Computing Systems in Engineering Vol. 3, Nos 1–4, pp. 525–538, 1992
Printed in Great Britain.

0956-0521/92 $5.00 + 0.00
Pergamon Press Ltd

NEURAL NETWORKS IN STRUCTURAL ANALYSIS AND DESIGN: AN OVERVIEW

P. HAJELA† and L. BERKE‡

†Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, U.S.A.

‡Structures Division, NASA Lewis Research Center, U.S.A.

Abstract—There has been considerable recent interest in the application of neural networks in problems of structural analysis and design. The present paper provides an overview of the state-of-the-art in this emerging field, including a survey of published applications in structural engineering. Such applications have included, among others, the use of neural networks in modeling non-linear analysis of structures, as a rapid reanalysis capability in optimal design, and in developing problem parameter sensitivity of optimal solutions for use in multilevel decomposition based design. While most of the applications reported in the literature have been restricted to the use of the multilayer perceptron architecture and minor variations thereof, other network architectures have also been successfully explored, including the adaptive resonance theory network, the counterpropagation network, and the Hopfield–Tank model. Applications of these architectures have included solution of inverse analysis problems, pattern matching in structural analysis and design, and as an optimization tool for generically difficult optimization problems, in particular, those characterized as being NP-complete.

1. INTRODUCTION

An artificial neural network (ANN) can be described as a massively parallel, interconnected network of basic computing elements, that demonstrate information processing characteristics similar to several hypothesized models of the functioning of the brain. Analogous to the biological model which consists of a large number of interconnected neurons, the ANN comprises a number of similarly connected computational elements referred to as artificial neurons. Such models of computation are distinctly different from traditional numerical computing and the more recently emergent strategies of symbolic processing. There has been significant recent activity in adapting this computational model in various fields of engineering. Applications have included image processing and pattern recognition, fault detection, diagnostic systems, the use of neural networks as function approximations and formulations which allow the use of neural networks as numerical optimization algorithms.

In several cases, the ANN models adapted for such applications have architectures that are patterned after one or another model of learning and/or association that has its origin in psychology. While the basic computing element in all such architectures remains the same, the pattern of connectivity is changed to accommodate differences in the models. The network may either have feedforward characteristics only, or may have feedback loops. Likewise, networks may be fully connected in that each neuron is linked to every other neuron in the network, or the processing elements may be sparsely connected. While the feedforward and feedback architectures influence network training, the property of connectivity is intimately related to the parallelism in the system.

Subsequent sections of this paper survey a sample of applications of artificial neural networks in problems of structural analysis and design. The multilayer perceptron model is perhaps the most widely used architecture and is discussed in some detail. This model has been used most commonly in a mode where a trained network provides a mapping ϕ between some input and output quantities, X and Y, respectively. In this context, another architecture referred to as the counterpropagation network, or more specifically its improved version, has been shown to be comparably effective. This network is considerably easier to train than the multilayer perceptron model and provides an additional capability of generating inverse mappings.

A second class of network architectures examined in this review can be broadly described as self-organizing networks, and includes the ART (adaptive resonance theory) network, the Hopfield network, and the elastic net or the generalized deformable template model. The ART network has been primarily used for vector classification, and its adaptation in problems pertaining to the conceptual design of structural systems is discussed in this review. Although the Hopfield network and the elastic net can also be used in vector classification, the present paper only examines their applications in direct optimization problems. These strategies are shown

to be particularly potent in combinatorial optimization problems, such as those resulting from the presence of discrete/integer variables in the design space. Near-optimal solutions to such NP-complete problems can be generated with significantly reduced investment of computational resource.

The paper is organized into two distinct sections. After a brief discussion of the basic components of a neural network and its biological counterpart, those network architectures for which the training requires a supervised presentation of input and its corresponding output are discussed in the first section. This discussion includes details of training strategies, the computational effort required in network training and a sampling of applications in problems of structural analysis and design. The next section is devoted to a similar discussion of networks that exhibit self-organization, and emphasizes applications in generically difficult problems in optimal design.

2. NEURAL NETWORKS—BIOLOGICAL AND ARTIFICIAL

Artificial neural networks were unquestionably inspired by the impressive cognitive and data processing capabilites that are characteristic of biological neural networks. While the functioning of the brain and its intricate system of sensors is not completely understood, there is some agreement that this biological machine comprises about 100 billion threshold logic processing elements also referred to as neurons or brain cells. The number of interconnections between these neurons is estimated to be about one quadrillion. With such a high degree of complexity, it is not surprising that the similarities between the biological and artificial systems are, at best, superficial. Nevertheless, models describing the cognitive processes that have originated in neurobiology and psychology, have been embraced by computer scientists in the development of different neural network architectures.

A simplified representation of a biological neuron and its principal components is shown in Fig. 1. The cell itself consists of a chamber partitioned by a semipermeable membrane and contains a fluid with varying concentrations of K^+ and N^+ ions. The movement of these ions across the membrane due to the electrical impulses received by the cell results in a voltage potential, and an ultimate "firing" of the neuron. This signal is then relayed to other cells to which the firing neuron may be connected. The electrical impulses are picked up by "dendrites" and the "synapses" or connections determine the strength of the signal. The stimulus is relayed to the cell by the "axon", where it may be further strengthened or inhibited. Of the brain functions patterned by various ANN architectures, learning is perhaps the most widely studied. This information is thought to be represented in a biological neural network by

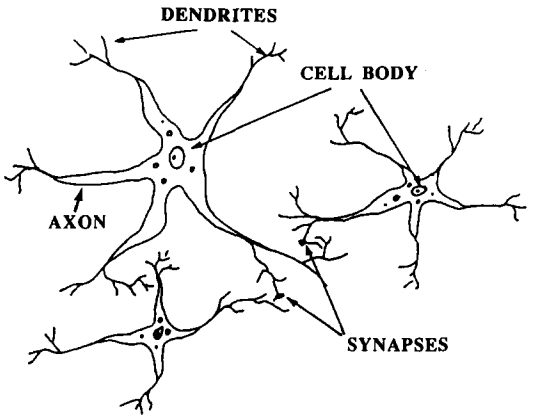

Fig. 1. Idealized representation of a biological neuron.

a pattern of synaptic connection strengths of the various neurons and the firing characteristics of the individual neurons.

It is perhaps more productive to veiw the ANN technology in much simpler terms. A computer science perspective of this field is provided in a two-volume compendium,[1] and has been referred to as parallel distributed processing (PDP). ANNs are typical of a case where a biological analogy was used as motivation for an artificial counterpart in the earlier stages of development and, once successful, the latter developed an evolutionary path of its own, departing from its biological counterpart. The artificial entity corresponding to a biological neuron is shown in Fig. 2. The processing element receives a set of signals X_i, $i = 1, 2, \ldots, n$, similar to the electrochemical signal received by a neuron in the biological model. In the simplest implementation, the modeling of synaptic connections is achieved by multiplying the input signals by connection weights w_{ij} (both positive and negative). The effective input to each processing element is therefore obtained as follows:

$$Z_j = \sum_i w_{ij} X_i.$$

In a neurobiological system, the neuron fires or produces an output signal only if the combined input stimulii to the cell builds to a threshold value. In the artificial counterpart, it is more common to process the weighted sum of the inputs by an activation function F to obtain an output signal $Y_j = F(Z_j)$. Various forms of activation functions have been proposed, typical examples of which are

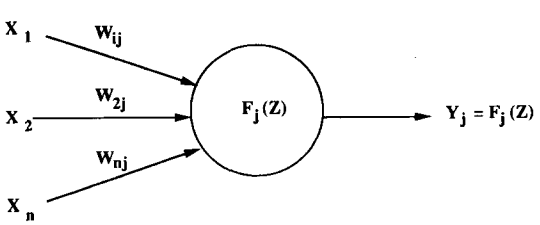

Fig. 2. A single processing element.

