Evolutionary Neural Network Model for the Selection of Pavement Maintenance Strategy

MAHMOUD A. TAHA AND AWAD S. HANNA

Neural networks are attracting an enormous amount of attention in many civil engineering disciplines, including transportation, because they represent a class of robust, nonlinear models capable of learning relationships from data. However, in the development of such models for a particular application, various parameter settings are left to the judgment of the network developer. The net result of poor parameter settings will be slow convergence and/or bad performance on unseen cases. Recently, genetic algorithms have emerged as a potential searching technique to design a neural network model that performs best on a specified task according to explicit performance criteria. Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. In this paper we present a genetic algorithm method that evolves a neural network model for the selection of the optimum maintenance strategy for flexible pavements. A hybrid evolutionary-learning system using gradient descent learning as well as a genetic algorithm to determine the network connections weights is described. The developed neural network model has an input vector of seven components and an output vector of seven components. The input vector represents the factors affecting the maintenance strategy selection, whereas the output vector represents the different pavement maintenance strategies available. Brainmaker Professional, a commercially available neural network simulator, was used in the development of the neural network model. The performance of the developed neural network model was validated by testing it using 100 unseen cases. The validation results showed that the system misclassified only six cases with an average error rate of 0.024.

Decision-making in most civil engineering disciplines, including transportation, frequently encounters complicated and unstructured problems for which solutions are devised based on analogy with previous cases with a mixture of intuition and experience. Selection of the appropriate pavement maintenance strategy represents one such problem. The ability of decision makers to find an adequate solution to this problem depends primarily on their accumulated experience. To lessen the dependency on experienced personnel and to improve the consistency of the decision-making process, decision-making aids are required. Neural networks have been recommended by many researchers as a suitable tool for developing such decision aids (*1*,*2*). This is attributable to their ability to learn mappings from a set of inputs to a set of outputs based on training examples and to generalize beyond the examples learned. However, experience with neural networks for learning different tasks has demonstrated the difficulty of selecting an appropriate functional structure for a network as well as appropriate values for learning rule parameters. This bottleneck may seriously impair neural networks progress in the coming years if it is left unaddressed.

M. A. Taha, Construction Administration Program, University of Wisconsin-Madison, 460 Henry Mall Street, Madison, Wis. 53706. A. S. Hanna, Department of Civil and Environmental Engineering, University of Wisconsin-Madison, 1415 Johnson Drive, Madison, Wis. 53706.

Recently, genetic algorithms (GAs) have emerged as a potential searching technique to craft a neural network application that performs best on a specified task according to certain explicit performance criteria. According to Austin (*3*), the GAs can be defined as:

> [A]n iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges. During each temporal increment (known as generation), the structures in the current population are rated for their effectiveness as domain solutions, and on the basis of these evaluations, a new population of candidate solutions is formed using specific "genetic operators" such as reproduction, crossover, and mutation.

In this paper we present a genetic algorithm approach that evolves a neural network model for the selection of the optimum maintenance strategy for flexible pavements. The backpropagation learning method (*4*) is combined with genetic algorithms to evolve the optimum interconnection weights. BrainMaker Professional, a commercially available neural network simulator, was used in the development of the neural network model.

THE PROBLEM OF NEURAL NETWORKS DESIGN

The process of developing a neural network model for a particular application usually involves four basic stages. First, a network developer selects a problem domain, such as pavement maintenance, based on his or her theoretical, empirical, or applied interests. Next, a network architecture is designed for capturing the underlying criteria from the problem domain. This architecture forms the configuration of the network including the number of units used, their organization into layers, learning parameters, and error tolerance. Third, given this chosen architecture and a chosen task, a learning paradigm is applied to train the network and develop the interconnection weights. Finally, the developer evaluates the trained network according to objective performance measures such as its ability to solve the specified task and its ability to predict the outcome of unseen cases.

Learning takes place in neural networks by adjusting the connection weights between simple processing units. This kind of computation is best understood as a kind of relaxation system (*4*). Most learning procedures perform a search over the weight space to minimize some performance function of the network. The "Boltzmann Machine," a widely used learning technique, uses simulated annealing. Unfortunately, simulated annealing is very slow. A much faster learning procedure is backpropagation. This method is getting the most attention in the neural network research community. However, this method also has limitations such as slow training and the possibility of getting a solution that is a local minimum, among others. This paper proposes an alternative technique to adjust the network's weights by using a biologically based optimization method.

