PREDICTING MINING MACHINE CUTTING TOOL WEAR USING NEURAL NETWORKS

A Thesis
Submitted to the College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

in the
Department of Electrical Engineering
University of Saskatchewan

by

DAVID MYERS

Saskatoon, Saskatchewan, Canada

December 1999

ACKNOWLEDGMENTS

The author sincerely wishes to thank his supervisor, Dr. H.C. Wood, for his guidance and encouragement throughout this research.

The author offers his thanks to his wife, Trudy, for her understanding and support, without which this work would not have been completed.

The efforts of the examination committee, comprised of professors R.J. Bolton, Dr. N.A. Chowdhury, Dr. S.O. Kasap, and Dr. G. Watson are also greatly appreciated.

The author would like to recognize the support, both financial and moral, of H.A Simons Limited, and Atomic Energy of Canada Limited.

The author acknowledges the input of Potash Corporation of Saskatchewan, who provided important background information regarding the subject of this research.

ABSTRACT

To improve safety and economic productivity, the mining industry has been striving towards completely unmanned underground operations. The potash sector has participated in this effort, and has succeeded in automating or remote operating the continuous mining machines used in these mines. However, the detection of worn cutting tools on these machines has remained a manual function performed by experienced operators.

In this thesis, research into a method of automatically scheduling cutting tool replacement outages is described. Recurrent neural networks were used to identify the dynamic process of mining machine revenue generation with tool wear. A genetic algorithm technique was employed to train the neural network on line.

The trained neural network was used to predict the productivity of the machine following a postulated outage, thereby allowing an informed decision as to whether an outage would be beneficial.

The results of this project show that an on line dynamic neural network system can be employed to schedule outages for a continuous mining machine for worn cutting tool replacement. The method approached the productivity realized with current outage scheduling. However, the potential for fully automating mining operations may be facilitated by this method.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Continuous Mining Machines

The key major equipment in the numerous potash mines in southern Saskatchewan is the continuous mining machine. The revenue of the mining operation is directly related to the availability and productivity of these machines.

Most potash mining in Saskatchewan is performed using continuous mining machines similar to the Marietta unit shown in Figure 1-1. The Marietta is a large track-wheeled machine with two to four cutting rotors on the front. The cutting rotors are driven by synchronous motors, while the traction drive is a DC motor. The DC drive motors are controlled to attain a constant cutting motor amperage. Each rotor has mounting slots for approximately thirty cutting tools. The rotors shown in the photo do not have bits installed.

Considerable effort has been expended to automate the operation of these mining machines. The primary objectives of this effort have been to increase productivity and safety, and ultimately, to remove the human operator from the underground environment. Fortney and Lewis [1] described in detail the automation of the four rotor Marietta miners in operation at the Potash Corporation of Saskatchewan Rocanville operation. The machines are controlled by an on board Programmable Logic Controller (PLC) and can be operated in a range of modes from manual to fully automatic. In the automatic mode,

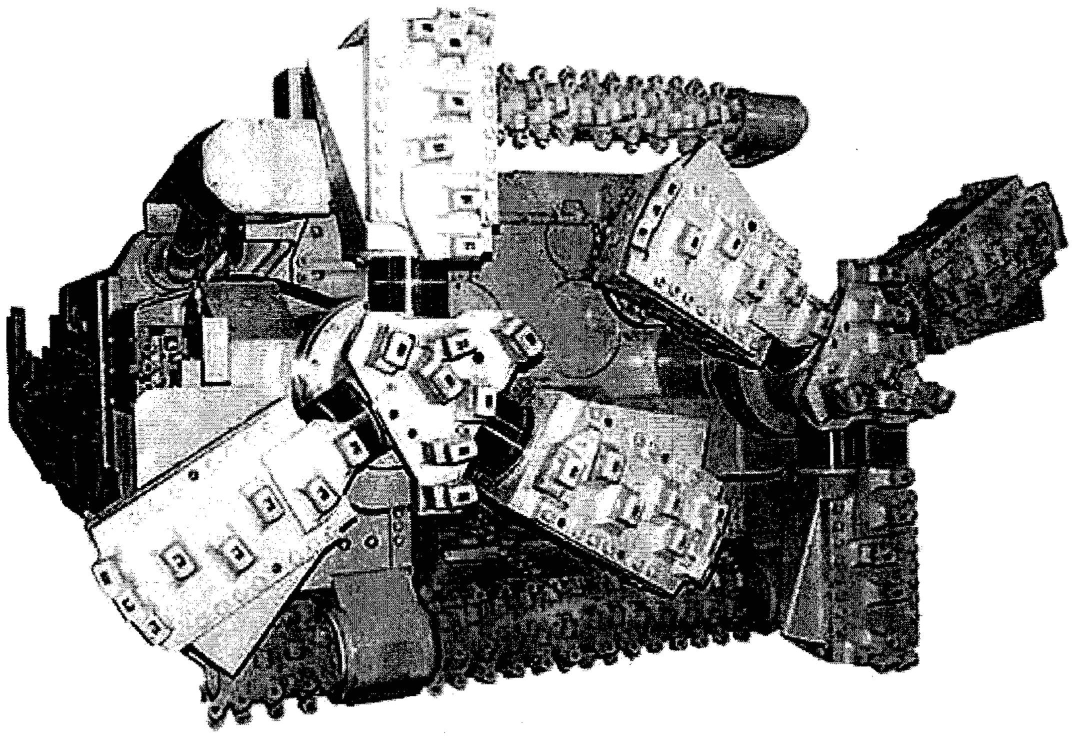


Figure 1-1 Two-Rotor Marietta Miner (photo courtesy Tamrock Inc.)

the machine is capable of tracking the ore body under computer control with only supervisory effort by the operator.

Although significant advances in machine automation have been achieved, the evaluation of cutting tool wear is essentially a subjective assessment performed by an experienced operator. The face of the machine is not visible during operation, and therefore the initial indication of poor cutting tool performance is derived from performance characteristics of the machine. If worn or damaged bits are suspected, the machine must be stopped and backed away from the mine face so the bits can be inspected. This can be a very dangerous action, which can require the worker to enter a confined space in front of the machine.

1.2. Objectives

The objective of this thesis is to determine if a neural network (NN) algorithm can be used to automate the scheduling of machine outages to replace worn cutting tools on a continuous potash miner. The NN method may also provide increased profit generated by the mining machine for the owner. Previous research on metal machining tools has shown that the non-linear wear of cutting tools can be modelled using neural networks. However, no published work applying this technology to potash mining machines has been performed to the knowledge of the author.

It shall be determined if a recurrent NN (with internal time-delayed feedback) will exhibit an ability to recognise time dependent patterns of machine behaviour and learn to predict optimal cutting tool replacement strategies. This project will be successful if the algorithm is able to control bit replacement scheduling, and if it improves the machine profitability compared to current scheduled replacement methods.

1.3. Organisation of the Thesis

This thesis is organised into five chapters. Chapter 1 introduces the current method of mining potash and defines the objectives of the research conducted. Chapter 2 summarises literature on related research into predicting cutting tool wear and more general work in rock cutting tool behaviour. Chapter 3 discusses the simulation of the mining process and the neural network method employed in this project. Chapter 4 describes the experimental procedure used to evaluate this method. Chapter 5 presents and discusses the results of the experiments. Chapter 6 concludes the thesis with a summary of the results and a discussion of the practical significance of the work.

2. BACKGROUND AND LITERATURE REVIEW

2.1. Cutting Tool Performance Prediction

There is little published work dealing specifically with continuous mining cutting tools used in potash. Some related research has been conducted in the mining area generally, including hard rock drilling and continuous coal mining. Extensive research has been performed in the area of metal fabrication regarding methods of monitoring and predicting cutting or drilling equipment performance.

Roswell and Waller [2] described an intelligent control system for rock drilling to optimise the cost per meter of the process. The costs were expressed as fixed and operational costs. The system used a knowledge based approach with past experience stored in decision tables, rather than NNs. Drill wear rate was considered in the control algorithm, as well as penetration rate, speed, thrust, and knowledge of the drilling cost. The paper acknowledges that bit wear rate is a very difficult parameter to measure while a rock drilling machine is in operation. To overcome this, the authors implemented a prediction method to estimate penetration and wear rates from a database. Their predictions are based on a limited entry decision table (LEDT) which contains some measured wear volumes for known drilling conditions. For new drilling conditions the wear volume is interpolated. The system was able to optimise drilling costs and overcome changes in rock strata.

Pazuchanics and Mowrey [3] investigated methods of discriminating between coal and rock cutting using NNs. The purpose of this research was to improve an automated coal-rock interface detector previously developed by the United States Bureau of Mines for guiding a continuous coal miner automatically. Cutting tool forces and vibration were measured in a laboratory apparatus while cutting known strata. A NN was trained to classify these signals as either coal or rock cutting. This work showed that back propagation NNs can be used to correctly classify the material being cut by a mining machine. The paper also compares neural networks to more conventional majority voting methods. The neural network method tended to perform better than the previously studied methods. This paper demonstrates the potential for NN applications using force and other non-intrusive measurements in mining. Mineral seam sensing is an issue in potash mining and has been investigated by researchers at the University of Saskatchewan [4] although neural network techniques were not studied in that case.

Ghasempoor, Moore and Jesweit [5] developed an on-line method of estimating tool wear in metal turning using neural networks. In [5] the metal turning process was defined with two components: a time-dependent wear function and a static function of wear, which gives the cutting forces on the tool. Neural networks were used for identifying these two components. A dynamic NN was trained to model the wear function, and a static NN was trained to identify the force function. Since the two NNs were cascaded, a difficulty with this method was that the error of the first NN could not be computed on-line unless the second NN was trained. The static NN was trained off-line for this purpose. The error of the first NN could then be calculated by back propagating the output error through the second NN. Training the dynamic NN was performed on-line using back propagation. This method demonstrated rapid learning of the dynamics of the wear process.

Ramamurthi and Hough [6] developed a real-time diagnostic system for automating the replacement of metal machining tools. The system used multiple sensory inputs to discriminate between good and worn tools. An expert system based on an influence diagram and a knowledge database was used. The influence diagram is a multilevel nodal system similar to NNs, but with a defined structure based on the input sensor selection and the ability of each parameter to predict tool wear and failure, rather than a generalized structure. The system was able to learn by adjusting the knowledge database during a training phase in which the test cases were presented and iterated off-line. It should be noted that in terms of real time operation, metal fabrication has much more stringent requirements than what is expected in rock cutting. The paper suggests a response time of 1.8 to 2 seconds is necessary to predict tool wear or failure in real time. It is encouraging that a cutting tool replacement system for metal fabrication appears possible under these conditions.

Leem, Dornfled and Dreyfus [7] developed a NN based sensor fusion system for monitoring metal cutting tool wear. The NN was divided into two stages: an unsupervised Kohonen's feature map; and an input feature scaling algorithm. Kohonen's feature map is an unsupervised method similar to competitive learning used to categorise input data sets into "clusters". The clustered categories must then be decoded using a small set of known inputs and outputs. However, Kohonen's method is more efficient because the order of the mapping of input value clusters to outputs is known. In the case of [7], the input values are from a number of instruments and the outputs represent degrees of tool wear. The input feature scaling stage is used to weight the sensory inputs to the NN to improve the classification abilities of the NN, and to map to resulting clusters based on some known inputs to define the categorisations of the unsupervised learning process. This system was tested and demonstrated a 92 percent success of correctly classifying tool condition.

Wilcox and Reuben [8] used a NN to discriminate metal milling tool degradation events using a variety of sensors including acoustic emission. A back propagation NN without feedback was used to classify acoustic emission samples into a number of tool degradation events. This system was designed to detect short time scale tool failure events which occur in less than one second. Neural networks were studied due to their very fast forward execution in comparison to the knowledge based methods studied by other researchers (e.g. [6]). The eight peak RMS acoustic levels for one or two tool rotations were input to the NN, based on off-line experimentation which suggested this would provide a means of discriminating various fault conditions. Training of the NN was performed off line rather than in real-time, due to the very fast response time required of the system while on line. The NN technique was deemed successful although a fairly small number of test cases were used for training.

Pepe, Looney, Hashimi and Galantucci [9] used neural networks to predict abrasive wear resistance of tungsten-carbide-cobalt coatings. A multiple regression system was also tested and the results compared to the neural network system. The multiple regression system involved tuning the parameters of a third order multivariate function to approximate the known coating wear test cases. The two systems were then tested using previously unknown data sets and their resulting errors compared. Using a

NN with 6 first layer, 20 hidden layer, and 3 output layer neurons and back propagation training, the NN system outperformed the regression technique with approximately half the error in typical cases. This suggests that neural networks are a useful tool for evaluating mechanical wear of materials, such as the cutting tools in the continuous mining machine.

2.2. Rock Cutting Tool Wear

A quantified model of tool performance degradation was required in this project to develop an approximate model of the potash mining process for simulation purposes. A review of past research on rock cutting tool wear was conducted to develop a basis for this simulation. No work specific to potash mining was found in this review. However, sufficient work in other industries, primarily coal mining, was found to form a basis for model development.

Plis *et al.* [10] conducted an extensive experimental procedure to quantify the performance of rock cutting tools as they wear. This work was developed for coal mining equipment, but the rock samples used in the experiment were highly siliceous and abrasive rock. A wide variety of tool designs were tested, including several which were similar to the tools used in potash mining. Each bit was used to cut the sample rock at a constant depth of cut. The applied forces and distance travelled by the bit were recorded. These data have been used in this project as a model of worn bit performance.

Aboujaoude *et al.* [11] developed a simulation for rotary blasthole drilling. As part of this work, the exponential relationship between penetration rate and applied force was experimentally determined for various rock strata. This relationship is required in the development of the potash mining model as described in section 3 of this thesis.

2.3. Summary of Previous Research

Neural network systems have been utilised successfully for cutting tool degradation analysis, particularly in metal fabrication. Both static and dynamic neural networks have been used with positive results. On line training of the neural networks has also been used successfully. Neural networks have been used in mining applications for categorisation and optimisation problems which make use of on line measurements.

The wear of cutting tools used in mining has been studied and mathematical models based on empirical data have been developed.

The past work suggests that analysing cutting tool wear for mining applications is within the possibility of current technology. In the next section, the NN method developed for potash mining is discussed.

3. THEORETICAL BASIS

3.1. Overview

In current mining operations, the decision to stop mining for cutting tool inspection and replacement is performed by the human operator. The decision is based on the operator's experience with the machine and ore body, using measured parameters such as advance rate and power draw, and more subjective parameters such as sound and vibration. To replace this decision making function successfully with a computer algorithm, the experience of the operator must be emulated. In this project, this was attempted by using a dynamic NN to identify the dynamic characteristics of the mining machine. Then, using the trained NN, the output of the machine a few time steps in the future was calculated and used to decide if a bit replacement outage was appropriate.

A block diagram of the overall test system is shown in Figure 3-1. The test system included the following components, each of which is discussed in more detail in the following sections:

- a numerical model of the mining process;
- a dynamic NN containing feedback which simulates the mining process;
- a training algorithm using a genetic algorithm method;
- two forward calculations of the NN which predict operation of the mining machine following an outage in the next step; and
- a rule which decides to either operate or shut down the machine.

