Parallel Implementation of Back-Propagation Algorithm in Networks of Workstations

S. Suresh, S.N. Omkar, and V. Mani

Abstract—This paper presents an efficient mapping scheme for the multilayer perceptron (MLP) network trained using back-propagation (BP) algorithm on network of workstations (NOWs). Hybrid partitioning (HP) scheme is used to partition the network and each partition is mapped on to processors in NOWs. We derive the processing time and memory space required to implement the parallel BP algorithm in NOWs. The performance parameters like speed-up and space reduction factor are evaluated for the HP scheme and it is compared with earlier work involving vertical partitioning (VP) scheme for mapping the MLP on NOWs. The performance of the HP scheme is evaluated by solving optical character recognition (OCR) problem in a network of ALPHA machines. The analytical and experimental performance shows that the proposed parallel algorithm has better speed-up, less communication time, and better space reduction factor than the earlier algorithm. This paper also presents a simple and efficient static mapping scheme on heterogeneous system. Using divisible load scheduling theory, a closed-form expression for number of neurons assigned to each processor in the NOW is obtained. Analytical and experimental results for static mapping problem on NOWs are also presented.

Index Terms—Multilayer perceptron, back-propagation, network of workstation, optical character recognition, performance measures, divisible load theory.

1 Introduction

In the last few decades, extensive research has been carried out in developing the theory and the application of Artificial Neural Networks (ANNs). ANN has emerged to be a powerful mathematical tool for solving various practical problems like pattern classification and recognition [1], medical imaging [2], speech recognition [3], [4], and control [5]. Of the many Neural Network (NN) architectures proposed, the Multilayer Perceptron (MLP) with Back-Propagation (BP) learning algorithm is found to be effective for solving a number of real world problems. The speed and storage space requirements to implement this can be circumvented using parallel computing techniques [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. Since ANNs are inherently suitable for parallel implementation, a lot of research is being carried out to implement ANN in commercially available parallel computers like Distributed-memory Message Passing Multiprocessor (DMPM) [6], Connection machine [7], the warp [8], the MPP [9], and the BBN Butterfly [10]. The MLP network trained using BP algorithm can be parallelized by partitioning the number of patterns [11], [12], or by partitioning the network [20], [21], [22], [23], [24], [25], [26], or by combination [11]. Several researchers have developed parallel algorithms for implementing in different types of parallel computers like transputers [16], systolic arrays [18], [20], multiple bus system [14], and hypercube [11], [27] for diverse range of applications. A detailed survey on parallel implementations of BP algorithm in a variety of computer architectures is given in [16].

As the cost of dedicated parallel machines is high, computing on a network-of-workstations (NOWs) is proven to be economical alternative for a wide range of engineering applications [24]. The most important feature for these computing networks is that it enables the use of existing resources. Furthermore, these resources can be shared with the other applications that require them. For a small application, computing on NOW is comparable with hypercube multiprocessor [25]. Mapping fully connected MLP on homogeneous SUN 3/50 workstation is considered in [26].

In [26], the vertical partitioning (VP) scheme is used to divide the MLP into m equal subnetworks and each subnetwork is mapped on a processor in the network. In many real world problems, MLP with an equal number of neurons in all layers is quite uncommon. From a practical point of view, the requirement of homogeneous NOW setup may not be viable and finding optimal mapping of NN architecture is computationally intensive. The intention of the present work is to remove these restrictions by focusing on the development of a very efficient scheme for general MLP network in heterogeneous NOWs. This paper also provides a framework for comparing the proposed method with existing scheme [26].

This paper presents a hybrid partitioning (HP) scheme for general MLP network with parallel BP learning algorithm on heterogeneous NOWs. The proposed HP scheme avoids recomputation of weights and require less communication cycle per pattern. The communication of data among the processors in the computing network is carried out using grouped All-to-All broadcast (GAAB) scheme [29]. We derive a closed-form expression for training single pattern in NOWs

• The authors are with the Department of Aerospace Engineering, Indian Institute of Science, Bangalore-560012, India.
E-mail: {suresh99, omkar, mani}@aero.iisc.ernet.in.

Manuscript received 28 Oct. 2003; revised 15 Mar. 2004; accepted 1 June 2004; published online 23 Nov. 2004.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0200-1003.

and also obtain the performance parameters like speed-up, space reduction factor, and optimal number of processors for homogeneous environment. In order to compare the proposed scheme with earlier VP scheme [26], the training time of VP scheme for single pattern in general MLP network is derived and also the closed-form expression for the performance parameters is obtained. The VP and HP scheme are evaluated using optical character recognition problems in homogeneous ALPHA workstations. The analytical and experimental results show that the proposed HP scheme performs better than the VP scheme.

Another important issue in parallel algorithm is mapping the NN architecture onto the computing network. The optimal mapping problem depends on the type of computing network and workloads. Based on the type of workloads, the mapping scheme can be divided into static mapping and dynamic mapping problems. In the case of static, the NN is partitioned and mapped on to the computing network and this partition remains until training process is completed. In the case of the dynamic mapping problem, the NN partition will change with time due to other workloads in the computing network. The mapping problem is a nonlinear mixed integer programming optimization problem with communication and memory constraints. The optimal static mapping of MLP network in multicomputers is formulated as mixed integer programming problem in [28]. The mapping problem can be solved using approximate linear heuristic optimization methods [14], [22]. An attempt to solve the static mapping problem using genetic algorithm is described in [16], [31]. The mapping problem using the approximate linear heuristic methods and genetic algorithm is computationally intensive.

In this paper, we also present a simple and computationally less intensive approach for static mapping problem in NOWs. For static mapping to be meaningful, the workstations are assumed to run in a single-user environment. The HP scheme provides a platform to formulate the static mapping problem as scheduling divisible loads in NOWs. In recent years, there is a great deal of attention focused on divisible load scheduling problem in a distributed computing system/network, consisting of a number of processors interconnected through communication links [33]. A divisible load can be divided into any number of fractions and can be processed independently on the processors as there are no precedence relationships. In other words, divisible load has the property that all the elements in the load require the same type of processing. Recent research papers in divisible load theory can be found in [34]. Using the concept of scheduling divisible loads, a closed-form expression is derived for the number of neurons assigned to each processor. The analytical and experimental studies are carried out in a heterogeneous ALPHA workstations. The results show that the HP scheme using scheduling divisible loads is computationally less intensive.

The use of computation intensive applications in NOWs has the following advantages. NOWS provide high-degree of performance isolation, i.e., they allow analysis of behavior on a node-by-node basis or factor-by-factor basis. NOWs provide incremental scalability of hardware resources. Well tuned parallel programs can be easily scaled to large configurations because additional workstations can always be added to NOW. The NOW also offer much greater communication speed at lower price using switch-based networks such as ATMs. NOWs also provide distributed service and support, especially for file systems. The technological evolution allows NOW to support a variety of disparate workloads, including parallel, sequential, and interactive jobs, as well as scalable computation intensive applications [35]. This paper provides a theoretical framework on analytical performance estimation and comparison of different parallel implementation of BP algorithm in NOWs.