MERGING GENETIC ALGORITHMS WITH NEURAL NETWORKS

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics (*5*). They imitate nature with their Darwinian survival-of-the-fittest approach. This approach allows genetic algorithms to speculate on new points in the search space with expected improved performance by exploiting historical information. A simple genetic algorithm responds to some function evaluation. It does not associate an output with an input (*6*). Neural networks, on the other hand, are good at associating different patterns once they have developed an internal representation. The problem with neural networks is developing the "proper" internal representation among the connections. Because GAs are best viewed as a function optimizer, it is suited to finding the set of weights that allow the neural network to solve a given problem.

The advantages of using GAs to evolve the neural network weights are twofold. First, GAs represent a global search method. Backpropagation, which is the most widely used search method to develop the network internal representations, is a local optimization method (*7*). Backpropagation involves a gradient descent in sum-squared error that minimizes the squares of the differences between the actual and the desired output. Second, GAs may be capable of much faster optimization. The procedures for evolving an optimum neural network's set of weights for a particular application are outlined in Figure 1 (*8*).

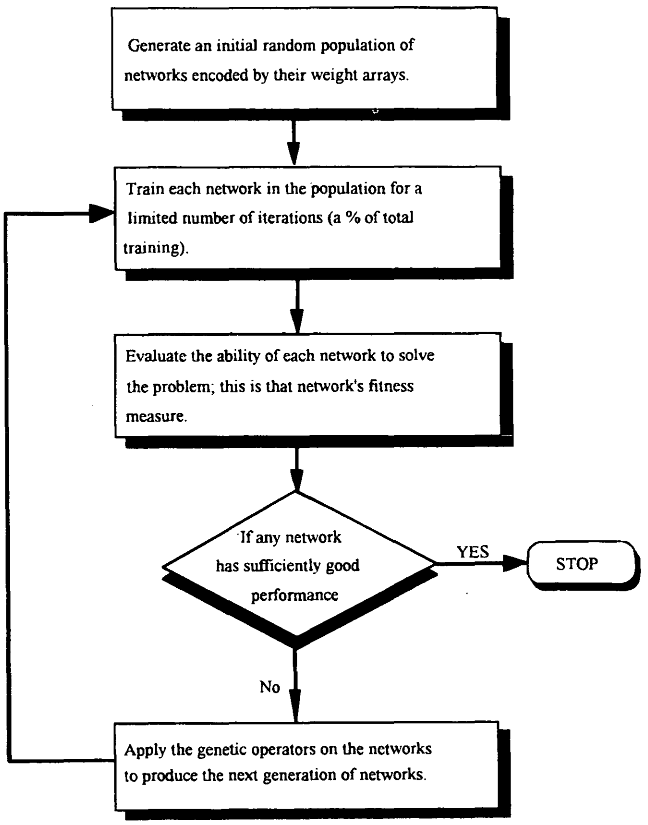

Generate an initial random population of networks encoded by their weight arrays.
Train each network in the population for a limited number of iterations (a % of total training).
Evaluate the ability of each network to solve the problem; this is that network's fitness measure.
If any network has sufficiently good performance
YES
STOP
No
Apply the genetic operators on the networks to produce the next generation of networks.

FIGURE 1 Procedures for optimizing weights of neural networks using GAs.

MODEL DEVELOPMENT

The following methodology has been used to illustrate the development of the present model. It includes three main phases: (a) problem definition; (b) model evolution; and (c) running the model for direct problem solving. Each of these phases is described in the following section.

Problem Definition

Pavement maintenance is defined as an action taken to correct deficiencies that are potentially hazardous and to repair defects that seriously affect serviceability to maintain or keep the pavement within a tolerable level of serviceability (*9*). Maintenance of most asphalt pavement involves repairing localized problem areas to prolong the pavement life. The proper maintenance of any highway system depends on the methods that the responsible agency uses to meet the climatic conditions in its jurisdiction.

Determining the best maintenance strategy starts by identifying the type, severity, and density of pavement distress. Pavement distress is defined as the condition of pavement structure that reduces serviceability or leads to a reduction in serviceability (*10*). Severity is measured using a three-point scale (slight, moderate, and severe) and density using a two-point scale (few and extensive). Typically, the decisions regarding the selection strategies are made by experienced senior practitioners. In order to lessen the dependency on these experts and to help less experienced practitioners to participate in this decision-making process, a neural network system is very useful.