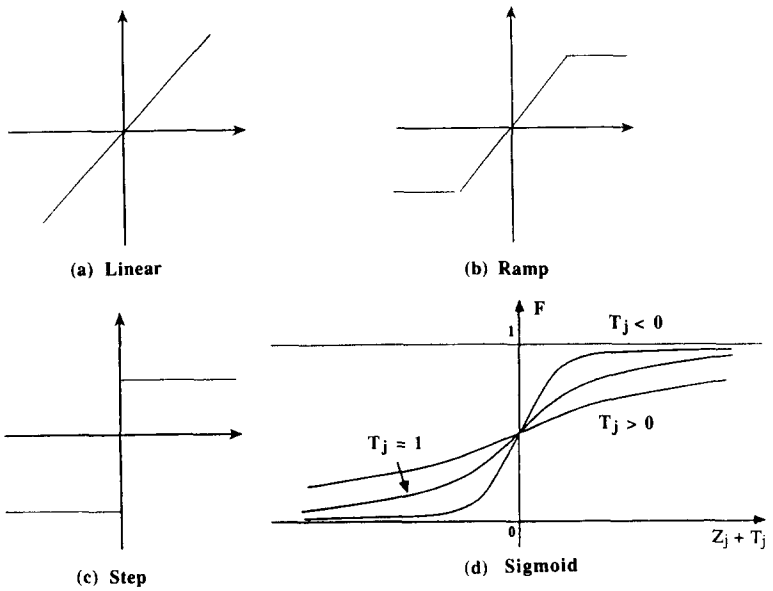

Fig. 3. Typical activation functions.

shown in Fig. 3. A sigmoid function, given by the expression

$$F(Z_j) = 1/\{(1 + \exp -(Z_j + T_j))\},$$

is perhaps one that is most widely used. Here, T_j is a bias parameter used to modulate the element output. The principal advantage of this function is its ability to handle both large and small input signals. The slope of the function is representative of the available gain. For both large positive and negative values of the input signal, the gain is vanishingly small; at intermediate values of the input signal, the gain is finite. Hence, an appropriate level of gain is obtained for a wide range of input signals. The output obtained from the activation function may be treated as an input to other neurons in the network.

In principle, artificial neurons can be interconnected in any arbitrary manner. In practice, however, these architectures are driven to a large extent by some analogous neurobiological model. One classification is based on whether the flow of the stimuli is from the input to the output nodes only, or if neurons can also relay stimuli backwards to activate or inhibit neuron firings. Feedforward networks, characterized by the flow of stimuli in one direction only, will be discussed under the category of networks where the training is supervised. The multilayer perceptron model and the counterpropagation networks are specific architectures to be explored. The second class of networks where information is permitted to flow backward as well, is the recurrent networks, and will be discussed in the second major section on self-organization in neural networks. Here, the Hopfield and elastic nets and the ART models will be examined.

2.1. *Supervised learning in ANNs*

The simple feedforward networks have a layer of neurons to which the external stimuli are presented, a series of hidden layers, and a layer of neurons at which the output is available. The input neurons do not process the input stimulus; they simply serve as "fan-out" points for connections to neurons in successive layers. The presence of the hidden layer and the non-linear activation functions enhance the ability of the networks to learn non-linear relationships between the presented input and output quantities. This "learning" or "training" in feedforward nets simply requires the determination of all interconnection weights w_{ij} and bias parameters T_j in the network. Once such a trained network is established, it responds to a new input within the domain of its training by producing an estimate of the output response which would otherwise be obtained from a computational or physical process. Variations of the generalized delta error back propagation algorithm have been used for this training; this scheme is essentially a special purpose steepest descent algorithm and indeed any optimization method can be used towards this end. The only concern would be the computational effort necessary for network training when the network size (number of independent network parameters) increases. Details of the backpropagation training algorithm are available in several sources,[1,2] and will not be presented in this paper. Nevertheless, a stepwise description of the training process is presented here for completeness:

a. First, a set of input–output pairs must be obtained from the real process that one is attempting to simulate. Determination of the number of such training pairs and how they should span the

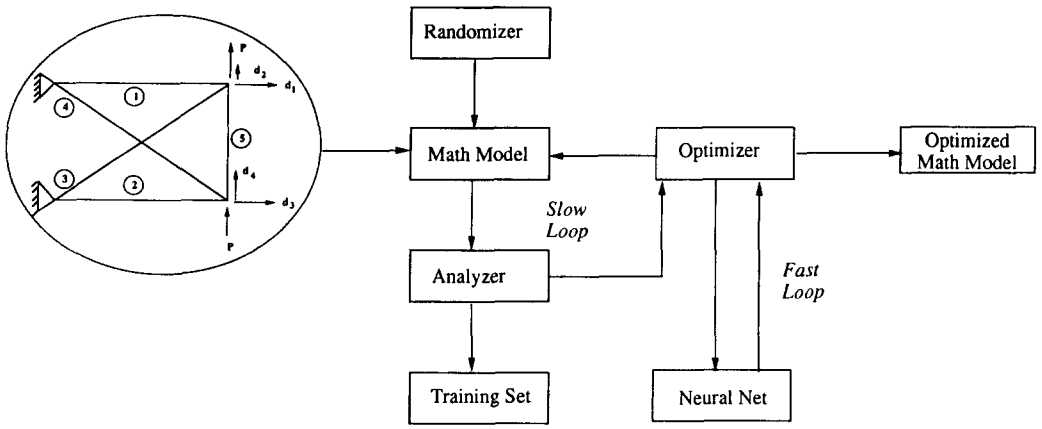

Fig. 4. Schematic illustration of use of neural networks in an optimization sequence.

intended domain of training requires experimentation and experience. The same statement is equally applicable to the selection of the network architecture.

b. The parameters of the network (w_{ij} and T_j) are first randomly selected. The input patterns are presented to the network and the network output is computed. This output is compared with the expected output and a sum of the squares of all errors is determined.
c. The network parameters are adjusted to minimize this error to some level of desired accuracy.

Some general comments can be made in reference to network training. While the number of input and output parameters is generally determined by the problem at hand, the number of hidden layers and hidden layer neurons is more difficult to establish. Wherever possible, the least number of neurons and layers should be used. In essence, this minimizes the number of design variables in the error minimization problem. The selection of the optimal number of neurons for a given data size[3] is an active area of research. Finally, the extent to which a network should be trained, i.e. the terminal value of the error for a given set of input–output patterns, is of some importance. As a general rule, if this value of error is taken much below 1%, the network becomes more of a memory device, with a lessened ability to generalize.

2.1.1. *Applications.* The network architecture discussed in the preceding section is perhaps the most widely used in problems of structural engineering and design. These applications have included, among others, the use of the networks to model non-linear structural processes, as structural engineering diagnostic systems, as a rapid reanalysis capability in optimal design, in multidisciplinary aircraft design, and as an optimal design estimation capability. Each of these applications has drawn upon the property of such networks to model a functional relationship between some input and output quantities. The use of ANNs as a rapid reanalysis capability will be presented here to illustrate this functional mapping ability. The salient aspects of other applications will also be included in the discussion.