The paper is organized as follows: In Section 2, we introduce the BP algorithm and mathematical model for execution of the algorithm in a uniprocessor simulation. Section 3 describes the proposed HP scheme and also derives the training time required to execute the parallel BP algorithm. In Section 4, we present an earlier VP scheme and expression for training time. We compare the analytical performances of both the schemes in Section 5. The optimal mapping of proposed scheme in heterogeneous NOWs using divisible load scheduling is presented in Section 6. Experimental performances of both the schemes on NOWs are presented in Section 7 and the results are discussed in Section 8.

2 Mathematical Model for BP

In this paper, we consider a fully connected MLP network trained using BP algorithm with patternwise approach. A single hidden layer MLP network with a sufficient number of hidden neurons are sufficient for approximating the input-output relationship [32]. Hence, in our analysis, we consider MLP with three layers ($l = 0, 1, 2$) having N^l neurons in each layer. The different phases in the learning algorithm and corresponding processing time are discussed in the following sections.

2.1 Back-Propagation Algorithm

The BP algorithm is a supervised learning algorithm, and is used to find suitable weights, such that for a given input pattern (U^0) the network output (Y_i^2) should match with the target output (t_i). The algorithm is divided into three phases, namely, forward phase, error BP phase, and weight update phase. The details on each of these phases and the time taken to process are discussed below.

Forward phase: For a given input pattern U^0, the activation values of hidden and output layer are computed as follows:

$$Y_i^l = f\left(\sum_{j=1}^{N^{l-1}} W_{ij}^l U_j^{l-1}\right), \; i = 1, 2, \cdots, N^1, \tag{1}$$

where $f(.)$ is a bipolar sigmoidal function. Let t_m, t_a, and t_{ac} be time taken for one floating point multiplication, addition, and calculation of activation value, respectively. The time taken to complete the forward phase (t_1) is by

$$t_1 = N^1\big(N^0 + N^2\big)M_a + t_{ac}\big(N^2 + N^1\big), \tag{2}$$

where $M_a = t_m + t_a$.

Error back-propagation phase: In this phase, the deviation between the network output and the target value is back-propagated to all the neurons through output weights. The δ_i^2 term for ith neuron in output layer is given by

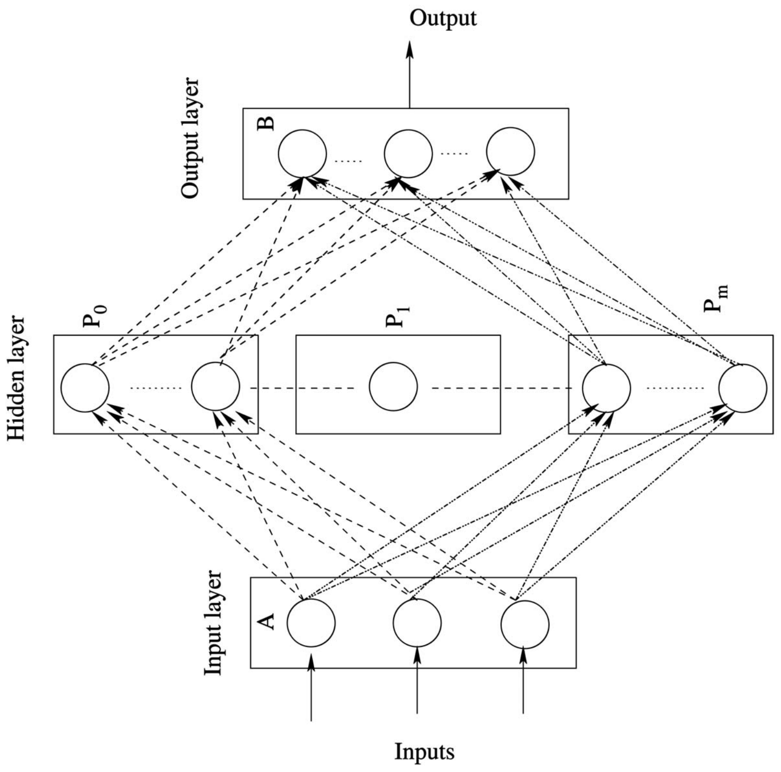

Fig. 1. Parallel architecture for MLP in HP scheme.

$$\delta_i^2 = \left(Y_i^2 - t_i\right) f'(.) \; i = 1, 2, \cdots, N^2, \tag{3}$$

where $f'(.)$ is the first derivative of the activation function. The term (δ_i^1) for ith hidden neuron is given by

$$\delta_i^1 = f'(.) \sum_{j=1}^{N^2} W_{ji}^2 \delta_j^2, \;\; i = 1, 2, \cdots, N^1. \tag{4}$$

The time taken to complete the error back-propagation phase is represented by t_2 and is calculated as

$$t_2 = N^1 N^2 M_a. \tag{5}$$

Weight update phase: In this phase, the network weights are updated and the updation process of any W_{ij}^l depends on the value of δ_j^l and Y_i^{l-1}.

$$W_{ij}^l = W_{ij}^l + \eta \delta_i^l Y_j^{l-1} \;\; l \; = \; 1, 2, \tag{6}$$

where η is the learning rate. The time taken to update the weight matrix between the three layers is represented by t_3 and it is equal to

$$t_3 = N^1 \left(N^2 + N^0\right) M_a. \tag{7}$$

Let $t_{ac} = \beta M_a$. The total processing time (T_{seq}) for training a single pattern is the sum of the time taken to process the three phases and is given as

$$\begin{aligned} T_{seq} &= t_1 + t_2 + t_3 \\ &= \left[N^1 K + \beta N^2\right] M_a, \end{aligned} \tag{8}$$

where $K = 2N^0 + 3N^2 + \beta$.

3 Distributed Parallel Algorithm

In this section, we describe the proposed parallel BP algorithm to train MLP in homogeneous NOWs.

3.1 Hybrid Partitioning Scheme

The Hybrid partitioning (HP) algorithm is a combination of neuronal level as well as synaptic level parallelism [16]. In case of *neuronal level parallelism or vertical slicing*, all incoming weights to the neurons local to the processor is kept in one processing element. In *synaptic level parallelism*, each processor will have only the outgoing weight connections of the neurons local to the processors. In the HP scheme, the hidden layer is partitioned using neuronal level parallelism and weight connections are partitioned on the basis of synaptic level parallelism. The parallel architecture of the MLP network used in the proposed scheme is shown in the Fig. 1. In case of a homogeneous m-processor network, the hidden layer is partitioned into $\frac{N^1}{m}$ neurons and the input and output neurons are common for all the processors, i.e., the blocks A and B in Fig. 1 are common for all the processors. The blocks P_0, A, and B are kept at processor p_0, blocks P_i, A, and B are kept at processor p_i, respectively. Each processor will store the weight connections between the neurons local to the processor.

3.2 Parallel Algorithm

Since we have partitioned the fully connected MLP network into m partitions and then mapped onto m processors, each processor is required to communicate with every other processors to simulate the complete network. Each of the processors in the network execute three phases of BP training algorithm. Parallel execution of the three phases and the corresponding processing time for each phases are calculated.