Neural networks can fit this problem because of their adaptable structure. This is because of the existence of hidden layers and the nonlinear activation function in their structure permits them to make reasonable generalization. This important property of neural networks enables the performance of complex multiattribute, nonlinear mapping for the selection of the optimum pavement maintenance strategy.

Model Evolution

The process of evolving the neural network model comprises five main aspects: (a) developing network training and testing examples; (b) development tool; (c) genetic operators; (d) fitness evaluation; and (e) training and testing (validating). Each of these aspects is described in the following sections.

Developing Network Training and Testing Examples

Training and testing examples represent the most important ingredient in the development of a neural network model. They consist of all input and output data used by the neural network's learning algorithm. Without those examples, the network would not be able to learn anything about the problem. A given training example consists of two components: (a) a data case consisting of a set of attributes, each with an assigned value, and (b) the corresponding correct class membership or the classification decision made by a domain expert according to the given data case. These examples can be assembled in several ways: (a) they may come from an existing database that forms a history of observations; (b) they may be a carefully culled set of tutorial examples prepared by domain experts; (c) they may be obtained from simulation results; or (d) they may be obtained from literature available from and interaction with domain experts.

In the present work we used the readily available knowledge acquired by Hanna et al. (*9*). This knowledge was acquired to build PMAS, a knowledge-based expert system for assisting highway engineers in planning effective flexible or asphalt concrete pavement maintenance strategies. This knowledge is available in the form of IF-THEN-ELSE rules. The examples used for developing the neural network model were compiled from these rules. The antecedents of the rules make the inputs, and the consequent of each rule makes the outputs of the examples.

In these examples the components of the input vector represent the factors affecting the maintenance strategy selection, which are identified as (a) distress type, (b) density of distress, (c) riding comfort index (RCI), (d) traffic volume, (e) climate, (f) crack type, and (g) severity of distress. Each of the first five input factors is represented by two binary neurons. Each of these neurons takes only one value, either 1 or 0. The sixth and the seventh input factors are represented by three binary neurons. Table 1 shows the possible values for the input neurons corresponding to each input factor. The components of the output vector represent the different pavement maintenance strategies available, which are identified as (a) do nothing, (b) crack seal coating, (c) route and seal, (d) cold mix patching, (e) hot mix patching, (f) hot mix recycled patching, and (g) reconstruction. Seven binary neurons were chosen to represent each output as shown in Table 2.

A total of 335 examples were developed from the available knowledge base. The whole set of examples was divided randomly into a training set of 235 (about 70% of the cases) examples and a test set of 100 examples (about 30% of the cases). A portion of the training and test examples is shown in Table 3.

Development Tool

The commercially available neural network simulator BrainMaker Professional was used for the development of the proposed neural network application. The Brainmaker package forms a complete

TABLE 1 Representation of Input Factors

Input Factor	Input Case	Representation		
		Neuron #1	Neuron #2	Neuron #3
(1)	(2)	(3)	(4)	(5)
Distress Type	Single	0	1	N/A
	Combined	1	0	
Distress Density	Few	0	1	N/A
	Extensive	1	0	
RCI*	< 4	0	1	N/A
	≥ 4	1	0	
Traffic Volume	$\leq$ 2000 VPL**	0	1	N/A
	$>$ 2000 VPL	1	0	
Climate	Coastal (DDI*** $<$ 600 C*days)	0	1	N/A
	Inland (DDI $\geq$ 600 C*days)	1	0	
Crack Type	Alligator	1	0	0
	Rutting	0	1	0
	Traverse	0	0	1
	Alligator + Rutting	1	1	0
	Rutting + Traverse	0	1	1
	Alligator + Traverse	1	0	1
	Alligator + Rutting + Traverse	1	1	1
Distress Severity	Slight	1	0	0
	Moderate	0	1	0
	Severe	0	0	1

* Riding Comfort Index   ** Vehicle Per Lane   *** Degree Days Index

TABLE 2 Output Representation

Maintenance Strategy (1)	Representation (2)						
Do Nothing	1	0	0	0	0	0	0
Crack Seal Coating	0	1	0	0	0	0	0
Rout and Seal	0	0	1	0	0	0	0
Cold Mix Patching	0	0	0	1	0	0	0
Hot Mix Patching	0	0	0	0	1	0	0
Hot Mix Recycled Patching	0	0	0	0	0	1	0
Reconstruction	0	0	0	0	0	0	1

