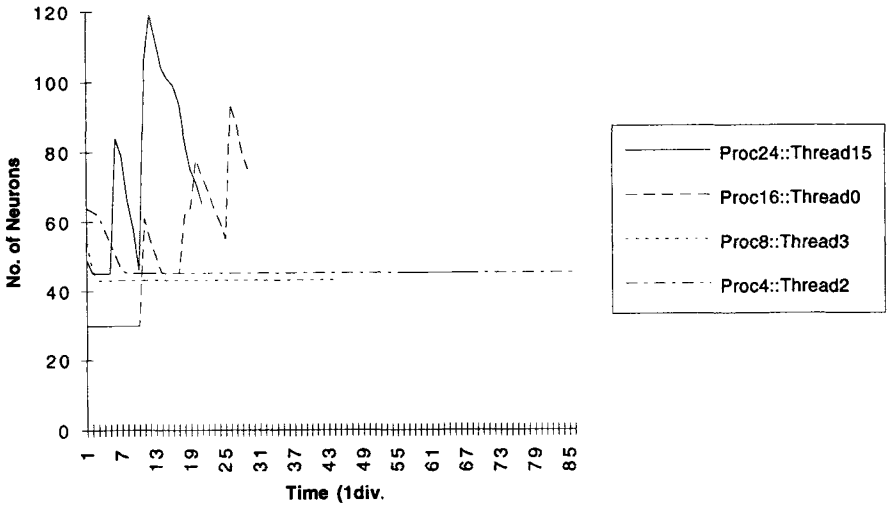

Figure 5: WaTor (15 iteration run) :: GIST Network Sizes
Proc$x$::Thread$y$ $\Rightarrow$ GIST network size for Application Thread#y in an x processor run

Correlation Interval lengths, plotted in Figure 7, indicate the variations in working set sizes (see Section 7). The large initial values always decreased to working-set sizes of $\approx$ 25 AMUs and hence, GEST network sizes were correspondingly smaller than the GIST networks.

The GEST network runs show that adaptive Correlation Intervals successfully track emergent *active regions* of the non-stationary SVS distribution (i.e. active working sets within local memory), providing an alternative to modeling the entire SVS distribution (i.e. entire local memory state).

Interestingly, the GIST network size variations are a good indicator of the locality characteristic variations in the WaTor application. Further, the relatively smaller variations in the network sizes in the GEST simulation indicate smaller working sets. In this sense, the network sizes and their variations indicate how well the GIST network tracks the the overall locality flux inherent in the applications' SVS distributions:

- Static locality characteristics and static dataset sizes in the Unstructured Mesh application produced negligible fluctuations in the corresponding GIST network sizes, as indicated in Figure 8.
- Slowly varying locality with static dataset sizes in the Barnes Hut application initially induced large GIST networks. Subsequent slow changes in locality are tracked via online weight adaptations and the GIST network shrinks to map only the working set, as shown in Figure 9.
- In WaTor, dramatic locality fluxes with slowly varying dataset sizes induced greater GIST network size variations than in other applications. This is evident in the multiple peaks in the 24 processor and 16