Forward phase: In this phase, the activation value of the neurons local to the processors are calculated. For a given input pattern, using (1), we can calculate the activation value for the hidden neurons. But, we need the activation values and the weight connections (w_{ij}^2) of neurons present in other processors to calculate the activation values of output neurons. Broadcasting the weights and activation values are circumvented by calculating the partial sum of the activation values of the output neurons. This partial sum is exchanged between the processors to calculate the activation values of the output neurons. The approximate time taken to process the forward phase (t_1^o) is given by

$$t_1^o = \frac{N^1}{m}\left(N^0 + N^2 + \beta\right)M_a + \beta N^2 M_a + T_{com}^o, \tag{9}$$

where T_{com}^o is time taken to communicate the partial sum to all the processors in the network.

Error propagation phase: In this phase, each processor calculates the error terms δ^l for the local neurons. The approximate time taken for processing error back-propagation phase (t_2^p) is

$$t_2^o = \frac{N^1}{m} N^2 M_a. \tag{10}$$

Weight update phase: In this phase, the weight connections between the neurons local to that processor alone are updated. To update the weight connection between the ith hidden neuron and jth input neuron, we need activation value at the jth input neuron and δ_i^1 term in the ith hidden neuron. Since both the activation value as well as the error term are present in the processor, no intercommunication is required. The processing time to update the weight connections (t_3^o) is given by

$$t_3^o = \frac{N^1}{m}\left(N^0 + N^2\right)M_a. \tag{11}$$

Let T_p^o be the time taken to train a single pattern in proposed HP scheme, and it is equal to the algebraic sum of time taken to process all the three phases.

$$T_p^o = t_1^o + t_2^o + t_3^o = \left[\frac{N^1}{m}K + N^2\beta\right]M_a + T_{com}^o. \tag{12}$$

In the GAAB scheme [29], all the values to be broadcast are grouped together and broadcasted as a single message. This reduces the overhead for processing each broadcast. The communication time to broadcast a message of size D in GAAB scheme is calculated below:

$$t_{com} = m(T_{ini} + T_c g(D)), \tag{13}$$

where T_c is time taken to send/receive one floating point number, T_{ini} is communication start-up time, and $g(D)$ is the scaling of grouped broadcast scheme. Normally, $g(D)$ is much less than D, which is the worst case for one-to-one broadcast scheme [29].

The communication time (T_{com}^o) required for broadcasting data of size N^2 is given by

$$T_{com}^o = m\left[T_{ini} + T_c g\left(N^2\right)\right] = mA_{com}^o M_a, \tag{14}$$

where $T_{ini} = M_a\alpha$, $t_c = M_a\gamma$, and $A_{com}^o = \alpha + \gamma g(N^2)$.

By substituting (14) in (12), we obtain

$$T_p^o = \left[\frac{N^1}{m}K + N^2\beta + mA_{com}^o\right]M_a. \tag{15}$$

The processing time in the HP approach shows that the computation time (the first term in the above equation) will decrease with the increase in number of processors and communication time (proportional to number of processors) as in (14) will increase with the increase in number of processors.

4 Distributed Parallel Algorithm

We now derive the expression for time taken to train the MLP with parallel BP algorithm described in [26].

4.1 Vertical Partitioning Scheme

In [26], the MLP network is vertically partitioned (VP) into m partitions and each partition is mapped on to a processor in NOWs. In the VP scheme, each layer having N^l neurons, is divided into $\frac{N^l}{m}$ neurons that are assigned to each processor in the computing network. The weight connections are partitioned on the basis of combination of inset and outset grouping scheme [16]. This leads to duplication of weight vectors in different processors and results in extra computation in weight update phase.

4.2 Parallel Algorithm

Parallel execution of the three phases of the BP algorithm and the corresponding training time for each phase are calculated for VP scheme.

Forward phase: In order to calculate the activation values of the output neurons, we need activation value of all the hidden neurons. Hence, before starting the calculation of activation values of output neurons, the activation values of neurons in the hidden layer are exchanged between the processors. Let the time taken to execute the communication process be equal to t_{com}^1. The time taken to process forward phase (t_1^s) is equal to the sum of time taken to compute activation values of the local neurons and communication time (t_{com}^1).

$$t_1^s = \frac{N^1}{m}\left(N^0 + N^2 + \beta\right)M_a + \frac{N_2}{m}\beta M_a + t_{com}^1. \tag{16}$$

Error propagation phase: In order to calculate the term δ^1 in the local hidden neurons, we need error term δ^2 of all the output neurons present in the other processors. Hence, the time taken to compute error propagation phase (t_2^s) is equal to the sum of the computation time required for calculating the error terms δ^l and communication time (t_{com}^2) required to broadcast the term δ^2.

$$t_2^s = \frac{N^1}{m} N^2 M_a + t_{com}^2. \tag{17}$$

Weight update phase: Since the VP scheme has a duplicated weight vector, we need an extra computational effort to keep the consistency in the weight connection. The time taken to process this phase (t_3^s) is equal to the sum of the time taken to update the weight connection between the neurons local to that processor and the time taken to update the duplicated weight connections.

$$t_3^s = \frac{N^1}{m}\left(2N^2 - \frac{N^2}{m} + N^0\right)M_a. \tag{18}$$

Let $T_{com}^s = t_{com}^1 + t_{com}^2$. The total processing time in VP algorithm (T_p^s) for training a single pattern is the sum of time taken to process the three phases and is given as

$$T_p^s = \left[\frac{N^1}{m}\left(K + N^2 - \frac{N^2}{m}\right) + \beta\frac{N^2}{m}\right]M_a + T_{com}^s. \tag{19}$$

Using the GAAB scheme given in (13), the total communication time (T_{com}^s) required to broadcast the data of size $\frac{N^1}{m}$ and $\frac{N^2}{m}$ is given as

$$T_{com}^s = mA_{com}^s M_a, \tag{20}$$

where $A_{com}^s = 2\alpha + \gamma\left[g\left(\frac{N^1}{m}\right) + g\left(\frac{N^2}{m}\right)\right]$.

The training time (19) can be modified as

$$T_p^s = \left[\frac{N^1}{m}\left(K + N^2 - \frac{N^2}{m}\right) + \frac{\beta}{m}N^2 + A_{com}^s\right]M_a. \tag{21}$$

The processing time for the VP scheme also shows that the computation time will decrease with increase in the number of processors and the communication time will increase with increase in the number of processors.

5 Analytical Performance Comparison

In this section, we calculate the various performance measures like speed-up, space reduction ratio, maximum number of processors, and optimal number of processors. We also provide the processing time difference between the two methods. Finally, we present the advantages of the HP scheme over the VP scheme.

5.1 Maximum Number of Processors

Let the maximum number of processors for the VP scheme be M_p^s and the HP scheme be M_p^o. The maximum number of processors for the VP scheme is equal to number of hidden neurons or number of output neurons, whichever is the minimum, i.e., $min(N^1, N^2)$. In most of the practical cases, $M_p^s = N^2$ since the number of output neurons is always less than the number of hidden neurons. The maximum number of processors M_p^o used in the HP scheme is equal to the number of hidden neurons N^1. Since $N^1 >> N^2$, the proposed HP scheme exploits parallelism very well.