The principal advantage of a trained neural network over the original computational process is that upon presentation of the input, the output can be generated with orders of magnitude less computational effort. Consequently, benefits can be substantial in those problem areas that are computationally very intensive, such as in numerical optimization. Note, however, that the cost of generating the training data itself must not be underestimated. The approach to simulate analysis through the rapid estimation capability of neural networks was motivated by approximation concepts in structural optimization. Both function and sensitivity information required by an optimization algorithm can be obtained from the neural network. Figure 4 describes the fundamental idea wherein a trained network is used to assist in a traditional non-linear programming based optimal search. Following a definition of the optimization problem, the selected design variables are randomly varied over their expected range of variation and the corresponding values of objective and constraint functions are obtained. These then comprise the input–output training pairs with which the network is trained.

Consider the 10-bar truss of Fig. 5 that is to be sized for minimal weight, with constraints on vertical

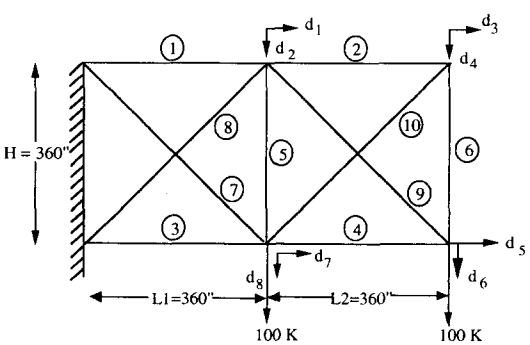

Fig. 5. A 10-bar planar truss structure.

displacements d_6 and d_8; the cross-sectional areas of the bar elements are the design variables in the problem. The presence of 10 bars requires an equal number of input nodes in the network. Similarly, since the two displacement constraints are the only output quantities of interest (weight is a linear function of cross-sectional area), the network must have two output nodes. As stated earlier, decision on the number of hidden layers and the number of neurons in each is still something of an art and is generally determined on the basis of past experience. Once the architecture definition is complete, the input–output data are presented to the network in the manner described earlier and a trained network is established. This trained network can then be used as an approximate analysis tool in lieu of exact finite element based analysis. A set of results for the 10-bar truss is presented in Table 1. Here, two hidden layers with six neurons each (see Fig. 6) were used. Numerous other example problems of both lower and higher dimensionality have been studied.[4,5] Swift and Batill have also conducted several numerical experiments in the use of neural networks in structural optimization problems.[6,7]

Experiments have also been performed to explore the idea of training a neural net to estimate optimal designs directly for a given set of design conditions and to bypass all the analyses and optimization iterations of the conventional approach. This idea draws on the concept of an "intelligent corporate memory", where several known optimal designs within some domain of variation in design conditions can be used to train a neural network. This network can be used to generalize on the basis of past experience to estimate optimal designs for changes in design conditions.

As an example, if one considers the 10-bar truss of Fig. 5 and would like to determine how the lengths L1, L2 and H affect the optimal design, it is possible to create a network which maps these three

Table 1.

	Network description	
Design variables (in^2)	(10–6–6–2) 100 training sets used in a range of $0.01 - 55.0\ in^2$ output scaled to reduce range of variation	Solution from exact analysis
X_1	30.508	30.688
X_2	0.100	0.100
X_3	26.277	23.952
X_4	11.415	15.461
X_5	0.100	0.100
X_6	0.413	0.552
X_7	5.593	8.421
X_8	21.434	20.605
X_9	22.623	20.554
X_{10}	0.100	0.100
Objective function:	5010.22 lb	5063.81 lb

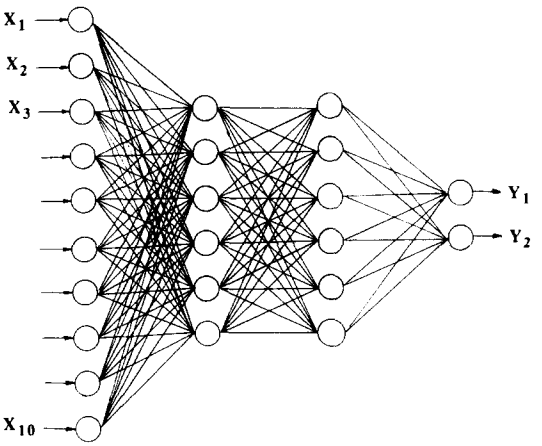

Fig. 6. Fully connected network architecture for the 10-bar truss structure.

quantities into the 10 cross-sectional areas and an optimal weight. A network with 14 neurons in the hidden layer was used to achieve this mapping.[8] For this simple problem, it was shown that 10–15 training patterns were sufficient to yield acceptable accuracy levels for such a mapping. This reference also describes larger problems where encouraging results were obtained, including an intermediate complexity wing structure and a space truss configuration.

The estimation of optimal designs for a set of design conditions is the central idea behind the computation of problem parameter sensitivities.[9] In a typical optimization effort, L1, L2 and H would be prescribed, and optimal values of the cross-sectional areas determined for minimal weight and prescribed constraints. The calculation of problem parameter sensitivities allows an estimation of new optimal solutions for changing design conditions (new values of L1, L2 and H), without actually performing a new search for this optimum. Such sensitivity calculations are critical in some multilevel decomposition strategies in optimal design. Hajela and Berke[10] demonstrated the use of neural networks in obtaining problem parameter sensitivities; this approach appears to be more robust in its estimation of optimal solutions for changed design conditions.

The multilayer perceptron network has also been used in the modeling of non-linear behavior of structural materials. Alam and Berke[11] discussed the use of this network as a non-linear stress–strain relationship, for representing material non-linearities in the analysis of truss structures. The modeling of non-linearity behavior of reinforced concrete has been similarly explored.[12] The motivating force behind these efforts is that for materials where the stress–strain behavior is not known, experimental data could be used to obtain a neural net based functional relationship. Brown *et al.* discussed the use of this network architecture to predict hygrothermal and mechanical properties of composites,[13] given the

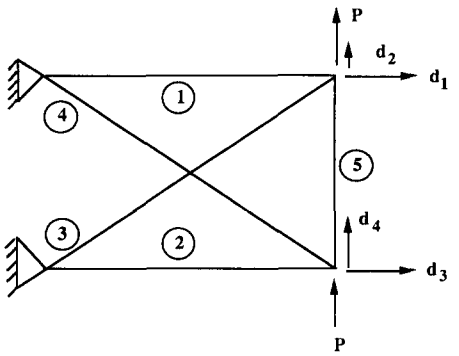

Fig. 7. A five-bar planar truss structure.

environmental conditions, constituent material properties, and volume fractions of constituent materials. Yet another application of such networks that has received attention is in the non-destructive damage assessment of structural systems.[14,15]

As a final note to this discussion on multilayer perceptron networks, it is worthwhile to emphasize the significance of the interconnection weights between the neurons. In a recent study, Hajela and Szewczyk[16] have shown that these weights can be used to establish causal relationships between input and output quantities. When a network is trained for a given set of input–output patterns, the weights can be analyzed to determine the influence of any input on a particular output quantity. As an example, consider the five-bar truss, loaded as shown in Fig. 7. If the cross-sectional areas of these bars are mapped into the five displacements, the weights can be analyzed to obtain the data shown in Table 2. The tabulated data are obtained by partitioning of weights in a manner that each row lists the normalized (to unity) influence of each cross-sectional area on a given displacement component. One clear conclusion is the lack of influence of area "a5" on any of the displacement components.