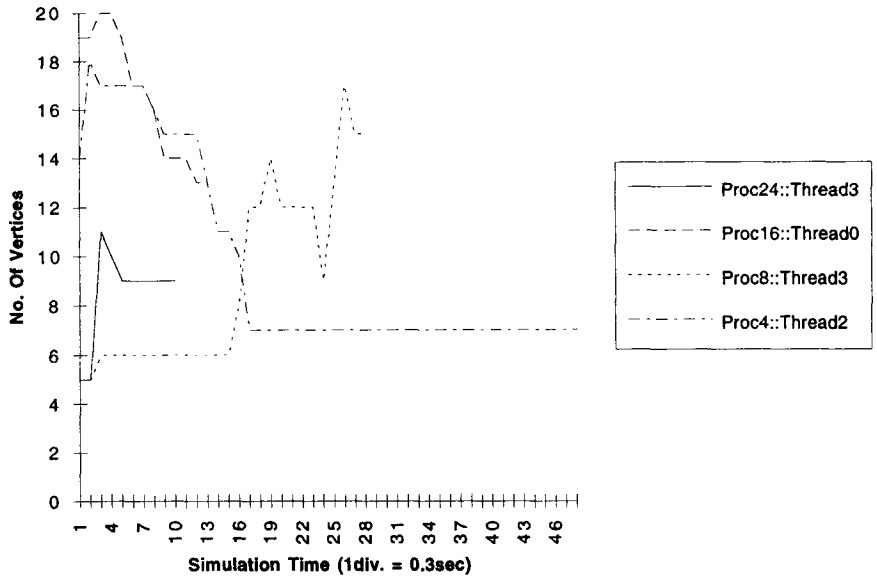

Figure 6: WaTor (10 iteration run) :: GEST Network Sizes
Proc$x$::**Thread**y $\Rightarrow$ GEST network sizes for Application Thread#y in an x processor run

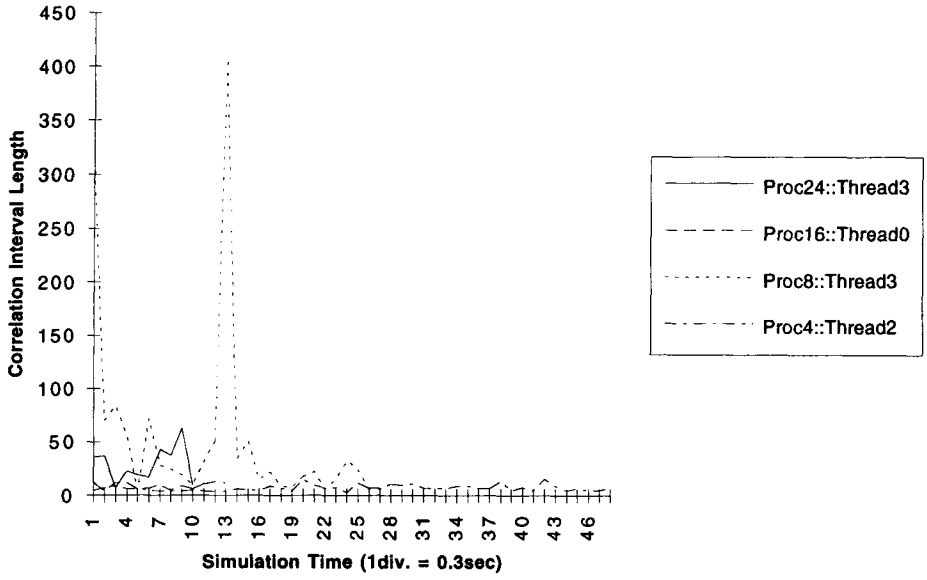

Figure 7: WaTor (10 iteration run) :: GEST network Correlation Intervals
Proc$x$::**Thread**y $\Rightarrow$ GEST Correlation Interval variations for Application Thread#y in an x processor run

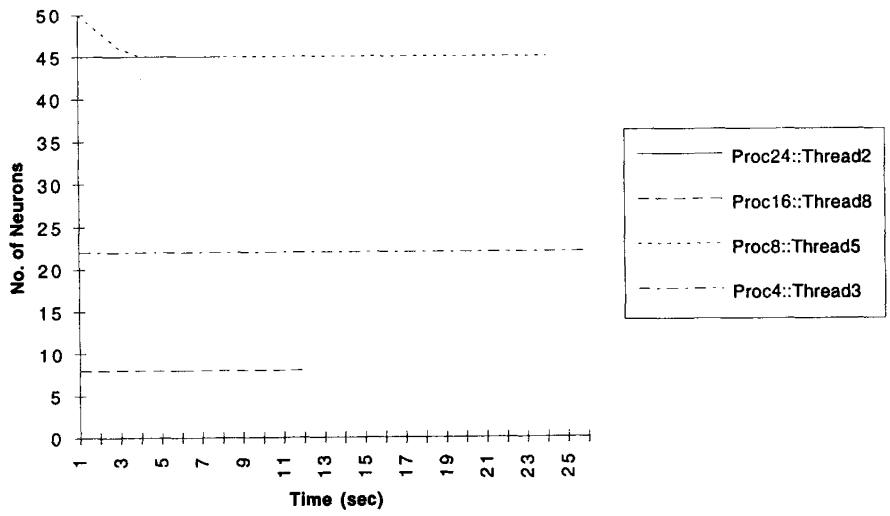

Figure 8: Unstructured Mesh :: GIST Network Sizes
Proc$x$**::Thread**y $\Rightarrow$ GIST network sizes for Application Thread#y in an x processor run

processor GIST runs as opposed to the single peak in the Barnes Hut runs.