5.2 Speed-Up Analysis

Speed-up for m-processor system is the ratio between the time taken by uniprocessor ($m = 1$) to the time taken by parallel algorithm in m-processor network.

$$S(m) = \frac{T_{seq}}{T_p}. \tag{22}$$

From (8) and (15), the speed-up ratio for the proposed HP scheme ($S^o(m)$) can be formulated as

$$S^o(m) = \frac{N^1K + N^2\beta}{\frac{N^1}{m}K + N^2\beta + mA_{com}^o}. \tag{23}$$

Similarly, from (8) and (21), speed-up ratio for the VP scheme ($S^s(m)$) is

$$S^s(m) = \frac{N^1K + N^2\beta}{\frac{N^1}{m}\left[K + N^2 - \frac{N^2}{m}\right] + \frac{N^2}{m}\beta + mA_{com}^s}. \tag{24}$$

In most of the practical problems, $N^1 >> N^2$, then the term $N^2\beta$ is small and neglected in the speed-up equation. Hence, the speed-up ratio for both the algorithm is modified as below:

$$S^o(m) = \frac{N^1K}{\frac{N^1}{m}K + mA_{com}^o} \tag{25}$$

$$S^s(m) = \frac{N^1K}{\frac{N^1}{m}K + \frac{N^1N^2}{m^2}(m-1) + mA_{com}^s}. \tag{26}$$

If the network size is extremely larger than the number of processors m, then the speed-up ratio will approach m in HP approach and it is less than m for VP scheme. This is due to extra computation required in weight updation phase and extra communication in exchanging the hidden neurons activation values.

Suppose, if $N^1 = N^2 = N$, then the above equations are modified as follows:

$$S^o(m) = \frac{N(5N + 2\beta)}{\frac{N}{m}(5N + (m+1)\beta) + mA_{com}^o}$$

$$S^s(m) = \frac{N(5N + 2\beta)}{\frac{N}{m}\left(6N + 2\beta - \frac{N}{m}\right) + mA_{com}^s}.$$

It is clear from the above equation that if $N >> m$, then $S^o(m)$ converges to m and $S^s(m)$ converges to $\frac{5}{6}m$. Hence, the speed-up factor for HP scheme is approximately 16 percent more than the VP scheme.

5.3 Storage Space Requirement

The total space requirement to execute the BP algorithm in single processor (M_s) is the sum of space requirement to store weight matrix, space required for activation and error terms.

$$M_s = N^1\left(N^0 + N^2 + 2\right) + 2N^2 + N^0. \tag{27}$$

The ratio between the space required by a single processor and the space required for the m-processor network is defined as space reduction ratio $M(m)$.

$$M(m) = \frac{M_s}{M_p}. \tag{28}$$

The total space required to execute the proposed HP scheme (M_p^o) in m processors system is the sum of space requirement to store weight matrix, space required for activation values, error terms, and the buffer size required to receive the data from different processors.

$$M_p^o = \frac{N^1}{m}(N^0 + N^2 + 2) + 3N^2 + N^0. \tag{29}$$

The total space required for VP approach (M_p^s) is the sum of space required for weights, error values, activation values of the neurons local to the processor, and the buffer requirement to communicate the data.

$$M_p^s = \frac{N^1}{m} L + 3N^2 + N^0 + 2N^1, \quad (30)$$

where $L = 2N^2 + N^0 + 1 - \frac{N^2}{m}$. Space reduction ratio for the proposed HP algorithm ($M^o(m)$) is given by the ratio between (27) and (29)

$$M^o(m) = \frac{N^1(N^0 + N^2 + 2) + 2N^2 + N^0}{\frac{N^1}{m}(N^0 + N^2 + 2) + 3N^2 + N^0}. \quad (31)$$

Similarly, space reduction ratio ($M^s(m)$) be for the VP approach

$$M^s(m) = \frac{N^1(N^0 + N^2 + 2) + 2N^2 + N^0}{\frac{N^1}{m} L + 3N^2 + N^0 + 2N^1}. \quad (32)$$

In case of a larger MLP network configuration and a small number of processors, the space reduction ratio will converge to number of processors (m) in the proposed HP approach, whereas it will be less than the number of processors for VP approach. This is due to the fact that the extra space required to store the duplicated weights and communication buffer required for activation values of hidden layer in the VP scheme.

If $N^1 = N^2 = N$, then the above equations are modified as follows:

$$M^o(m) = \frac{m(2N+5)}{2N + 4m + 2}$$
$$M^s(m) = \frac{m(2N+5)}{3N + 6m + 1 - \frac{N}{m}}.$$

It is clear from above equations that if $N >> m$, then $M^o(m)$ will converge to m, whereas $M^s(m)$ will converge to $\frac{2}{3}m$. The results indicate that the proposed HP scheme has better space reduction ratio.

5.4 Optimal Number of Processors

From (15), it is clear that if we increase the number of processors, the time taken to communicate will also increase and the time taken for computation will decrease. The total processing time will decrease first and then increase after a certain number of processors. So, there exists an optimal number of processor m^*, for which processing time is minimum. We calculate the closed form expression for optimal number of processors for the proposed HP algorithm by partially differentiating the training time expression T_p^o with respect to m and then equating it to zero.

$$\frac{\partial T_p^o}{\partial m} = \left[-\frac{N^1}{m^2} K + A_{com}^o\right] M_a. \quad (33)$$

Hence, optimal number of processors m^* is equal to

$$m^* = \left(\frac{N^1 K}{A_{com}^o}\right)^{\frac{1}{2}}. \quad (34)$$

Similarly for the VP algorithm, we derive a condition for optimal number of processors by partially differentiating the equation for processing time T_p^s with respect to m and equating to zero.

$$A_{com}^s m^3 - A^1 m + 2N^1 N^2 = 0, \quad (35)$$

where $A^1 = N^1(N^2 + K) + N^2\beta$. From the above equation, we can observe that obtaining closed-form expression for m^* is difficult.

5.5 Difference between the Processing Times

From (15) and (21), the difference in processing time is calculated as follows:

$$T_p^s - T_p^o = \left[A^2 + m\left(A_{com}^s - A_{com}^o\right)\right] M_a, \quad (36)$$

where $A^2 = N^2 \frac{m-1}{m}\left(\frac{N^1}{m} - \beta\right)$. The first term in the above equation is due to recomputation of weights in the VP algorithm, the second term is due to extra calculation of activation function for output neurons in HP approach, and the third term is due to difference between the communication time. Now, we consider two different conditions for network architecture with grouped AAB scheme, namely, 1) the number of neurons in each layers are equal and 2) the number of neurons in the hidden layer is greater than the output layer.

Case 1: Now, consider an MLP network with an equal number of neurons in all the layers, i.e., $N^l = N$, $l = 0, 1, 2$. First, we will find out the difference between the communication term ($A_{com}^s - A_{com}^o$) by substituting $N^l = N$ in (20) and (14) as

$$A_{com}^s - A_{com}^o = \left[\alpha + 2\gamma g\left(\frac{N}{m}\right) - \gamma g(N)\right]. \quad (37)$$

Since the communication time will not increase with the increase in the size of the message size for a grouped AAB scheme, the terms $\gamma g(N)$ and $\gamma g(\frac{N}{m})$ are neglected. Hence, the difference between the communication time is approximately equal to $m M_a \alpha$.

