Neural Networks for Reliability-Based Optimal Design

Ming J Zuo, Zhigang Tian

Department of Mechanical Engineering, University of Alberta, Canada

Hong-Zhong Huang

School of Mechanical and Electronic Engineering, University of Electronic Science and Technology of China, P.R. China

7.1 Introduction

7.1.1 Reliability-based Optimal Design

Today's engineering systems are sophisticated in design and powerful in function. Examples of such systems include airplanes, space shuttles, telecommunication networks, robots, and manufacturing facilities. Critical measures of performance of these systems include reliability, cost, and weight. Optimal system design aims to optimize such performance measures.

The traditional system reliability theory assumes that a system and its components may only experience one of two possible states: working or failed. As a result, we call it binary reliability theory. Under the binary assumption, the reliability of a system is defined to be the probability that the system will perform its functions satisfactorily for a certain period of time under specified conditions. The reliability of a system depends on the reliabilities of the constituent components and the configuration of the system. A design of a system provides a specification of the reliabilities of the components and the system configuration. In optimal system design, one aims to find the best design that optimizes various measures of performance of the system.

M.J. Zuo et al.: *Neural Networks for Reliability-Based Optimal Design*, Computational Intelligence in Reliability Engineering (SCI) **40**, 175–196 (2007)
www.springerlink.com

One of the most studied system configurations in the literature is the series-parallel system configuration. A series-parallel system consists of N subsystems connected in series such that the system works if and only if all the subsystems work wherein subsystem i ($1 \leqslant i \leqslant N$) consists of n_i components connected in parallel such that the subsystem fails if and only if all the components in this subsystem fail. Fig. 1 shows such a series-parallel configuration. The reliability of such a series-parallel system is expressed as:

$$R_s = \prod_{i=1}^{N} \left(1 - \prod_{j=1}^{n_i} (1 - p_{ij}) \right), \tag{1}$$

where p_{ij} is the reliability of component j in subsystem i. For such a system, a typical optimization problem involves finding the number of parallel components in each subsystem to maximize system reliability subject to constraints on budget, volume, and/or weight. Requirement on system reliability may be treated as a constraint while one of the constraints may be treated as the objective function to be maximized or minimized. It is a nonlinear programming problem involving integer variables. Since the focus is on finding the optimal redundancy level in each subsystem, such an optimal design problem is also referred to as a redundancy allocation problem.

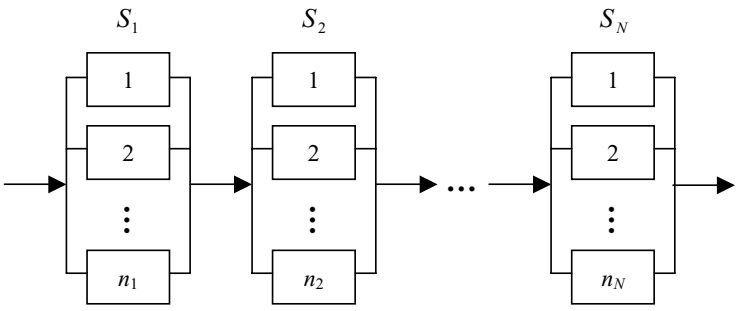

Fig. 1. Structure of a series-parallel system

Many variations of the redundancy allocation problem have been studied in the literature. The design variables may include the number of redundant components in each subsystem, the reliability value of each component, and the selection of component versions that are available on the market. The redundancy structure in each subsystem may be in the form of k-out-of-n (Coit and Smith 1996) or in the form of standby (Zhao and Liu

2004). The constituent components have also been modeled as being multi-state (Liu et al 2003) or having fuzzy lifetimes (Zhao and Liu 2004).

Another well studied system configuration in the literature is a general network configuration. A network consists of nodes and links. One is interested in determining what links should be present between pairs of nodes. The measure of performance of the network to be optimized may be cost, two-terminal reliability, or all-terminal reliability (AboElFotoh and Al-Sumait 2001 and Srivaree-ratana et al 2002). Such optimization problems are non-linear integer programming problems.

7.1.2 Challenges in Reliability-based Optimal Design

A major challenge in reliability based optimal design problems is the evaluation of system reliability given a system design. This is a time-consuming task for large systems. In optimal system design, system reliability has to be evaluated frequently for each candidate design. Thus, efficient algorithms for system reliability evaluation are essential for solving these problems.

To search for optimal solutions of reliability-based optimal design problems, efficient optimization algorithms are needed. Kuo et al. (2001) surveyed and classified optimization techniques for solving redundancy allocation problems. They compared the pros and cons of the following classical optimization techniques: integer programming, transforming nonlinear to linear functions, dynamic programming, the sequential unconstrained minimization technique (SUMT), the generalized reduced gradient method (GRG), the modified sequential simplex pattern search, and the generalized Lagrangian function method. Other examples of integer programming solutions to the redundancy allocation problems are presented by Misra and Sharma (1991), Gen et al. (1990), and Gen et al. (1993). In the process of searching for more efficient optimization algorithms, researchers have used artificial neural networks (ANN) as a function approximator and as an optimizer for solving all kinds of reliability based design problems.

