EFFICIENT MAPPING OF NEURAL NETWORKS ON MULTICOMPUTERS

Benjamin W. Wah and Lon-Chan Chu

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801
wah%aquinas@uxc.cso.uiuc.edu

ABSTRACT. In this paper, an efficient mapping of multilayer artificial neural networks on multicomputers is formulated and analyzed. The objective is to minimize the completion time of parallel neural-network simulations. This optimization problem is NP-hard. By noting that the computation time is predominant over the communication time in most cases, a simplified algorithm with negligible error is developed and analyzed. Experimental results for mapping on an Intel iPSC/2 computer and a network of Sun computers are shown and are found to be very close to those predicted by analysis.

1. INTRODUCTION

Artificial neural networks (ANNs) show promising potentials in artificial intelligence applications. However, the technologies for implementing them in hardware are not mature; only simple and small-scale ANNs are implemented in VLSI at this time. ANNs are usually studied by simulations on existing computer systems. These simulations require large amount of computational time and can be carried out more efficiently by parallel processing.

We assume in this paper that the ANNs are multilayer feedforward networks. We further assume that the Back-Error Propagation (BEP) learning algorithm is used. The results we develop are applicable to feedback networks since all iterations of feedback are identical and would result in the same mapping. Our results apply to "static" learning algorithms in which the corresponding task graph is not dynamically changing; they do not apply to competitive learning algorithms in which the activities of neurons are dynamically changing. Our results exploit the regular structure of multilayer ANNs with clusters in order to reduce the complexity of the mapping algorithm, although ANNs with an arbitrary interconnection can also be mapped. Processors in the target multicomputers are assumed to have local memory. Due to space limitations, only static mapping algorithms are described in this paper. For static mappings to be meaningful, the target computer is assumed to run in a single-user environment.

We formulate the optimal static mapping of learning the weights of an ANN on a multicomputer as an integer programming problem. The objective is a function of computation and communication times. Constraints on feasibility, configuration, resource, and dependency are considered. The mapping problem is NP hard. To reduce its complexity, the multicomputer is decomposed into disjoint partitions of processors according to the ratio of communication to computation times. Each partition represents a conceptual processing resource. Experiments on multicomputers are carried out to validate the correctness of the mappings predicted by analysis.

Related works on this problem include parallel software simulations on multiprocessors, design of generic multicomputers for ANN simulations, and implementations of computers or VLSI chips for ANNs.

H. T. Kung, *et al.*, map layered ANNs to WARP, a linear array of ten processing cells [8]. They propose two approaches: network partitioning and data partitioning. Their partitioning scheme is optimal for a ring of processors and a multilayer ANN. However, it may be suboptimal when ANN simulations are mapped to an arbitrary network of processors.

Hwang, *et al.*, design generic multicomputers for ANN simulations [4, 5]. They discuss design issues on the processing elements and the communication-bandwidth requirements, and propose several guidelines for designing generic multicomputers for ANN simulations. Their method is based on datagram routing, which may result in unpredictable network congestion. The performance of their scheme is also dependent on the system-supported routing algorithm.

The weight-update process in a multilayer ANN can be considered as a sequence of matrix-vector multiplications. Using this approach, S. Y. Kung, *et al.*, transform ANN learning to recursive matrix operations, then to data dependency graph, and finally to a linear systolic array with a fast interconnection network [6, 7]. Active neurons in each layer are evenly distributed to the processing cells of the systolic array, and full resource utilization is obtained in many cases. In fact, we show in this paper that their scheme is optimal when the ANN is layered and the interconnection network is fast. However, they did not consider the case when the links and processors are nonuniform. In this case, active neurons do not have to be evenly distributed to all processing cells.

A number of other multiprocessor simulations have been reported. Researchers at Edinburgh simulate ANN learning on a transputer-based Computing Surface with 42 processors [3]. Researchers at Rochester use a 128-node BBN Butterfly multicomputer for simulating ANNs [2].

This paper addresses some of the deficiencies found in previous studies, which either assume a tightly coupled system, such as a systolic array with homogeneous processing cells, or present a heuristic

This research was partially supported by National Science Foundation Grant MIP 88-10584, National Aeronautics and Space Administration Contract NCC 2-481, and Joint Services Electronics Program Contract N00014-90-J-1270.

1990 International Conference on Parallel Processing, St. Charles, IL.

mapping algorithm for a set of heterogeneous processors. The mapping problem can be formulated using integer programming, and an optimal mapping can be found for a network of heterogeneous computers. By recognizing that neurons in a multilayer ANN are clustered and that their communication patterns are uniform within a cluster, we show efficient approximation schemes for mapping clusters of neurons to partitions of processors. In this paper, heuristic methods for partitioning processors are assumed. Due to space limitation, we do not show the algorithm which decomposes processors into optimal partitions. In the latter case, the complexity of finding the optimal mapping can be reduced by exploiting isomorphism of processors. Our results are important for designing special-purpose computers for ANN simulations and for determining the suitability of an existing multicomputer for ANN applications.

The paper is organized into six sections. Section 2 defines the models of the ANN and the multicomputer. Section 3 examines the mapping problem. Section 4 discusses the solution strategies and techniques. Section 5 describes our experimental results, and conclusions are drawn in Section 6.

2. MODELS

An ANN is characterized by a set of neurons, pattern of interconnection, propagation rule, activation rule, output function, and learning rule.

A neuron is the basic processing unit, which is characterized by its state, an activation function, and an output function. The activation function transforms the input signals associated with the corresponding weights and its state value to a new state value. The output function transforms the state value to an output signal.

The pattern of interconnection determines the dependence of signal flows in an ANN. The propagation rule specifies the composition of the *net input* of a neuron. The activation rule specifies the transformation into a new state from the net input of a neuron, its global signal, and its current state. The output function transforms the state of a neuron into an output signal. The learning rule specifies the mechanism for modifying the weights. An ANN learns by incremental modifications of its connection weights.