By substituting the difference between the communication terms and $N^l = N$ in (36), the difference in processing time is reduced to

$$T_p^s - T_p^o = \left[N \frac{m-1}{m}\left(\frac{N}{m} - \beta\right) + m\alpha\right] M_a. \quad (38)$$

For analytical study purpose, we assume the value of β equal to 40 and α equal to 55 as mentioned in [26]. Suppose each and every processor contains one neuron per layer ($N = m$), then the difference between the processing time is equal to $[16m + 39] M_a$. From this it is clear that the time taken by HP scheme is less than the VP scheme. In the general case, the proposed HP scheme is better than the VP scheme if $\alpha > (m-1)\frac{\beta - 1}{m}$. For a larger number of processors, the fraction $\frac{m-1}{m}$ is almost equal to 1. Hence, the condition is further simplified to $\alpha > (\beta - 1)$.

Case 2: In general, most of the practical problems will have more number of hidden neurons than the output neurons, i.e., $N^1 >> N^2$, hence maximum number of processors for VP scheme described in Section 5 is equal to N^2 and N^1 for the HP approach. Let m be equal to N^2, the above equation can be simplified as

$$T_p^s - T_p^o = \left[N^1 - \frac{N^1}{N^2} + \beta + N^2(\alpha - \beta)\right] M_a. \quad (39)$$

Time taken to train the MLP using the HP scheme is better than the VP scheme if $\beta < \frac{N^1}{N^2} + \frac{N^2}{N^2 - 1}\alpha$. The communication

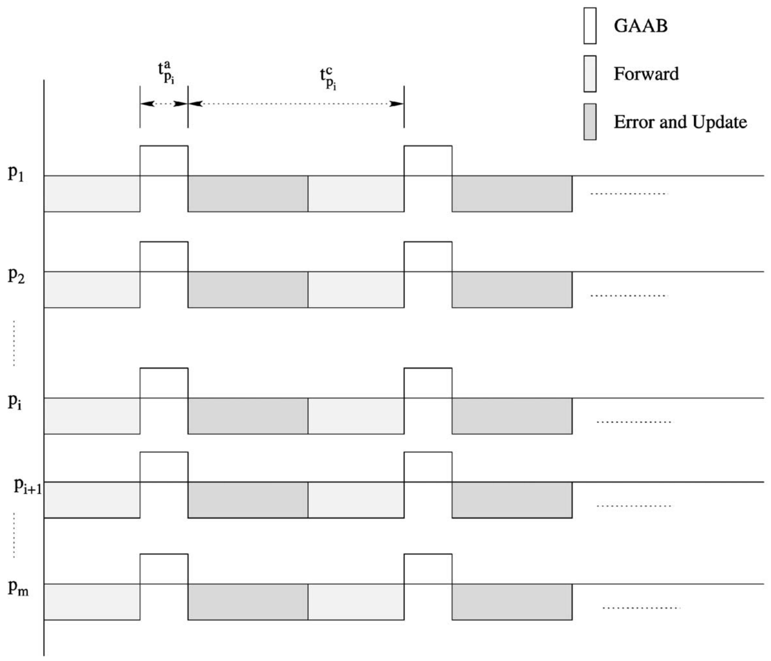

Fig. 2. Timing diagram for the HP algorithm.

time factor (α) will be greater than the activation factor β and, hence, the HP scheme is more efficient to train the MLP network using distributed BP algorithm.

5.6 Improvement over a VP Scheme

The improvements of the proposed HP scheme over the VP scheme [26] are:

- The proposed algorithm uses only one set of communication, when compared with two sets of communication used in VP approach.
- Recomputation of weights is avoided in HP Scheme.
- The proposed HP approach can exploit the parallelism by using more number of processors than the VP scheme.
- Mapping the proposed HP scheme on NOWs is simpler than the VP scheme. This aspect will discussed in the next section.

6 Mapping MLP Network on NOWs

The mapping problem of MLP network on a NOWs entails the search for an optimal mapping of the neurons to the processors so that the training time is minimum. The mapping involves schemes for assigning neurons to processors and schemes for communicating data among the processors in the computing network. A mapping scheme depends on the network architecture, computing network, and communication scheme. Here, we develop a closed-form expression for mapping HP scheme on heterogeneous NOWs.

Let m be the number of workstations in the computing network. In the HP partitioning scheme, mapping the network on NOWs means dividing the hidden layer arbitrarily, i.e., the number of neurons in the hidden layer is divided into m parts and mapped onto the processors. The timing diagram for different phases of training process in different processors is shown in Fig. 2. Let n_i be the number of hidden neurons assigned to the processor p_i. The time taken to train single pattern in the processor p_i is expressed as

$$T^o_{p_i} = t^a_{p_i} + t^c_{p_i}, \qquad (40)$$

where $t^a_{p_i}$ and $t^c_{p_i}$ are the time taken to complete communication and the computation process

$$t^a_{p_i} = m\left[\alpha^i + \gamma^i f(N^2)\right] M^i_a$$
$$t^c_{p_i} = \left[n_i\left(2N^0 + 3N^2 + \beta^i\right) + N^2\beta^i\right] M^i_a.$$

The M^i_a, β^i, α^i, γ^i are corresponding values of the parameters in processor p_i. For a given MLP architecture, the communication time is linearly proportional to the number of neurons assigned to the processors and it is independent of the number of hidden neurons assigned to the processors. The computation time for any processor has two components. The first component depends on the number of hidden neurons assigned to the processor and the other component is a constant for given MLP architecture. Hence, the computation time taken for processor p_i can be rewritten as

$$t^c_{p_i} = n_i t^i_1 + t^i_2, \qquad (41)$$

where $t^i_1 = (2N^0 + 3N^2 + \beta^i)M^i_a$, $t^i_2 = N^2\beta^i M^i_a$.

Let us assume that all the processors in the network will stop computing at the same time instant. This assumption has been shown to be necessary and sufficient condition to obtain optimal processing time [33]. From the timing diagram, we can write the following equations:

$$n_i t^i_1 + t^i_2 + t^a_{p_i} = n_{i+1} t^{i+1}_1 + t^{i+1}_2 + t^a_{p_{i+1}}, \qquad (42)$$

$i = 1, 2, \cdots, m-1$.

TABLE 1
Time Parameter in ALPHA Machines

Workstation group	t_m in μs	t_a in μs	t_{ac} in μs	T_c in μs	T_{ini} in μs
1	1.732	1.732	53.5	0.3	64.5
2	0.637	0.637	13.5	0.1	44.2

TABLE 2
Network Architectures

	Binary number representation	Network structure $N^0 \times N^1 \times N^2$	Total weights
N_1	12 × 12	144 × 144 × 12	22464
N_2	6 × 8	48 × 48 × 6	2592

The above equations can be rewritten as

$$n_i = n_{i+1} f_{i+1} + g_i,\ i = 1, 2, \cdots, m-1, \tag{43}$$

where $f_{i+1} = \frac{t_1^{i+1}}{t_1^i}$ and $g_i = \frac{t_2^{i+1} + t^a_{p_{i+1}} - t_2^i - t^a_{p_i}}{t_1^i}$.