7.1.3 Neural Networks

Neural networks consist of simple elements called neurons operating in parallel. The structure of neural networks is inspired by biological neurological systems. According to Rojas (1996), McCulloch and Pitts introduced the first abstract model of neurons by mimicking biological neurons and Hebb presented a learning law so that a network of neurons can be

trained. The research on neural networks achieved significant progress in the 1980s. Neural network models have found many applications in the past fifteen years.

Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision and control systems (Rojas, 1996). Neural networks can be trained to solve problems that are difficult for conventional computers or human beings. The advantages of neural networks include: (1) Adaptive learning: an ability to learn how to do tasks based on the data given for training or initial experience. (2) Self-organization: an neural network can create its own organization or representation of the information it receives during learning time. (3) Real time operation: neural network computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability. (4) Fault tolerance via redundant information coding: partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Two types of neural networks are most widely used in reliability-based optimal design: feed-forward neural networks as a function approximator, and Hopfield networks as an optimizer. These two types of neural networks and their applications will be discussed in details in the following sections.

7.1.4 Content of this Chapter

In this chapter, we explore the applications of artificial neural networks for solving reliability-based optimal design problems. The remaining part of this chapter is organized follows. In Section 2, we summarize the advantages of artificial neural networks that are specifically useful for solving reliability based optimal design problems. The use of ANN as a function approximator is presented in Section 3 while the use of ANN as an optimizer is given in Section 4. Section 5 provides a summary and points out future research topics in application of ANN for solving reliability based design problems.

7.2 Feed-forward Neural Networks as a Function Approximator

7.2.1 Feed-forward Neural Networks

The most widely used type of neural networks is the feed-forward neural network. The structure of a feed-forward neural network with three layers is shown in Fig. 2. It has one input layer, one hidden layer, and one output layer. A feed-forward neural network is used for nonlinear mapping. That is, based on the available data sets of input and output pairs, a neural network can be trained to model the mapping relationship between inputs and outputs.

For example, from function $y = x^2$, we generate five input/output pairs: [1, 1], [2, 4], [3, 9], [4, 16], and [5, 25]. The neural network we are using should have one neuron in the input layer and one neuron in the output layer, since there are only one input and one output. In this example, we can simply use one hidden layer with 3 hidden neurons. After training with the provided training pairs, the neural network can pretty much model the hidden mapping relationship $y = x^2$, and thus we can calculate what is the output value when the input is say 1.5.

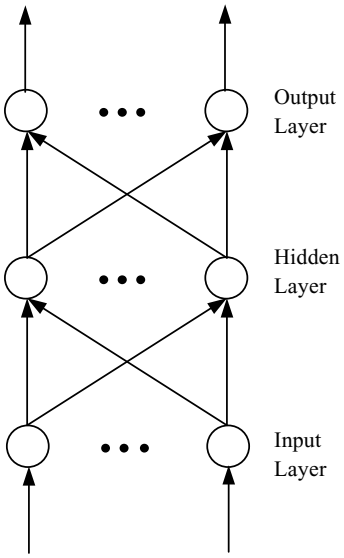

Fig. 2. Structure of a feed-forward neural network

A three-layer feed-forward neural network is capable of modeling any nonlinear mapping (Rojas, 1996). A feed-forward neural network may have more than three layers. However, too many hidden layers make the model more complex and the generalization capability of the network will become worse. Thus, feed-forward neural networks with one or two hidden layers are the most widely used ones in practical applications. Building nonlinear mapping relationship is a major advantage of feed-forward neural networks. We do not have to know the interior mechanism of the system to be modeled. As long as we have a set of input and output pairs, the feed-forward neural network can be trained to approximate the relationship between the output and the input to any specified degree of accuracy.

The function represented by a neural network model is determined largely by the connections between neurons. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between neurons (Rojas, 1996). Typically many input/output pairs are used in the process called supervised learning to train a network. The back-propagation (BP) algorithm is a widely used training algorithm for feed-forward neural networks. The BP algorithm aims at minimizing the following error function:

$$E = \frac{1}{2}\sum_{j}\left(T_j - v_j\right)^2 \tag{2}$$

where j represents all the neurons in the output layer, T_j is the desired output in the training pair, and v_j is the actual output from the current neural network. The procedure of BP algorithm is shown as follows (Fu, 1994).

The Backpropagation Algorithm

A. ***Initialization***. Set all weights and node thresholds to small random numbers. Give each neuron an index (including the input neurons).

B. ***Feed-forward calculation***. Use v_j to denote the output of neuron j. (1) The output of an input neuron is equal to the input value. (2) The output of a hidden neuron or output neuron is:

$$v_j = F\left(\sum_i w_{ji} v_i - \theta_j\right) \tag{3}$$

where w_{ji} is the weight from neuron i to j, θ_j is the threshold, and F is the so-called activation function. In a feed-forward neural network, a neuron only gets inputs from the immediately preceding layer. A commonly used activation is the sigmoid function given by: $F(x)=1/(1+e^{-x})$.