TABLE 3 Portion of Training and Testing Examples

Distress Type		Distress Density		RCI		Traffic Volume		Climate		Crack Type			Distress Severity			Maintenance Strategy
(1)		(3)		(2)		(4)		(5)		(6)			(7)			(8)
(a) Training Examples																
0	1	0	1	1	0	1	0	0	1	0	1	0	0	1	0	Do Nothing
1	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	Do Nothing
0	1	1	0	1	0	0	1	1	0	0	0	1	0	1	0	Crack seal coating
1	0	0	1	1	0	0	1	0	1	0	1	1	0	0	1	Crack seal coating
0	1	0	1	0	1	1	0	1	0	0	0	1	0	0	1	Rout and Seal
0	1	1	0	0	1	0	1	1	0	0	0	1	0	0	1	Rout and Seal
0	1	0	1	1	0	1	0	0	1	1	0	0	0	1	0	Cold Mix Patching
0	1	1	0	1	0	0	1	0	1	0	1	0	0	0	1	Cold Mix Patching
0	1	0	1	1	0	0	1	0	1	0	0	1	0	0	1	Hot Mix Patching
1	0	1	0	0	1	1	0	0	1	1	1	1	0	1	0	Hot Mix Patching
0	1	1	0	1	0	0	1	0	1	0	1	0	0	0	1	Hot Mix Recycled Patching
0	1	1	0	1	0	1	0	0	1	0	1	0	0	0	1	Hot Mix Recycled Patching
0	1	1	0	0	1	0	1	0	1	1	0	0	0	0	1	Reconstruction
1	0	1	0	0	1	0	1	1	0	1	1	0	0	0	1	Reconstruction
(b) Test Examples																
0	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	Do Nothing
1	0	0	1	0	1	1	0	0	1	0	1	1	0	0	1	Crack Seal Coating
0	1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	Rout and Seal
0	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	Cold Mix Patching
0	1	0	1	1	0	0	1	0	1	1	0	0	0	0	1	Hot Mix Patching
1	0	1	0	0	1	0	1	1	0	0	1	1	1	0	0	Hot Mix Recycled Patching
1	0	1	0	1	0	1	0	0	1	1	1	0	0	0	1	Reconstruction

system for designing, building, training, testing, and running neural networks on IBM personal computers and compatibles. The Genetic Training Option (GTO) of Brainmaker is used to optimize the weights of the developed neural network model. It applies Darwin's theories of genetic mutation and natural selection. GTO creates a large number of subtly different networks to do the same job. It then tests, trains, and ranks them to find the network(s) that perform(s) the best overall according to the user definition of the "best." It has the capability to keep up to 10 best networks at the end of each run. The GTO "genetic evolution" works on the neuron connections of a trained network using genetic operators (*11*).

Genetic Operators

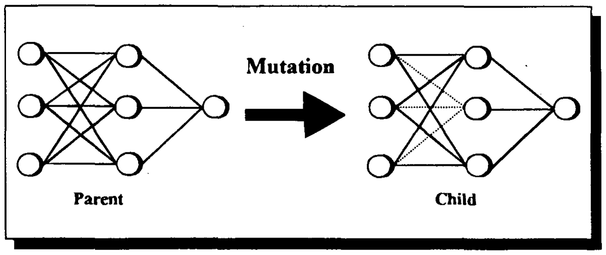
The genetic operators used in this study were mutation and crossover. The mutation requires only one parent. The mutation operator allows new genetic sequences to be introduced. This is done by changing a random percentage of the neurons by modifying the weights associated with them. In the example shown in Figure 2*a*, the child is produced from the parent network by changing the connection weights from the input layer to the second hidden neuron. On the other hand, the crossover operator allows "sexual" reproduction by combining two parents to produce an offspring. It is implemented by taking some neurons from one "parent" network and some from another to produce an offspring. In the example shown in Figure 2*b*, the child receives the second hidden neuron and output neuron from the second parent and the first hidden neuron from the first parent.