10.1.2 GST Sampling Efficiency

Sampling efficiency is consistently high in the GIST runs for all three applications, ranging between **90%** to **100%**. This indicates that the training phases, during which the references from the processor were not processed, are of relatively short durations. This is seen in the sampling efficiency plot of the GIST network in the Barnes-Hut simulation in Figure 10, for example. Due to constraints on space, the sampling efficiencies of the WaTor and Unstructured Mesh runs using the GIST network are not presented. The graph for the WaTor simulation is similar to the Barnes-Hut simulation, while the static locality characteristics of the Unstructured Mesh application induce little variations in the GIST network sampling efficiency.

The large amount of computation per reference in the Barnes-Hut application produces larger inter-reference times which allows the network to achieve 100% sampling efficiencies (Figure 10). This curve evinces high-frequency variations caused by frequent but short stints of network retraining. This effect is due to the locality flux which induced variations in the emergent SVS distributions over time. For example, at the *14 sec.* point in the 32 processor run (Figure 10), the network size (Figure 9) increases from about 130 neurons to about 200 neurons, indicating network retraining activity, since neuron additions occur only during retraining.

It is interesting to note that the GEST network sampling efficiencies are always consistently high in

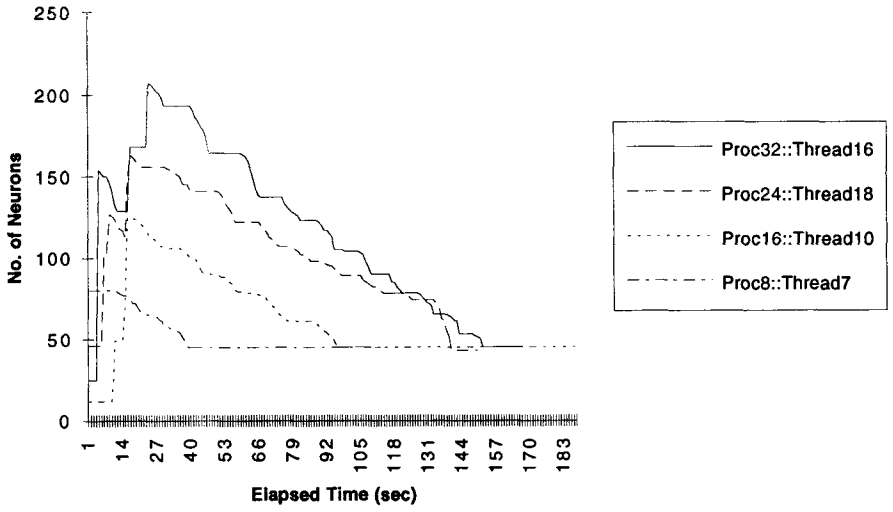

Figure 9: Barnes-Hut :: GIST Network Sizes
Proc$x$::Thread$y$ $\Rightarrow$ GIST network sizes for Application Thread#y in an x processor run

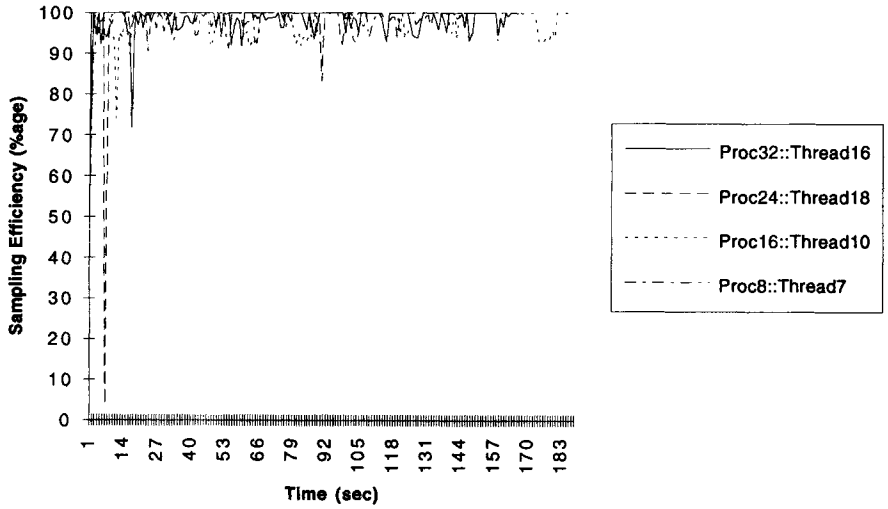

Figure 10: Barnes Hut :: GIST Sampling Efficiency
Proc$x$::Thread$y$ $\Rightarrow$ GIST network for Application Thread#y in an x processor run