C. ***Backpropagation weight training.*** (1) Start from the output layer and work backward to the hidden layers recursively to calculate error δ_j. For output neurons:

$$\delta_j = v_j(1-v_j)(T_j - v_j) \tag{4}$$

For hidden neurons:

$$\delta_j = v_j(1-v_j)\sum_k \delta_k w_{kj} \tag{5}$$

where δ_k is the error at neuron k to which a connection points from hidden neuron j. (2) Adjust the weights as follows:

$$\begin{aligned} w_{ji}(t+1) &= w_{ji}(t) + \Delta w_{ji} \\ \Delta w_{ji} &= \eta \delta_j v_i \end{aligned} \tag{6}$$

where η is the positive scalar and called learning rate.

D. Repeat the feed-forward calculation and backpropagation weight training until the stopping criterion in terms of output errors is met.

Other techniques can be applied to improve the BP algorithm, like using the momentum terms. There are also other training algorithms based on other optimization methods, such as quasi-Newton methods and conjugate gradient methods (Rojas, 1996).

The ability of ANN to approximate a function of many variables to any degree of accuracy has been put into good use in solving reliability based optimal design problems. Coit and Smith (1996) used ANN to estimate the reliability of a series-parallel system wherein each subsystem has a k-out-of-n configuration in order to solve the optimal redundancy allocation problem. Zhao and Liu (2004) considered a series-parallel system wherein the redundancy configuration may be either parallel or standby and the lifetime of the system and that of each component is modeled as a fuzzy random variable. They used an ANN model to approximate the expected system lifetime and system reliability as a function of the redundancy levels and component lifetimes. Liu et al (2003) used ANN to approximate the expected system utility of a series-parallel system wherein the state of the system and that of each component is modeled as a continuous multi-

state random variable. Huang et al (2005) used an ANN model to represent the relationship between a designer's preference score and the performance measures such as cost, reliability, and weight of the system given a specific design as a part in their integrated interactive multi-objective optimization approach. Srivaree-ratana et al (2002) used an ANN model to represent the relationship between the all-terminal reliability of a network and the failure-prone links to be installed in the network. Papadrakakis and Lagaros (2002) used an ANN model to represent the relationship between performance measures such as stress and failure probability and the design variables in optimal design of large-scale 3-D frame structures. These uses of ANN as a function approximator will be discussed in details in this section.

In a feed-forward ANN, the number of layers, the number of neurons in each layer, and the connection weights between neurons define the structure of the ANN. Training data specify the desired relationship between output and input. Through training, a feed-forward ANN can be used to approximate any continuous function to any degree of accuracy (Cybenko 1989). This capability has been used in many reliability based optimization problems. In this section, we summarize applications of ANN as a function approximator.

7.2.2 Evaluation of System Utility of a Continuous-state Series-parallel System

Liu et al (2003) report a study on optimal redundancy allocation for a continuous-state series-parallel system. The structure of the considered multi-state series-parallel system can also be represented by Fig. 1. It consists of N subsystems, S_1 to S_N, connected in series. Each subsystem, say S_i, has n_i identical components connected in parallel. The state of each component and the system may be modeled as a continuous random variable taking values in the range of [0, 1]. The definition of a multi-state series-parallel system provided by Barlow and Wu (1978) is used here. That is, the state of a parallel system is the state of the best component in the system while the state of a series system is the state of the worst component in the system. Let x_{ij} denote the state of component j in subsystem S_i. Then, the system state can be expressed as

$$\varphi(\boldsymbol{x}) = \min_{1\le i\le N} \max_{1\le j\le n_i} x_{ij}$$

When the system is in state s, the utility of the system is denoted by $\mu(s)$. Given the state density function of each component, namely $f_{ij}(s)$, we can evaluate the state distribution of the system. With the system state distribution obtained, we can then find the expected utility of the system.

The design problem concerned is maximization of the expected system utility subject to cost constraints through determination of the optimal redundancy level in each subsystem. The optimization model is as follows:

$$\begin{aligned} &\text{Maximize :} \\ &U(n_1,n_2,\cdots,n_N) = -\int_0^1 \mu(s)\left(\frac{\mathrm{d}}{\mathrm{d}s}\left(\prod_{i=1}^{N}\left(1-\prod_{j=1}^{n_i}\left(\int_0^s f_{ij}(t)\mathrm{d}t\right)\right)\right)\right)\mathrm{d}s \\ &\text{Subject to :} \\ &\sum_{i=1}^{N}\sum_{j=1}^{n_i} C_{ij} \le C_T \end{aligned} \tag{7}$$

where C_{ij} is the cost of component j in subsystem i and C_T is the total budget allowed.

This optimization model includes both integration and differentiation in the objective function. The objective function is very complicated and the problem is very difficult to solve using a classical optimization algorithm. This situation arises when (1) the number of subsystems, N, is large; (2) the state distributions of components in the same subsystem are not identical; and/or (3) component state distribution is not a simple distribution. In addition, in some cases, the component state distribution function may have to be expressed in an empirical form and, as a result, no analytical expression of $U(n_1,n_2,\cdots,n_N)$ is available. The critical problem is that evaluating this utility function directly is very time-consuming, and thus evaluating it repetitively in the optimization process is very hard and sometimes impossible.