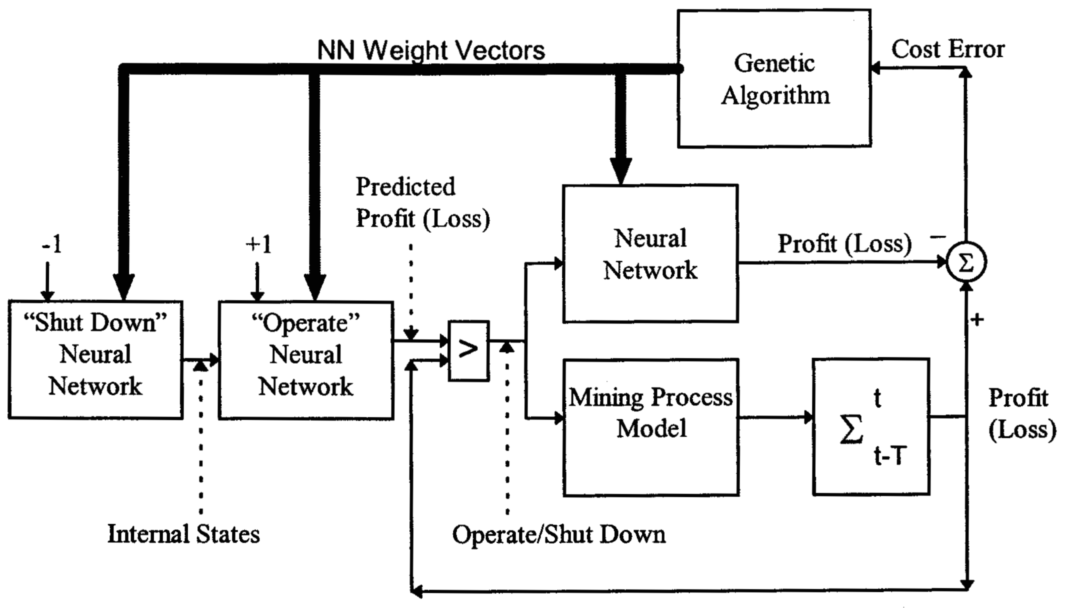


Figure 3-1 Test System Block Diagram

Discussions with operating potash mine staff revealed that training data from a real operating mining machine would not be available in quantities sufficient for this project. To continue the research, an approximate dynamic computer simulation of the mining process was developed. This model was based on known specifications of the Marietta machine and typical cutting bits, and published work dealing with cutting tool wear and performance in mining.

The model was intended to simulate the process of cutting soft rock with worn tools in sufficient detail to demonstrate that the NN prediction technique can be applied successfully. However, the simulation is not a precise duplication of any particular machine, cutting tool or potash ore body.

The input to the mining process model is a binary value that indicates whether the machine will be operated or shut down for bit changing during the current time increment. This input was based on the human operator's decision to operate the machine or shut it down for maintenance. The machine output is a net profit or loss for that time increment in dollars, and is dependent on the ability of the cutting tools to penetrate the potash and produce ore and the costs associated with operating the machine. On the real machine, this output would be calculated from instrument signals such as ore recovery rate, power consumption, and other fixed and variable operating costs.

A dynamic NN was placed in parallel with the mining process model. The purpose was to train the NN to mimic the behaviour of the mining machine. A dynamic network was employed because the mining process output is dependent on past states, such as the time when new bits were last installed.

The NN weights were adjusted by a genetic algorithm training technique in response to the error between the mining process and the NN. A genetic algorithm was selected for this project because, unlike the more classical back propagation method, the genetic algorithm does not require partial derivatives of the NN to solve for the weights. This allowed some experimentation with various non-linear functions without the need to re-write the programming. The genetic algorithm was also selected to determine its value in solving this problem.

As the NN learns to approximate the dynamic behaviour of the machine, it can be used to predict the mining process output in future time increments. This allows the future outcome to be predicted if the machine is either operated or shut down for maintenance in the current period. Since the bit performance and therefore the machine output always decrease if the machine is operated, it is only necessary to test the future machine output if maintenance is performed during the current period. This was achieved with two forward calculations of the trained network, first with an input of -1 (shut down) then with an output of +1 (operating). If the final output was greater than the previous period by some arbitrary value, a shutdown was assumed to be the appropriate action to take.

3.2. Mining Process Model

The mining process model simulates the revenue and cost of the machine in the operating and shut down states. The sole revenue producing parameter is the product recovery rate achieved by the machine while operating. All other factors are costs, either while operating or shut down, which influence the total profit or loss for a given time period. A block diagram of the machine model is shown in Figure 3-2.

The recovery rate is dependent on the condition of the cutting bits and the forces applied by the machine. A worn bit will require more mechanical force to recover product at a given rate than a new tool. Therefore the machine model must account for increased bit wear with use, and product recovery with bit wear and applied forces.

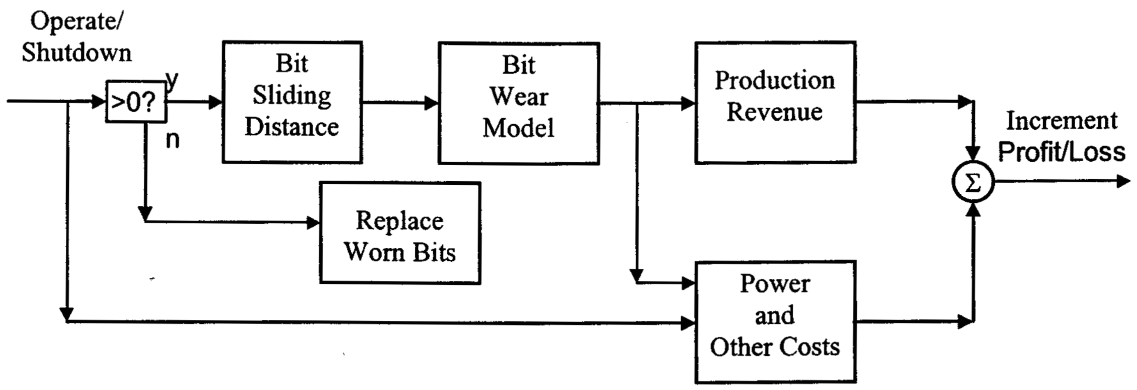


Figure 3-2 Mining Process Model

The "Operate/Shut Down" input is a binary value, where +1 indicates the machine is running and -1 indicates the machine is shut down for the current time increment. It is assumed that during the one hour shut down, operators can identify and replace all worn bits and return the machine to operation, and therefore the outage is always one time step in duration.

3.2.1. Tool Wear and Performance

The relationship between cutting rate, wear and applied forces is required to model the mining process. Relevant work done for various mineral types (e.g. coal, hard rock) was evaluated and critical parameter relationships were extracted. From this an approximate model of worn tool performance was developed. The following describes the development of this model from published experimental results.

The forces on the bit are termed normal and cutting forces, as shown in Figure 3-3 [10].

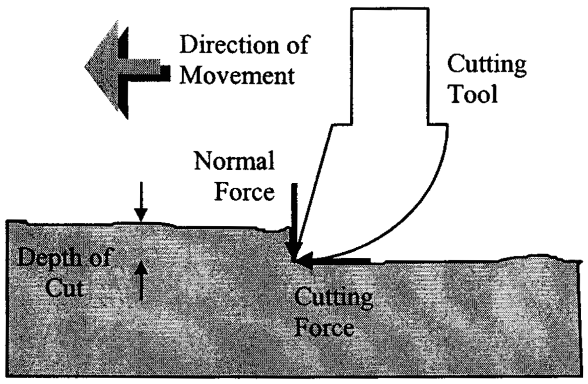


Figure 3-3 Cutting Tool Forces

Plis *et al.* [10] tested the wear and performance characteristics of a number of cutting tool designs in the laboratory. The bits were used to cut a known rock specimen at a constant depth of cut. The normal and cutting forces required to maintain the cutting rate were recorded versus the total linear distance cut (referred to as the sliding distance of the tool on the rock face). The radial bits with PDC inserts tested by Plis *et al.* [10] are very similar to the bits used on Saskatchewan potash operations. Data for the 0° and -20° polycrystalline diamond compact (PDC) bits are used as a basis for this research.

The work of Plis *et al.* [10] is based on a constant depth of cut. The Marietta machine is controlled to maintain constant rotor torque [1], and thus constant cutting force. The normal force applied by the tramming motor is varied to achieve this constant torque, and thus depth of cut is variable depending on the condition of the individual bit. Aboujaoude [11] experimentally determined the relationship between rock drill penetration rate and normal force follows an power law relationship of the form

$$R \propto W^{x} \tag{3.1}$$

where

$R =$ penetration rate; and

$W =$ weight on bit (normal force); and

$x =$ constant.

The exponent x depends on drilling conditions and was found in [11] to be between 0.4889 and 0.9. For this project a value of .707 (1/2 of square root of 2) is assumed. Using this relationship, the penetration rate used in [10] is scaled from the measured normal and cutting forces to obtain a penetration rate versus sliding distance curve for constant cutting force. A combination of 0 degree and -20 degree PDC radial bit data from [10] is plotted in Figure 3-4.

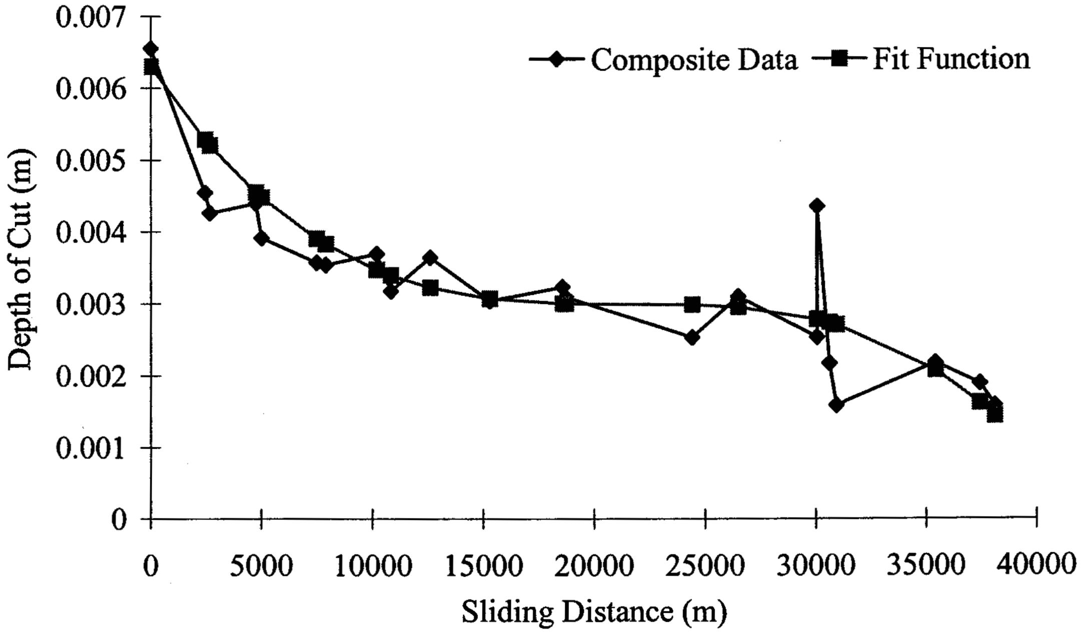


Figure 3-4 Empirical Cutting Tool Productivity and Functional Approximation

For modelling purposes, a continuous function of the form

$$P = 0.033 \cdot \left(1 - \frac{D}{21340}\right)^3 + 0.003 \qquad (3.2)$$

where D is sliding distance in meters and P is depth of cut in meters, is used which approximately models the empirical relationship shown in Figure 3-4. This function is plotted in Figure 3-4 and referred to as the fit function. From this relationship, the productivity of each bit can be determined from the total sliding distance of each bit against the face.

A single rotor with 30 bits is simulated. The sliding distance D for each bit is dependent on the radial position of the bit. This explains why operators observe that the bits near the center of the rotor do not wear as quickly as bits near the periphery. The distance cut for each bit is then expressed as

$$D_j = R_j \cdot \pi \cdot (\omega \cdot T) \qquad (3.3)$$

where

$j =$	bit number;
$D_j =$	sliding distance for bit j;
$R_j =$	radial position of bit j;
$\omega =$	rotational speed of rotor (RPM);
$T =$	time period of simulation increment (minutes).

The total productivity model for the machine can then be expressed as

$$Q(t) = \sum_j P_j(t) \cdot D_j \qquad (3.4)$$

where

$$P_j(t) = 0.0033 * \left[1 - \frac{R_j \cdot \pi \cdot \varpi \cdot \left(t - T_{repl}\right)}{21340} \right] + 0.003 \qquad (3.5)$$

and

$Q(t) =$ the excavation productivity in cubic meters for the simulation time period;

$T_{repl} =$ the time at which each bit was last replaced.

3.2.2. Uncertainty Modelling

The mining process model as described to this point is non-linear due to the worn bit productivity function and the binary nature of the bit replacement process. However, the model does not simulate the uncertainties in bit quality that would be observed in the real world. These are introduced to test the NN's ability to cope with new situations. This is achieved by adding an amount of random variation to the constant factors in Equation 3.5 as shown in Equation 3.6.

$$P_j(t) = 0.0033 \cdot \left[1 - \frac{R_j \cdot \pi \cdot \varpi \cdot \left(t - T_{repl}\right)}{21340 + \delta_j} \right] + 0.003 \qquad (3.6)$$

where

$\delta_j =$ random variation in bit quality within a batch.

The bit variation factor δ_j represents a variation in the expected life of the bit and is applied to each newly replaced bit. A normally distributed value with a mean of ten percent the expected bit life is used for this factor.

3.2.3. Revenue and Cost Model

The revenue derived from mining can be measured in tons of ore extracted. In this project, the objective is to attempt to optimise the net profit of operating the miner. This requires that the output of the mining process model be equated to dollars rather than tonnage, and the cost factors both of operating and maintaining the machine be considered in the model.

The revenues and costs modelled in this study are given in Table 3-1.

Table 3-1 Revenue and Cost Factors

Variable Description	**Operating Value**	**Outage Value**
Ore recovery revenue	grade (%) x recovery rate (t/hr) x market value ($/t)	0
Electricity usage	usage (kW-hr) x rate ($/kW-hr)	0
Operator salary	hours of operation x wage	0
Maintainer salary	0	hours of outage x wage
Parts, particularly new cutting tools	0	cost/tool x number replaced
Machine wear (placement value)	hours of operation (hr) ÷ expected life (hr) x machine cost ($)	0

In the simulation, these revenues and costs are normalised such that the machine output with all new cutting bits is equal to 1. For simulation purposes, this maximum revenue rate was set to $5000 per hour. The other costs were normalised using this factor.

It is assumed that the salaries of the operators and maintainers partially cancel out and the relative cost of labour is small compared to the other costs for the purpose of this study.

Further, it is assumed that external variables such as ore grade, market value of product, electricity rates, and parts cost are constant for the period of the simulation. These factors could be included readily in a real installation using instrumentation or manual data entry.

The cost of electricity can be assumed to be constant for the purposes of this study. The large 400 horsepower rotor motors are the dominant consumer, and as they are operated at constant torque, will draw constant current. The smaller 50 horsepower DC tramming motors will consume variable power levels. However, the tramming motors account for only a small fraction of the total machine power draw. Therefore the simulation in this project treats power cost as a constant.

3.3. Dynamic Neural Network

Artificial neural networks are numerical algorithms that are based on our understanding of the functionality of biological neural systems [13]. Artificial neural networks (abbreviated as NN in this thesis) utilise parallel, highly connected networks with adjustable interconnection coefficients and non-linear mapping functions in attempt to emulate the learning capability of biological neural networks. This is achieved typically using a numerical model of a neuron, with many inputs, weighting factors on each input, a bias or offset which also has an adjustable weight, summation, and a non-linear saturation function. These neurons are organised in layers, with the neuron outputs of each layer connected to the inputs of subsequent layers.

The neural network used in this project is a dynamic network, which means the neuron outputs are delayed in time and fed back to the neuron inputs. This type of NN exhibits dynamic behaviour in the time domain. Since the output of the mining process is dynamic in that the current output is dependent on past states, a dynamic NN was deemed necessary for this study. Dynamic neural networks have been used for this purpose in other research [5].

The form of the network used in this project is shown in Figure 3-5. This structure was developed based on a two-layer static architecture with time-delayed feedback only within the first layer. Other possible structures could include feedback from the output of the final layer back to the first layer, or feedback distributed among the neurons in the first layer. The structure shown was based on its similarity with the bit wear process, in which the wear of each bit is time dependent but not closely coupled with the wear of other bits.

The network consists of two layers of neurons Nx,y, where x is the layer number and y is the neuron number within a layer. The number of first layer neurons, y, is adjustable for experimental purposes. The first layer includes delayed feedback from each neuron output to the respective neuron input. An adjustable number of delayed feedback states are used with increasing delay cycles k. This allows some experimentation to find the appropriate order of the dynamic system. The second layer is a summing layer that does not include feedback.