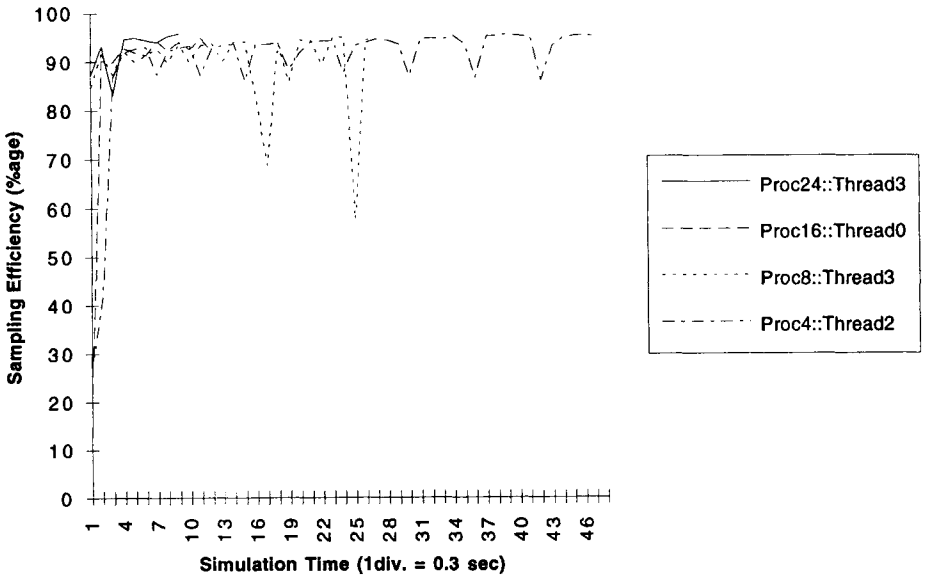

Figure 11: WaTor (10 iteration run) :: GEST Network Sampling Efficiency
Proc$x$**::Thread**y $\Rightarrow$ GEST network for Application Thread#y in an x processor run

the WaTor simulation (Figure 11); further, they do not suffer the high frequency variations evident in the Barnes-Hut simulation runs using the GIST network.

10.2 WaTor Application Performance

This section presents the WaTor execution performance in terms of the locality achieved and maintained as well as the overall execution timings. The results for the Barnes-Hut and the Unstructured Mesh application performances are not presented here due to the limitations of space.

The Minnow update loop execution timings are considered here since these were responsible for 75% to 90% of the total execution time. This is evident from Table 2 presenting the execution timings for the runs, which are seen to scale well in performance. A plot of the locality measure (defined in Section 9) for the GIST network runs (presented in Figure 12) shows increasingly better execution locality. In each of the GIST runs, the inter-task distances decrease from an initial high of 75 to level off at about 40 as shown in Figure 12. This indicates that the local partitions achieved increasing contiguity and compactness *in spite of the increasing dataset sizes.*

This is also evident in the locality characteristics achieved in the WaTor runs using the GEST network as illustrated in Figure 13. The larger correlation intervals (evidenced in Figure 7) and subsequent capture of higher-order temporal correlations results in an almost monotonic decrease in the inter-task distances. This follows, as mentioned in Section 5.4, from the long-range locality characteristics extracted in the GEST

	Minnow Loop		Shark Loop	
Run Size	GIST Run	GEST Run	GIST Run	GEST Run
4 Processors	23.695952 sec	21.768551 sec	4.679540 sec	3.528160 sec
8 Processors	13.460341 sec	13.248189 sec	2.312860 sec	2.040482 sec
16 Processors	8.473986 sec	6.6604046 sec	1.409917 sec	1.273765 sec
24 Processors	6.034877 sec	4.6708921 sec	1.140522 sec	1.208808 sec