The system state distribution $g(s,n_1,n_2,\cdots,n_N)$ is defined as:

$$g(s,n_1,n_2,\cdots,n_N) = \prod_{i=1}^{N}\left(1-\prod_{j=1}^{n_i}\left(\int_0^s f_{ij}(t)\mathrm{d}t\right)\right) \tag{8}$$

Liu et al (2003) use feed-forward neural networks to approximate this system state distribution. They use a three-layer neural network, with sigmoidal activation functions used in the hidden layer, and linear activation functions used in the output layer. Training and testing (validation) data sets are generated as follows: first, a suitable number of input vectors, $(\mathrm{s},n_1,n_2,\cdots,n_{\mathrm{N}})$, are chosen or generated randomly from the allowed

value ranges of s, n_1, n_2, …, and n_N; next, the input vectors are normalized and for each input vector, the desired output, $g(s,n_1,n_2,\cdots,n_N)$, is calculated with equation (8). The training and testing data sets consist of different pairs of the input vector and its corresponding output.

No matter how complicated $g(s,n_1,n_2,\cdots,n_N)$ might be, the approximate analytical expression of system distribution function, $\hat{g}(s,n_1,n_2,\cdots,n_N)$, is always a linear combination of a finite number of sigmoidal functions. The approximate objective function $\hat{U}(n_1,n_2,\cdots,n_N)$ constructed from $\hat{g}(s,n_1,n_2,\cdots,n_N)$ usually has analytical expression (Liu et al 2003), and thus much less time-consuming to calculate. And the redundancy allocation of continuous-state series-parallel systems can be implemented more efficiently. The following example is given to illustrate this approach (Liu et al 2003).

Example 1

In this example, we use a 4-stage series-parallel system, in which $N = 4$; $\mu(s)=10s$; $C_{1i}=3200$; $C_{2i}=1700$; $C_{3i}=830$; $C_{4i}=2500$; and $C_T=160{,}000$. The probability density functions of components in the four subsystems are three commonly used distributions: unit distribution, triangular distribution and Beta distribution.

$f_{1i}=1$: unit distribution;

$f_{2i}=2s$: triangular distribution;

$f_{3i}(s)=\dfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}s^{\alpha-1}(1-s)^{\beta-1}$: Beta distribution where $\alpha = 2$, $\beta = 3.5$;

$f_{4i}(s)=\dfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}s^{\alpha-1}(1-s)^{\beta-1}$: Beta distribution where $\alpha = 5$, $\beta = 2$.

The density functions of the components in the 4 subsystems are plotted in Fig. 3.

The desired output targets for ANN training, i.e., actual system distribution function, are calculated with

$$g(s, n_1,n_2, n_3,n_4) = \prod_{i=1}^{4}\left(1-\prod_{j=1}^{n_i}\left(\int_0^s f_{ij}(t)\mathrm{d}t\right)\right) \tag{9}$$

$$= (1-s)^{n_1}(1-s^2)^{n_2}(1-(3.5(1-s^{4.5})-4.5(1-s^{3.5}) + 0.0635)^{n_3}(1-6s^5+5s^6)^{n_4}$$

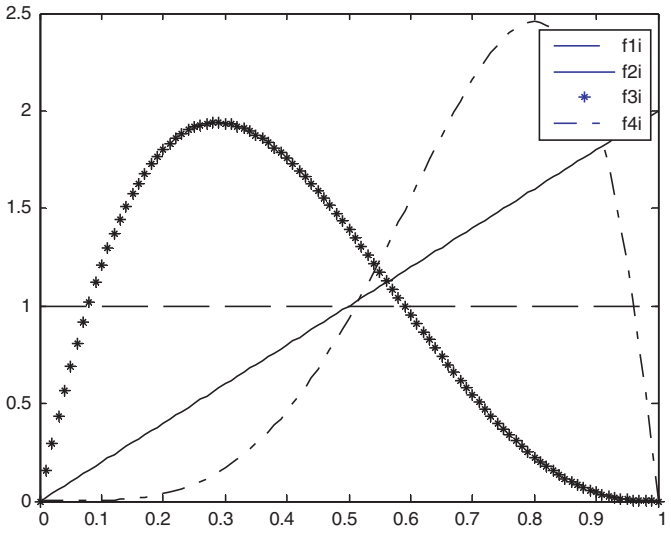

Fig. 3. Component state distribution functions for Example 1

We can see that although equation (9) is also complicated, it is calculable. But the objective function $U(n_1,n_2,n_3,n_4)$ as in Equation (7) is too complex and seems impossible to calculate analytically.

On the contrary, it is straightforward to solve this optimization problem using the ANN approximation. After training the neural network and obtaining the approximate system distribution function $\hat{g}(n_1,n_2,n_3,n_4)$, we can construct the approximate objective function $\hat{U}(n_1,n_2,n_3,n_4)$, which is an analytical expression (Liu et al 2003).