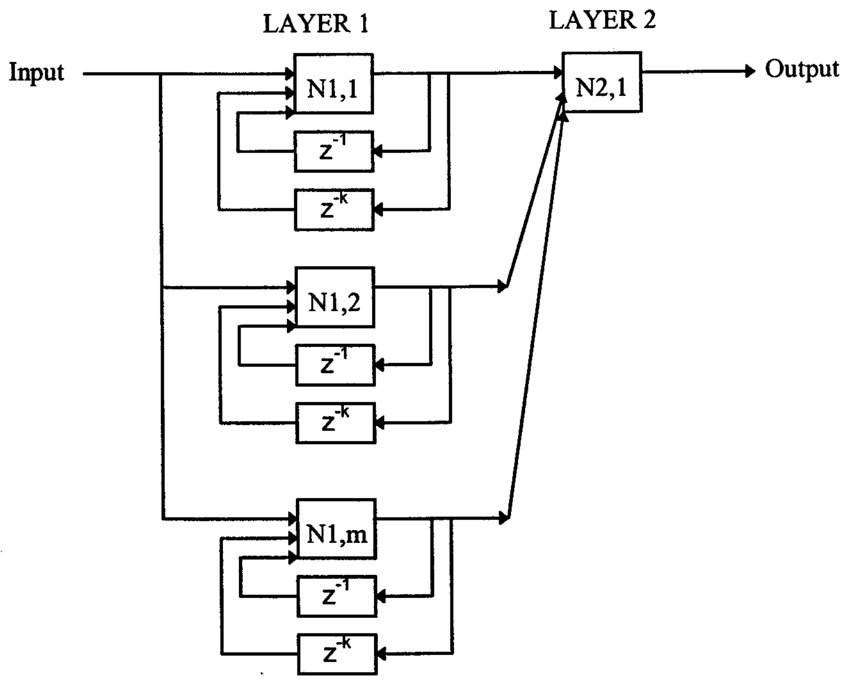


Figure 3-5 Neural Network Topology

All neurons include an offset component, input weighting variables, summation of the weighted inputs and offset, and a non-linear saturation function. The form of the neurons is shown in Figure 3-6.

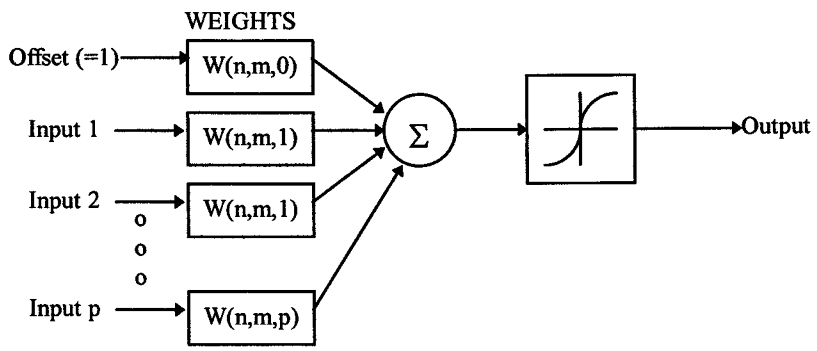


Figure 3-6 Neuron Model

The input weights in each neuron are adjusted to achieve a given relationship between NN inputs and outputs. The use of delayed feedback in the first layer introduces time and history dependence to this relationship. This dynamic behaviour can be adjusted to approximate the dynamics of a known specimen; in this case, mining machine operation.

In dynamic terms, this network is a sum of difference equations. However, the system is not linear due to the saturation function in the neurons and the offset included in the inputs. It is proposed that the summation of a sufficient number of nonlinear dynamic neurons will adequately model the machine dynamics for the purposes of this project.

3.4. Genetic Algorithm Training

Training is the process by which a NN "learns" to map from a set of known inputs to a corresponding set of known outputs. The known data is referred to as training data. The weighting factors in the NN are adjusted through training to adapt the input-output map of the NN to match the training data.

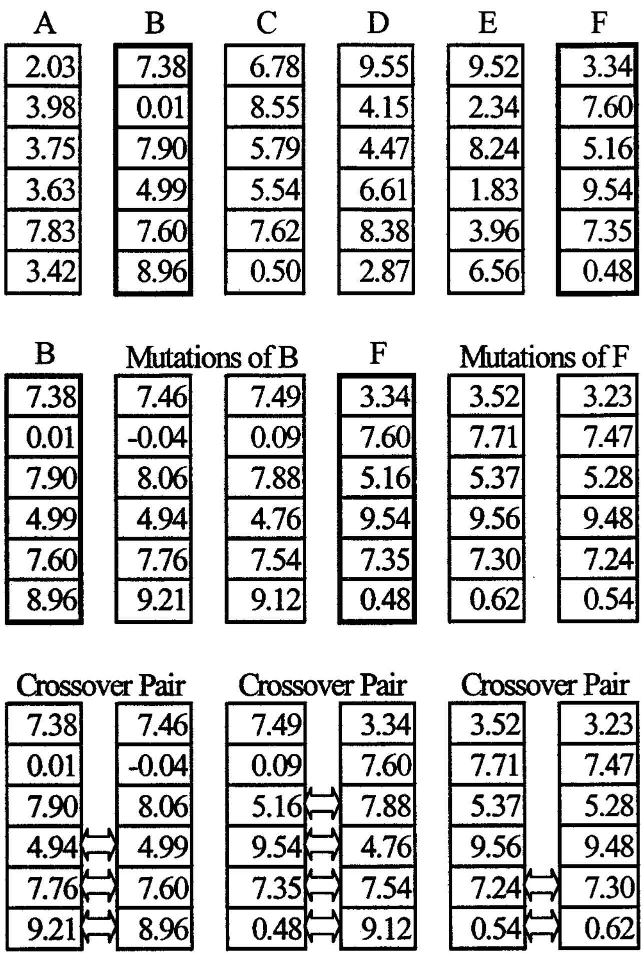
In this project, the NN is a dynamic system with time dependence and memory of past events. The training data is obtained in real time from the machine simulation. The weights of the NN are continuously adjusted in response to the error between machine output and NN output.

A genetic algorithm (GA) is used to find the NN weights. The genetic algorithm is an optimisation technique based on the principles of biological evolution [14]. Plant and animal species have evolved into highly optimised forms through a continuous cycle of crossover, mutation, and survival of the fittest. This cycle results in effective optimisation of the gene structures that determine the characteristics of the organism. In a similar manner, natural selection can be applied to parameter optimisation problems such as searching for optimal weighting factors in the NN. Figure 3-7 provides an overview of the genetic algorithm method.

A population of test weight vectors is created at random. The test vectors are applied to the NN individually along with the input for the current time step. The outputs of each test are compared to the actual machine output. The vectors that produce the least NN error are allowed to survive. The remaining vectors are replaced to maintain the original test population.

Some of the vectors are replaced with normally distributed random values to ensure a wide search domain. The rest are replaced with copies of the survivor vectors, mutated slightly by adding a normally distributed random value to each element. The mean value of the mutation factor is reduced as a function of the neural network output error. This enhances the stability of the training algorithm while allowing a probability of searching a broad weight gain locus due to the use of the normally distributed mutation factor.

To cross over elements, pairs of vectors are selected. The pairs then exchange a random number of elements. The exchange of elements aids in the search for a single vector containing all optimised weight values.



SURVIVAL
In this example, assume vectors B and F produce the least error among the weight vector population. These vectors survive to the next generation.

MUTATION
The surviving vectors are copied to maintain the population. Random values are added to the duplicate vector elements to simulate mutation.

CROSSOVER
The mutated vectors are paired. A random number of elements are exchanged between each pair.

Figure 3-7 Genetic Algorithm Process

Although the application of genetic algorithms to neural network training is relatively new, several authors (e.g. [15], [14]) have successfully used the method for NN training. Some of this work has focused on dynamic NNs or control system problems.

Pham and Karaboga [16] successfully trained two types of dynamic NNs using GA techniques to identify non-linear second and third order plants. The NNs studied were the Elman network, in which feedback is from the hidden layer to the output layer, and Jordan network, which includes feedback from the output layer to the input layer. The Elman network is most like the NN used in this project. In [16], a GA was used to train both types of NN to identify three different plants. In addition, modified versions of the two NN structures were tested which included self feedback in the input layer, and each of these four structures were tested both with constant and trainable feedback connection weights. This paper concluded that allowing the feedback weights to be trainable significantly decreased the training time. Also, it was found that the NN structure with feedback improved the ability of the NN to identify the dynamic system.

Seng, Khalid, and Yusof [15] utilised GA's to train neuro-fuzzy controllers and successfully controlled unstable and non-linear plants better than conventional fuzzy controller methods or PID controller tuned by GA. The hidden layer of the controller contained the fuzzy membership functions. Both the membership function parameters and the input and output weights were tuned using GA. The GA utilised binary parameter coding, with 8 bits per value trained or 360 bits per chromosome. A flexible position strategy was used for the membership function parameters that allowed re-ordering of these parameters within a chromosome.

In order to apply genetic algorithms to NN training, the weights for the first layer, second layer, and feedback are treated as vectors. The classical genetic algorithm (e.g. [14],[17]) codes the parameters as binary values and treats each bit as an element of the gene. In this project, the floating-point weight values were treated as the discrete elements of the gene vector. This coding scheme was selected for computational simplicity, since there were already many elements to handle without converting each weight value to a binary string.

The basic GA technique is enhanced by two methods. First, the best pair of vectors are not subjected to the crossover and mutation processes, and carry forward to the next generation unchanged. This is known as elitist survival, and ensures the result of the next iteration will be at least as good as the current iteration. This elitist technique does not remove the best vectors from the crossover process entirely, because slightly mutated duplicates are used to replace other vectors that do not survive the iteration.

Second, the random mutation factor is scaled according to the least error for the iteration. This helps the dynamic training process settle when reasonable weights have been found. The genetic algorithm can still avoid local minima because some of the replacement vectors are completely random, and the mutation factor is a normally distributed value that has some probability of adding a large change to the vector.

3.5. Performance Prediction

The final element of the test system is forward prediction of the mining process output. This was the most challenging component to develop during this project, and several methods were attempted.

The goal of the prediction algorithm is to project the machine output after a hypothetical shut down. It is not necessary to predict the output if no shutdown is performed, since the bits will always wear out from use and produce less ore. However, if the machine is shut down and inspected, some of the bits might need to be replaced. New bits will always increase performance. The objective is to automatically schedule outages which optimise the output over a period of time, or at least lead to operating the machine economically.

The most successful method, and the one employed through the remainder of this thesis, was to run the trained NN forward in time. If the NN were perfectly trained, it would provide a correct prediction of machine output. Therefore the NN is first tested with an input of -1 (shut down), and then +1 (operating). The output of the final operating stage is compared with the current operating output. The hypothetical shut down is deemed successful if the predicted output after shutting down exceeds the current level by an arbitrary limit.

If the machine is currently shut down, it is automatically operated in the next period, since all bit change outages are assumed to be completed in one hour.

Of the other prediction methods that were tested the following were notable:

1. The NN was trained to produce the output from the *current* time period based on the control input from the *previous* time period. Using this time-shifted mapping, it was expected the machine output for a *future* period could be obtained using the *current* period input. This method was abandoned when training was not successful.

2. The NN was trained to map current inputs to current outputs, and was then inverted to provide a map from the output to the input. The desired output was entered to see if the inverted NN would indicate the required control input. The difficulty with this method was that it was highly sensitive to variations in the desired output, and became unstable easily.

4. SIMULATION PROCEDURE

4.1. Overview

Simulation testing was performed in two stages. In the first stage, open loop tests were conducted to determine the ability of a NN to model the machine behaviour and forecast machine output. The number of neurons in the first layer, the number of time-delayed feedback states per neuron, and the number of genetic algorithm test vectors were all variable.

Based on the conclusions of open loop system testing, closed loop tests were carried out in the second stage to gauge the ability of the NN to control the outage schedule of the machine.

4.2. Program Description

The test simulation shown in Figure 3-1 was developed using Matlab® version 4.0 by The Mathworks Inc. Structured text programs were created using the Matlab language library. The text programming method was used instead of the graphical programming available in Matlab to permit more complete control and understanding of the internal workings of the simulation.

The programs are structured to allow the numbers of first layer neurons, feedback delays, and genetic algorithm test vectors to be defined at run time. This allows for experimentation to find the appropriate number of these parameters required to achieve the project objectives. These program parameters include:

- the number of 60 minute time steps to simulate;
- the number of neurons in the first layer;
- the size of the genetic algorithm "population" of test weight vectors;
- the number of cutting bits to simulate (30 in all tests described below);
- the number of past state values within the NN; and
- the number of previous steps over which revenue is averaged.

Two main programs are used in this project: "openloop.m", which executes the open loop test case; and "clseloop.m", which is very similar to the former but completes the closed loop control algorithm shown in Figure 3-1. These programs call subroutines as required for the neural network model, "net2.m", the genetic algorithm, "genalg.m", and the mining machine model "tram3.m". The following sections define these subroutines to promote discussion of the main program algorithms which are described in section 4.2.4.

4.2.1. Neural Network Function

The neural network function "net2" used in this project is called using the form

$$[z, y_t] = net2(a1, a2, b, x, y_{t-1}) \qquad (4.1)$$

where vectors

x = the NN input

y_t = the outputs of each first layer neuron at time step t

z = the output of the neural network

a_1= the first layer input weights

a_2= the first layer feedback weights

b = the final layer input weights

In this project the NN learns the behaviour of the mining machine model. Therefore the input x is the Boolean command to operate or shut down the machine, and the output z is the estimate of the machine revenue for the current time step. The NN is perfectly trained if, for any sequence of operating states, the correct revenue is produced at the output at each time step.

The NN function in Equation (4.1) is dynamic because the value of y from the previous time step is fed back into the network as shown in Figure 3-5. The value of the first layer output y must be stored for use during the next time step.

The neural network subroutine describes each layer of the NN as a matrix equation. The first layer of the neural network is expressed as

$$y = \frac{2}{\pi} \cdot \mathrm{a\,tan}\left[\left| x \quad 1 \right| \cdot a_1 + y \cdot a_2 \right] \tag{4.2}$$

and the final layer is given by

$$z = \frac{4}{\pi} \cdot \mathrm{atan}\left[\left| y \quad 1 \right| \cdot b' \right] \tag{4.3}$$

Each neuron includes an offset component with an associated weight factor in vectors a_1 and b. The value of 1 is appended to each of the first layer input vectors x and y to provide an input to the offset in each neuron.

The factors $2/\pi$ and $4/\pi$ were introduced due to the use of the inverse tangent functions in Equations 4.2 and 4.3. The hyperbolic tangent function is more commonly used in neural networks. However, it was found that the inverse tangent function required much less computing time than the hyperbolic function, including the multiplication of the gain factor, and produced acceptable function approximation results.

4.2.2. Machine Model Function

The machine model function call is expressed as

$$[amps, revenue, wv, bc] = \mathrm{tram}3(repl, t, wv, bc, bits)\,, \tag{4.4}$$

where

amps	= the tramming motor current
revenue	= the net revenue of the machine for the hour
wv	= a vector containing the accumulated sliding distance of each bit
bc	= a vector containing the bit life of each bit, equivalent to the total sliding distance at which the bit is changed
repl	= boolean operate (+1) / shutdown (-1) state
t	= time step index
bits	= the number of bits on the machine.

The machine model begins by calculating the radial position of each bit, assuming the bits are equally spaced on a 1.5 meter radius rotor.

The remainder of the machine model is an if-then branch structure. If the machine is operated in the current time step, the radial position is then used to determine the incremental sliding distance for each bit for the current time increment. The productivity of each bit is found from the individual sliding distances multiplied by the total machine penetration, as given by Equation 3.6. This in turn relates to revenue from the machine based on the market value of potash.