Neurons in a layered ANN can be *clustered* in such a way that if one cluster is connected to another cluster, then all neurons in the first cluster are connected to all neurons in the second cluster. A special case is a multilayer ANN with one cluster in each layer. Note that all neurons in a cluster are homogeneous and receive identical inputs. As a result, it is only necessary to know the number of neurons in a cluster that are mapped to a processor but is not important to know the mapping of each individual neuron in the cluster.

The ANN studied in this paper operates in two phases: the *feed-forward phase* and the *error-propagation phase*. In the feed-forward phase, the ANN receives input signals and produces output signals. In the error-propagation phase, the ANN receives teaching inputs, if they are provided, and modifies the weights according to the learning (or error-propagation) rule. By alternating between these two phases, an ANN adapts itself to the training inputs.

The multicomputer assumed in this paper consists of a set of processors with local memory, a set of communication links connecting the set of processors, and a queuing discipline. For static mappings studied in this paper, the multicomputer is assumed to run in a single-user mode. The queuing discipline is scheduled by the mapping algorithm. Due to space limitations, dynamic mapping algorithms are not presented here.

A processor consists of a CPU, its associated local memory, and a set of communication ports through which this processor communicates with other processors. A processor may also have an I/O facility for communicating with the external environment. The computation power of a processor is characterized by the execution time per unit computation, which may include CPU and memory-access activities. The size of local memory in each processor is a constraint in our mapping scheme.

3. THE MAPPING PROBLEM

The mapping problem of an ANN on a multicomputer entails the search for an optimal mapping of the neurons to the processors so that the completion time of the ANN simulations is minimum and that constraints on feasibility, dependency, resource, and configuration are satisfied.

The mapping involves schemes for assigning neurons to processors and schemes for routing data in the interconnection network. The assignment of neurons must meet constraints on integrity, feasibility, and resource. The routing scheme must meet constraints on dependency, feasibility, resource, and configuration. Note that the routing of data can only be determined after neurons have been assigned to processors.

A mapping scheme Φ is defined formally as a 4-ary tuple $\Phi(M_N, M_M, \Phi_A, \Phi_R)$, where M_N is the ANN model, M_M is the multicomputer model, Φ_A is the assignment scheme, and Φ_R is the routing scheme. Φ_A and Φ_R are mutually related. All feasible mapping schemes constitute the mapping space $\Omega(M_N, M_M)$. The optimal mapping problem entails finding a mapping $\Phi^* \in \Omega(M_N, M_M)$ such that

$$T_{EXEC}(\Phi^*) = \min_{\Phi \in \Omega} T_{EXEC}(\Phi), \qquad (3.1)$$

where T_{EXEC} is the completion time for the given mapping.

The assignment scheme Φ_A can be represented as an integer-valued *assignment matrix* **A** of size N-by-P, where N is the number of neuron clusters and P is the number of processors. Component a_{ij} of matrix **A** indicates the number of neurons in neuron cluster i assigned to processor j. When $a_{ij}>0$, processor j is called a *home processor* of neuron cluster i.

The routing scheme Φ_R can be represented as a *routing vector* **R** of set Ξ of 4-ary tuple $<p,f,t_s,t_u>$, where p denotes the processor that transmits frames via a specified link, f denotes the data frame transferred via this link, t_s denotes the start time for transmission (*time stamp*), and t_u denotes the time period for utilizing this link. The cardinality of the routing vector **R** is the number of links. The ith component of vector **R** is denoted by Ξ_i. A member ξ_{ij} of Ξ_i defines link i to be allocated to processor $p_{\xi_{ij}}$ starting at time $t_{s_{\xi_{ij}}}$ for a period of $t_{u_{\xi_{ij}}}$.

The ANN operations can be decomposed into *segments* according to the dependency constraints of the ANN. A segment represents the computations at a processor for the neurons in a cluster assigned to this processor. Note that a cluster of neurons may be decomposed into multiple segments of computations. Each segment can start only after all segments in the predecessor clusters have finished. A segment of computation is defined by its *entry point* and its *exit point*. An entry point of a segment at a particular processor is the time when this processor receives the necessary inputs from processors with predecessor segments. An exit point of a segment at a particular processor is the time when this processor completes the computations in this segment and starts sending its results to processors having successor segments. Between the exit point of one segment and the entry point of the next segment that follows, a processor is either sending results to other processors or waiting for data from other processors. Note that this model can accommodate systems with either overlap or no overlap between computations and communications. If overlap is allowed, then computation of a segment may start once the necessary inputs have been received. To allow the model to be tractable, we assume that the transmission of results to other processors begins immediately after the computation of a

segment is completed.

Formally, the jth segment at processor i is denoted by $\sigma_{ij}=<\sigma_{1ij},\sigma_{2ij}>$, where σ_{1ij} is the entry time and σ_{2ij} is the exit time. To facilitate finding a routing scheme, each processor i is associated with a Boolean data matrix D_i of size N-by-P. This matrix keeps information on the availability of the output signals of neurons such that $(D_i)_{xy}=1$ if the output signals of neurons in neuron cluster x assigned to processor y are available at processor i; otherwise, it is 0.

Using the above notation, the mapping problem can be formulated by nonlinear integer programming, and the optimal mapping can be found by a branch-and-bound algorithm. A node in the search tree represents either a possible assignment of a certain neuron cluster or a possible routing between two clusters. One important feature of this representation is that the search branches on neuron clusters instead of individual neurons.

The search iteratively chooses an unassinged cluster, decomposes it into segments, and maps the segments to processors. The routing between neurons in this cluster and neurons in clusters mapped earlier are then determined. The routing is done under dependency, resource, and configuration constraints. After each selection, updates are made on the data matrix $\mathbf{D}_i$, the entry time $t_{\sigma_{1ij}}$, and the exit time $t_{\sigma_{2ij}}$. These selections continue until all processors receive all necessary frames for their computations. By backtracking on clusters assigned earlier, the branch-and-bound algorithm can determine the optimal mapping of clusters to processors. Due to space limitation, the integer programming formulation and branch-and-bound algorithm are not shown here [1]. Experimental results using the branch-and-bound algorithms are presented in Section 5.