The training data set containing 1600 training pairs is generated randomly. The size of the hidden layer is 19. After 2500 epochs of training iterations, the actual percent training error is 0.9859%. The maximum validation error is 2.4532%. After an exhaustive search, the final optimal solution is

$$M_1=14;\ M_2=9;\ M_3=36;\ M_4=15$$

and the corresponding approximate utility is $\hat{U}(12,7,35,32) = 8.4384$.

7.2.3 Other Applications of Neural Networks as a Function Approximator

7.2.3.1 Reliability Evaluation of a k-out-of-n System Structure

Coit and Smith (1996) consider the optimal design problem of a series system with *k*-out-of-*n* redundancy. In such a system structure, there are *N* subsystems connected in series and each subsystem adopts a *k*-out-of-*n* structure. The optimal design model aims to find the optimal number of components n_i in subsystem i ($n_i \geq k$) such that the system cost is minimized subject to system reliability constraints. For each subsystem, several types of components are available for selection. The number of components of each available type needs to be determined so that we can find the total number of components of possibly different types for each subsystem. The decision variable values are determined through an optimization process using genetic algorithms. However, the reliability of the system for each candidate design must be evaluated. This may be a time-consuming process.

In developing the ANN model for the purpose of reliability estimation of a *k*-out-of-*n* system, Coit and Smith (1996) use full factorial design of the critical parameters *k*, *n*, and three underlying distributions (uniform, quadratic skewed-left and quadratic skewed-right) in equal proportions for component reliabilities. The skewed distributions are used to make sure that the developed ANN model is accurate for high component reliabilities or low component failure probabilities. Analytical methods are used to find the training data for the neural networks.

Since the ANN model is developed for estimation of the reliability of a *k*-out-of-*n* system structure, it actually includes the parallel redundancy as a special case. This makes the ANN model more general than for a simple parallel structure. However, caution has to be taken in assessing the error in the estimate of the reliability of the *k*-out-of-*n* subsystem. When these reliability values of the subsystems are multiplied together to get the system reliability, these errors may be magnified (Coit and Smith 1996). One needs also to take into consideration the optimizer to be used in the training of the ANN model.

7.2.3.2 Performance Evaluation of a Series-parallel System Under Fuzzy Environment

Zhao and Liu (2004) consider the problem of redundancy allocation of a series system with parallel redundancy or standby redundancy. The system has n subsystems connected in series. Subsystem i ($1 \leq i \leq n$) has n_i components either connected in parallel or connected in standby. The lifetime of each component can be represented by a fuzzy random variable. The measures of performance of the system may be the expected lifetime of the system, system reliability, or the so-called (α,β)-system lifetime (Zhao and Liu 2004).

The key for solving this optimal allocation problem is to evaluate the expected lifetime of the system, system reliability, and the so-called (α,β)-system lifetime. For a given design, a random fuzzy simulation approach was proposed by Zhao and Liu (2004) to evaluate these performance measures. Since this approach is very time-consuming, they used it to generate training data to train a feedforward ANN which will then be used to approximate these measures of performance during the design optimization process. The used ANN model has one input layer, one hidden layer, and one output layer. The number of neurons in the input layer is equal to the number of decision variables. The number of output neurons is equal to the number of performance measures of interest. The number of neurons in the hidden layer is determined by the pruning algorithm of Castellano et al (1997).

Once the ANN model is trained, it is used as a function approximator in the optimization model. Genetic algorithms are used to solve the optimization problems. Examples are used to illustrate this approach for solving redundancy allocation problems including parallel redundancy and standby redundancy.

7.2.3.3 Evaluation of All-terminal Reliability in Network Design

Srivaree-ratana et al. (2002) consider a network design problem in which the nodes are fixed and perfect while the links are failure prone. The question to be answered is what links should be installed to minimize the total cost of installing these selected links subject to requirement on all-terminal reliability of the network.

The most time-consuming task in this network design problem is the evaluation of all-terminal reliability given a network design. Approaches such as enumeration and Monte Carlo simulation can be used for evaluation or approximation of network reliability, but they are very time-consuming. Srivaree-ratana et al (2002) decide to use ANN to estimate the

all-terminal reliability as a function of a selected design. A feedforward ANN model is adopted and the backpropagation training algorithm is used. A hyperbolic activation function is used for all neurons and a learning rate of 0.3 is used for hidden neurons and of 0.15 is used for the output neurons. The total number of hidden neurons is chosen to be identical to the number of input neurons.

Experiment results provided by Srivaree-ratana et al (2002) show that the ANN model works very well for estimating all-terminal reliability. Future research topics may include fine tuning the ANN model and imbed occasional evaluation of the exact all-terminal reliability of preferred designs.

7.2.3.4 Evaluation of Stress and Failure Probability in Large-scale Structural Design

Reliability based optimal design of large-scale structural systems is extremely computation intensive. Papadrakakis and Lagaros (2002) address the optimal design of multi-storey 3-D frames. The goal is to minimize the weight of the structure subject to constraints on allowed stress, displacement, and failure probability. Due to the randomness in loads to be applied, material properties, and member geometry, evaluation of the stress, displacement, and failure probability given a structure designed is very time-consuming.