2.2. *Counterpropagation network*

The counterpropagation (CP) neural network was first introduced by Hecht-Nielsen[17] as a combination of two basic architectures—the Kohonen's self-organizing neural network and Grossberg's outstar neurons. This architecture required less computational effort to train than the multilayer perceptron architecture described in the previous section. Training times are of considerable importance when one considers modeling of extremely large structural systems. However, the original version of the network did not receive widespread attention due to its unimpressive generalization performance, particularly in comparison with the multilayer perceptron model.

Table 2. Weight analysis for the five-bar truss

	Areas				
Displacements	a1	a2	a3	a4	a5
d1	0.5130	0.1418	0.133	0.124	0.0874
d2	0.2670	0.1375	0.3925	0.1832	0.0198
d3	0.1264	0.4655	0.1779	0.2050	0.0252
d4	0.1231	0.2655	0.2588	0.330	0.0196

The idea behind the CP network, the architecture of which is shown in Fig. 8, is relatively simple. This figure shows the full version of this network, where both the input and output vectors, $\mathbf{X}$ and $\mathbf{Y}$, are presented to the input neurons, and their best approximations given as $\mathbf{X}'$ and $\mathbf{Y}'$ are available at the output Grossberg outstars. In this form, the network functions as an identity mapping; it has the property that if only a portion of the vector $\{\mathbf{X}, \mathbf{Y}\}^T$ is presented to the network, the best approximation to the complete $\{\mathbf{X}', \mathbf{Y}'\}^T$ is obtained at the output. The network can thus be used for pattern completion and for inverse mappings (i.e. $\mathbf{Y}$ presented at input and estimate of $\mathbf{X}$ obtained at output). The solid lines in Fig. 8 represent the feed forward version of the network, where the input $\mathbf{X}$ is mapped directly into an output $\mathbf{Y}$.

Each input vector presented to the network is classified as belonging to a certain cluster or category. Each neuron in the Kohonen layer represents one such cluster and the interconnection weights between the input nodes and this neuron are representative of an average of all input patterns of the cluster. Similarly, the interconnection weights between each Kohonen neuron and the output or Grossberg layer neurons are representative of an averaged output of all patterns belonging to the cluster. If the radius of each cluster is infinitesimally small, then each Kohonen neuron will only represent one pattern, and the network will be a memory record of all input–output patterns. On the other hand, if the radius of the cluster is large, then several patterns will be classified in the same cluster. A larger radius generally results in significant errors during generalization, as an input presented to the network is classified as belonging to a cluster on the basis of its similarity to the stored cluster weights for each Kohonen neuron. Even if the input pattern were

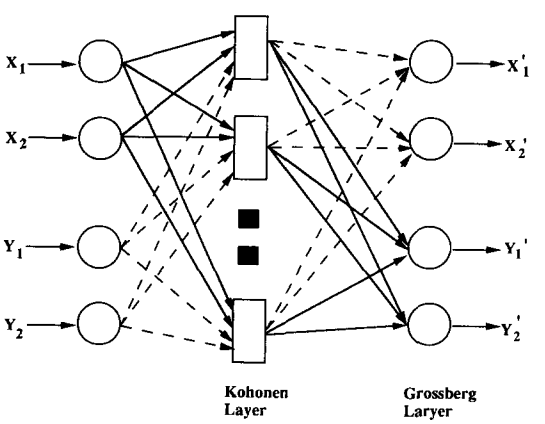

Fig. 8. Architecture for the counterpropagation network.

classified into the right cluster, the output would only be an average of the outputs of all input patterns belonging to that cluster, and that were used in network training.

More recently, a modified version of this network has been developed,[18] wherein the rapid training feature of this class of networks has been preserved and its generalization capability has been substantially improved over the original version. The size of this network, as determined by the number of Kohonen neurons, is dynamically determined by the definition of a user specified network resolution parameter.

Assume that for the training of the Kohonen layer neurons, the data set contains **M** input vectors whose distribution over the domain of interest is adequate for the mapping that the neural network is expected to represent. Additionally, the value of a network resolution parameter δ which is considered central to determining the network size, is also known. At any given stage of the training process, the algorithm increases the size of the network whenever a new input vector presented to the network is more than a prescribed distance away from all currently defined neurons. If the input vector is classified as belonging to a neuron group, the weights of that neuron are modified by a simple delta rule. This training process eventually returns the number of Kohonen neurons necessary to classify all input vectors, and their corresponding weight sets w_{ij}^{k}.

The layer of neurons to which the Kohonen layer neurons connect are referred to as the Grossberg outstars. The connection weights of these neurons are trained to produce an output approximation to the input vectors $\mathbf{X}_j$. In the approach proposed and implemented in the modified version of the network, the outstar neuron weights were set at values representing the cluster center for a given Kohonen neuron. The output from the network is simply the weight vector $\mathbf{z}_i$ for the connections between the ith neuron of the Kohonen layer and all the outstars. To this extent, the network functions like a look-up table with k entries. Since only k distinct outputs are available in this network architecture, for many applications the required accuracy of network estimates may result in thousands of Kohonen neurons. Even though this increased dimensionality network would still be relatively easy to train, its use as a look-up table would become overly cumbersome.

In an effort to provide improved estimates without an inordinate increase in network size, a fuzzification of Kohonen layer neuron outputs was introduced in the modified version. The essence of this approach was to consider a given input vector as belonging to more than one Kohonen layer neuron, albeit to different degrees. Once these contributing Kohonen layer neurons were identified, an approximation of the output was obtained as a combination of the connection weight vectors of these neurons and the Grossberg outstar neurons. This combination was facilitated by the use of a non-linear membership function. Intuitively, this non-linear averaging should yield better results as it removes the sharp boundaries between different clusters.

2.2.1. *Applications of the CP network.* The CP network has been used in a number of applications in structural engineering, including as a rapid reanalysis tool in optimal design,[19] and as an inverse mapping capability in system identification problems.[20,21] One example of this latter class of problems was the use of the CP network in structural damage assessment. The static displacement response under applied loading was used to assess the extent and location of damage in the structural components. A model of structural damage proposed by Soeiro,[22] based on stiffness reduction in structural components of truss and frame assemblies, was used in this study. In a typical example in this study, for a three-bay frame with nine beam elements (see Fig. 9), 2×10^6 discrete damage states were considered. Of these possible states, 3600 randomly generated damage patterns were used to obtain the displacement response under the applied loading. A mapping between the displacements and the structural component stiffnesses was then represented in a CP network. For a prescribed network resolution parameter, a network with 504 Kohonen neurons was established and was tested for 400 patterns of displacement that were not part of the network training. With a complete set of displacement measurements, damage estimates were established with errors ranging from 6 to 10%. Even with incomplete measurements, as would be the case in real online damage detection where all nodal displacements cannot be measured, the pattern completion characteristics of the network allowed acceptable damage diagnostic performance.