4. PROBLEM SIMPLIFICATION METHODS

In the ANN simulations considered in this paper, the computation overhead generally dominates the communication overhead, and the number of neurons are generally larger than the number of processors. This dominance is characterized by the ratio of the communication to computation times. Hence, the mapping problem can be greatly simplified with negligible error by first heuristically decomposing the multicomputer into partitions of tightly coupled processors (in which the communication speed is fast), optimally mapping the clusters of neurons to partitions of processors, and heuristically routing the communications within partitions. Note that the optimal assignment and optimal routing are interrelated problems, and that each cannot be solved independently. The scheme we propose here is heuristic because it is based on a heuristic method for partitioning processors in the multicomputer system. We show in this section that negligible error is incurred in mapping clusters of neurons on partitions of processors rather than individual processors. The optimal partitioning of processors will be discussed in a subsequent paper.

The symbols we use in this section are summarized in Table 4.1. These symbols are explained briefly here.

For a given partitioning of processors (the heuristic partitioning method will be discussed later), the optimal mapping and routing of neurons on these partitions can be found by evaluating a branch-and-bound algorithm based on the nonlinear integer programming formulation of the mapping and routing problems. As stated before, these two problems are related and cannot be solved independently. t^*_{iQ}, the computation and intra-partition times for cluster i in the optimal case, satisfies the following equation.

$$t^*_{iQ} = t^*_{c_{iQ}} + t^*_{ri\mathcal{Q}} \tag{4.1}$$

Figure 4.1 illustrates the neural network, the decomposition of clusters of neurons into partitions of processors, the mapping of clusters within a partition, and the mapping of clusters on the entire mul-

Table 4.1a. Summary of symbols used in mapping neuron clusters to partitions of processors, based on given heuristic partitioning and optimal inter- and intra-partition routing schemes

Symbol	Meaning
n_{iQ}	number of neurons in cluster i assigned to partition Q of processors
t^*_{iQ}	for the optimal mapping and optimal routing, time interval during which one or more processors are performing computations or communications for cluster i on partition Q of processors and are not overlapped with computations of the next cluster that follows
$t^*_{c_{iQ}}$	for the optimal mapping and optimal routing, time interval during which one or more processors are performing computations for cluster i on partition Q of processors
$t^*_{ri\mathcal{Q}}$	for the optimal mapping and optimal routing, time interval during which all processors are performing intra-partition communications for cluster i on partition Q of processors and are not overlapped with computations in this cluster or the next cluster that follows
$t^*_{R_{iQ}}$	for the optimal mapping and optimal routing, same as $t^*_{r_{iQ}}$ except that inter-partition communications are concerned

Table 4.1b Summary of symbols used in mapping neuron clusters to partitions of processors, based on given heuristic partitioning, heuristic intra-partition routing schemes, and optimal inter-partition routing schemes

Symbol	Meaning
t_{iQ}	for the optimal mapping of given subset of neurons in cluster i to partition Q (without considering intra-partition communication delay) and heuristic intra-partition routing, time interval during which one or more processors are performing computations or communications for cluster i on partition Q of processors and are not overlapped with computations in the next cluster that follows
$t_{c_{iQ}}$	ignoring the intra-partition communication overhead and assuming optimal mapping of given subset of neurons in cluster i to partition Q, time interval during which one or more processors are performing computations for cluster i on partition Q of processors
$t_{ri\mathcal{Q}}$	for heuristic intra-partition routing, time interval during which all processors are performing intra-partition communications for cluster i on partition Q of processors and are not overlapped with computations in this cluster or the next cluster that follows
$t_{R_{iQ}}$	for heuristic intra-partition routing, same as $t_{ri\mathcal{Q}}$ except that inter-partition communication overheads are concerned
$t_{comp_{iQ}}$	$= t_{c_{iQ}} \times \lvert Q \rvert / n_{iQ}$
$t_{comm_{iQ}}$	$= t_{ri\mathcal{Q}} \times \lvert Q \rvert / n_{iQ}$
r_{iQ}	$= t_{ri\mathcal{Q}} / t_{c_{iQ}} = t_{comm_{iQ}} / t_{comp_{iQ}}$

ticomputer. In Figure 4.1b, the three processors represent one partition Q. The three blocks on the left represent the three segments for cluster 1, which are processed concurrently by the processors in Q. Note that $t_{c_{4,1}}$ includes all times during which one or more processors are performing computations for cluster 1, and that $t_{r_{4,1}}$ represents the unoverlapped intra-partition communication times between computations in cluster 2 and cluster 4. In case that overlaps between communications and computations are allowed, $(t_{r_{4,1}} + t_{R_{4,1}})$ represents the minimal interval between the time when the computations of the last segment in cluster 2 is completed and the time when the first computation in one of the segments of cluster 4 can begin. Figure 4.1c shows the timing diagrams for simulating the five clusters of neurons in two partitions of processors.

Similarly, the definition of t_{iQ} satisfies the following equation.

$$t_{iQ} = t_{c_{iQ}} + t_{r_{iQ}} = \frac{n_{iQ}}{|Q|} \times \left(t_{comp_{iQ}} + t_{comm_{iQ}} \right) \qquad (4.2)$$

In this case, the neurons in a cluster are first allocated by ignoring their communication requirements. It is obvious that an even distribution of the neurons according to the computation power of processors in partition Q of processors will result in the minimal completion time $t_{c_{iQ}}$ (a more general result will be proved in Theorem 4.2). $t_{c_{iQ}}$, the corresponding computation overhead incurred for cluster i is, therefore minimum.