Table 2: WaTor Execution Timings (using GIST and GEST networks)

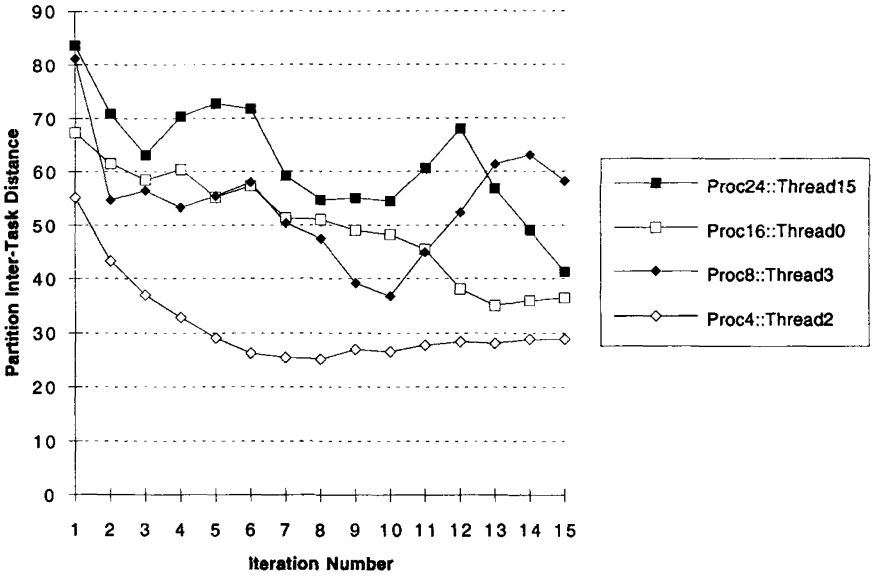

Figure 12: WaTor Locality Measure using GIST networks(15 iteration run) :: Thread Intra-Task Distances **Proc**x::**Thread**y $\Rightarrow$ Locality Characteristics of Thread#y in an x processor run

network, as opposed to the non-monotonic decreases in the GIST run resulting from the use of first-order correlation information containing short-range locality characteristics (noise) as well.

11 Conclusions and Discussions

In this paper we have extended the GCS network into the GIST and GEST networks (collectively referred to as the GST network) for solving spatio-temporal problems. The GST network has been shown to successfully extract spatio-temporal correlation information dynamically from real-time incoming signals. The results of applying the network to the data mapping problem demonstrated the feasibility of hardware implementations. The network sizes that evolved are easily within the grasp of state-of-the-art hardware neural network implementations which routinely deal with network sizes of the order of thousands of neurons.

The main results (in Section 10) show network convergence to stable maps of emergent SVS distributions without undue loss in the Sampling Efficiency, in both the GIST and the GEST networks. This further implies that temporal correlations in the evolution of modes in emergent SVS distributions (local working-