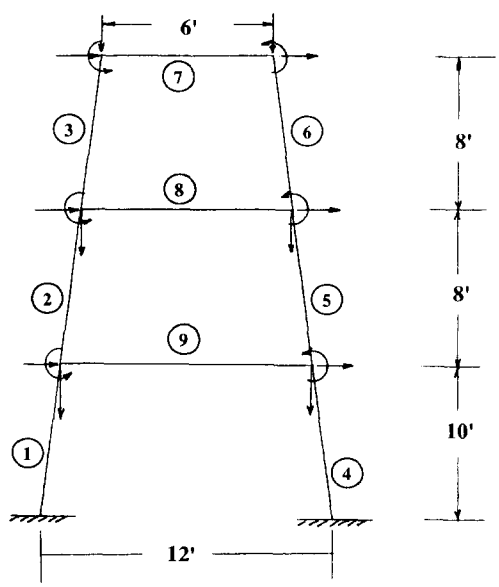

Fig. 9. A three-bay frame assembly.

The network architecture was also used in an inverse mapping mode to determine the set of physical construction parameters such as mass, stiffness, and damping,[20] to yield a desired set of eigenvalues and eigenvectors of a dynamical system. This problem has been approached through a variety of analytical and numerical techniques, each with its own difficulties and limitations. The CP network approach was tested for a three degree of freedom, spring–mass–damper system that represented an automotive suspension (Fig. 10). Five system parameters were mapped into the three complex eigenvalues (a total of six real and imaginary parts). For the 480 samples used in training, 43 Kohonen neurons were required. Tests with this trained network showed prediction of eigenvalues with average errors of 6%. This example was also used to predict parameter changes necessary to produce selective shifts in eigenvalues with extremely encouraging results.

The use of the CP network as a rapid reanalysis tool[19] was tested for a particularly computationally intensive problem in structural optimization. Space truss structures, with discrete variations in design variables, were sized for minimal weight and constraints on allowable stress levels in the members. Furthermore, a simulated-annealing based optimization approach was used for weight minimization; this selection was motivated by the ease with which discrete variables could be included in the optimal search. Of the 28 bars in the space truss shown in Fig. 11, only 15 were allowed to vary from minimum gage, and could take on any one of nine discrete values. This resulted in a design space with approximately 2×10^{14} possible design points; of these, 5000 random patterns were selected and used in the network training. A network with 1516 Kohonen neurons was established and used in the preliminary optimal search. The design space was then biased towards the preliminary design and 5000 new patterns used to generate a second network with only about 180 Kohonen neurons. The optimization based

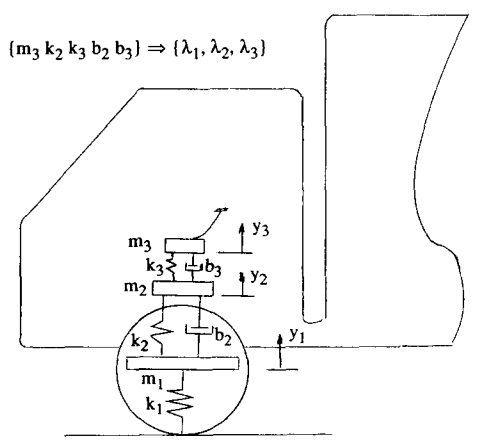

Fig. 10. Spring–mass–damper arrangement of an automotive suspension system.

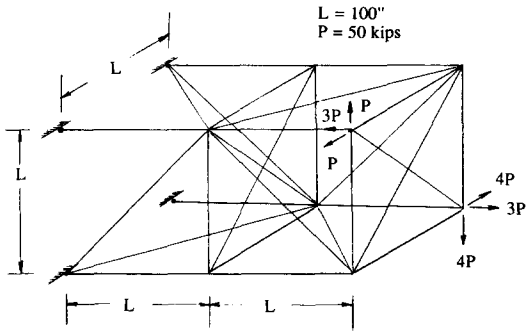

Fig. 11. A 28-bar spatial truss.

on function estimates from the CP network required less than 10% of the CPU time that was necessary for optimal design with exact analysis.

3. SELF-ORGANIZATION IN NEURAL NETWORKS

Networks that have a feedback path between the output and input neurons are described as recurrent networks. Although unconditional stability is not assured in these architectures, a more realistic modeling of the memory process can be attained. Such networks also exhibit a form of learning referred to as self-organization, such as may be required if no known output is available to train the network. The ART network and the Hopfield and elastic nets have the recurrence property and are also classified as belonging to the general category of self-organizing networks.

3.1. *ART networks*

The representation of human memory, albeit in a limited form, by an artificial neural network, has been the subject of extensive research.[23] A special feature that must be incorporated in this model of memory is that it must allow for recognition of previously encountered patterns and at the same time accommodate and store new input without destroying the existing contents. This stability–plasticity requirement has contributed to the proposition of adaptive resonance theory, or ART, networks. Variants of this architecture, distinguished on the basis of the form of input information they can accept and process (binary or continuous), have been proposed. The essence of the approach can be explained most simply by considering a network that can only process binary input patterns. Discussions in this paper will be limited to this model.

ART networks are configured to recognize invariant properties of a given problem domain; when presented with data pertinent to the domain, the networks can categorize it on the basis of these features. This process also includes the ability to create new categories when distinctly different data are presented. ART networks accommodate these requirements through interactions between different subsystems, designed to process previously encoun-

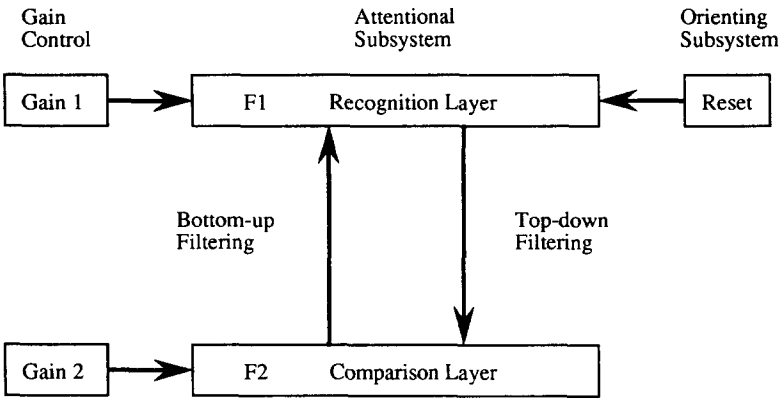

Fig. 12. Architecture of the ART network.

tered and unfamiliar events, respectively. A brief description of the different components of this network and their operation is included here for completeness.

As shown in the schematic sketch of Fig. 12, there are essentially five interacting subsystems that comprise the ART network—these include a set of two gain controls, an attentional subsystem consisting of two layers of neurons referred to as the comparison and the recognition layers, respectively, a vigilance control, and a reset control, also termed as the orienting subsystem. As stated earlier, these networks function as vector classifiers, determining if an input vector presented to the network has features that are similar to those of one of the stored patterns. The attentional subsystem is designed to process familiar events. By itself, however, it is unable to process new patterns and at the same time retain stability of existing patterns. This shortcoming is addressed by the orienting subsystem or the reset control.

The attentional subsystem consists of two layers of artificial neurons—the comparison (F1) and the recognition (F2) layers. The input pattern is first presented to the F1 layer neurons. Here, the signal is processed by the signal from Gain 1 to produce an output signal that is presented to the F2 layer neurons. Note that during the first presentation of the input to the F1 neurons, Gain 1 is set to unity and the pattern goes through unchanged. Each F2 layer neuron has associated with it a set of weights which are representative of stored patterns that have been previously classified. The neuron with weights closest to the input pattern fires in a "winner-take-all" strategy. This is enforced further by a lateral inhibition mechanism, wherein the firing of all other neurons is suppressed. The output of this activated neuron then generates a template pattern for comparison with the input pattern. During this comparison, the Gain 1 signal is set to zero if any component of the returned binary pattern is unity. A vigilance or tolerance level is assigned to this comparison, and if the two patterns are similar the input pattern is automatically classified as belonging to the category of the firing F2 neuron. Otherwise, a reset wave is generated from the orienting subsystem.