If the machine is commanded to shut down, the total sliding distance of each bit is compared to the defined maximum the bit may withstand. Any bits whose current sliding distance exceed their expected life are replaced. This is achieved by resetting the total sliding distance to zero, and redefining the maximum life of the bit in the *bc* vector.

The amperage used by the machine is treated as a constant when operating, and zero when shut down. The constant operating amperage is based on the control strategy for the machine that regulates tramming motor torque to a constant value. After some experimentation, it was found the power consumption of the machine is not relevant for the purpose of this study since it does not vary significantly with bit wear. Amperage was retained in the model only because it impacts the economics when the machine is shut down. The amperage output of the function was not used elsewhere in the main programs.

The revenue is normalised using a constant equal to the maximum revenue the model can produce with all new bits. This was done because the NN functions properly only with normalised inputs.

4.2.3. Genetic Algorithm Function

The genetic algorithm function was designed to accept three matrices of population vectors and a vector containing the absolute error associated with each set of three column vectors. The form of the function is:

$$[a1,a2,b,er,ind] = genalg(a1,a2,b,er) \tag{4.5}$$

where

er = the error vector for the GA vector population;

ind = a vector containing an index of the sorted error vector.

The function first sorts the population vectors according to the error vector, and performs the survival, mutation and crossover on the resulting sorted data set.

Survival and mutation are achieved during the same step. The quarter of the vectors with the least error are retained without mutation. The next best quarter are replaced with a copy of the best quarter and are mutated according to the following formula:

$$a1(i + V/4) = a1(i) + 0.1 \cdot er(1) \cdot \text{randn}() \tag{4.6}$$

where

i = values from 1 to ¼ of the number of vectors in a1;

V = total number of vectors in a1;

$er(1)$ = the least error passed to the subroutine.

The function *randn* in Matlab generates normally distributed random values with a mean of 1 and standard deviation of 1. By mutating the weight vectors in this way, the root search can be narrowed as the error reduces, but can still have some probability of searching a wider range to jump out of local minima.

The third best quarter of the vectors is also a duplicate of the best quarter, mutated by a normal random value multiplied by the least error. However, the gain factor of 0.1 is replaced with a gain of 1 in order to widen the root search.

The final quarter of the weight vectors is replaced with completely random values. The normal function is also used here.

The crossover step of the genetic algorithm is performed by exchanging a random number of elements between pairs of vector sets. The best two vectors are excluded from this operation. It was found that this elitist survival technique reduced the possibility of the system diverging, since the error in the next step would be at least as good as the previous step. The pairs are selected sequentially according to the error sort, rather than randomly. This also seemed to improve the root finding exercise, as the better vectors exchange weights only amongst themselves. If the purely random vectors produce low error in the next time step, they become part of the elitist crossover program and their better weight factors become part of the surviving pool.

It is worthwhile to note the indexing of the first layer weight matrices *a1* and *a2* proved to be a challenge in the programming. Matlab version 4 only supports up to two-dimensional matrix indexing. Both the first layer weight matrices require three dimensional indexing. This was solved using a custom dual index technique in the program. The syntax in the genetic algorithm section becomes very complex particularly in the crossover loops, where a random number of elements must be exchanged.

4.2.4. Main Program Algorithms

The main programs begin with definitions of constants and the initialisation of all program variables. The initial neural network weights are selected at random using the normal distribution function available in Matlab. The initial history of bit changes is also selected using random functions.

During each simulated time step, the following sequence of calculations occurs.

1. **Forward Prediction Test**

Using the best weight factors found in the previous time step, the neural network is calculated twice to simulate two time steps into the future. The first step assumes the machine is shut down, and the second assumes it is operated. Care is taken to carry the internal state of the neural network to the second time increment, along with internal states from past time steps if required.

The two NN calculations into the future take the form

$$[z_t, y_t] = net2(a1, a2, b, -1, y_{t-1}) \qquad (4.7)$$

and

$$[z_{t+1}, y_{t+1}] = net2(a1, a2, b, 1, y_t) \qquad (4.8)$$

Note that the neural network input in Equation (4.7) is -1, indicating the machine is shut down, and in Equation (4.8) it is +1 indicating the machine should operate. Also note that the internal feedback states from the neural network are carried forward. This is essential to ensure the NN follows the predicted dynamics of the mining process;

however, the values are later discarded when the NN is recalculated to compute output error. The best weight vectors from the previous step are used to calculate both NN's.

2. Decision to Operate or Shut Down (Closed Loop Only)

In the closed loop test, the revenue output predicted by the neural network is compared to the current known output of the machine. This routine is omitted in the open loop test. The function used in this decision is:

$$repl_t = \frac{revenue_{t-1} - predicted\_rev_{t+1}}{\text{abs}(revenue_{t-1} - predicted\_rev_{t+1})} + \frac{0.05}{ave} \qquad (4.9)$$

This function simply determines if the predicted revenue exceeds the current revenue by a preset value of 5%. The machine is stopped for bit replacement in the upcoming time step if the predicted revenue is sufficiently high.

3. Calculation of Machine Model Output

The mining machine model output is calculated in either the operate or shutdown state. In the open loop test, the sequence of operate or shutdown states is predetermined. In the closed loop test, the state is determined by the decision calculated in step 2. Since the machine model may replace bits while shut down, the new bit replacement history data is saved for the next time step.

In the open loop test, the actual machine model revenues from this step are stored for comparison with the predicted revenues found in step 1. The results of the comparison are presented and discussed in section 5.1.

4. Calculation of Revenue with Standard Replacement

In the closed loop test, a parallel simulation is conducted using one shutdown at the beginning of each shift. This is referred to as standard replacement later in this document. The standard replacement strategy is used as a benchmark for evaluation of the closed loop NN control strategy.

5. Neural Network Calculations

The NN is calculated once for each of the genetic algorithm weight vector sets.

The error between the neural network output and the actual machine revenue for each of the genetic algorithm cases is calculated.

This step is the basis for training the NN. It is important to note that only forward calculations of the NN are used in this program. This resulted in a fast execution time for the simulation. For example, a complete simulation of 500 operating hours with 60 vector sets in the GA population would take approximately 5 minutes on an Intel Pentium 75 MHz processor, including the NN training which occurs in "real" time as the simulation proceeds.

6. Genetic Algorithm Weight Tuning

The genetic algorithm routine calculates a new population of weight vectors based on the current weights and the errors found in step 5.

7. Internal State Handling

The neural network internal state vector from the best test case is saved for use in the next time increment.

4.3. Open Loop Testing

A Matlab program was created which would run the NN training routine multiple times, each time using different NN structure and training parameters. The results of each test run were saved to text files on computer disk. This system permitted generation of hundreds of test cases. The data files were then analysed to determine relationships among the NN size, number of past state values, number of genetic algorithm training vectors, and the resulting average error and rate of training.

All test cases used a standard set of sample outage schedules that were generated prior to the open loop testing. Initially, randomly generated test cases were used for the open loop tests. The variation between cases due to the variation in the random input data made later comparisons difficult. Consistent input data produced more consistent results, allowing the correlation between network size and training parameters on total system error and training rate to be studied.

Five test runs were performed for each of the cases indicated in Table 4-1. Based on trial and error experimentation, a range of network size, past state values, and number of genetic algorithm test vectors was selected which would give a range of results for analysis. Each of the five test runs used one of five predefined input sequences as described earlier.

Table 4-1 Open Loop Test Cases

Neurons	Past State	Number of Genetic Algorithm Test Vectors				
	Values	60	120	180	240	360
2	2	X	X	X	X	X
2	4	X	X	X	X	X
2	6	X				
2	8	X				
4	2	X	X	X	X	X
4	4	X	X	X	X	X
4	6	X				
4	8	X				
6	2	X	X	X	X	X
6	4	X	X	X	X	X
6	6	X				
6	8	X				
8	2	X	X	X	X	X
8	4	X	X	X	X	X
8	6	X				
8	8	X				

4.4. Closed Loop Testing

The NN structure and training algorithm for all closed loop testing was based on the results of the open loop tests. Four first layer neurons and two past states were used, based on the relatively low RMS error in Figure 5-3 and Figure 5-4 and the reasonable convergence rates in Figure 5-6 and Figure 5-7. Cases for 60 and 240 GA test vectors were generated, based on the higher convergence rates in Figure 5-8.

The closed loop stage of testing was performed by running 20 simulations of the complete system shown in Figure 3-1 for each of the 60 and 240 GA test vector cases. One thousand operating hours were simulated in each run.

In parallel with each simulation run, a standard test case was simulated using one outage hour per eight-hour work shift. This was intended to provide a basis for evaluating the performance of the NN control technique. The standard replacement schedule assumes operators spend one hour once during a shift to inspect and replace worn bits. This schedule does not account for unplanned outages for bit inspection, and therefore the standard outage schedule will produce an optimistic result. Bit failure and machine stoppages for purposes other than bit maintenance are assumed to be equal for both conventional and NN control and are neglected.

5. RESULTS AND DISCUSSION

5.1. Open Loop Tests

For each of the five runs for each of the cases indicated in Table 4-1, data files were created for the actual machine output per time step, machine output as forecast by the NN, and the error of the NN prediction. The error files were loaded into a Microsoft Excel spreadsheet for analysis.

Figure 5-1 and Figure 5-2 show typical results for the open loop test. In this case the NN was structured with 4 neurons in the first layer and two past states of feedback. The GA used a population of 60 vectors for training. Figure 5-1 displays the complete 500 hour simulation, while Figure 5-2 shows the final 100 hours in more detail. The predicted values are the output of the NN two steps into the future. This is possible in the open loop test because the input sequence is predetermined.

Generally, the NN appears to be capable of predicting the output of the machine simulation. The NN output tracks the machine almost from the beginning of the simulation. This result is encouraging, although also suspicious since we would expect the NN to take much longer to train. In fact, the training process is able to adjust the weights quickly enough in real time to follow the machine output [17]. Learning the dynamic behaviour of the machine will take much more time. However, in this project a perfect identification of the machine may not be necessary. The NN only needs to model the machine dynamics for the current time and conditions so that a prediction of the next output can be made.

By the five hundredth hour in Figure 5-2, the NN is correctly predicting the output of the machine in the shut down state most of the time. The system is also tracking the output generally. Of interest in this project is the ability of the NN to correctly predict the output immediately after a shutdown. In Figure 5-2, the NN appears to be capable of this if the time since the last shutdown is short. If it is longer, the NN seems to loose its knowledge of the system dynamics during the transition from shut down to operating.

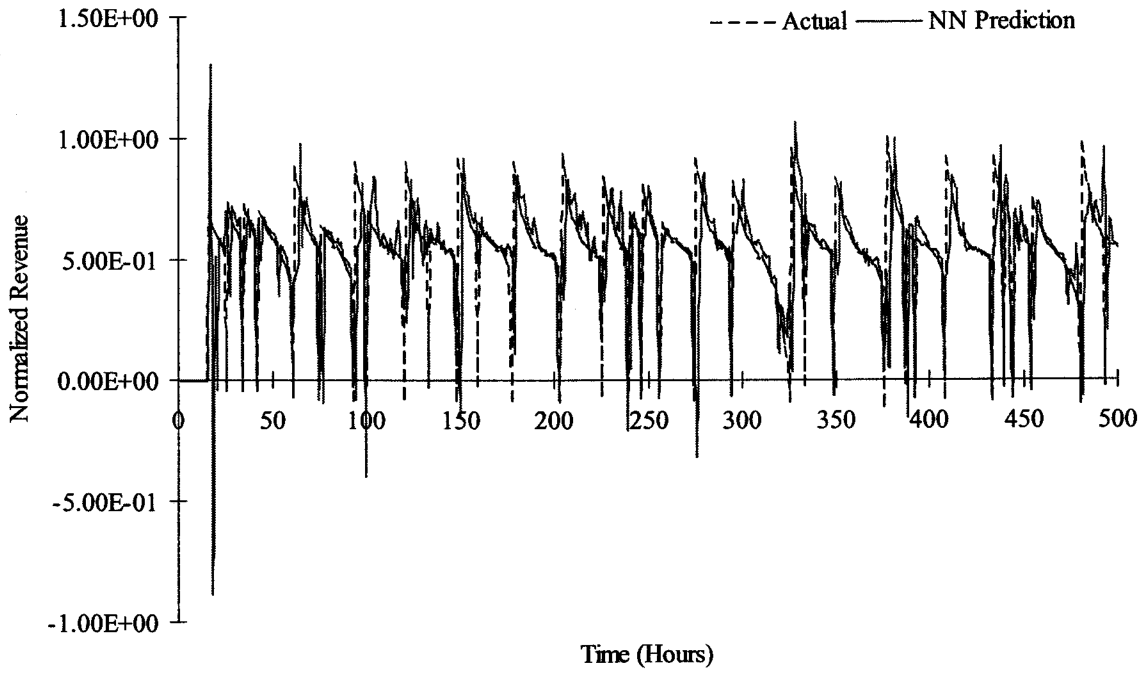


Figure 5-1 Actual and Predicted Revenue
4 Neurons, 2 Past States, 60 GA Population

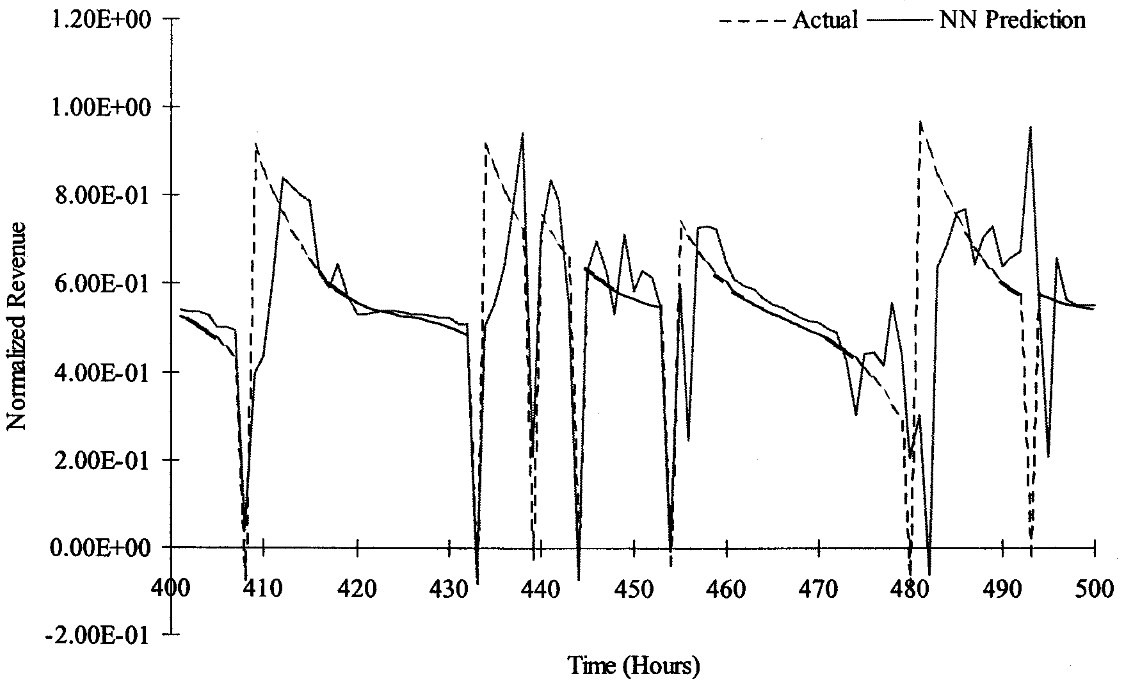


Figure 5-2 Actual and Predicted Revenue
4 Neurons, 2 Past States, 60 GA Population, Final 100 Hours

The average RMS error and the average linear slope of error versus time of the five test runs in each case were calculated and plotted versus each of the three run-time parameters in Table 4-1 above. These plots are given in Figure 5-3 through Figure 5-8 below. In each plot, the error or slope of the error versus time is plotted for the last 250 of 500 one-hour time steps in the test. This was done to allow the neural network some time for real-time training before testing the performance of each parameter set.