From this, it can be seen that there are $m-1$ linear equations with m variables and together with normalization equation ($\sum_{i=1}^{m} n_i = N^1$) we have m equations. Now, closed-form expression to find the number of neurons assigned to each processor can be formulated. Each of the n_i in (43) is expressed in terms of n_m as

$$n_i = n_m L_i + N_i,\ i = 1, 2, \cdots, m-1, \tag{44}$$

where $L_i = \prod_{j=i+1}^{m} f_j$ and $N_i = \sum_{p=i}^{m-1} g_p \prod_{j=i+1}^{p} f_j$, and the number of neurons (n_m) assigned to processor p_m is

$$n_m = \frac{N^1 - X(m)}{Y(m)}, \tag{45}$$

where

$$X(m) = \sum_{i=1}^{m-1} \sum_{p=i}^{m-1} g_p \left(\prod_{j=i+1}^{p} f_j \right)$$

$$Y(m) = 1 + \sum_{i=1}^{m-1} \prod_{j=i+1}^{m} f_j.$$

Equation (45) shows that there exists an optimal number of processors (m^*) beyond which the training time will not decrease with increase in the number of processors. The necessary and sufficient condition for optimal training time using all the m processors in the network is given by

$$X(m) = \sum_{i=1}^{m-1} \sum_{p=i}^{m-1} g_p \left(\prod_{j=i+1}^{p} f_j \right) < N^1.$$

From (44) and (45), we can easily calculate the number of hidden neurons assigned to each processor. We will show the results obtained and advantage of this DLT approach through some numerical examples for easy understanding.

6.1 Numerical Example

Let us consider a two-group of workstations connected by an Ethernet, one of which is a file server. The parameters of workstation groups is shown in Table 1. The values of actual parameters are larger than the values shown in Table 1 because calculations are performed by software. To illustrate the partitioning and mapping algorithm, we consider a three layer MLP network as described in Table 2. The MLP architectures are mapped on to eight workstations (four in each group). Using the closed-form expression for partitioning the hidden neurons given in (45), the analytical training time is calculated for the heterogeneous NOWs. In case of network N_1, nine hidden neurons are assigned to each processor in the first group ($p_1 - p_4$) and 27 hidden neurons are assigned to each processor in the second group ($p_5 - p_8$). Similarly, for network N_2, two hidden neurons are assigned to each processor in first group and 10 hidden neurons are assigned to each processor in second group. The time taken to complete the training process for network N_1 and N_2 are 0.0118 and 0.0020 seconds, respectively.

7 Experimental Study

In this section, we present the performance of the proposed HP algorithm in NOWs and compare the results with the VP approach. Both the algorithms are implemented in ALPHA machines connected through Ethernet. Eight workstations are selected from group "1" for experimental studies. The values of system parameters are shown in Table 1. In order to verify the performance of both the methods, we have chosen optical character recognition problem. The hand-written numbers 0 through 9 forms the character set. Each number is represented as 16×12 (192 bits) bitmap images. Several hand-written images are used for each number to train the neural network. The training set consists of 6,690 images and the testing set consists of 3,330 images, slightly different from the images used for training.

For comparative analysis, we solve this problem using the proposed method and the VP method given in [26]. The neural network architecture (N_3) used for classifying the digital numbers is $192 \times 192 \times 10$, i.e., 192 input nodes, 192 hidden nodes, and 10 output nodes. The network is trained using both the methods. Network training is stopped if the 95 percent of patterns are classified accurately or the number of epoch is equal to 5,000. After the training process, the network is tested with 3,330 patterns. Among the 3,330 test patterns, 90 percent of the patterns are classified correctly. The analytical speed-up factors for both the algorithms are shown in the Fig. 3a. We can see that the HP scheme has better speed-up factor than the VP scheme. This is due to the fact that computation and communication overheads are more in the VP approach than in the proposed HP method. The analytical space reduction ratio is shown in the Fig. 3b. The space reduction factor is more for proposed method than for the VP scheme. The total analytical and experimental time taken for training the MLP network with different number of processors for HP and VP schemes are shown in Fig. 4. The analytical training time is always less than the experimental results because the synchronization, other workloads and overheads are not included in our formulation. As we can observe from the figure, the training time for HP scheme is less than that of the VP scheme.

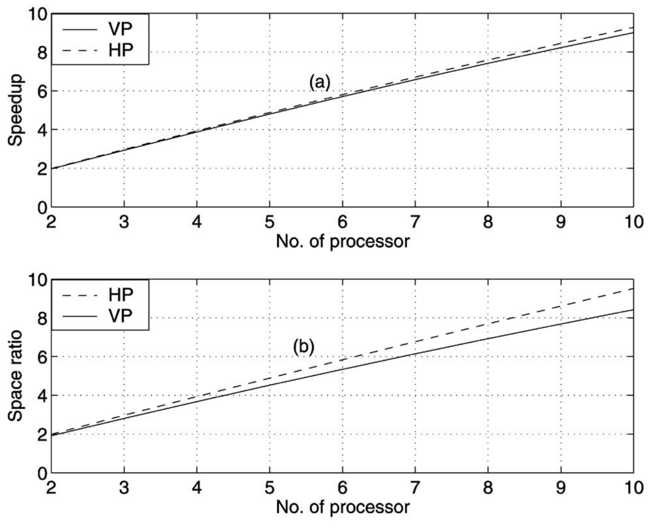

Fig. 3. Analytical results for the OCR problem.

The same experiment is carried out in heterogeneous NOWs for the HP scheme. For this purpose, four workstations from each group are selected. The mapping of network N_3 is calculated using the closed-form expression given in previous section. In mapping the network on two workstations group (four in each group), the first group of workstations assigned 13 hidden neurons each and second group assigned 37 hidden neurons each. The experimental time and analytical time taken to simulate the training process in 8 processors network are 163.48 seconds and 115.24 seconds, respectively. The difference between the analytical training time and experimental time are due to synchronization and overheads.

8 Results and Discussion

In this section, we discuss the analytical performance, such as space requirement, speed-up, and optimal number of processors for a given network configuration. We have considered two different networks, namely, N_1 and N_2, for classifying binary image of numbers as stated in [26] to evaluate the performances. The major performance factors like speed-up, storage factor, optimal number of processors, and processing time at m^* are calculated for both the algorithms. The network configuration for classifying binary numbers is shown in the Table 2. Table 3 shows the comparison between performance measures for both the algorithms. For the network N_1, the optimal number of processors for VP algorithm is less than or equal to the number of output neurons 12, whereas in the proposed HP scheme it can be any value between 1 and N^1. Optimal number of processors m^* for a given network configuration N_1 is equal to 50 in the HP approach and it is equal to 6 for the VP algorithm. At teh optimal number of processors, the processing time for proposed HP scheme is approximately 54 percent less than for the VP approach. Since the maximum number of processors is restricted to N^2 in the VP approach optimal number processors is less than or equal to N^2. A similar observation can be made for speed-up, where speed-up is heavily influenced by the communication time and extra computation is required to execute the parallel algorithm.