The measures such as stress, displacement, and failure probability can be evaluated using finite element method, the limit elasto-plastic method, and Monte Carlo simulations given certain design parameters. These calculated values and the corresponding design parameters can then be used as the training data set for training of a feedforward ANN which can then be used for approximation of these measures during the optimization process. The actual optimization algorithm used is the genetic algorithm. Numerical results are presented to illustrate the effectiveness of the proposed approaches.

There are more application examples of neural networks as function approximator in the literature, like the representation of the preference structure of the designer in multi-objective design optimisation (Huang et al 2005).

7.3 Hopfield Networks as an Optimizer

7.3.1 Hopfield Networks

Another widely used structure of ANN is the Hopfield network (Hopfield 1982, Hopfield and Tank 1985). In a Hopfield network, all neurons are in a single layer (Fig. 4). Every pair of neurons are connected with the same connection weights. That is

$$w_{ji} = w_{ij}, \text{ for } i \neq j$$
$$w_{ii} = 0 \tag{10}$$

Each neuron may represent a binary variable because its output may take values of 0 or 1 only. An activation function is used to map the total input to a neuron to a 0-1 output value.

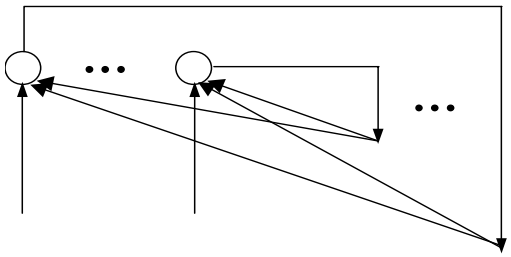

Fig. 4. Structure of a Hopfield network

Through a dynamic update equation of the input value of each neuron, the Hopfield ANN converges to a state that minimizes an energy function of the neural network. Consider a Hopfield ANN with n neurons wherein w_{ij} denotes the connection weight between neurons i and j, u_i the input to neuron i, $v_i = F(u_i)$ the output of neuron i through activation function $F(.)$, and θ_i the bias of neuron i. Then, the energy function of the ANN is given by Hopfield and Tank (1985) as:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} v_i v_j - \sum_{i=1}^{n} \theta_i v_i \tag{11}$$

The dynamics of the Hopfield ANN is defined as

$$\frac{du_i}{dt} = \sum_{j=1}^{n} w_{ij} v_j + \theta_i \tag{12}$$

Equation (12) actually represents the steepest descent direction of the energy function given in equation (11). If the input to neuron i, u_i, is updated following the direction given by equation (12), the energy function will converge to a local minimum. This is why the Hopfield ANN can be used to solve optimization problems.

A major advantage of Hopfield networks is their efficiency in solving optimization problems (Nourelfath and Nahas, 2003). Such an ANN as an optimizer was first introduced by Hopfield and Tank (1985). The concept of quantized neurons was introduced by Matsuda (1999). AboElFotoh and Al-Sumait (2001) used Hopfield networks for solving a network design problem. Nourelfath and Nahas (2003) used quantized Hopfield networks for selection of the components in a series system for system reliability maximization. These uses of ANN as an optimizer will be discussed in details in this section.

The key in the use of a Hopfield ANN for solving reliability optimization problems is in formulation of the energy function and definition of decision variables v_i. In this section, we summarize the work reported by AboElFotoh and Al-Sumait (2001) and Nourelfath and Nahas (2003) for this purpose.

7.3.2 Network Design with Hopfield ANN

AboElFotoh and Al-Sumait (2001) considered a network design problem. There are n perfect nodes in the network. The question to be answered is what links should be installed to minimize the total cost of the network subject to all-terminal reliability requirement. The reliability and the cost of each possible link are given as data.

Notation:

n	Number of nodes in the network
i, j	Network nodes
(i, j)	The link between node i and node j
$p_{i,j}$	Reliability of link (i, j)
$c_{i,j}$	Cost of link (i, j)
R_0	All-terminal reliability requirement of the network
R_S	All-terminal reliability of the network
$v_{i,j}$	Takes the value of 1 if link (i, j) is selected and 0 otherwise

A, *B*, *C* Positive constants that may be adjusted in the optimization process

The optimization model for the network design problem is

$$\begin{aligned} &\text{Minimize } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j} v_{i,j} \\ &\text{Subject to: } \quad R_S \geq R_0 \end{aligned} \tag{13}$$

To use a Hopfield ANN to solve this optimization problem, AboElFotoh and Al-Sumait (2001) use (*i*, *j*) to denote a neuron and the following to present the energy function:

$$E = -A \cdot R_S + \mathrm{B} \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{i,j} v_{i,j} + C \cdot \left| R_S - R_0 \right| \tag{14}$$