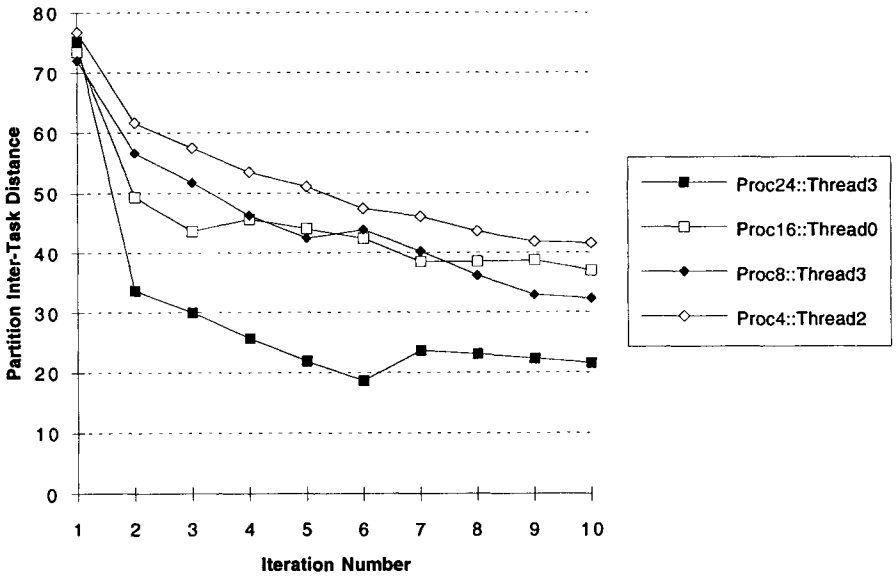

Figure 13: WaTor Locality Measure using GEST networks(10 iteration run) :: Thread Intra-Task Distances **Proc**x::**Thread**y $\Rightarrow$ Locality Characteristics of Thread#y in an x processor run

sets in the WaTor application) are tracked very well by both variants of the network, as shown by the improvements in locality in Figure 12 and Figure 13. The GEST network sampling efficiency is consistently high although lower than that of the GIST network. But the GEST network sampling efficiency does not suffer the high-frequency variations evident in the GIST network. This indicates that the GEST networks are more stable (relatively infrequent and/or short offline retraining stints) and track evolving active regions in the SVS more efficiently (using smaller network sizes).

GIST networks needed large offline mappings of the entire SVS distribution which were refined online to track the emergent active regions (i.e. evolving data-clusters) of the SVS. In contrast, the adaptive correlation intervals of the GEST networks successfully tracked the active regions without needing offline estimates of the entire SVS distribution.

As expected, the higher-order dependencies captured by the GEST network prove to be better indicators of the long-range evolution of the SVS distribution than the first-order dependencies used in the GIST network. This is manifest in the almost monotonic improvements in locality in the GEST network runs (Figure 13). However, the price paid takes the form of lower sampling efficiency in comparison with the GIST network.

In data mapping problems, load-balancing and locality maintainence are often seen as top-down problems in the sense that the **user** maps domain knowledge about domain inspired data abstractions into the corresponding execution locality characteristics about architectural level data abstractions. However, the

current efforts shows excellent prospects for a bottom-up approach wherein locality/load-balance are seen as ultimately to do with architectural-level abstractions and successfully extracts locality and load information at this level thereby easing programming burden on the user.

In summary, the results demonstrate the successful online use of the GST network for extracting spatio-temporal correlation information among emergent features of previously unknown distributions. Further, execution driven simulations of the GIST and GEST networks on the challenging data mapping problem indicate the feasibility of their hardware implementations. Between the two networks, the GEST network trades a lower sampling efficiency (in comparison with the GIST network) for increased network stability, better dependency predictions, enhanced tracking abilities and smaller network sizes evolved.

References

[1] D. L. Snyder and P. M. Fishman, "How to Track a Swarm of Fireflies by Observing Their Flashes," *IEEE Transactions on Information Theory*, pp. 692–695, November 1975.

[2] R. Ponnusamy, *Run-Time Support and Compilation Methods for Irregular Computations on Distributed Memory Parallel Machines.* Phd thesis, Syracuse University, May 1994.

[3] N. Mansour, *Physical Optimization Algorithms for Mapping Data to Distributed-Memory Multiprocessors.* PhD thesis, Syracuse University, August 1992.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability.* Freeman, 1979.

[5] T. Ibrakai and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches.* Cambridge, Mass.: MIT Press, 1988.

[6] B. M. Fritzke, "Growing Cell Structures - A Self-organizing Network for Unsupervised and Supervised Learning," Tech. Rep. TR-93-026, International Computer Science Institute, Berkely, May 1993.

[7] B. Fritzke, "Kohonen Feature Maps and Growing Cell Structures - A Performance Comparison," *Advances in Neural Information Processing Systems 5*, 1993.

[8] T. Kohonen, *Self-Organization and Associative Memory.* Heidelberg, Germany: Springer-Verlag, third ed., 1989.

[9] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, Massachusetts: Addison-Wesley, 1981.

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York, NY: Academic Press, 1972.

[11] W. S. Searle, "Neural Networks and Statistical Models," in *Proc. of the 19th Annual SAS Users Group International Conference*, April 1994.

[12] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," in *Advances in Neural Information Processing Systems* (D. S. Touretzky, ed.), vol. II, pp. 524–532, Morgan Kaufmann, 1990.