TABLE 3
Analytical Comparison

	Factor	VP	HP	% Saving
N_1	S(5)	4.7686	4.8774	2.2827
	$^*t_{comp}$	0.0350	0.0345	1.2707
	$^*t_{com}$	0.0007	0.0003	50.7806
	M(5)	4.3524	4.8453	10.1723
	m^*	12	50	-
	$^*T_{com}\ at\ m^*$	0.0146	0.0040	72.4474
	$^*T_p\ at\ m^*$	0.0162	0.0074	54.1589
N_2	S(5)	4.2092	4.4066	4.6907
	$^*t_{comp}$	0.0045	0.0046	2.1460
	$^*t_{com}$	0.0007	0.0003	49.8639
	M(5)	3.7333	4.5527	17.9980
	m^*	6	18	-
	$^*T_{com}\ at\ m^*$	0.0038	0.0015	59.8669
	$^*T_p\ at\ m^*$	0.0046	0.0027	40.6901

*All time in sec

We can observe from Table 3 that the proposed HP scheme has a space reduction ratio of 4.8453 in case of 5-processor network, whereas 4.3524 is the space reduction ratio for VP scheme for a given network N_1. The 10 percent less space reduction is achieved by avoiding the duplicated weight vector. The same is reflected in computation time. Since the proposed HP scheme is using single communication set, the communication time is comparatively less than that of the VP scheme. This is clear from Table 3, where the communication time in the proposed HP scheme is equal to 0.0003 for given network N_1 with $m = 5$. For the same number of processors and network, the communication time is equal to 0.0007 in the VP algorithm. The same can be observed in case

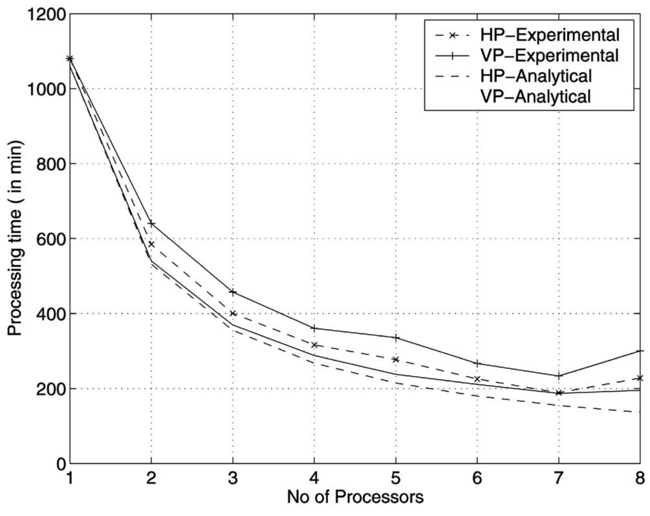

Fig. 4. Experimental training time for in NOWs.

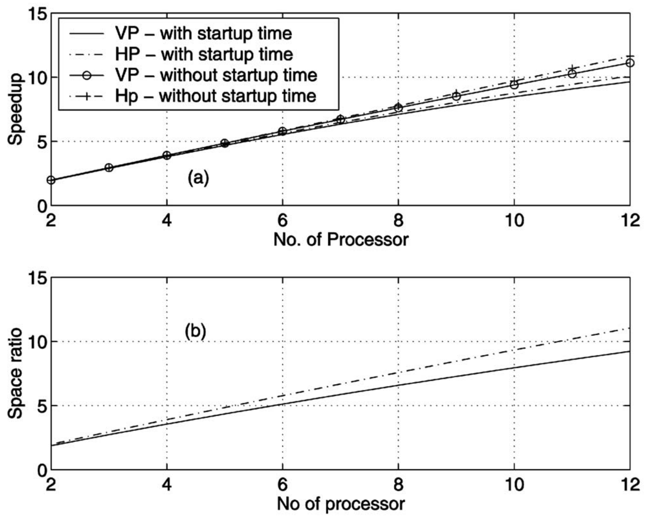

Fig. 5. Comparison of speed-up and space ratio between two algorithms for network N_1.

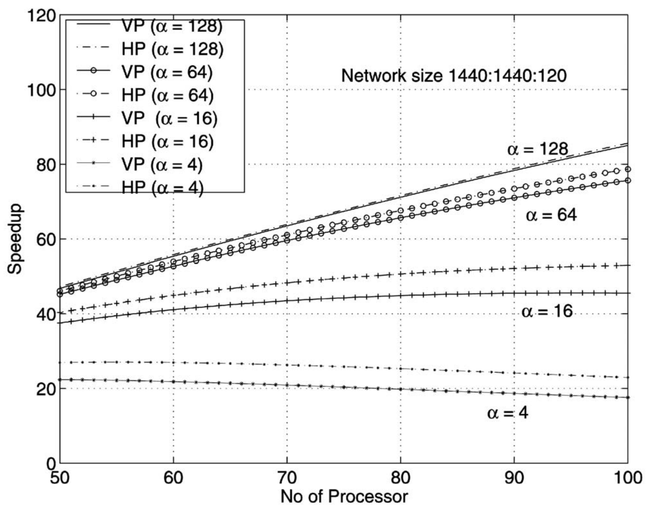

Fig. 6. Comparison between two algorithm for various α: speed-up versus number of processors.

of network N_2. From the above observations, we can say that the proposed HP scheme is better than the VP scheme for parallel implementation of BP algorithm.

Fig. 5a shows the speed-up factor (with and without start-up time) and Fig. 5b shows the space reduction factor for different number of processors for the network N_1 using both the methods. From the above figure, we can observe that the HP scheme has better speed-up and space reduction factor than the VP scheme. In order to verify the performance of both the methods, we present the analytical speed-up obtained for various values of α, as shown in Fig. 6. The figure indicates that the HP method performs better than the VP scheme.

9 Conclusion

In this paper, an optimal implementation of distributed BP algorithm to train MLP network with single hidden layer on a ALPHA NOWs is presented. Hybrid partitioning technique is proposed to partition the network. The partitioned network is mapped onto NOWs. The performance of the proposed algorithm is compared with the vertical partitioning [26]. Using the hybrid partitioning scheme, recomputation of weights is avoided and the communication time is reduced. Another advantage in the proposed scheme is that, a closed-form expression for optimal number of processors can be formulated. From this, it is possible to calculate the optimal number of processors for any given network configuration. This paper also presents a simple mapping scheme for MLP network on NOWs using divisible load scheduling concept. The mapping scheme is computationally less intensive. Analytical results for the benchmark problems and the experimental results for optical character recognition problem show that the proposed HP method is efficient and computationally less intensive than the VP scheme.

Acknowledgments

The authors would like to thank the reviewers for their valuable comments and suggestions, which has improved the quality of the presentation.

References

[1] Y.L. Murphey and Y. Luo, "Feature Extraction for a Multiple Pattern Classification Neural Network System," *Pattern Recognition Proc.,* vol. 2, pp. 220-223, 2002.

[2] M. Nikoonahad and D.C. Liu, "Medical Ultra Sound Imaging Using Neural Networks," *Electronic Letters,* vol. 2, no. 6, pp. 18-23, 1990.

[3] "Explorations in the Micro Structure of the Cognition," *Parallel and Distributed Processing,* D.E. Rumelhart and J.L. McClelland, eds. Cambridge, Mass.: MIT Press, 1986.

[4] T.J. Sejnowski and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems,* vol. 1, pp. 145-168, 1987.

[5] K.S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Network," *IEEE Trans. Neural Networks,* vol. 1, pp. 4-27, 1990.

[6] H. Yoon, J.H. Nang, and S.R. Maeng, "Parallel Simulation of Multilayered Neural Networks on Distributed-Memory Multiprocessors," *Microprocessing and Microprogramming,* vol. 29, pp. 185-195, 1990.