The orienting subsystem assists in the processing of unfamiliar patterns. If the returned pattern from the F2 layer is dissimilar to the input pattern as indicated by the vigilance control, the orienting subsystem generates a reset signal, whereby the fired neuron is disabled for a second round of presentation of the input patterns. In this situation, the second closest neuron in the F2 layer fires as a result and the process of similarity assessment is repeated until a match is located. If no similar category can be found, as would be true for an unfamiliar pattern, a new category is established. The gain controls Gain 1 and Gain 2 are vital for the process of training and classification. They play a central role in regulating the firing of F1 and F2 layer neurons.

Details of the network training are beyond the scope of overview. For the present development, it is sufficient to know that the dynamics of network learning are based on models found in psychology. It is also important to indicate that ART networks are indeed amenable to parallel processor implementation. Since the stored patterns must be compared with the input pattern in sequence, the bottom-up filtering which is required can be performed simultaneously on a large number of processors.

3.1.1. *ART applications in structural design.* Hajela *et al.*[24] described the use of ART networks in conceptual design of structural systems. Two distinct design processes were considered in this work; in both problems, a significant amount of prior experience in the problem domain was assumed to be available. The ART network simply classifies the input information presented to the network on the basis of identifiable critical features and suggests the use of the appropriate procedural solution approach.

The first example considers the design/analysis problem for a given structural layout. In this class of problems, the structural geometry is assumed to be known; however, the loading and support conditions can vary. For each loading or support condition, the optimal sizing of the structural system can be relegated to a distinct procedural design process. The

problem may therefore be best understood as a use of ART networks to provide a memory capacity or a knowledge base for design, from which information can be recovered upon presentation of relevant features—associative reasoning principles.

As a simple illustration of this concept, consider a problem in which the optimal cross-section of a beam is to be determined for uniform allowable strength and for minimum weight. This determination of optimal section properties is based entirely on the value of the bending moment at that section, where the latter is determined by the end supports and the type of loading that is applied. The types of load and support can thus be considered as critical features of the problem. As a simple illustration, the problem features can be represented by a four-digit binary string; these features relate to whether the beam is simply supported or clamped, whether the load is a force or moment, and whether this force or moment is concentrated or distributed uniformly over the beam. The binary strings and the associated physical process are shown in Table 3 and the 16 possible combinations of these problem features are depicted in Fig. 13.

For the comparison layer in the attentional subsystem, this problem requires a total of four nodes (one node per digit of the input pattern). Note that this discussion is confined to binary coded ART networks only. The recognition layer in this subsystem would typically have as many nodes as the number of distinct patterns presented for categorization. The results on numerical experiments with this simple problem, designed to study the influence of the vigilance parameter on the network's ability to learn and to classify the given patterns, are described in Ref. 25. As is to be expected, stricter levels of the vigilance parameter result in the recognition of minor differences in the input pattern. This finer distinction is of increased significance as the number of problem features is increased. In a modification of this example problem, the beam was divided into a

Table 3. Binary coding for the load and support conditions

A	0	The left end is a simple support
	1	The left end is a clamped support
B	0	The right end is a simple support
	1	The right end is a clamped support
C	0	The load is a force
	1	The load is a moment
D	0	The load is concentrated
	1	The load is uniformly distributed

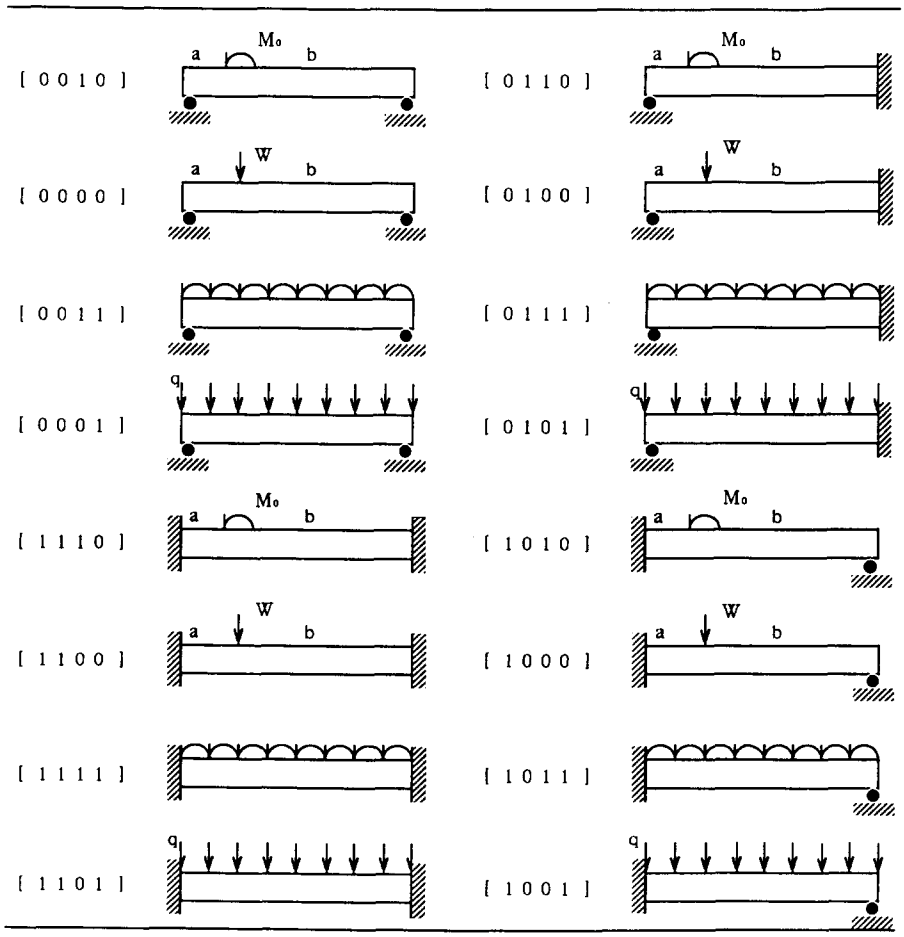

Fig. 13. Binary coding for possible load and support conditions.

number of smaller segments and the presence or absence of a load on that segment was described by a binary variable. This subdivision may be necessary as the problem definition becomes more elaborate. Another level of refinement would be to include the magnitude of loads and moments as part of the problem features.

The second class of problems is one where the loads and support points were assumed to be given, and the object of the design was to generate a near optimal structural topology. This topology generation is considered as a problem at the first level of feature abstraction and may be stated as follows:

"Given a set of external loads, their location, orientation, and magnitudes, and a set of supports and their locations, obtain a near optimal topology to transmit the loads to the available supports."

The characteristic features of the problem that are considered in making the design decisions are first identified. As an example, consider a problem where a structural topology must by generated from cable, truss and beam elements. One set of assumptions for this problem may be summarized as follows:

1. The structural topology can either be established for equilibrium under the given system of loads and supports, or structural system/component stability be required in addition.
2. Weight of elements used in the design may be ordered as cable < truss < beam.
3. Element type is of greater significance in "optimizing" topology than element length.
4. Element weight is proportional to element length alone.
5. Only simple supports can be used to stabilize the load systems.
6. The problem is two-dimensional.