Fitness Evaluation

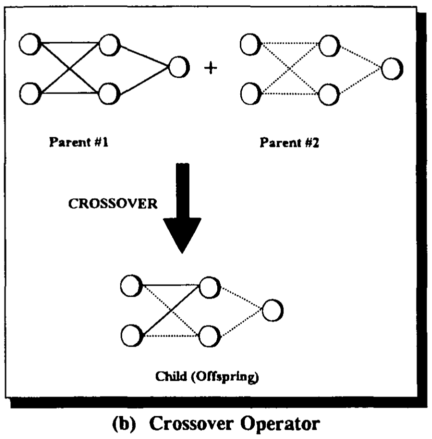
The GAs use an objective function to evaluate the performance of the members of each generation. In our case the members of each generation are neural networks, and the task is to select the optimum pavement maintenance strategy. The objective function is a mapping from the weight space of a particular neural network to a single value. This objective function should possess some degree of smoothness in the region about the solution point in the weight space (*7*). This means that any change in the weights in the direction of the optimum should yield a higher performance value.

In our work the fitness of each network is measured by testing its performance on unseen cases. The network with the lowest average error on all test cases will have a higher chance to survive and to produce the next generation. This average error is calculated using the following equation:

$$\text{Average Error} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{o} |O_{ij} - P_{ij}|}{N} \qquad (1)$$



(a) Mutation Operator



(b) Crossover Operator

FIGURE 2 Genetic operators.

where

O_{ij} = Calculated output value
P_{ij} = Desired output value
N = Number of test cases
o = Number of output neurons

Training and Testing

Before any particular run, the network topology is specified. A neural network model with 16 input neurons in the input layer, 16 hidden neurons in the middle layer, and 7 neurons in the output layer is used. The number of hidden neurons is determined based on the rule of thumb that suggests that this number is equal to the number of neurons in the input layer. The learning rate parameter is assumed to be constant over all of the network connections with a value of 1.0.

The training procedure starts by generating an initial randomized weight matrix using BrainMaker Professional. This initial weight matrix is loaded into GTO. The genetic evolution process is used to save the best 10 networks over 50 generations. During this evolution process, the mutation and crossover operators are applied to 10 and 70 percent of the hidden neurons, respectively. The evolution process is started by applying the mutation operator to the initial network to create another parent network. The two parent networks are then crossed over to produce an offspring. These networks are trained using backpropagation for 100 epochs (an epoch is one presentation of the whole training set). Because network generalization is a main criterion for optimization, the networks are tested using 100 unseen cases. The test results are presented in the form of average error rate. If the child network outperforms its parents, it will be saved and used to replace its parents. Otherwise, the best parent will be saved and used to start a new generation.

Because the program can keep up to 10 networks, the best networks in the first 10 generations will be saved automatically. For the following generations, the average error rate on unseen cases of the best network of each generation (ϵ_{new}) will be compared with the highest average error rate on unseen cases of the 10 saved networks (ϵ_{old}). If ϵ_{new} is found to be lower than ϵ_{old}, the new network will replace that one. This procedure will be repeated until no improvement in the average error rate is achieved. In our case no improvement was achieved after 50 generations.

The runs were conducted on a powerful DX2-66 MHz IBM compatible. The results of these runs are summarized in Figure 3, which shows the best-of-generation average error for three independent runs with different initial populations. The plot indicates that the error value decreases with successive generations. Moreover, the best 10 networks generated in this evolutionary process are tested by presenting test examples to them. The test results are summarized in Table 4. These results show that network number 5 provided a better prediction ability (average error rate attributable to test on unseen cases was 0.024; misclassified only 6 out of 100 unseen cases) than the other nine networks. On the basis of these results, this neural network model was chosen as a final solution to our problem.

Also, the initial neural network model used in the evolution process is trained using backpropagation only using the same training set. After presenting the training examples 1706 times, the training error converged to 0. This network is saved, and its generaliza-

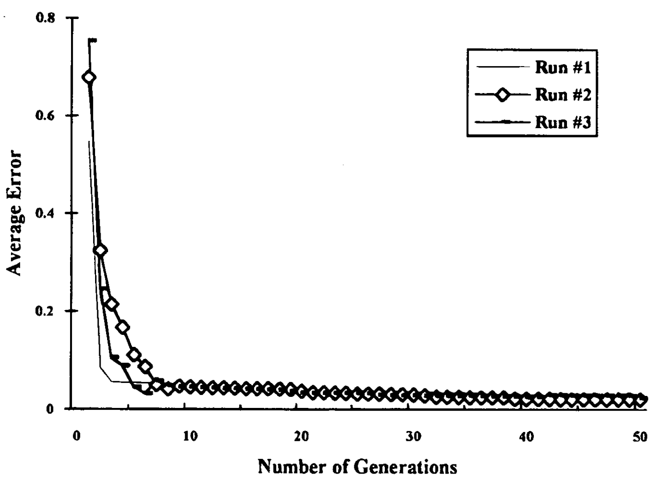


FIGURE 3 Best-of-generation average error rate versus generation number.

tion ability is checked using the same test cases used above. The test results show that the generalization ability of this network was worse than that of the evolutionary network; its average error rate was 0.043 (about 2 times more), and it misclassified 20 out of 100 test cases.