In Figure 5-3, the error for each case is plotted versus the number of neurons in the first layer. It is expected that a larger NN will be needed to model the machine accurately. While the minimum error does appear to decrease slightly with increasing NN size, it is clear that the error can increase dramatically with the number of first layer neurons. This may simply be due to the longer time required for a larger NN to be trained using a fixed number of GA test vectors, since the GA is searching for the optimal value of many more weights. For the purpose of the closed loop tests, a NN structure with 4 first-layer neurons will be used.

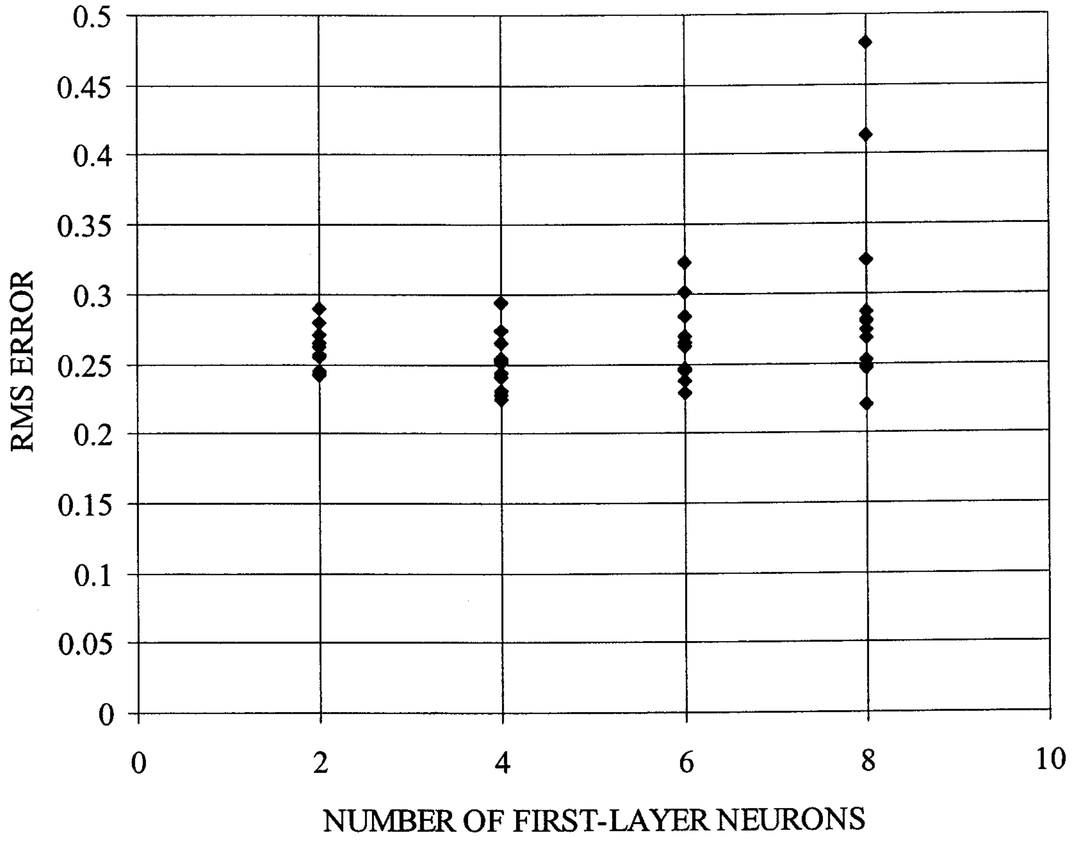


Figure 5-3
RMS Error of Final 250 Data Points Versus Number of First Layer Neurons for All Cases of Past States and GA Vector Population

In Figure 5-4, the error is plotted versus the number of past states in the first layer. We expect that a higher number of past states are needed to model the dynamic complexity of the machine. This is not suggested by the graph; in fact, the error increases almost linearly as the past states increase. Again, this may be attributed to the larger number of weights that must be trained as the order of the NN increases. However, for this result it seems reasonable to use either 2 or 4 past states in the closed loop system.

Note that while a general trend is apparent in Figure 5-4, some error values were observed which were well outside the trend, such as the error of 0.48 for the case of 4 past states, 8 first layer neurons and 60 GA test vectors. The genetic algorithm training method is a random process and can produce unexpected results. In this case, there were a relatively large number of parameters to train, and a small number of test vectors were used to solve for all the parameters. It is reasonable to assume that larger numbers of test vectors are needed to solve for larger numbers of parameters.

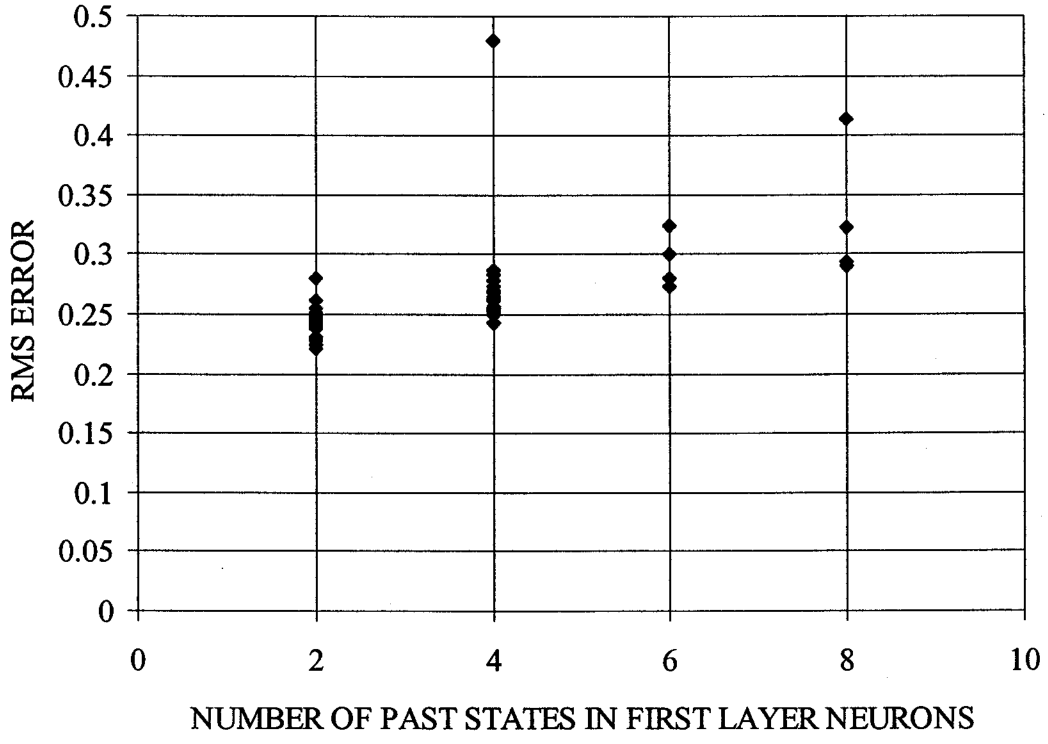


Figure 5-4
RMS Error for Final 250 Data Points Versus Number of Past States for All Cases of First Layer Neurons and GA Vector Population

In Figure 5-5, error is plotted versus the population of GA test vectors. We expect that the error would decrease as more test vectors are used. This is observed in Figure 5-5. However, the improvement in error is not as dramatic as expected. This figure suggests that over a period of 500 training hours, there may be an absolute minimum error that can be achieved. If this is the case, excessive computer operations and time may be wasted if large populations are used. Reduced computation is of great interest when designing software for a real-time environment.

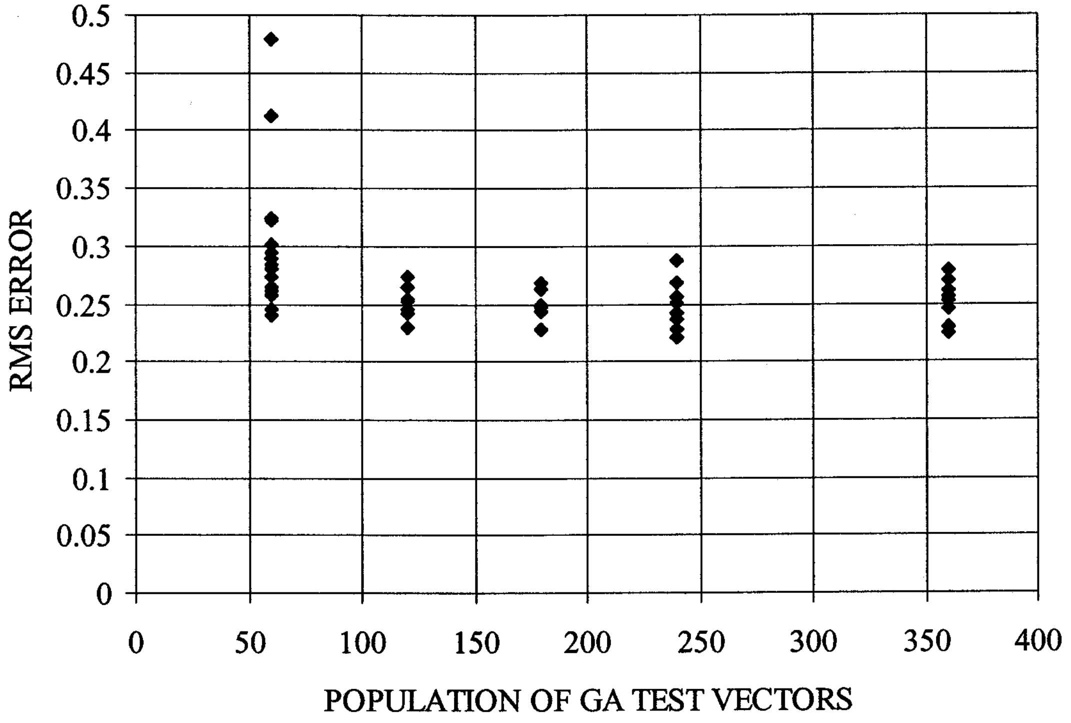


Figure 5-5
RMS Error for Final 250 Data Points Versus Population of GA Test Vectors for All Cases of Neurons and Past States

The linear slope of the RMS error is plotted in Figure 5-6 through Figure 5-8. The slope value is used to indicate the rate of NN training for the test case. A negative slope indicates the error is converging towards zero. Positive slope suggests the training process is diverging.

In Figure 5-6, the rate of training with variation in the number of first-order neurons is given. We expect that the smaller NNs train more rapidly since there are fewer weights to adjust. In fact, we observe generally faster training rates for larger NNs. This is an interesting result, since in Figure 5-3 the absolute error appeared to increase with the size of the NN. A possible explanation is that the smaller NNs achieved some training prior to the 250th time step, and therefore were beginning to converge, while the larger NNs were just beginning to converge and therefore made more progress in the final 250 steps.

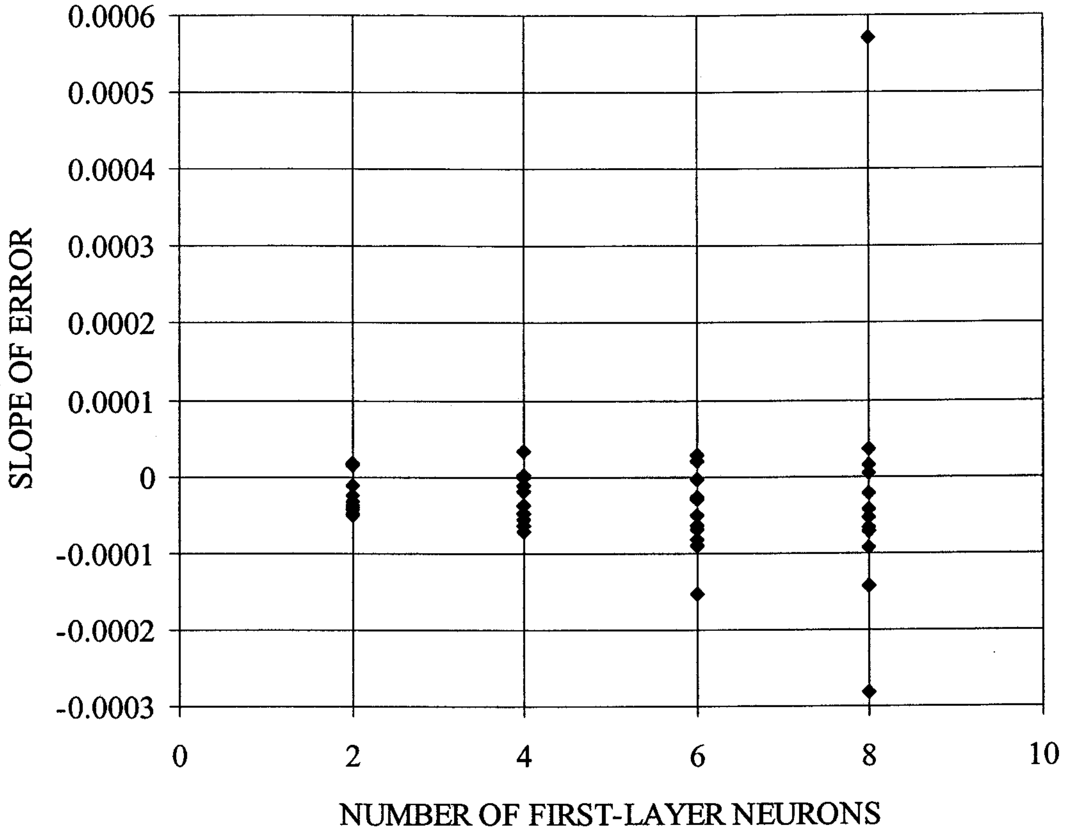


Figure 5-6
Slope of Error for Final 250 Data Points Versus Number of First-Layer Neurons for All Cases of Past States and GA Vectors

In Figure 5-7, the rate of training is given versus the order of the dynamic NN. There is no obvious relationship. There are many more data points for the 2nd and 4th order NNs, and for these orders a negative slope is observed in most cases, indicating the training is convergent. From this data is appears that a NN of any dynamic order will tend to converge.

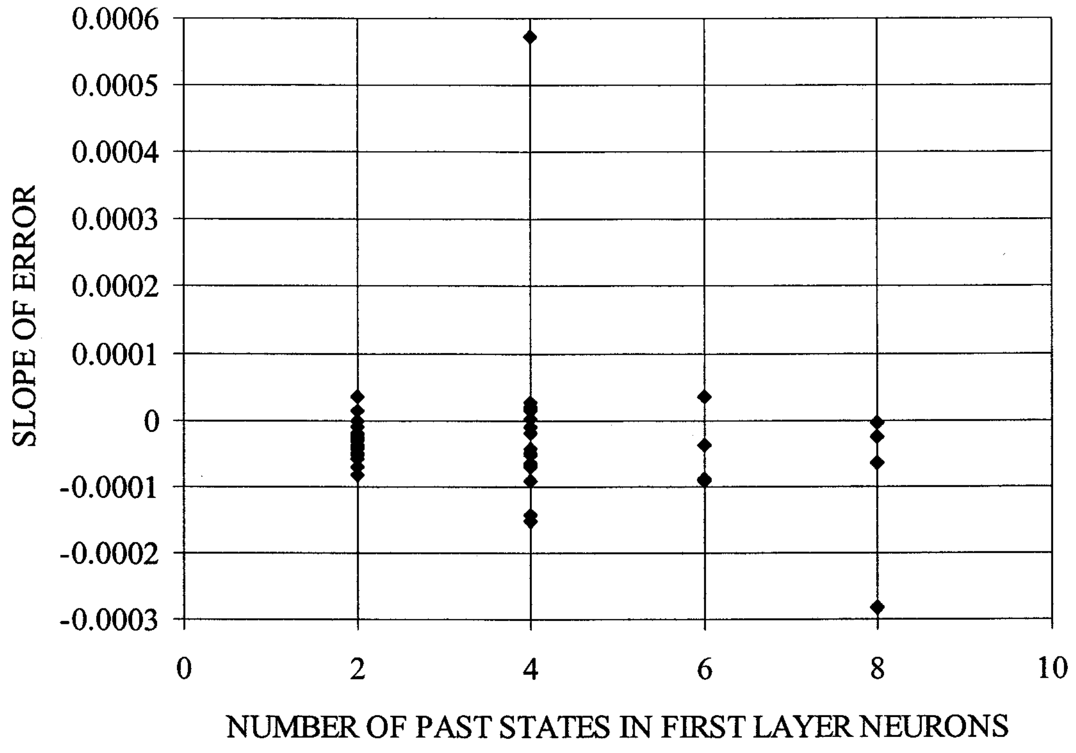


Figure 5-7
Slope of Error for Final 250 Data Points Versus Number of Past States in First-Layer Neurons for All Cases of Neurons and GA Vectors

In Figure 5-8, the training rate is plotted against the number of test vectors in the GA population. It is expected that the rate of convergence would increase with larger test vector populations. There is a gradual trend evident in Figure 5-8, but not as dramatic as expected. Throughout the experimentation for this project, it seemed that larger populations were not as helpful in training the NN as expected. In fact the opposite was observed: smaller populations of test vectors seemed to promote system identification in a given time period.