A distinct procedural process can be identified for this set of assumptions and problem features.[25]

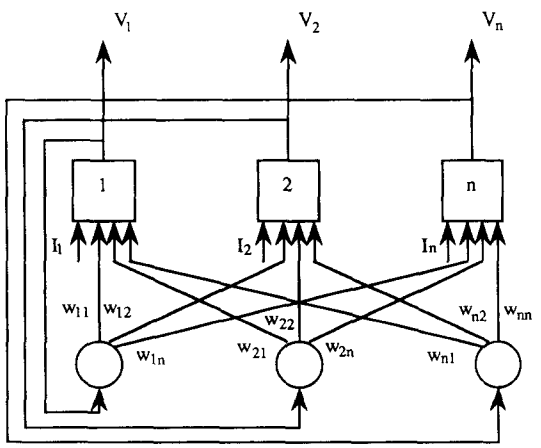

Fig. 14. Architecture of the Hopfield network.

For purposes of ART implementation, one approach of coding the assumptions and problem features is as follows:

Assumption 1
only equilibrium ⇒ 1
stability also ⇒ 0

Assumption 2
element weight order cable < truss < beam ⇒ 1
otherwise ⇒ 0

Assumption 3
element type is primary consideration ⇒ 1
element length is primary consideration ⇒ 0

Assumption 4
weight is proportional to length ⇒ 1
otherwise ⇒ 0

Assumption 5
only simple supports available ⇒ 1
both simple and fixed supports available ⇒ 0

Assumption 6
two-dimensional ⇒ 1
three-dimensional ⇒ 0.

Binary representation of typical input patterns for this problem can be obtained from the assumption stated above. A six-digit binary string would be required for a typical pattern. The above representation can be extended to include decisions concerning element selection on the basis of load and support locations and magnitudes of load. Both of these features are available as real-valued numerical data and can be converted to binary form for use in the present ART implementation. Alternatively, ART networks capable of accepting real-valued input patterns can be used. In either case, with an increase in the number of problem features, the number of categories for classification also increases. When the boundaries between the features are not too sharp, a pattern for which no exact match is available can be classified on the basis of a "best-match" fit.

3.2. *Hopfield networks*

The Hopfield network is another widely studied network architecture where the self-organizing behavior is exhibited. A schematic sketch of this network architecture is shown in Fig. 14. There is a single layer of processing neurons and the output of each neuron is returned through a set of weighting coefficients w_{ij} to the input of all neurons in the network. Each neuron also receives a bias input I_i, as shown in the figure. For the binary Hopfield network, each neuron can have a value of 0 or 1 and the outputs of all neurons in the network describe its state. For a network with n neurons, there are 2^n states for the network, each characterized by a set of weights and bias inputs. In network training,

these connection weights and bias inputs must be established.

As is true of the ART network, the Hopfield network can also be used as an associative memory for vector classification problems in structural design. The memory is first constructed and stored in the form of interconnection weights. When a new input vector is presented for classification, the various stored states are searched to determine the one closest to the input. The dynamics of this search are derived on the basis of determining the minimum of an energy functional, where the latter is a function of the network state, the interconnection weights and the bias input. For stable network dynamics, the energy functional must be of the Lyapunov form. The input pattern to be classified is presented to the network and the latter is allowed to relax along the energy contours, settling to a state that is most similar to the presented input. Since the network follows the contours of the energy function, there is a possibility of getting trapped in a local minimum. The Boltzmann machine, a stochastic variant of the Hopfield model, and one based on the simulated annealing concept, has a better chance of locating the global optimum of the energy function.

A second application of the Hopfield network is its use as an optimization tool. Since the dynamics of the network are determined by the minimization of an energy functional, where the latter is dependent upon the state of the network, the following analogy can be made. If the design variables of an optimization problem are represented by the state of the network and the objective function (which is a function of the design variables) is considered to be equivalent to the energy functional, then the evolution of the network to a minimal energy state would yield the optimal values of the design variable. The success of the approach is dependent upon the ability to write the objective and constraint functions into the energy functional that is of the Lyapunov form. This is not a trivial problem by any stretch of the imagination.

3.2.1. *Applications in structural optimization.* The present discussion is directed towards illustrating the mapping of an optimal design problem in the framework of the Hopfield networks[25] and is restricted to a discrete variable, optimal assignment problem. Consider a typical truss assembly that may be assumed to consist of a few groups of equal length members. In each group of elements and joints, there may exist errors in lengths and diameters, where such errors are typically small in relation to nominal values of these dimensions. Further, it is assumed that the member lengths and joint diameters for a given truss can be determined precisely and that errors in joint diameters can be represented as equivalent member length errors. For the truss assembly shown in Fig. 15, the problem is simplified further in that the nominal lengths of all structural elements are the same.

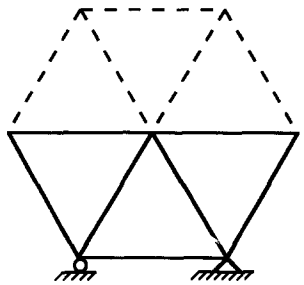

Fig. 15. A planar 7/12-bar truss structure.

This problem is essentially one of optimal assignment, where a particular member is assigned to a special position so as to reduce the overall shape distortion and to minimize the member pre-loads. To map this problem into the energy space corresponding to a Hopfield network, a variable V_{xi} is defined, the value of which is unity when member x is assigned to position i, and zero otherwise. If each neuron (binary) in the Hopfield network is assigned to one design variable, the state of the network describes the design variable set at any stage of the network evolution. As shown in Ref. 26, the energy functional of the Hopfield network can be expressed in terms of the neuron outputs and the interconnection weights in the desired Lyapunov form. This energy functional contains four appropriately weighted terms, one related to the total distortion produced from errors in element lengths and three penalty terms associated with violation of constraints that each member is uniquely assigned to one position in the truss structure.

Note that this assignment problem is NP-complete and that the computational effort necessary to generate an optimal solution increases exponentially with an increase in the number of structural members to be assigned. For a relatively small structure with seven elements as indicated by the bold lines in Fig. 15, the number of possible assignments are 5040; for the full 12-bar structure, this number increases to 479,001,600. The Hopfield network was able to generate near-optimal assignments for these problems, requiring only a fraction of the effort required by exhaustive enumeration.

This approach has also been used[26] as an optimization tool to perform node numbering in a finite element analysis to minimize the bandwidth of the resulting stiffness matrix. The energy function used in this formulation was similar to that of the previous problem. It was defined in terms of a variable V_{ij}, the value of which was one when node i was in the jth place in the numbering scheme, and zero otherwise. It included a weighted sum of the constraints and the objective of requiring a minimal bandwidth. Good results were reported for relatively small problems; however, even for these problems, the network behavior was shown to be extremely sensitive to the choice weighting constants used in the definition of the energy functional.

3.3. *Elastic nets and deformable templates*

This network is a further illustration of the self-organizing behavior in neural networks, with an architecture that is very similar to the recurrent Hopfield network. The approach has applications in optimal assignment and scheduling problems, in integer and discrete variable problems in structural optimization, and due to its strong geometrical flavor could also be extended to shape optimization problems. The concept behind elastic nets and deformable template models is similar; here, the deformable template model will be discussed as an exemplar. In this approach a template, the shape of which is controlled by some parameters, is first defined. The template and its parameters are obviously chosen to provide the template the necessary degrees of freedom to deform to an expected final form. An energy function is formulated in terms of these parameters and the latter are adjusted to minimize this function.[27] The essence of the method is described here with reference to the classical quadratic assignment problem.