Running the Model for Direct Problem Solving

The developed neural network model is just a set of interconnection weights that are simple real numbers. It can be used to select the optimum maintenance strategy for new cases in two ways:

- If the user owns a copy of the Brainmaker simulator, he/she can prepare a fact file and then run Brainmaker and use this network to predict new cases included in the fact file.
- For a user who does not own a copy of this simulator, a query-and-answer program was developed, using C programming language, to help the end user to run the neural network model. The program will first prompt the user to input the factors affecting the selection process. The program then will translate these factors into the network input values as 1 or 0, propagate these inputs through the developed neural network with the obtained interconnection weights, and prompt the end user with the final recommendation. A copy of the data input screen is given in Figure 4.

SUMMARY AND CONCLUSIONS

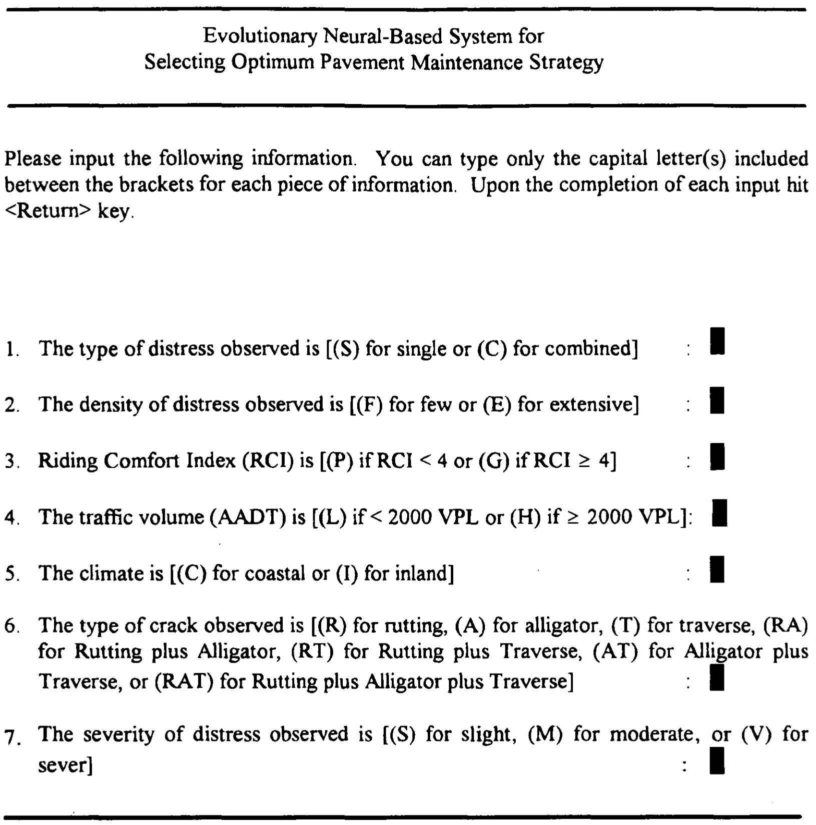
Genetic algorithms are search procedures that are able to locate near-optimal solutions after examining only portions of the search space. They are gaining increasing popularity as a valuable tool to optimize many engineering problems. In this paper a simple genetic algorithm made up of mutation and crossover was combined with the backpropagation algorithm to find the optimum weights for a neural network model. The objective of the model is to select the optimum pavement maintenance strategy. The use of backpropagation in conjunction with genetic algorithms is not seen as just a way of fine tuning the rough solution generated by a genetic algorithm but, rather, as a way of sustaining diversity in the population in a nondestructive fashion. This is because the use of backpropagation will change the connection weights and thus introduce new genetic material that can be exploited by the genetic algorithm. The genetic training option of BrainMaker Professional neural network simulator was used in the development process. The details of the development and implementation of the model were presented. The results showed that combining genetic algorithms and backpropagation to develop the neural network weights helps in improving the network generalization ability compared with that obtained using backpropagation alone.