$t_{r_{iQ}}$, the intra-partition communication overheads in computing t_{iQ} is computed using a heuristic routing scheme. For simplicity, it is assumed that each processor broadcasts its results according to a minimum spanning tree, and that broadcasts of different processors are done sequentially. As a result, there is never any congestion involved in this communication scheme. It is, therefore, simple to compute $t_{r_{iQ}}$, the interval between the time when the last inter-partition communication in cluster i is completed and the time when the first computation in cluster i begins. Note that $t_{r_{iQ}}$ represents a worst-case communication delay.

Another observation about the definitions in Table 4.1 is that $t_{comp_{iQ}}$ is the per-neuron average computation time for cluster i, and that $t_{comm_{iQ}}$ is the per-neuron average communication time for cluster i (based on a heuristic routing scheme). Since $t_{c_{iQ}}$ is a lower-bound estimate and $t_{r_{iQ}}$ is an upper-bound estimate, r_{iQ} consequently represents a *worst-case* communication-to-computation overhead ratio that can be experienced in partition Q for processing cluster i.

The last observation is that both t^*_{iQ} and t_{iQ} include the execution times in the feed-forward and error-propagation phases.

The following lemma and theorem show the upper bound on the error due to a heuristic routing scheme for a given partitioning of processors.

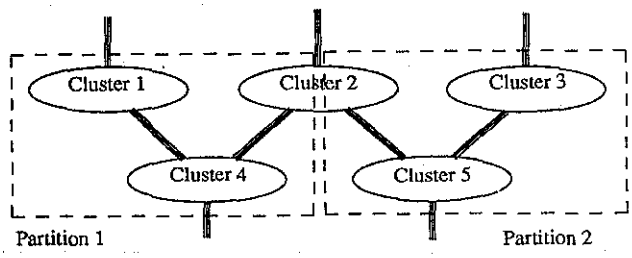
Lemma 4.1. The difference between t_{iQ} and t^*_{iQ} is bounded from above by t^*_{iQ} times the comm/comp ratio r_{iQ}. That is,

$$t_{iQ} \le \left(1 + \frac{t_{comm_{iQ}}}{t_{comp_{iQ}}} \right) t^*_{iQ} \qquad (4.3)$$

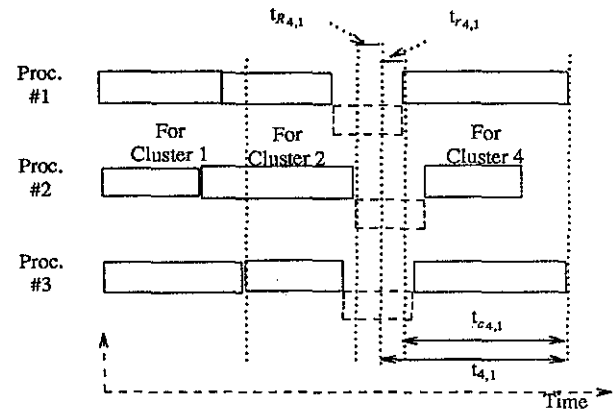
Proof. Since t^*_{iQ} is the optimal execution time, the following relation holds

$$t_{c_{iQ}} = \frac{n_{iQ}\, t_{comp_{iQ}}}{|Q|} \le t^*_{iQ} \le t_{iQ} \qquad (4.4)$$

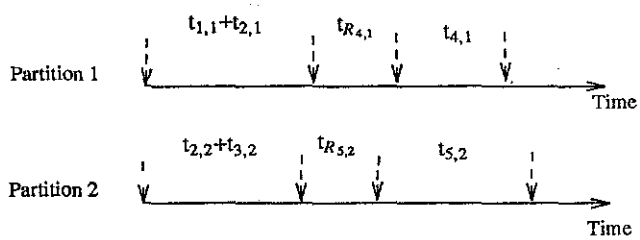
The optimal assignment without considering communication overheads will result in a shorter completion time than a mapping in which the communication overheads are included. In the latter case, the assignment is functionally constrained by the routing. Therefore, we have



(a) Mapping of 5 clusters on 2 partitions of processors.



(b) Timing diagram showing mapping within Partition 1 ($|Q|=1$).



(c) Timing diagram showing overall mapping

Figure 4.1. An example illustrating the symbols in Table 4.1

$$t_{c_{iQ}} \le t^*_{c_{iQ}} = t^*_{iQ} - t^*_{r_{iQ}} \qquad (4.5)$$

Simple algebraic manipulations on Eq's (4.4) and (4.5) results in the following inequality, which proves the lemma.

$$\begin{aligned} t_{iQ} &\le t^*_{iQ} - t^*_{r_{iQ}} + t_{r_{iQ}} \\ &= t^*_{iQ} \left(1 + \frac{n_{iQ}\, t_{comm_{iQ}} / |Q| - t^*_{r_{iQ}}}{t^*_{iQ}} \right) \\ &\le t^*_{iQ} \left(1 + \frac{n_{iQ}\, t_{comm_{iQ}} / |Q| - t^*_{r_{iQ}}}{n_{iQ}\, t_{comp_{iQ}} / |Q|} \right) \\ &\le t^*_{iQ} \left(1 + \frac{t_{comm_{iQ}}}{t_{comp_{iQ}}} \right) \quad \square \end{aligned} \qquad (4.6)$$

To study the effect of mapping neurons within the given partitions, the following heuristic mapping scheme is applied.

Mapping Heuristic 4.1. Neurons within a cluster can be mapped by a branch-and-bound algorithm to one or more partitions of processors with the following assumptions: (a) partitions of processors are known and do not change, (b) routing across partitions is optimal (with time $t_{R_{iQ}}$), and (c) routing within partitions is suboptimal (with time $t_{r_{iQ}}$).

The above heuristic results in a suboptimal mapping because we do not consider the case in which different partitions of processors are used. The following theorem generalizes the concept of Lemma 4.1 to the whole multicomputer system and ANN.