A hypothesis for this observation is as follows. With GA-based training, the NN rises to the absolute output level quickly. Larger test vector sets help the NN track more quickly. However, this affords little opportunity for the NN to learn the dynamic content of the system. With smaller test vector sets, the absolute error is larger, and training occurs more slowly. It seems possible that slower training could promote recognition of the dynamic characteristics of the system.

Mars *et al.* [17] support this observation with relation to nonlinear dynamic system identification. The training process continuously adjusts the weights in time and therefore contributes to the overall dynamic characteristics of the system. In [17], a recurrent neural network is used for system identification of a dynamic system in which the non-linearity is static. Like the results in this dissertation, Mars et. al. note that the NN quickly tracks the system output closely. However, if training is stopped after a short time, the NN fails to track the system output.

Very long training times of between 100,000 and 200,000 training iterations were required in [17] for complete system identification. In this project, with training iterations occurring in real time once per hour, it would not be feasible to train the system on line (100,000 hours is over 11 years). However, the NN in this project does not need to be trained until it correctly replicates the mining process under all conditions without weight adjustment. It is important to realise that:

1. the NN must only correctly identify the system output for two time steps into the future; and
2. continuous weight adjustment of the NN is tolerable in real time, given the one hour cycle time and the fast GA-based training system.

It is felt that these two factors contribute to the feasibility of the method employed in this project.

During open loop testing, random system inputs were employed. The results in [17] show that random inputs force the dynamic system into action which leads to better system identification.

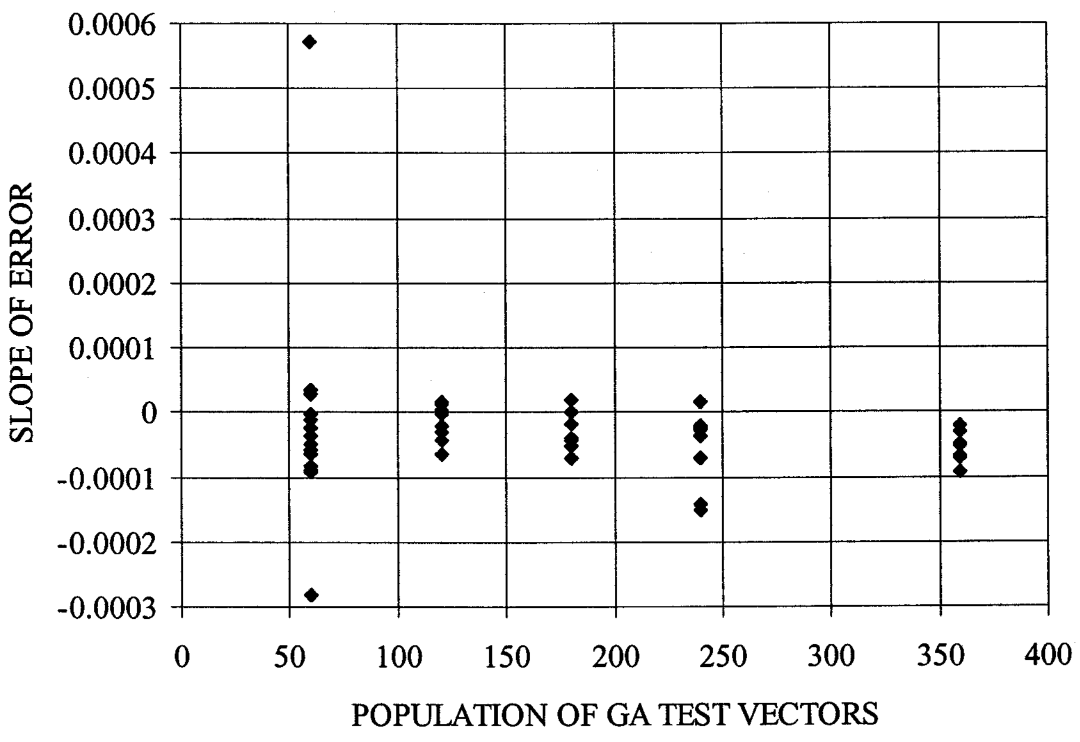


Figure 5-8
Slope of Error for Final 250 Data Points Versus Population of GA Test Vectors for All Cases of Neurons and Past States

5.2. Closed Loop Tests

Twenty cases of 1000 simulated hours were generated using 60 and 240 GA vectors. For each of these 40 cases, the average normalised revenue for the final 100 hours was calculated. The mean, maximum and minimum of these averages are summarised in Table 5-1. Note that in the best cases, the neural network approached the revenue achieved using the standard replacement schedule of one outage per shift. On average, the NN produced about 90% of the revenue of the standard, and at worst, the NN produces 80%.

Table 5-1 Normalised Revenue in Closed Loop Tests, Final 100 Hours

	60 GA Vectors	240 GA Vectors
Standard Replacement		
Mean	0.559	0.558
Max	0.562	0.562
Min	0.556	0.556
Neural Network Replacement		
Mean	0.495	0.497
Max	0.546	0.550
Min	0.441	0.453

This result may be deemed successful if other benefits of complete tele-remote operation is considered. In a potash mine, the travel time for workers to and from the production area is a significant fraction of one shift. If the mining machine could be fully automated and operated from the surface, the machine could be less efficient but yield a higher total revenue per shift since the non-productive travel time would be eliminated. This has been observed in other mining automation projects [18], [19] in which equipment controlled remotely from surface produce better than their manually operated counterparts, despite slower operating speeds and more frequent maintenance requirements.

One typical simulation run is shown in Figure 5-9 for 60 GA test vectors and Figure 5-10 for 240 GA test vectors. The final 100 simulation hours are shown. In these cases, the neural network method appears to become more successful in the latter stages of the simulation, suggesting the NN is learning to model the mining process better and produce a better bit replacement schedule. Other typical output plots are given in Appendix B.

All 40 test cases tend to approach the standard revenue benchmark. Complete divergence appears to be unlikely. This is partly due to the objective function that determines if the predicted revenue following a simulated shutdown is sufficient to warrant a real shutdown. If unrealistic expectations are programmed into this objective function, the controller may never elect to stop the machine and the revenue will diverge to negative infinity.

Generally, the performance of the test cases using 60 GA test vectors are not noticeably better than those using 240 GA test vectors. It is possible that 1000 simulated hours are not enough to visualise the difference. However, the cases using 240 GA vectors appear to correctly predict a rise in productivity following an outage more correctly. In Figure 5-9, note that between hours 935 and 960 the NN shuts the machine down several times, but there is no increase in revenue following the outage. During this period, the NN is incorrectly predicting at least a 5% increase. In Figure 5-10, a productivity increase generally does occur following the shutdown. This suggests that with more GA vectors the predictive capability of the NN system is improved.

An interesting feature in most cases is the peak revenue developed under neural network control. The peaks are generated after most of the bits are replaced in an outage, and the machine operates near maximum efficiency. These peaks suggest that it should be possible to operate the machine at a higher state of production than the standard method. However, because of the design of the machine model used in this project, the NN controller cannot choose to replace the bits, only to stop and allow an operator to inspect the bits and replace those that are worn. The model assumes that the bits are changed only after their pre-defined life expectancy has been reached. The only means the NN has to achieve high production rates is to first operate the machine long enough to wear out many of the bits. This tends to limit the ability of the NN method to maximise average machine revenue over time.

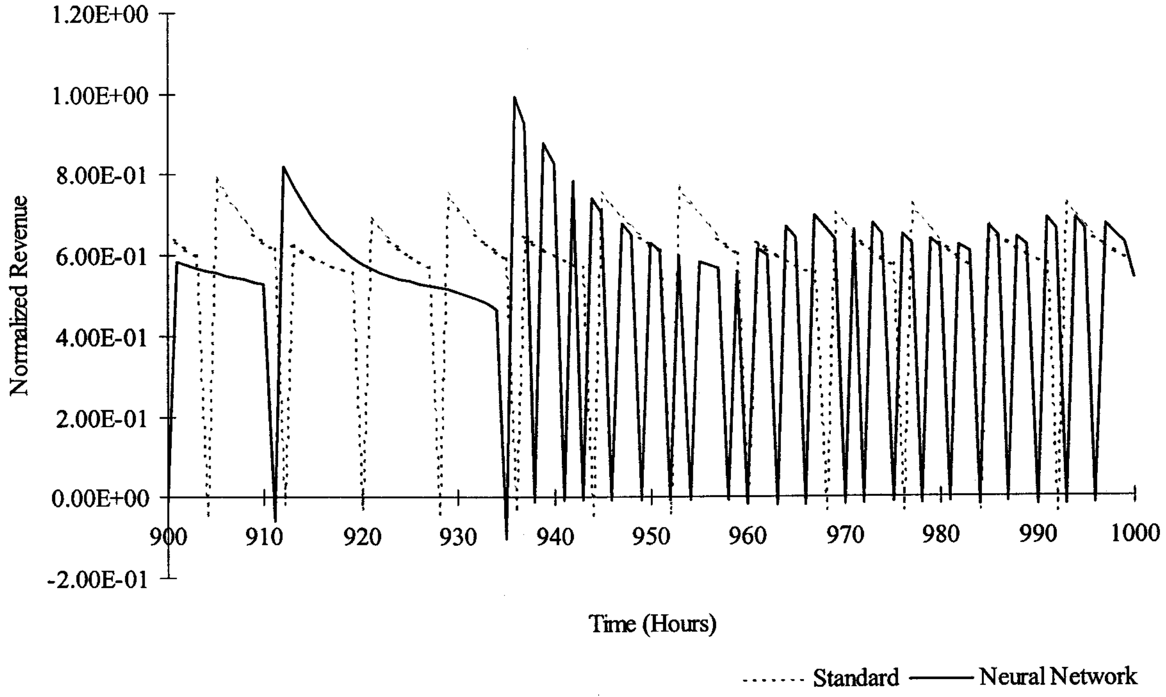


Figure 5-9 Revenue for Standard Versus Neural Network Replacement, Case 6, GA Test Vector Population of 60

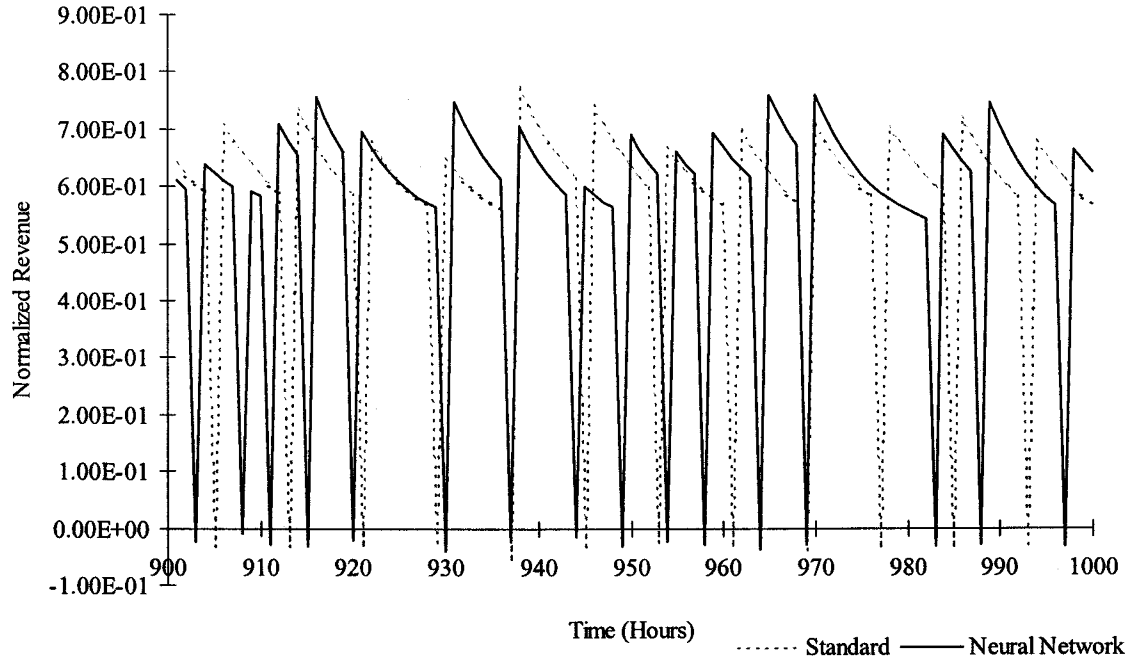


Figure 5-10 Revenue for Standard Versus Neural Network Replacement, Case 10, GA Test Vector Population of 240

6. CONCLUSIONS

The detection of worn cutting tools in continuous mining processes is inherently non-linear, dynamic, and is performed intuitively by experienced operators. Much effort has been put into automating mining operations to remove the human operator from the underground environment. A system that could automate the bit wear detection function would place the mining industry one step closer to removing the need for underground human operators.

In this project, a dynamic neural network (NN) system for automatically scheduling cutting tool replacement was developed. The NN was used to identify the dynamic mining process, including the revenue and cost of operating the system. The NN was then used to predict revenue or cost for upcoming time increments. Using the predicted revenue, a decision to shut down the machine for cutting tool replacement was made based on economics.

Neural network training was performed using genetic algorithms (GA) on line during the simulations. The GA technique did not require back propagation of the NN. This resulted in fast forward computation during training, and allowed for some experimentation with the NN structure and non-linear functions without the need for rewriting software. The combination of a dynamic NN with GA training proved to be very effective for real time system identification.

The system was tested in two stages. Open loop tests were conducted to determine the ability of the NN to predict future system outputs. The system was able to calculate system outputs two time steps into the future. A range of NN structures and GA populations were tested to determine the best combination for closed loop testing.

The closed loop tests showed the neural network method could be utilised to determine economically if the mining machine should be stopped for bit replacement. The result is interesting since the NN did not achieve complete system identification during the simulations. However, the NN does provide a sufficient model at any given time to predict the machine output a short time into the future.

The method was capable of automatically scheduling cutting tool replacement outages cost effectively, though it did not outperform conventional bit replacement. However, the reduced level of underground staffing and associated increased productivity could justify the implementation of this method.

REFERENCES

[1] Stephen J. Fortney, James L. Lewis, "Advanced Mining Machine Automation in Potash", *Mining Automation: 4th Canadian Symposium,* Saskatchewan Research Council, August 1990, pp 9-18.

[2] Philip J. Rowsell, Martin D. Waller, "An Intelligent Drill", *Mining Automation: 4th Canadian Symposium,* Saskatchewan Research Council, August 1990, pp 135-142.

[3] Micheal J. Pazuchanics, Gary L. Mowrey, "Recent Progress in Discriminating between Coal Cutting and Rock Cutting with Adaptive Signal Processing Techniques", *Report of Investigations (United States Bureau of Mines) 9475,* United States Department of the Interior, 1993.

[4] K.G. Wesa, J.N. Wilson, D.J. Gendzwill, "Clay Seam Sensing in Potash Mines", *Mining Automation: 4th Canadian Symposium*, Saskatchewan Research Council, August 1990, pp219-230.

[5] A. Ghasempoor, T.N. Moore, J. Jeswiet, "On-line Wear Estimation Using Neural Networks", *Journal of Engineering Manufacture, Proceedings of the Institution of Mechanical Engineers*, vol 212 no. B2 1998.