This paper provided a solution for the problem of finding the interconnection weights such that the network can compute a desired input to output mapping. It suggests that a fundamental area of research involves the problem of finding a suitable network topology. This includes number of hidden layers, number of hidden neurons per hidden layer, type of activation function, and learning parameters. Genetic algorithms may also be a good candidate for this problem. Finally, the approach described in this paper could be applied to the development of neural networks in other civil engineering domains.

TABLE 4 Validation Results (Unseen Cases)

Network Number (1)	Missclassified Cases (2)	% of Missclassified Cases (3)	Average Error (4)	Root Mean Square Error (5)
1	8	8	0.0250	0.1065
2	6	6	0.0244	0.1039
3	8	8	0.0252	0.1060
4	8	8	0.0252	0.1063
5*	**6**	**6**	**0.0240**	**0.1035**
6	8	8	0.0258	0.1079
7	6	6	0.0248	0.1044
8	12	12	0.0284	0.1255
9	7	7	0.02449	0.1051
10	8	8	0.0550	0.1062

*** Best Network**



Evolutionary Neural-Based System for
Selecting Optimum Pavement Maintenance Strategy

Please input the following information. You can type only the capital letter(s) included between the brackets for each piece of information. Upon the completion of each input hit <Return> key.

1. The type of distress observed is [(S) for single or (C) for combined] : █
2. The density of distress observed is [(F) for few or (E) for extensive] : █
3. Riding Comfort Index (RCI) is [(P) if RCI < 4 or (G) if RCI ≥ 4] : █
4. The traffic volume (AADT) is [(L) if < 2000 VPL or (H) if ≥ 2000 VPL]: █
5. The climate is [(C) for coastal or (I) for inland] : █
6. The type of crack observed is [(R) for rutting, (A) for alligator, (T) for traverse, (RA) for Rutting plus Alligator, (RT) for Rutting plus Traverse, (AT) for Alligator plus Traverse, or (RAT) for Rutting plus Alligator plus Traverse] : █
7. The severity of distress observed is [(S) for slight, (M) for moderate, or (V) for sever] : █

FIGURE 4 Data input screen.

REFERENCES

1. Flood, I., and N. Kartam. Neural networks in Civil Engineering. I. Principles and Understanding. *Journal of Computing in Civil Engineering*, Vol. 8, No. 2, 1994, pp. 131–148.
2. Moselhi, O., T. Hegazy, and P. Fazio. Neural Networks as Tools in Construction. *Journal of Construction Engineering and Management*, Vol. 117, No. 4, 1991, pp. 606–625.
3. Austin, S. An Introduction to Genetic Algorithms. *AI Expert*, March 1990, pp. 49–53.
4. Rumelhart, D. E., G. E. Hinton, and R. J. Williams. Learning Internal Presentations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Micro Structures of Cognition* (D. E. Rumelhart and J. L. McClelland, eds.), Foundations, MIT Press, Cambridge, Mass. 1986, pp. 318–362.
5. Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Co., New York, 1989.
6. Whitley, D. *Applying Genetic Algorithms to Neural Network Learning*. Technical report CS-88-105, Department of Computer Science, Colorado State University, Fort Collins, 1988.
7. Smith, T. R., G. A. Pitrey, and D. Greenwood. Calibration of Neural Networks Using Genetic Algorithms with Application to Optimal Path Planning. *First Annual Workshop on Space Operations Automation and Robotics* (SOAR 87) (Sandy Griffin, ed.), 1987, pp. 519–526.
8. Caudill, M. Evolutionary Neural Networks. *AI Expert*, March 1991, pp. 28–33.
9. Hanna, P. B., A. S. Hanna, and T. A. Papagiannakis. Knowledge-Based Advisory System for Flexible Pavement Routine Maintenance. *Canadian Journal of Civil Engineering*, Vol. 20, 1993, pp. 154–163.
10. Thomas, W. K., L. R. Freddy, and J. B. Rauhut. Distresses and Related Material Properties for Premium Pavements. In *Transportation Research Record 715*, TRB, National Research Council, Washington, D.C., 1987, pp. 15–25.
11. California Scientific Software. *Getting Started With Brainmaker: User's Guide and Reference Manual*. California Scientific Software, Nevada City, 1993.

Publication of this paper sponsored by Committee on Artificial Intelligence.