Theorem 4.1. Consider a multilayer ANN with L layers and a multicomputer with K disjoint partitions of processors. Assume that every neuron cluster i in every partition of processors Q has comm/comp ratio $r_{iQ} = t_{comm_{iQ}}/t_{comp_{iQ}}$ no greater than a pre-defined value ε. Let $C_{\Phi_1^*}$ and C_{Φ_1} be the completion times based on Mapping Heuristic 4.1 with respect to optimal and heuristic intra-partition routing. Then $C_{\Phi_1} \le (1+\varepsilon) \times C_{\Phi_1^*}$.

Proof. Assume the symbols defined in Table 4.1, and let N_l be the set of neuron clusters in layer l. The time in each processor allocated to cluster i can be computed by the summation of t^*_{iQ} and $t^*_{R_{iQ}}$, for optimal intra-partition routing, and by the summation of t_{iQ} and $t_{R_{iQ}}$, for heuristic intra-partition routing. If a cluster is not allocated to a partition, then its time is zero. The completion time $C_{\Phi_1^*}$ of mapping Φ_1^* can be expressed by

$$C_{\Phi_1^*} = \max_{Q} \sum_{l=0}^{L-1} \sum_{i \in N_l} \left(t^*_{R_{iQ}} + t^*_{iQ} \right) \tag{4.7}$$

The completion time C_{Φ_1} of mapping Φ_1 is

$$C_{\Phi_1} = \max_{Q} \sum_{l=0}^{L-1} \sum_{i \in N_l} \left(t_{R_{iQ}} + t_{iQ} \right) \tag{4.8}$$

According to Lemma 4.1, $t_{iQ} \le (1+\varepsilon) t^*_{iQ}$ holds. Simple algebraic manipulations show that

$$\begin{aligned} C_{\Phi_1} &= \max_{Q} \sum_{l=0}^{L-1} \sum_{i \in N_l} \left(t_{R_{iQ}} + t_{iQ} \right) \\ &\le \max_{Q} \sum_{l=0}^{L-1} \sum_{i \in N_l} \left(t^*_{R_{iQ}} + (1+\varepsilon)\, t^*_{iQ} \right) \\ &\le \max_{Q} \sum_{l=0}^{L-1} \sum_{i \in N_l} \left(1+\varepsilon \right) \left(t^*_{R_{iQ}} + t^*_{iQ} \right) \\ &\le (1+\varepsilon)\, C_{\Phi_1^*} \qquad \square \end{aligned} \tag{4.9}$$

According to Theorem 4.1, given an error bound ε of the comm/comp ratio, the multicomputer can be partitioned into several disjoint groups such that the comm/comp ratio of each group for simulating part of a given neuron cluster is less than the threshold ε. Hence, the optimal mapping on the partitioned multicomputer with a heuristic routing scheme within each group will have a completion time no greater than $(1+\varepsilon)$ times the completion time of the optimal mapping on the non-partitioned multicomputer. The maximum of all comm/comp ratios of the partitioned multicomputer is called the *error degree*. A small error degree will result when the number of neurons in all neuron clusters are large or when the partitions are small. The following theorem shows that a proportional distribution of neurons within a partition according to the computational power of processors within the partition is optimal.

Theorem 4.2. Assume that n_{iQ} neurons in neuron cluster i are assigned to a partition of processors Q. The optimal assignment on Q can be obtained by distributing the n_{iQ} neurons evenly according to the processing power of processors. Processor j completes at approximately $x_{ij}\tau_j + t_j$, where x_{ij} is the number of neurons in neuron cluster i assigned to processor j, τ_j and t_j are, respectively, the execution time of unit computation and the amount of time that processor j is not available for ANN simulations.

Proof. Since the computation time dominates the communication time in this partition, only the computation time has to be considered in the proof. Let X_{iQ} be the i'th possible mapping of cluster i on Q. The optimal execution time can be written as

$$t_{iQ} = \min_{\{X_{iQ}\}} \max_{j \in Q} \{ x_{ij}\, \tau_j + t_j \} \tag{4.10}$$

Let z_{ij} $(= x_{ij}\tau_j + t_j)$ be the computation time of processor j for cluster i and $C_{<X_{iQ}>}$ $(= \max_{j \in Q} \{x_{ij}\tau_j + t_j\})$ be the completion time of mapping X_{iQ}. Assume the assignment as stated in the theorem such that $z_{ij} = C_{<X^*_{iQ}>}$ for every j, where $C_{<X^*_{iQ}>}$ is the completion time of the optimal mapping in Q. Since $\sum_{j \in Q} x_{ij} = n_{iQ}$, $C_{<X^*_{iQ}>}$ can be easily derived as

$$C_{<X^*_{iQ}>} = \frac{n_{iQ} + \sum_{j \in Q} t_j/\tau_j}{|Q|/\tau_j}, \tag{4.11}$$

where $|Q|$ is the cardinality of Q. By assuming another assignment X'_{iQ} such that $C_{<X'_{iQ}>} = \max_{j \in Q} \{x_{ij}'\tau_j + t_j\} \le C_{<X^*_{iQ}>}$, then for every j, assignment x'_{ij} satisfies an inequality $x'_{ij}\tau_j \le C_{<X'_{iQ}>} - t_j$. By summing all x'_{ij}, we have

$$n_{iQ} = \sum_{j \in Q} x'_{ij} = \sum_{j \in Q} \frac{C_{<X'_{iQ}>} - t_j}{\tau_j} < \sum_{j \in Q} \frac{C_{<X^*_{iQ}>} - t_j}{\tau_j} = n_{iQ}$$

A contradiction! Consequently, $C_{<X'_{iQ}>} \ge C_{<X^*_{iQ}>}$ must hold; that is, the optimal execution time is $C_{<X^*_{iQ}>}$. $\square$

According to Theorem 4.2, x_{ij} can be calculated by using the following equality.

$$x_{ij}\,\tau_j = \left\lfloor \frac{n_{iQ} + \sum_{j \in Q} t_j/\tau_j}{|Q|/\tau_j} - t_j \right\rfloor \tag{4.12}$$

Note that if $t_j = 0$ for every j, then a uniform distribution according to the computation power of processors in Q follows from Theorem 4.2. Also note that if

$$\frac{n_{iQ} + \sum_{j \in Q} t_j/\tau_j}{|Q|/\tau_j} < t_j \tag{4.13}$$

is true, then the most negative x_{ij} can first be set to zero and x_{ik} can be recomputed for every $k \ne j$ in the set of processors Q. This process may have to be repeated several times in the worst case.