This energy function consists of three terms added together. Since we are to minimize this energy function, the first encourages network reliability maximization, the second term encourages cost minimization, and the third term discourages the ANN from adding new links to increase network reliability unnecessarily over R_0. The negative derivative of the energy function given in equation (14) with respect to each decision variable $v_{i,j}$ is given by

$$-\frac{\partial E}{\partial v_{i,j}} = \begin{cases} (A-C) \cdot \dfrac{\partial R_S}{\partial v_{i,j}} - Bc_{i,j}, & \text{if } R_S > R_0 \\ (A+C) \cdot \dfrac{\partial R_S}{\partial v_{i,j}} - Bc_{i,j} & \text{if } R_S < R_0 \end{cases}, \tag{15}$$

The update equation for the input $u_{i,j}$ is then given by

$$u_{i,j}(t+1) = u_{i,j}(t) + \frac{\partial E}{\partial v_{i,j}} \tag{16}$$

To use the above equations, one has to have an algorithm to calculate the all-terminal reliability for each given network design. Since this is an NP-hard problem, AboElFotoh and Al-Sumait (2001) provide a lower bound and upper bound on this network reliability. Either bound may be used in equation (14) to approximate the reliability of the network. The activation function used is a simple threshold function, namely

$$v_{i,j} = \begin{cases} 1, & \text{if } u_i > \text{UTP} \\ 0, & \text{if } u_i < \text{LTP}, \\ \text{unchanged}, & \text{otherwise} \end{cases} \tag{17}$$

where UTP and LTP are the threshold values.

Many network cases were generated to test the Hopfield ANN approach for this network design problem. AboElFotoh and Al-Sumait (2001) conclude that this approach is very efficient for design of large networks but does not guarantee global optimal solutions. Possible future research work include consideration of node failures, more efficient algorithm for evaluation of all-terminal network reliability, and better rules for selection of parameters of the energy function.

7.3.3 Series System Design with Quantized Hopfield ANN

Nourelfath and Nahas (2003) considered a series system with N components. For component j ($1 \le j \le N$), there are M_j choices available. These choices correspond to different costs, reliabilities, weights, and possibly other characteristics. We are interested in making a choice for each of the N components such that system reliability is maximized subject to cost and other constraints. The optimization model for this problem can be expressed as

$$\begin{aligned} \text{Maximize} \quad & R_S = \prod_{j=1}^{N} \left(\sum_{i=1}^{M_j} x_i^j R_i^j \right) \\ \text{Subject to:} \quad & \sum_{j=1}^{N} \sum_{i=1}^{M_j} C_i^j x_i^j \le B \\ & \sum_{i=1}^{M_j} x_i^j = 1, \quad \forall j = 1,2,\cdots,N \end{aligned} \tag{18}$$

where R_S is the system reliability, R_i^j is the reliability of choice i for component j, C_i^j is the cost of choice i for component j, B is the budget for the system, and x_i^j is a 0-1 variable that takes the value of 1 if choice i is selected for component j. This is a 0-1 non-linear programming problem. However, the objective function can be transformed into a linear function as follows (Nourelfath and Nahas 2003):

$$\text{Minimize} \qquad \psi = -\ln R_S = \sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j \left| \ln R_i^j \right| . \tag{19}$$

With this transformation, the optimization problem becomes a 0-1 linear programming problem.

Without loss of generality, the budget amount B is assumed to be an integer value. After introducing a slack variable t to convert the inequality constraint into an equality constraint, the optimization model becomes:

$$\begin{aligned} &\text{Minimize} \quad \psi = -\ln R_S = \sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j \left| \ln R_i^j \right| \\ &\text{Subject to} \quad \sum_{j=1}^{N} \sum_{i=1}^{M_j} C_i^j x_i^j + t = B \\ &\qquad\qquad \sum_{i=1}^{M_j} x_i^j = 1, \quad \forall j = 1,2,\cdots,N \\ &\qquad\qquad x_i^j \text{ are 0-1 variable and } t \text{ is integer.} \end{aligned} \tag{20}$$

Though the Hopfield ANN in its originally proposed form allows only 0-1 variables, the quantized Hopfield ANN developed by Matsuda (1999) can be used to deal with integer variables too. Applying this model, Nourelfath and Nahas (2003) use the following energy function of the quantized Hopfield ANN for solving the series system optimization problem:

$$\begin{aligned} E = &\frac{A_1}{2} \left(\sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j \left| \ln R_i^j \right| \right)^2 + \frac{A_2}{2} \left(\sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j C_i^j + t - B \right)^2 \\ &+ \frac{A_3}{2} \sum_{j=1}^{N} \left(\sum_{i=1}^{M_j} x_i^j - 1 \right)^2 \end{aligned} , \tag{21}$$

where A_1, A_2, and A_3 are positive parameters.

Simulation studies are conducted to test the quantized Hopfield ANN model for the series system optimization problem. The following two forms of optimization objectives other than that in equation (20) were tested as well

$$\begin{aligned} &\text{Minimize} \quad \psi = \sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j \frac{1}{\ln R_i^j}, \text{ and} \\ &\text{Minimize} \quad \psi = \sum_{j=1}^{N} \sum_{i=1}^{M_j} x_i^j \left(1 - \ln R_i^j\right), \end{aligned} \tag{22}$$

and simulation results showed that no effects occur when considering these forms of objective functions.