[7] E. Deprit, "Implementing Recurrent Back-Propagation on the Connection Machines," *Neural Network,* vol. 2, pp. 295-314, 1989.

[8] D.A. Pomerleau et al., "Neural Network Simulation at Warp Speed: How We Got 17 Million Connection Per Second," *Proc. IEEE Second Int'l Conf. Neural Networks II,* vol. 3, pp. 119-137, 1989.

[9] J. Hicklin and H. Demuth, "Modeling Neural Networks on the MPP," *Proc. Second Symp. Frontiers of Massively Parallel Computation,* pp. 39-42, 1988.

[10] J.A. Feldman et al., "Computing with Structured Connection Networks," *Comm. ACM,* vol. 31, no. 2, pp. 170-187, 1998.

[11] B.K. Mak and U. Egecloglu, "Communication Parameter Test and Parallel Backpropagation on iPSC/2 Hypercube Multiprocessor," *IEEE Frontier,* pp. 1353-1364, 1990.

[12] K. Joe, Y. Mori, and S. Miyake, "Simulation of a Large-Scale Neural Network on a Parallel Computer," *Proc. 1989 Conf. Hypercubes, Concurrent Computation Application,* pp. 1111-1118, 1989.

[13] D. Naylor and S. Jones, "A Performance Model for Multilayer Neural Networks in Linear Arrays," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 12, pp. 1322-1328, Dec. 1994.

[14] A. El-Amawy and P. Kulasinghe, "Algorithmic Mapping of Feedforward Neural Networks onto Multiple Bus Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 2, pp. 130-136, Feb. 1997.

[15] T.M. Madhyastha and D.A. Reed, "Learning to Classify Parallel Input/Output Access Patterns," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 8, pp. 802-813, Aug. 2002.

[16] N. Sundararajan and P. Saratchandran, *Parallel Architecture for Artificial Neural Networks.* IEEE CS Press, 1998.

[17] T.-P. Hong and J.-J. Lee, "A Nearly Optimal Back-Propagation Learning Algorithm on a Bus-Based Architecture," *Parallel Processing Letters,* vol. 8, no. 3, pp. 297-306, 1998.
[18] S. Mahapatra, "Mapping of Neural Network Models onto Systolic Arrays," *J. Parallel and Distributed Computing,* vol. 60, no. 6, pp. 667-689, 2000.
[19] V. Kumar, S. Shekhar, and M.B. Amin, "A Scalable Parallel Formulation of the Back-Propagation Algorithm for Hypercubes and Related Architectures," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 10, pp. 1073-1090, Oct. 1994.
[20] S.Y. Kung and J.N. Hwang, "A Unified Systolic Architecture for Artificial Neural Networks," *J. Parallel and Distributed Computing,* vol. 6, pp. 357-387, 1989.
[21] W.M. Lin, V.K. Prasanna, and K.W. Przytula, "Algorithmic Mapping of Neural Network Models onto Parallel SIMD Machines," *IEEE Trans. Computers,* vol. 40, no. 12, pp. 1390-1401, Dec. 1991.
[22] J. Ghosh and K. Hwang, "Mapping Neural Networks onto Message Passing Multicomputers," *J. Parallel and Distributed Computing,* Apr. 1989.
[23] Y. Fujimoto, N. Fukuda, and T. Akabane, "Massively Parallel Architecture for Large Scale Neural Network Simulation," *IEEE Trans. Neural Networks,* vol. 3, no. 6, pp. 876-887, 1992.
[24] V. Sundaram, "PVM: A Framework for Parallel and Distributed Computing," *Concurrency, Practice, Experience,* vol. 12, pp. 315-319, 1990.
[25] S.Y. Kung, *Digital Neural Networks.* Englewood Cliffs, N.J.: Prentice Hall, 1993.
[26] V. Sudhakar, C. Siva, and R. Murthy, "Efficient Mapping of Back-Propagation Algorithm onto a Network of Workstations," *IEEE Trans. Man, Machine, and Cybernetics—Part B: Cybernetics,* vol. 28, no. 6, pp. 841-848, 1998.
[27] D.S. Newhall and J.C. Horvath, "Analysis of Text Using a Neural Network: A Hypercube Implementation," *Proc. Conf. Hypercubes, Concurrent Computers, Applications,* pp. 1119-1122, 1989.
[28] L.C. Chu and B.W. Wah, "Optimal Mapping of Neural Network Learning on Message-Passing Multicomputers," *J. Parallel and Distributed Computing,* vol. 14, pp. 319-339, 1992.
[29] T. Leighton, *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufmann Publishers, 1992.
[30] X. Zhang and M. McKenna, "The Back-Propagation Algorithm on Grid and Hypercube Architecture," Technical Report RL90-9, Thinking Machines Corp., 1990.
[31] S.K. Foo, P. Saratchandran, and N. Sundararajan, "Application of Genetic Algorithm for Parallel Implementation of Backpropagation Neural Networks," *Proc. Int'l Symp. Intelligent Robotic Systems,* pp. 76-79, 1995.
[32] S. Haykins, *Neural Networks—A Comprehensive Foundation.* Prentice Hall Int'l, 1999.
[33] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems,* IEEE CS Press, 1996.
[34] http://www.ee.sunysb.edu/tom/dlt.html#THEORY, 2004.
[35] R. Pasquini and V. Rego, "Optimistic Parallel Simulation over a Network of Workstations," *Simulation Conf. Proc., Winter,* vol. 2, pp. 5-8, 1999.

S. Suresh received the BE degree in electrical and electronics engineering from Bharathiyar University in 1999 and the ME degree in aerospace engineering from Indian Institute of Science, Bangalore in 2001. Currently, he is working toward the PhD degree in the Department of Aerospace Engineering at the Indian Institute of Science, Bangalore. His research interests includes parallel and distributed computing, intelligence control, data mining, genetic algorithm, and neural network.

S.N. Omkar received the BE degree in mechanical engineering from the University Viswesvarayya College of Engineering in 1985, the MSc (Engg) degree in aerospace engineering from Indian Institute of Science in 1992, and the PhD degree in aerospace engineering from the Indian Institute of Science, Bangalore, in 1999. He joined the Department of Aerospace Engineering at the Indian Institute of Science, Bangalore, where he is currently a senior scientific officer. His research interest includes helicopter dynamics, neural network, fuzzy logic, and parallel computing.

V. Mani received the BE degree in civil engineering from Madurai University in 1974, the MTech degree in aeronautical engineering from the Indian Institute of Technology, Madras, in 1976, and the PhD degree in engineering from the Indian Institute of Science (IISc), Bangalore, in 1986. From 1986 to 1988, he was a research associate at the School of Computer Science, University of Windsor, Windsor, ON, Canada, and from 1989 to 1990 at the Department of Aerospace Engineering, Indian Institute of Science. Since 1990, he has been with IISc, Bangalore, where he is currently an associate professor in the Department of Aerospace Engineering. His research interests include distributed computing, queuing networks, evolutionary computing, neural computing, and mathematical modeling. He is the coauthor of the book *Scheduling Divisible Loads in Parallel and Distributed Systems* (IEEE Computer Society Press).