Corollary 4.1. In a system with homogeneous processors connected by a fast interconnection network (such as a linear systolic array assumed by S. Y. Kung, *et al* [6,7].), an even distribution of neurons in a cluster to all processing cells results in the minimal completion time of simulations.

Proof. Since the interconnection network is fast, the computation overhead dominates the communication overhead. According to Theorem 4.1, the entire system can be considered as one partition with negligible error in the optimal mapping. Further, according to Theorem 4.2, neurons should be mapped evenly to all processing elements. $\square$

The theorems proved in this section assume that the partitions of the processors are given. Due to space limitation, the search algorithm for finding the optimal partitions is not presented in this paper. A heuristic partitioning algorithm is described here.

Heuristic Partitioning Algorithm 4.2

(1) Select one processor not included in any partition to form a new partition. If all processors have been partitioned, then exit.

(2) For a given partition and a processor not included in any other partition, if r_{iQ} for all processors in this partition (including the newly selected processor) does not exceed the error allowance ε, then include the new processor into this partition. This step is repeated for all partitions already formed and all processors not included in any partition. Go to (1).

The complexity of the mapping problem depends on the number of partitions formed, the interconnection of multicomputers, and the resource parameters. Figure 4.2 shows the execution time on a Sun 3/60 computer for solving the optimal mapping of neurons on various partitions of a 16-processor iPSC/2 computer using a branch-and-bound algorithm. Note that the execution time grows exponentially with respect to the number of partitions, since a branch-and-bound algorithm is used to decompose clusters to partitions.

5. EXPERIMENTAL RESULTS

In this section, we show experimental results on two multicomputers: a 16-node Intel iPSC/2 Hypercube computer and a heterogeneous network of Sun workstations connected by an Ethernet.

5.1. Experimental Results Based on iPSC/2

The iPSC/2 uses datagrams for inter-processor communication. Each processor uses concurrent asynchronous broadcasts for routing, which allow frames to be routed more efficiently by the network server. As shown in Figure 5.1, a linear relation can be used to approximate the communication times of frames of different sizes. Table 5.1 shows the communication parameters obtained by a linear approximation of the curves in Figure 5.1. Note that the setup time includes all overheads for communication processing and that the exact communication model is very hard to characterize. Further, note that the broadcast parameters are measured under the situation when all processors broadcast concurrently and asynchronously (in which the host initiates a start signal for all processors to broadcast) rather than one processor broadcasting at a time. The computation parameters of a processor are shown in Table 5.2.

To illustrate the partitioning and mapping algorithm, consider a 3-layer ANN with one cluster in each layer and having 256, 1024, and 256 neurons in layer 1, 2, and 3, respectively. The ANN is to be mapped on a 16-node iPSC/2 hypercube computer. By setting ε, the predefined error allowance, to 0.04, the 16 processors are decomposed into two equal partitions, each of which has 8 processors. Any processor in a partition have a maximum distance of two hops to any other processor in the partition. Partitions 1 and 2 are each assigned 128, 512, and 128 neurons in layer 1, 2, and 3, respectively. The reason that the mapping to the two partitions are identical is a consequence of the symmetric and homogeneous properties of the Hypercube interconnection network. The inter-partition routing is trivial, since only two partitions are concerned. The predicted completion time for one iteration (including feed-forward and error-propagation phases) is 1.053 seconds, while the experimental completion time is 1.109 seconds. The relative error is 5.05%. The reason for the experimental relative error to be greater than the predefined ε is due to the approximation in the communication model.

The predicted results, the experimental results and the underestimation error for various ANN sizes are shown in Figure 5.2. All

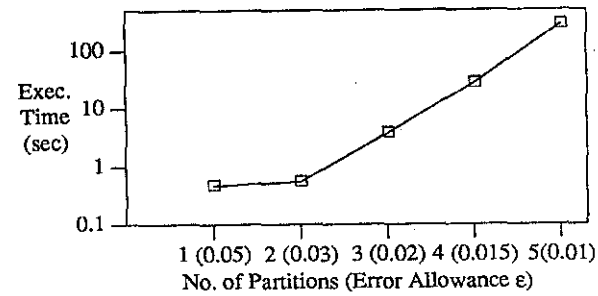


Figure 4.2. Execution times on Sun 3/60 for solving the optimal mapping on various numbers of partitions of 16-processor iPSC/2 (the error allowance ε is supplied by the users).

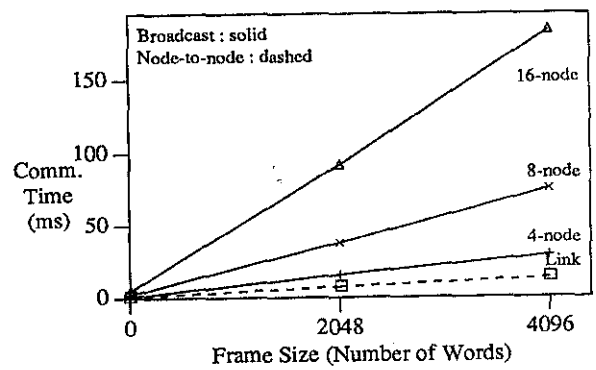


Figure 5.1. Communication time for different frame sizes.

Table 5.1. Communication parameters of Intel iPSC/2 Computer.

Parameter	Node-to-node Link	16-node Broadcast	8-node Broadcast	4-node Broadcast
τ_s (ms) per transmission	0.73	4.6	2.2	1.0
τ_t (μs) per word	3.08	43.6	17.8	6.84

Table 5.2. Computation parameters of Intel iPSC/2 Computer.