[13] J. Sietsma and R. J. F. Dow, "Neural Net Pruning - Why and How," in *Proc. of International Conference on Neural Networks*, pp. 325–333, 1988.

[14] S. Jockusch and H. Ritter, "Self-Organizing Maps: Local Comptetition and Evolutionary Optimization," *Neural Networks*, vol. 7, no. 8, pp. 1229–1239, 1994.

[15] E. Wan, "Time Series Prediction by Using a Connectionist Network with Internal Delay Lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past* (A. S. Weigend and N. A. Gershenfeld, eds.), pp. 195–217, Reading, Massachusetts: Addison-Wesley, 1994.

[16] D. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1. MIT Press, 1986.

[17] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," in *Neural Computation*, vol. 1, pp. 270–280, 1989.

[18] M. Caudill, "Neural Networks Primer, Part VII," *AI Expert*, pp. 55–58, May 1989.

[19] R. P. Lippman, "An introduction to computing with neural nets," *IEEE ASSP magazine*, pp. 4–22, April 1987.

[20] P. K. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations*, ch. 5, pp. 110–111. Pergamon Press, 1990.

[21] B. Kosko, "Differential Hebbian Learning," in *Procs. American Institute of Physics: Neural Networks for Computing*, pp. 277–282, April 1986.

[22] A. H. Klopf, "Drive-Reinforcement Learning: A Real-Time Learning Mechanism for Unsupervised Learning," in *Proc. International Conference on Neural Networks*, pp. II–441:445, 1987.

[23] J. Wang and G. Lin, "On the Learning Dynamics of Spatiotemporal Neural Networks," in *Proc. International Conference on Neural Networks*, pp. V–3201:3206, 1994.

[24] K. S. Fu, *Syntactic Pattern Recognition and Applications.* Englewood Cliffs, New Jersey: Prentice Hall, 1982.

[25] A. Papoulis, *Probability, Random Variables and Stochastic Processes.* McGraw Hill, second ed., 1984.

[26] P. Stigall, C. Dagli, and C. Sen, "A Neural Network Cache Controller," in *Intelligent Engineering Systems through Artificial Neural Networks, Proc. of ANNIE '91*, pp. 561–566, November 1991.

[27] M. Cena, M. L. Crespo, and R. Gallard, "Transparent Remote Execution in LAHNOS by means of a Neural Network Device," *Operating Systems Review*, pp. 17–28, January 1995.

[28] P. Mehra, *Automated Learning of Load-Balancing Strategies for a Distributed Computer System.* Phd thesis, University of Illinois at Urbana Champaign, December 1992.

[29] P. J. Denning, "Working Sets past and present," *IEEE Transactions on Software Engineering*, vol. SE-6, pp. 64–84, January 1980.

[30] J. P. Singh, W.-D. Weber, and A. Gupta, "SPLASH: Stanford Parallel Applications for Shared Memory," *Computer Architecture News*, vol. 20, pp. 5–44, March 1992.

[31] J. E. Barnes and P. Hut, "A Hierarchical O(NlogN) Force Calculation Algorithm," *Nature*, vol. 324, no. 4, pp. 446–449, 1986.

[32] J. K. Salmon, *Parallel Hierarchical N-Body Methods.* Phd thesis, California Institute of Technology, December 1990.

[33] D. J. Mavriplis, R. Das, R. E. Vermeland, and J. Salz, "Implementation of a Parallel Euler Solver on Shared and Distributed Memory Architectures," in *Proc. SUPERCOMPUTING '92*, pp. 132–141, November 1992.

[34] A. K. Dewdney, "Computer Recreations," *Scientific American*, pp. 14–22, December 1984.

[35] I. Angus, G. Fox, J. Kim, and D. Walker, *Solving Problems on Concurrent Processors*, vol. II. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[36] H. Speckmann, P. Thole, and W. Rosenstiel, "A COprocessor for KOhonen's Selforganizing Map (COKOS)," in *Procs. 1993 International Joint Conference on Neural Networks*, pp. 1951–1954, 1993.

[37] R. H. Saavedra, R. S. Gaines, and M. J. Carlton, "Micro Benchmark Analysis of the KSR1," in *Proc. of SUPERCOMPUTING '93*, pp. 202–213, September 1993.

[38] Kendall Square Research, *KSR/Series Principles of Operation*, March 1994.