[6] K. Ramamurthi, C.L. Hough, Jr., "Intelligent Real-Time Predictive Diagnostics for Cutting Tools and Supervisory Control of Machining Operations", *Transactions of the ASME*, Vol. 115, August 1993, pp 268-277.

[7] Choon Seong Leem, D.A. Dornfeld, S.E. Dreyfus, "A Customized Neural Network for Sensor Fusion in On-Line Monitoring of Cutting Tool Wear", *Transactions of the ASME*, Vol. 117, May 1995, pp 152-159.

[8] S.J. Wilcox, R.L. Reuben, "The Detection of Short Time-scale Tool Degradation Events During Face Milling Operations Using Cutting Force and Acoustic Emission", *Proceedings of the Institution of Mechanical Engineers*, Vol. 208, 1994, pp 205-215.

[9] G. Pepe, L. Looney, M.S.J. Hashimi, L.M. Galantucci, "Predicting the Wear Resistance of WC-Co Coatings using Neural Networks", *Proceedings of the IASTED International Conference on Artificial Intelligence, Expert Systems, and Neural Networks*, 1996, pp 260-263.

[10] Matthew N. Plis, Carl F. Wingquist, Wallace W. Roepke, "Preliminary Evaluation of the Relationship of Bit Wear to Cutting Distance, Forces, and Dust Using Selected Commercial and Experimental Coal- and Rock-Cutting Tools", *United States Department of the Interior Bureau of Mines RI 9193*, 1988.

[11] C.E. Aboujaoude, L.K. Daneshmend, J.P. Peck, “Control of Rotary Blasthole Drills (I): Modeling and Simulation”, *Proceedings of the 5th Canadian Symposium on Mining Automation*, 1992, pp 12-32.

[12] C.E. Aboujaoude, L.K. Daneshmend, J.P. Peck, “Control of Rotary Blasthole Drills (II): Controller Design and Validation”, *Proceedings of the 5th Canadian Symposium on Mining Automation*, 1992, pp 33-52.

[13] M. M. Gupta, “Neural Computing Systems (Instroduction to Theory and Applications of Neural and Neuro-Fuzzy Systems)”, *Lecture Notes, Intelligent Systems Research Laboratory*, University of Saskatchewan, 1996

[14] L. Davis, “Genetic Algorithms and Simulated Annealing”, Pitman Publishing, 1987.

[15] T.L. Seng, M.B. Khalid, R. Yusof, “Tuning of a Neuro-Fuzzy Controlled by Genetic Algorithm”, *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybrenetics*, vol. 29 no. 2, April 1999, pp 226-237.

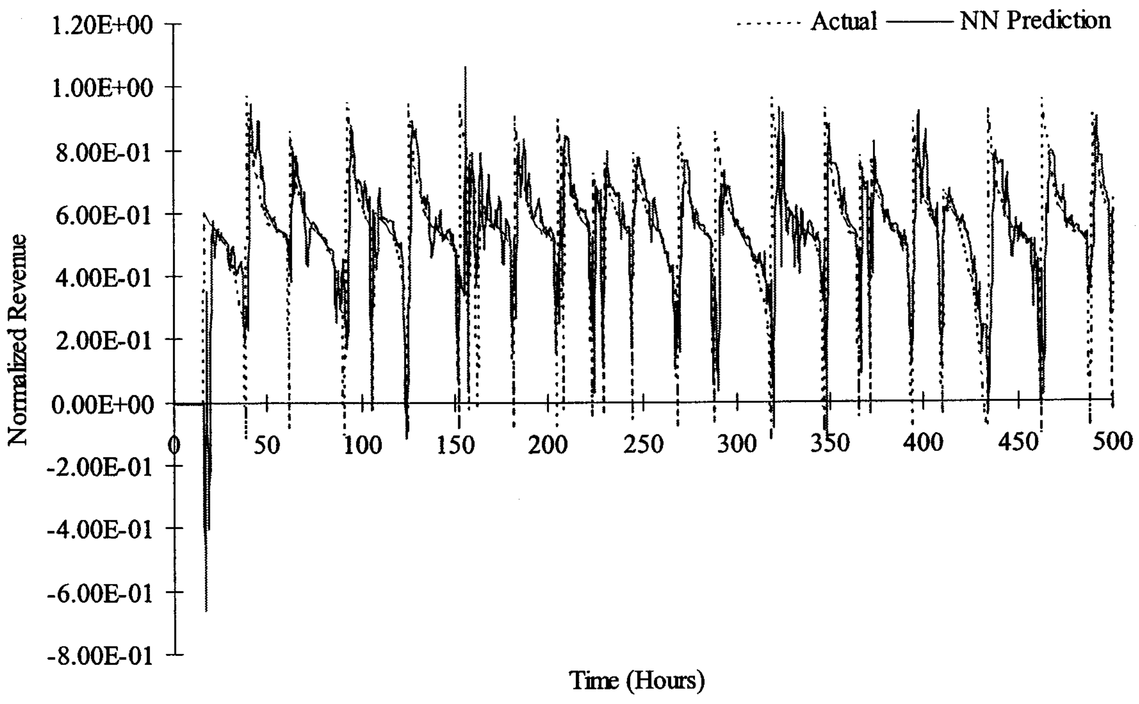
[16] D.T. Pham, D. Karaboga, “Training Elman and Jordan Networks for System Identification Using Genetic Algorithms”, *Artificial Intelligence in Engineering*, vol. 13 no. 2, April 1999, pp 107-117.

[17] P. Mars, J.R. Chen, R. Nambiar, “Learning Algorithms: Theory and Applications in Signal Processing, Control, and Communications”, CRC Press, 1996.

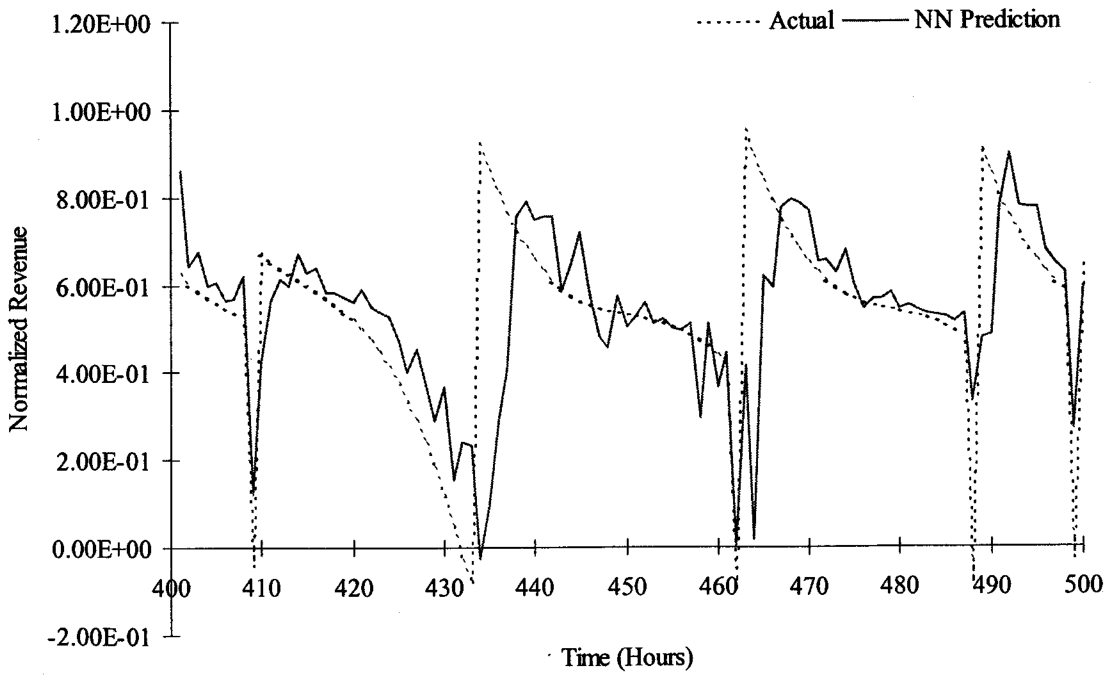
[18] Sverker Hartwig, Christoph Mueller, “Totally Remote Controlled Production Drill Rigs at LKAB: Experience from the first year of remote operation and consequences for future machine teleoperation”, *Proceedings of the 5th International Symposium on Mine Mechanization and Automation*, 1999 (CD-ROM).

[19] Greg Baiden, “Update on the Teleremote Operation from Surface of Scooptrams at INCO’s Stobie Mine”, *Proceedings of the 5th International Symposium on Mine Mechanization and Automation*, 1999 (CD-ROM).

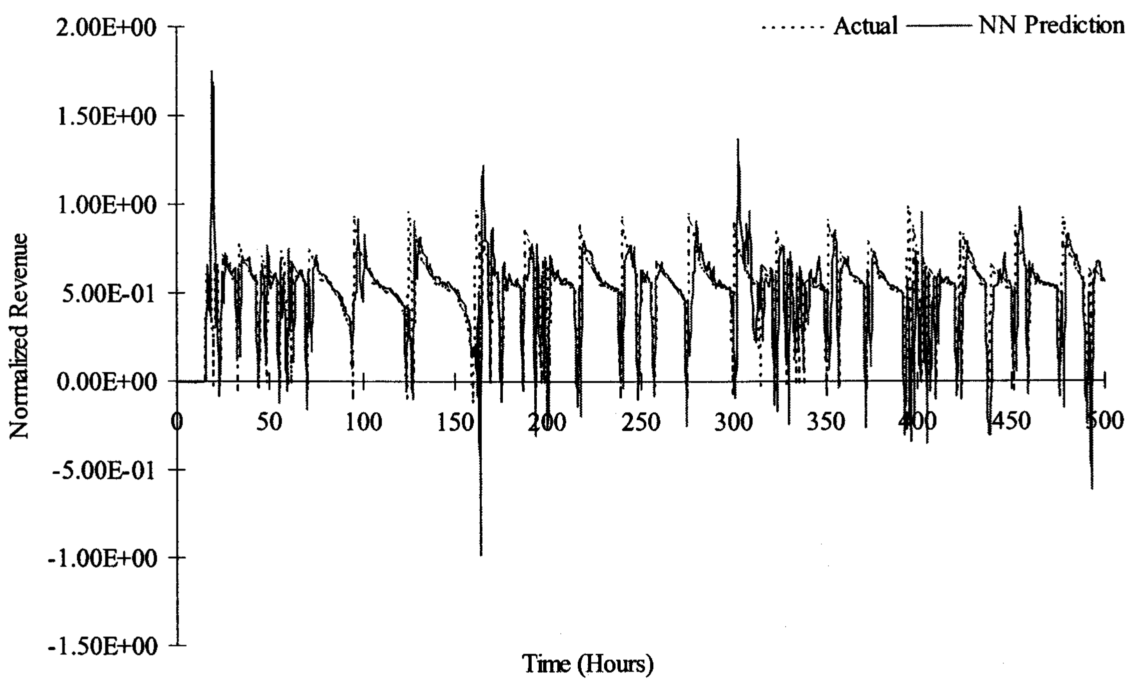
Appendix A - Selected Simulation Results of Open Loop Tests



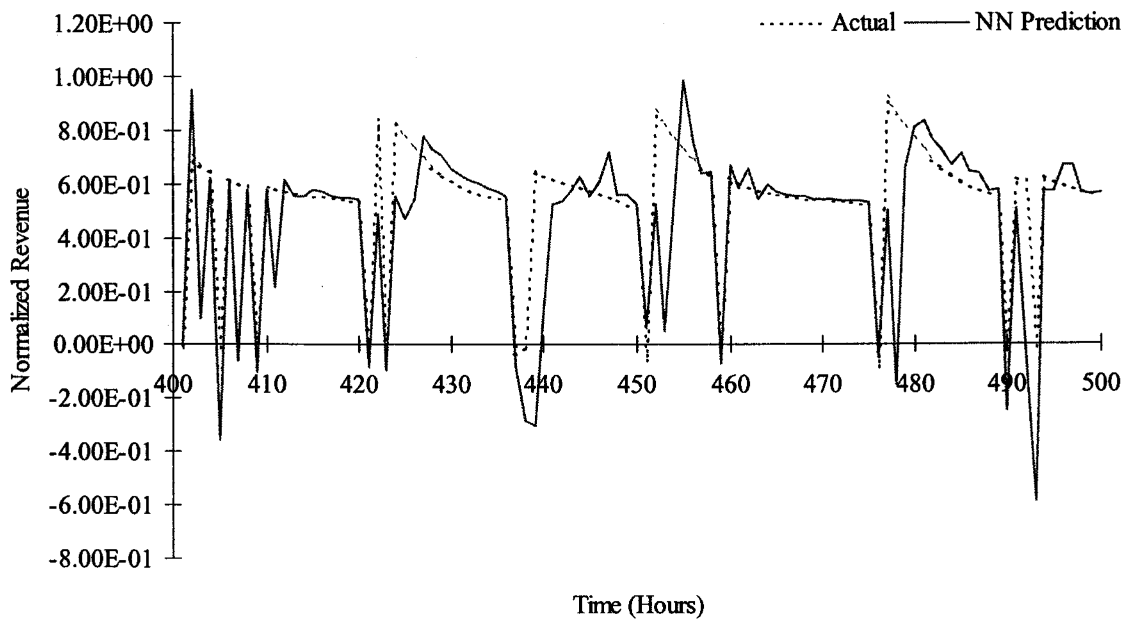
Open Loop Test, 4 First Layer Neurons, 240 GA Vectors



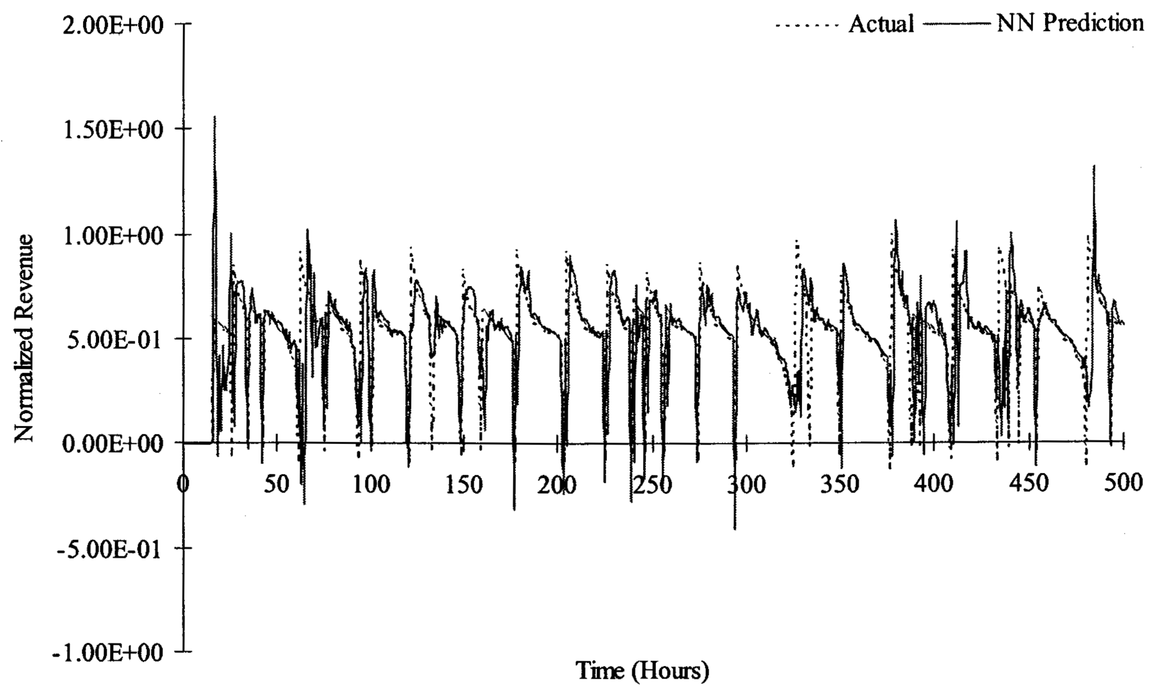
Open Loop Test, 4 First Layer Neurons, 240 GA Vectors, Final 100 Hours