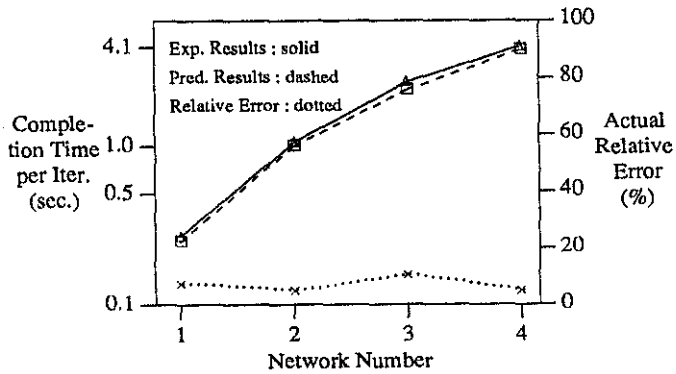
Parameter	MUL	ADD	MUL/ADD	Sigmoid
Time	9.52 μs	9.28 μs	11.96 μs	50.29 μs

* All operands are floating point numbers.

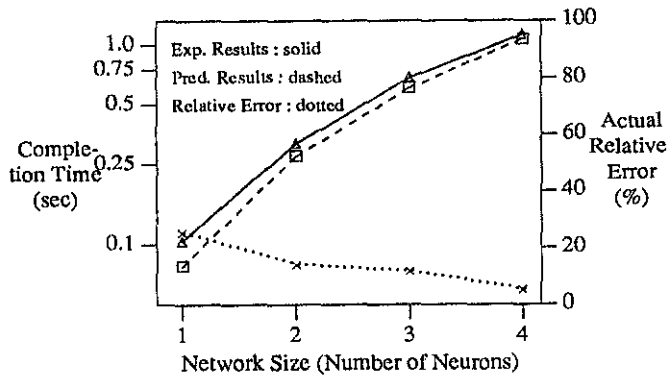
** All time parameters include memory access time.

experiments are simulating one iteration of the ANN operations, since all iterations are identical. The predicted results always under-estimate the actual completion time because the synchronization and problem-partitioning overheads are not included in our formulation. However, they produce the same effects on all mappings and do not affect their optimality.

Table 5.3 shows experimental results of simulating ANNs and hypercubes of various sizes using ε=0.05. All these ANNs have three layers and are fully connected between adjacent layers. It is interesting to observe that the efficiency (or the ratio of the speedup to the number of processors) is higher for smaller cubes. This happens because larger cubes need more time for inter-processor



(a) 4-node Hypercube partition.



(b) 16-node Hypercube partition.

Figure 5.2. Experimental results *vs.* predicted results on two Hypercube partitions (network configuration 1 is 64-256-64, configuration 2 is 128-512-128, configuration 3 is 192-768-192, and configuration 4 is 256-1024-256).

communication, and a larger portion of the execution time is wasted for this purpose. It is also interesting that larger ANNs have higher speedups. This is true because broadcasts are allowed in the system, and computation becomes more dominant over communication in larger ANNs.

A major limitation in using the iPSC/2 for ANN simulations is that the memory space is restricted in each processor. A processor lacks direct access to the common secondary memory, and all secondary-memory accesses must be handled by the Cube Manager. When the number of neurons mapped to each processor is larger than the capacity of its local memory, part of the data must be kept in the Cube Manager. This results in a high volume of traffic between the Cube Manager and the rest of the system.

5.2. Experimental Results Based on Workstations

The ANNs studied here have similar configurations as those defined in the last section. The target machine is a network of 3 heterogeneous Sun workstations connected by an Ethernet, one of which is a file server. The computation power is the lowest in node 0 and is the highest in node 2. Each processor is assumed using virtual-circuit communication, which has a one-time setup cost. Broadcasts on Ethernet using datagrams is not used in our experiments due to the small number of processors. However, they are useful when the number of processors is large. Since the communication overhead is relatively low, the three computers are grouped into one partition.

Table 5.3. Experimental results of simulating various ANNs on hypercubes of different sizes. Times are in seconds. $\varepsilon = 0.05$.

Network Configurations	Cube Size	Seq. Time	Parallel Time	Speedup	Efficiency
256,1024,256	16	15.10	1.12	13.48	0.843
256,1024,256	8	15.10	2.11	7.16	0.895
256,1024,256	4	15.10	4.14	3.64	0.910
192,768,192	8	9.09	1.27	7.16	0.895
128,512,128	8	3.74	0.55	6.80	0.850
64,256,64	8	0.95	0.15	6.33	0.791

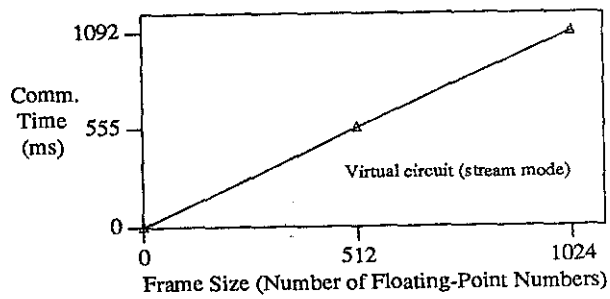


Figure 5.3. Communication time for different frame sizes in a network of 3 workstations.

Table 5.4. Communication parameters for a network of 3 Sun workstations.

Parameter	τ_s per transmission	τ_t per word
Time	108.36 ms	5.33 μs

Table 5.5. Computation parameters for a network of 3 Sun workstations.

Parameter	MUL	ADD	MUL/ADD	Sigmoid
Workstation 0	113.9 μs	97.7 μs	191.2 μs	451.7 μs
Workstation 1	73.2 μs	57.0 μs	122.1 μs	276.7 μs
Workstation 2	52.9 μs	44.8 μs	77.3 μs	195.3 μs

* All operands are floating point numbers.

The communication times for different frame sizes are shown in Figure 5.3. Using a linear approximation model, we can derive the communication parameters easily, which are summaried in Table 5.4. Note that the setup time is a one-time cost. The average computation parameters are shown in Table 5.5. These values are larger than the values shown in Table 5.2 because floating-point calculations were performed by software rather than by the floating-point coprocessor.