In the quadratic assignment problem, one is given a set of locations x_s, $s = 1, 2, \ldots, n$, and hypothetical factory locations y_i, $i = 1, 2, \ldots, n$, and the problem is one of assigning the factories to the cities in a manner which minimizes a predefined cost function. In order to apply the deformable template approach, the QAP is formulated as a matching problem, where an objective function to be minimized is defined in terms of a matching variable V_{is}, and the coordinates of factory locations y_i. The binary matching variable V_{is} takes on a value unity when factory i is placed at locations, and is zero otherwise. The terms of the objective function include an initial cost related to assigning a factory to a location, costs incurred during manufacturing and related to an exchange of material between the factories, and a penalty term ensuring that the cities and factories are collocated during the assignment. As shown in Ref. 27, the assumption of a Gibbs distribution for y_i and V_{is} results in the elimination of the latter from the cost function. The resulting cost function is analogous to an elastic energy function, the value of which changes with variations in factory locations, y_i.

A deformable template can be defined in terms of the geometry coordinates and adapt to a shape that results in a minimization of the energy functional. This approach results, rather elegantly, in a recursive equation for changing the position coordinates that is similar in form to the equation describing the dynamics of a Hopfield network. Near-optimal solutions to moderate sized problems are presented in Ref. 27.

4. CLOSING REMARKS

The present paper provides a broad overview of neural computing applications in problems of structural analysis and design. Neural networks which require a supervised training approach were first discussed, with special emphasis on their applications in modeling functional relationships between some input and output quantities. Both the widely used multilayer perceptron architecture and a more recently emergent improved CP network were discussed in this context. The latter architecture was also shown to be effective in inverse mapping problems. Neural network models that exhibit self-organizing behavior were then discussed. Of these, the ART and Hopfield networks can both be used for vector classification and examples of this type of problem in structural design were presented with reference to the ART model. Applications of the Hopfield network and elastic nets in direct optimization problems were also presented. Both examples discussed are representative of combinatorial optimization problems, a class of generically difficult problems encountered in structural design. Although neural computing models are applicable to such problems, they do not necessarily provide a general solution procedure, as the mapping of an optimization problem to the network domain is not trivial. However, the computational advantages available from a distributed processing implementation of these methods cannot be ignored in light of the computational requirements of combinatorial optimization problems.

Acknowledgement—The first author would like to acknowledge support received under research grants NAG 3-1196 and NAG 1-1269 from the National Aeronautics and Space Administration.

REFERENCES

1. D. E. Rumelhart and J. L. McClelland J. L., *Parallel Distributed Processing*, Vols 1 and 2, The MIT Press, Cambridge, Massachusetts, 1988.
2. P. D. Wasserman, *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, New York, 1989.
3. S. Geman, E. Bienenstock and R. Dourstat, "Neural networks and the bias/variance dilemma," *Neural Computation* **4**, 1–58 (1992).
4. P. Hajela and L. Berke, "Neurobiological computational models in structural analysis and design," *Computers and Structures* **41** (4), 657–667 (1991).
5. L. Berke and P. Hajela, "Application of artificial neural networks in structural mechanics," *Journal of Structural Optimization* **3** (1), 90–98 (1992).
6. R. Swift and S. Batill, "Application of neural networks to preliminary structural design," AIAA Paper No. 91-1038, Proceedings of the 32nd AIAA/ASME/ASCE/AHS/ASC SDM Meeting, Baltimore, Maryland, April 1991.
7. R. Swift and S. Batill, "A recursive learning approach for neural networks in preliminary structural design," Proceedings of the 28th Society of Engineering Science Meeting, Gainesville, Florida, 1991.
8. L. Berke and P. Hajela, *Applications of Neural Networks in Structural Optimization* (edited by G. Rozvany), Springer, Berlin, 1992.
9. J. Sobieski-Sobieszczanski, J.-F. Barthelemy and K. M. Riley, "Sensitivity of optimum solutions of

problem parameters," *AIAA Journal* **20** (9), 1291–1299 (1982).
10. P. Hajela and L. Berke, "Neural network based decomposition in optimal structural synthesis," *Computing Systems in Engineering* **2** (4), 473–481 (1991).
11. J. Alam and L. Berke, "Application of artificial neural networks in nonlinear analysis of trusses," NASA TM, to be published.
12. J. Ghaboussi, J. H. Garrett, Jr and X. Wu, "Knowledge-based modeling of material behavior with neural networks," *Journal of Engineering Mechanics* **117** (1), 132–153 (1991).
13. D. A. Brown, P. L. N. Murthy and L. Berke, "Computational simulation of composite ply micromechanics using artificial neural networks," *Microcomputers in Civil Engineering* **6**, 87–97 (1991).
14. M. R. Ramirez and D. Arghya, "A faster learning algorithm for back-propagation neural networks in NDE applications," Proceedings of the International Conference on AI and Civil Engineering (edited by B. H. V. Topping), pp. 275–284, Civil Comp Press, 1991.
15. P. Hajela and Y. Teboub, "A neural network based damage analysis of smart composite beams," Proceedings of the 4th AIAA/NASA/Air Force Symposium on Multidisciplinary Analysis and Optimization, Cleveland, Ohio, September 1992.
16. P. Hajela and Z. Szewczyk, "On the use of neural network interconnection weights in multidisciplinary design," Proceedings of the 4th AIAA/NASA/Air Force Symposium on Multidisciplinary Analysis and Optimization, Cleveland, Ohio, September 1992.
17. R. Hecht-Nielsen, "Counterpropagation networks," *Journal of Applied Optics* **26**, 4979–4984, 1987.
18. Z. Szewczyk and P. Hajela, "Feature sensitive neural networks in structural response estimation," Proceedings of the ANNIE'92, Artificial Neural Networks in Engineering Conference, November 1992.
19. Z. Szewczyk and P. Hajela, "Neural network approximations in a simulated annealing based optimal structural design," Proceedings of the 28th Society of Engineering Science Meeting, Gainesville, Florida, 1991.
20. Z. Szewczyk and P. Hajela, "Neural network based selection of dynamic system parameters," Proceedings of the CSME 1992 Forum, Montreal, Canada, 1–5 June 1992.
21. Z. Szewczyk and P. Hajela, "Neural network based damage detection in structures," Proceedings of the ASCE 8th Computing in Civil Engineering Conference, Dallas, Texas, 6–8 June 1992.
22. F. J. Soeiro, "Structural damage assessment using identification techniques," Ph.D. Dissertation, University of Florida, 1990.
23. G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *Computer* **21**, 77–88 (1988).
24. P. Hajela, B. Fu and L. Berke, "ART networks in automated conceptual design of structural systems," Proceedings of the 2nd International Conference on AI in Civil and Sructural Engineering, Oxford, AI & SE, pp. 263–274, 1991.
25. B. Fu and P. Hajela, "Minimizing distortion in truss structures: a Hopfield network solution," Proceedings of the 33rd AIAA/ASME/ASCE/AHS/ASC SDM Conference, Dallas, Texas, 13–15 April 1992.
26. M. M. Hakim and J. H. Garrett, Jr, "A neural network approach for solving the minimum bandwidth problem," Proceedings of the 28th Society of Engineering Science Meeting, Gainesville, Florida, 1991.
27. B. Fu and P. Hajela, "The quadratic assignment problem using a generalized deformable model," *Journal of Engineering Optimization*, submitted.