Nourelfath and Nahas (2003) conclude that the quantized Hopfield ANN reduces the number of neurons needed to represent the series system optimization model and as a result reduces the computation time in finding optimal solutions. Unfortunately, the quantized Hopfield ANN does not guarantee global optimal solutions either. Other possible future research topics include application of this model to solving other reliability based optimization problems.

7.4 Conclusions

Reliability based optimal design presents challenging optimization problems. These problems often involve time-consuming tasks of evaluation of various system performance measures such as reliability, expected utility, lifetime, stress, displacement, and failure probability. Neural network models have been used for the purpose of function approximation to significantly reduce the computation needs in the on-line optimization process because ANN models can be trained off-line. The Hopfield ANN model has also been used as a local optimization routine in search for optimal solutions.

When ANN is used for function approximation, the main concern is its accuracy. Usually the ANN approximation can not be 100% accurate. To have a better accuracy, a larger training sample size is required, which leads to more computation efforts. The users need to verify the accuracy of ANN approximation, and find out whether or not the accuracy is acceptable. When ANN is used as an optimizer, the main concern is its local optimisation characteristic. It is possible that the global optimum can never be reached.

Future research directions for application of ANN models in reliability based design includes improvement of the global search ability of the Hopfield neural networks, systematic selection of the parameters of the energy function of the Hopfield neural networks, combination of ANN function

approximator with occasional evaluation of the exact values of the functions being approximated, the issue of error propagation in the function approximators, and the interaction between the function approximator and the actual optimization routine used.

References

H.M.F. AboElFotoh and L.S. Al-Sumait, (2001) A neural approach to topological optimization of communication networks, with reliability constraints. IEEE Transactions on Reliability, Vol. 50, No. 4, pp. 397-408.

R.E. Barlow and A.S. Wu, (1978), Coherent systems with multi-state Components, Mathematics of Operations Research, Vol. 3, No. 4, pp. 275-281.

G. Castellano, A.M. Fanelli and M. Pelillo, (1997) An iterative pruning algorithm for feedforward neural networks, IEEE Transactions on Neural Network, Vol. 8, pp. 519–537.

G. Cybenko, (1989), Approximation by superpositions of a sigmoidal function. Mathematics of control, signals, and systems, Vol. 2, No. 4, pp. 303-314

D.W. Coit and A.E. Smith. (1996) Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. Computers & Operations Research, Vol. 23, No. 6, pp. 515-526.

L. Fu, (1994), Neural Networks in Computer Intelligence, McGraw-Hill, Inc., New York.

M. Gen, K. Ida and J. U. Lee, (1990), A computational algorithm for solving 0-1 goal programming with GUB structures and its applications for optimization problems in system reliability, Electronics and Communication in Japan: Part 3, Vol. 73, pp. 88-96.

M. Gen, K. Ida, Y. Tsujimura and C. E. Kim, (1993), Large scale 0-1 fuzzy goal programming and its application to reliability optimization problem, Computers and Industrial Engineering, Vol. 24, pp. 539-549.

J.J. Hopfield. (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America-Biological Sciences, Vol. 79, No. 8, pp. 2554-2558.

J. J. Hopfield and D.W. Tank, (1985) Neural computation of decisions in optimization problems, Biological Cybernetics, vol. 52, pp. 141–152.

H. Huang, Z. Tian, and M.J. Zuo, (2005) Intelligent interactive multi-objective optimization method and its application to reliability optimization". IIE Transactions, Vol. 37, No. 11, pp. 983-993.

W. Kuo, V. R. Prasad, F. A. Tillman and C. L. Huang, (2001), Optimal Reliability Design, Cambridge University Press, New York.

P.X. Liu, M.J. Zuo and M. Q-H Meng, (2003) A neural network approach to optimal design of continuous-state parallel-series systems. Computers and Operations Research, Vol. 30, pp. 339–352.

S. Matsuda, (1999) Quantized Hopfield networks for integer programming. Systems and computers in Japan, pp. 1354–64.

K. B. Misra and U. Sharma, (1991), An efficient approach for multiple criteria redundancy optimization problems, Microelectronics and Reliability, Vol. 31, No. 2, pp. 303-321.

M. Nourelfath and N. Nahas. (2003) Quantized hopfield networks for reliability optimization. Reliability Engineering & System Safety, Vol. 81, No. 2, pp. 191-196.

M. Papadrakakis and N.D. Lagaros. (2002) Reliability-based structural optimization using neural networks and Monte Carlo simulation. Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 32, pp. 3491-3507.

R. Rojas, (1996). Neural networks: a system introduction. Berlin: Springer.

C. Srivaree-Ratana, A. Konak and A.E. Smith. (2002) Estimation of all-terminal network reliability using an artificial neural network. Computers & Operations Research, Vol. 29, No. 7, pp.: 849-868.

R.Q. Zhao and B.D. Liu. (2004) Redundancy optimization problems with uncertainty of combining randomness and fuzziness. European Journal of Operational Research, Vol. 157, No. 3, pp. 716-735.