The predicted and experimental results are shown in Figure 5.4. The predicted results always under-estimate the actual completion time because the asynchronous network overheads are not included in our model. Further, unexpected page faults may also cause errors in our estimation since the parameters in Table 5.5 do not include

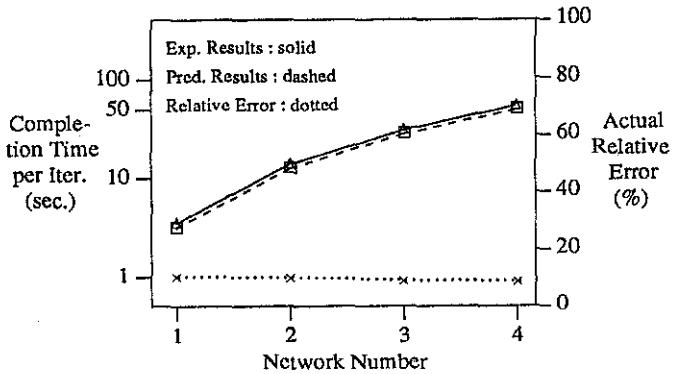


Figure 5.4a. Experimental results *vs.* predicted results on a set of 3 Sun workstations (network configuration 1 is 64-256-64, configuration 2 is 128-512-128, configuration 3 is 192-768-192, configuration 4 is 256-1024-256).

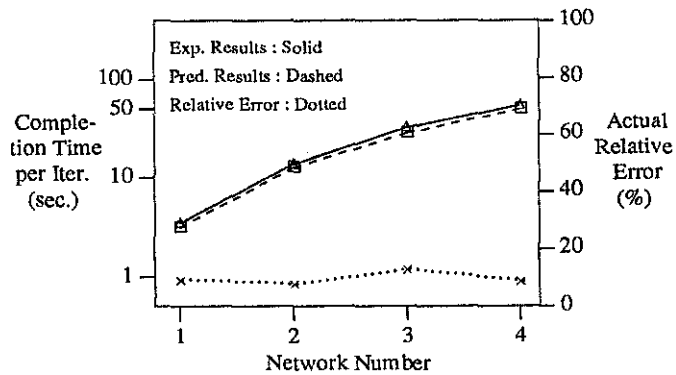


Figure 5.4b. Experimental results *vs.* predicted results on a set of 3 Sun workstations (network configuration 1 is 128-128-128, configuration 2 is 256-256-256, configuration 3 is 384-384-384, configuration 4 is 512-512-512).

Table 5.6. Experimental results of simulations of different ANNs on a network of 3 Sun workstations. Times are in seconds. $\varepsilon = 0.05$.

Network Size	Seq. Time	Parallel Time	Speedup
256,1024,256	94.70	54.60	1.73
192,768,192	54.03	30.83	1.75
128,512,128	24.28	13.88	1.75
64,256,64	6.23	3.50	1.78

these overheads.

Experimental results for various ANNs and ε=0.05 are shown in Table 5.6. All these networks have three layers and are fully-connected between adjacent layers. It is observed that the speedup does not change with increasing ANN size, as in the case with the iPSC/2. This is true because larger ANNs incur more memory traffic over the network, which reduce the corresponding speedup.

6. CONCLUSIONS

In this paper, we have studied the heuristic mapping of neurons in the learning process of a multilayer ANN on a multicomputer system. Processors in the multicomputer may be heterogeneous and may be connected by communication links of different speeds. The mapping problem is NP-hard in general. We derive a number of results for reducing the complexity of the mapping problem. By observing the dominance of the computation time over the communication time in the learning operations within a layer of the ANN, we decompose the set of processors into partitions in such a way that the error deviation of a heuristic intra-partition routing scheme from the optimal intra-partition one can be bounded. Experimental results using a 16-processor Intel iPSC/2 computer and a network of three Sun workstations are shown and are found to be very close to the analytical results predicted. Our results are useful for designing special-purpose computers for ANN simulations and for determining the suitability of an existing multicomputer for ANN applications.

REFERENCES

[1] L.-C. Chu, *Optimal Mapping of Neural Networks on Multicomputers*, M.Sc. Thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, Aug. 1990.

[2] J. A. Feldman, M. A. Fanty, N. H. Goddard, and K. J. Lynne, "Computing with Structured Connectionist Networks," *Communications of the ACM*, vol. 31, no. 2, pp. 170-187, Feb. 1988.

[3] B. M. Forrest, D. Roweth, N. Stroud, D. J. Wallace, and G. V. Wilson, "Implementing Neural Network Models on Parallel Computers," *The Computer Journal*, vol. 30, no. 5, pp. 413-419, British Computer Society, Cambridge University Press, 1987.

[4] J. Ghosh and K. Hwang, "Mapping Neural Networks onto Message-Passing Multicomputers," *J. Parallel and Distributed Computing*, vol. 6, pp. 291-330, Academic Press, 1989.

[5] K. Hwang and J. Ghosh, "Critical Issues in Mapping Neural Networks on Message-Passing Multicomputers," *Int'l Symp. on Computer Architecture*, pp. 3-11, ACM/IEEE, 1988.

[6] S. S. Kung, "Parallel Architectures for Artificial Neural Nets," *Proc. Int'l Conf. on Systolic Arrays*, pp. 163-174, IEEE, 1988.

[7] S. Y. Kung and J. N. Hwang, "A Unified Systolic Architecture for Artificial Neural Networks," *J. Parallel and Distributed Computing*, vol. 6, pp. 358-387, Academic Press, 1989.

[8] D. A. Pomerleau, G. S. Gusciora, D. S. Touretzky, and H. T. Kung, "Neural Network Simulation at Warp Speed: How We Got 17 Million Connections Per Second," *Proc. Int'l Conf. on Neural Networks*, vol. II, pp. 143-150, IEEE, July 1988.