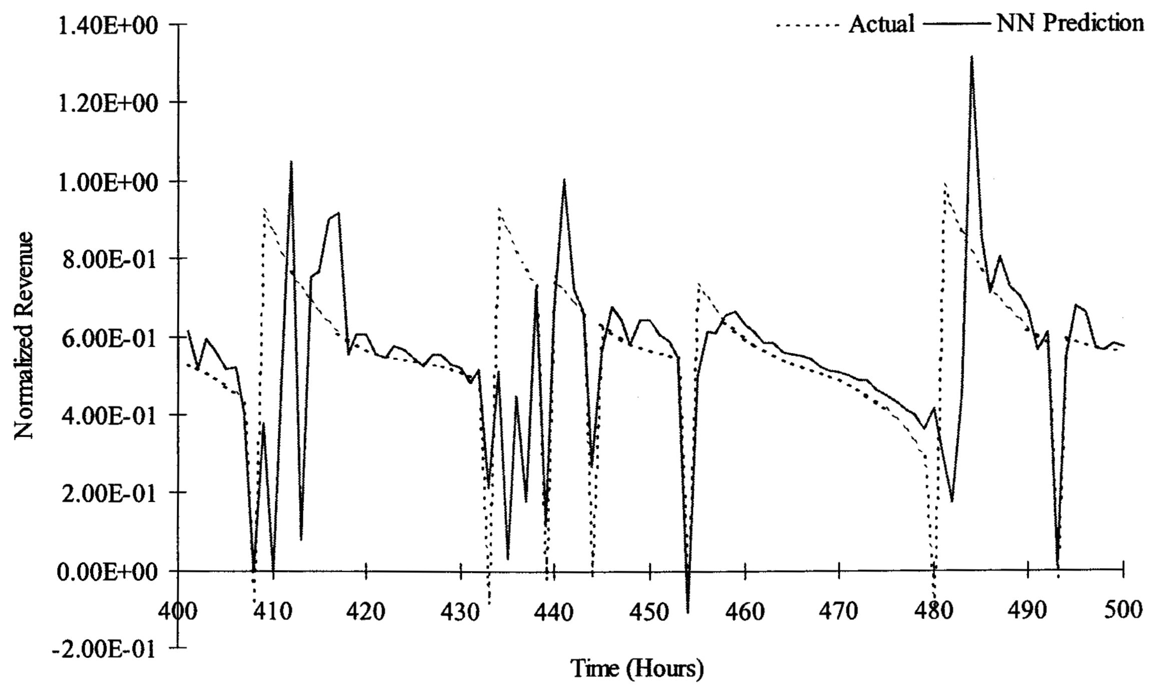
Open Loop Test, 8 First Layer Neurons, 60 GA Vectors



Open Loop Test, 8 First Layer Neurons, 60 GA Vectors, Final 100 Hours

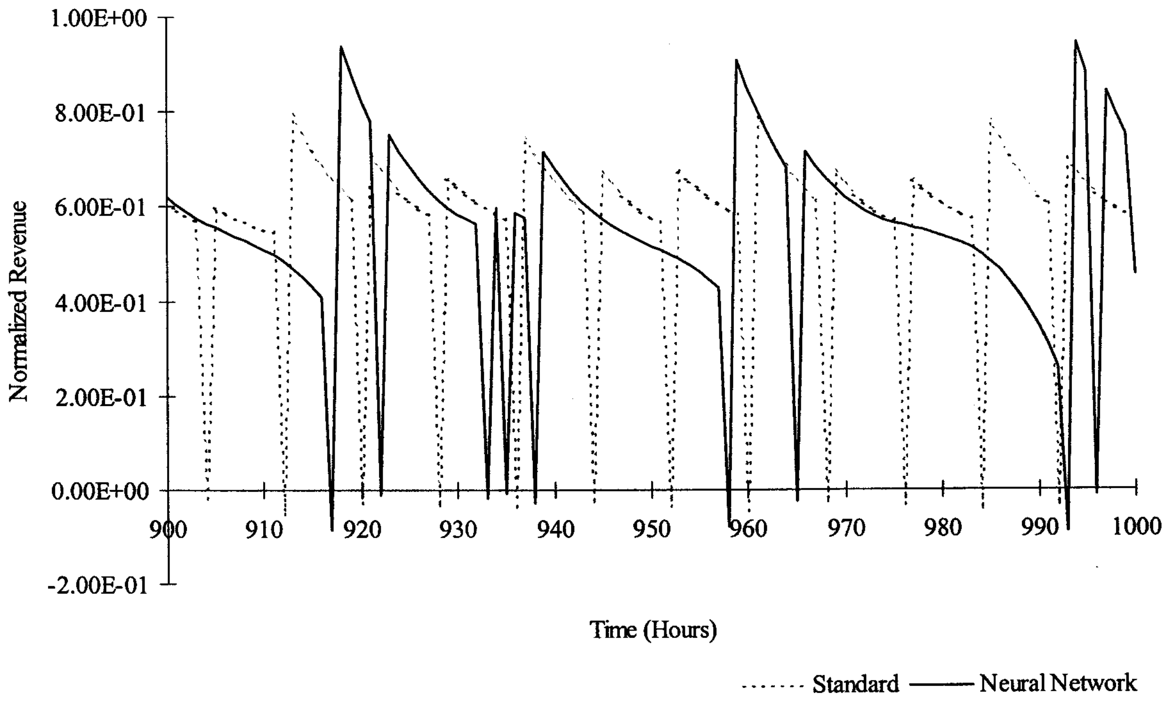


Open Loop Test, 8 First Layer Neurons, 120 GA Vectors

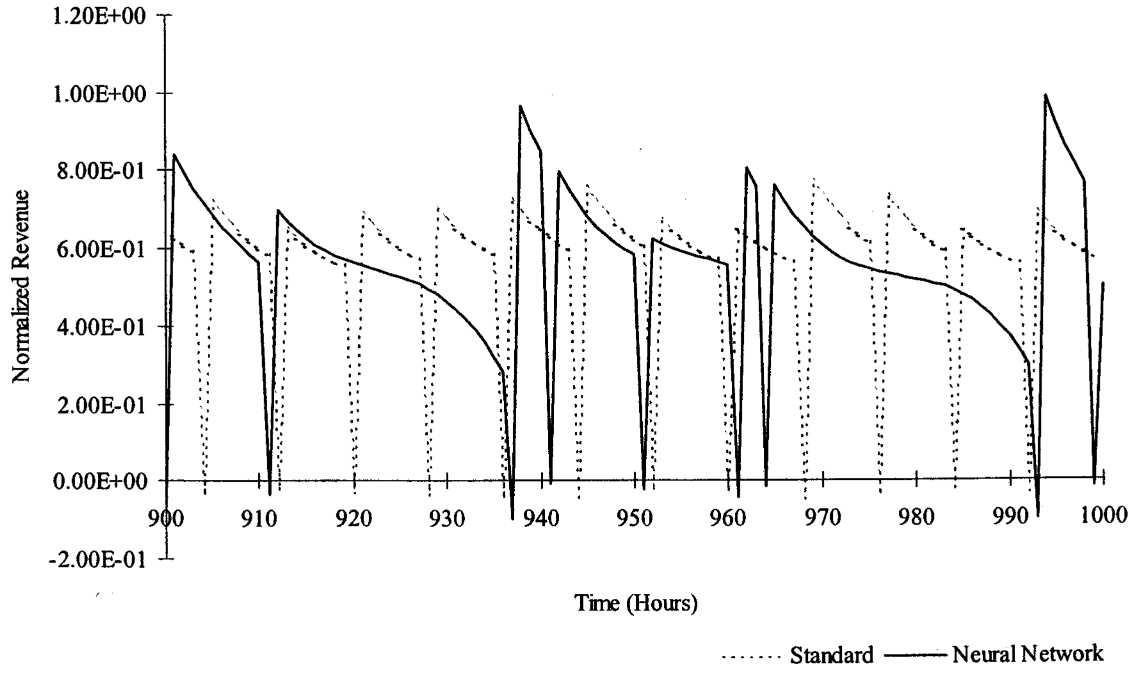


Open Loop Test, 8 First Layer Neurons, 120 GA Vectors, Final 100 Hours

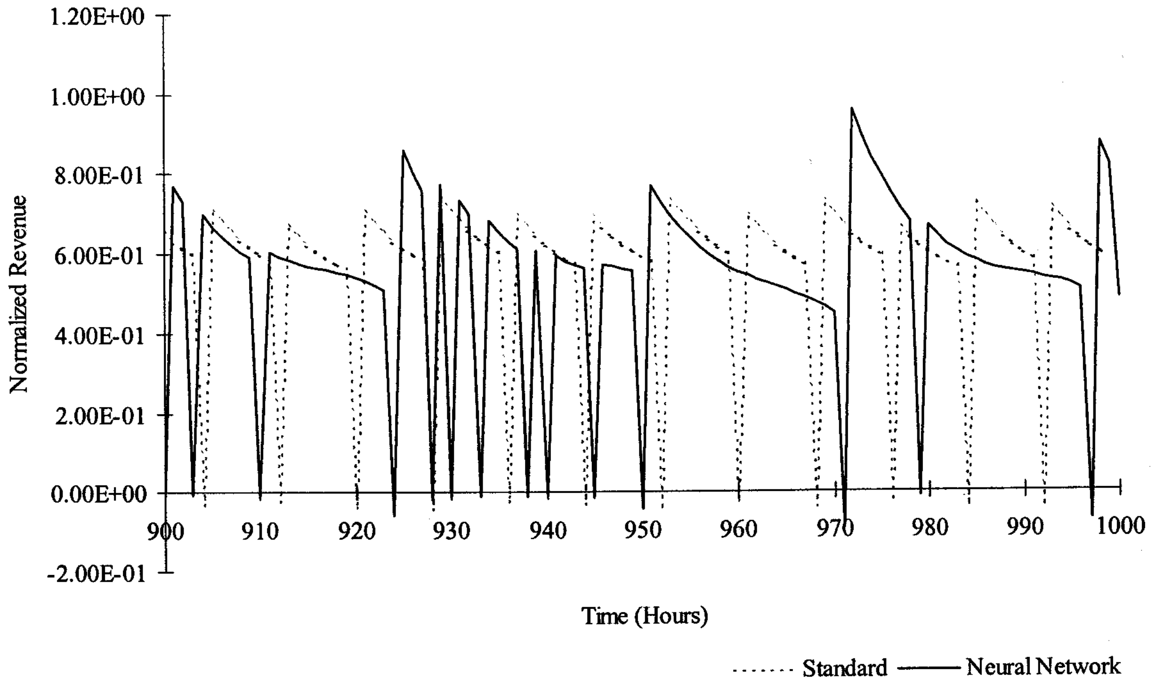
Appendix B - Selected Simulation Results of Closed Loop Tests



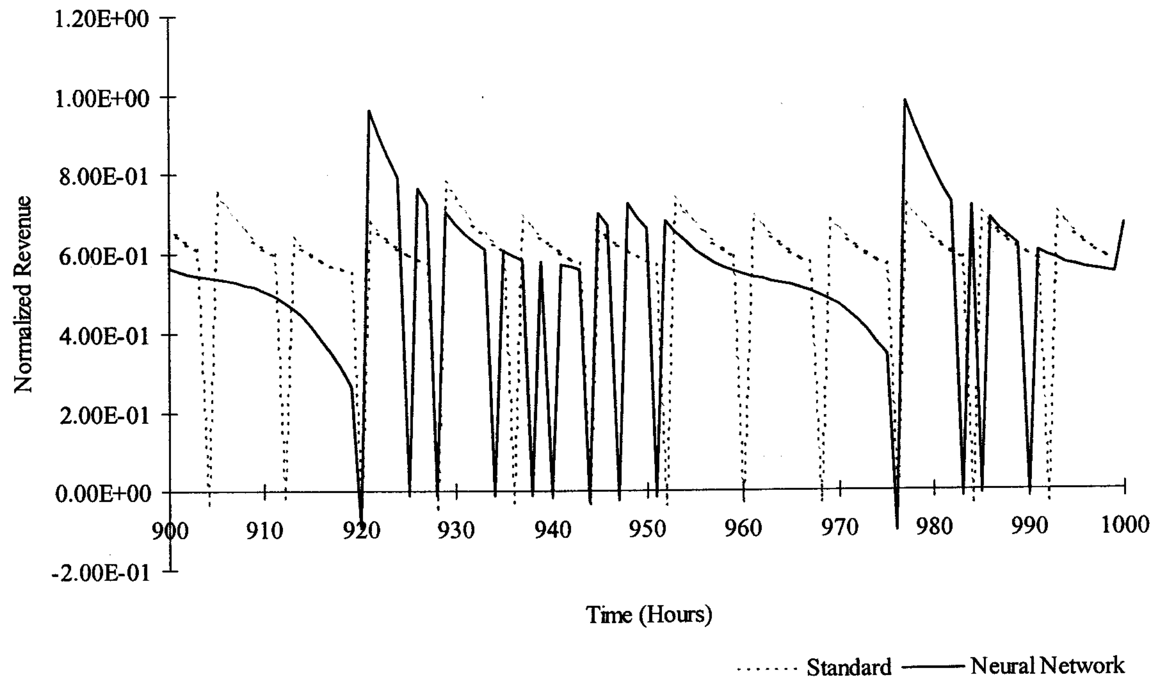
Closed Loop Test, 4 First Layer Neurons, 60 GA Vectors, Case 7



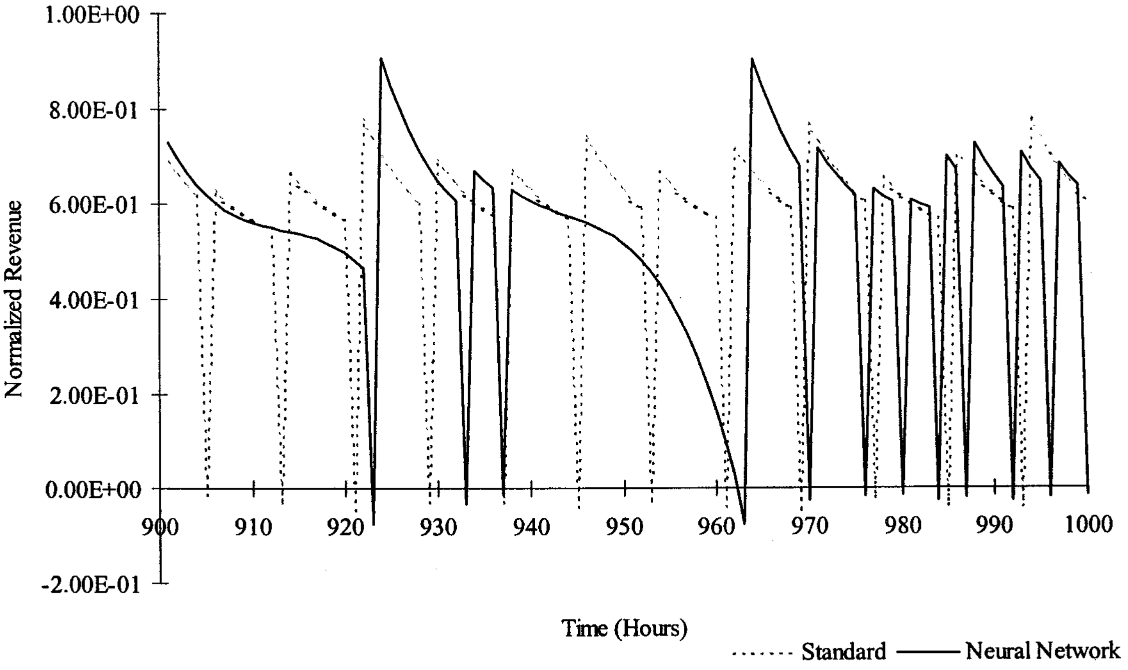
Closed Loop Test, 4 First Layer Neurons, 60 GA Vectors, Case 10



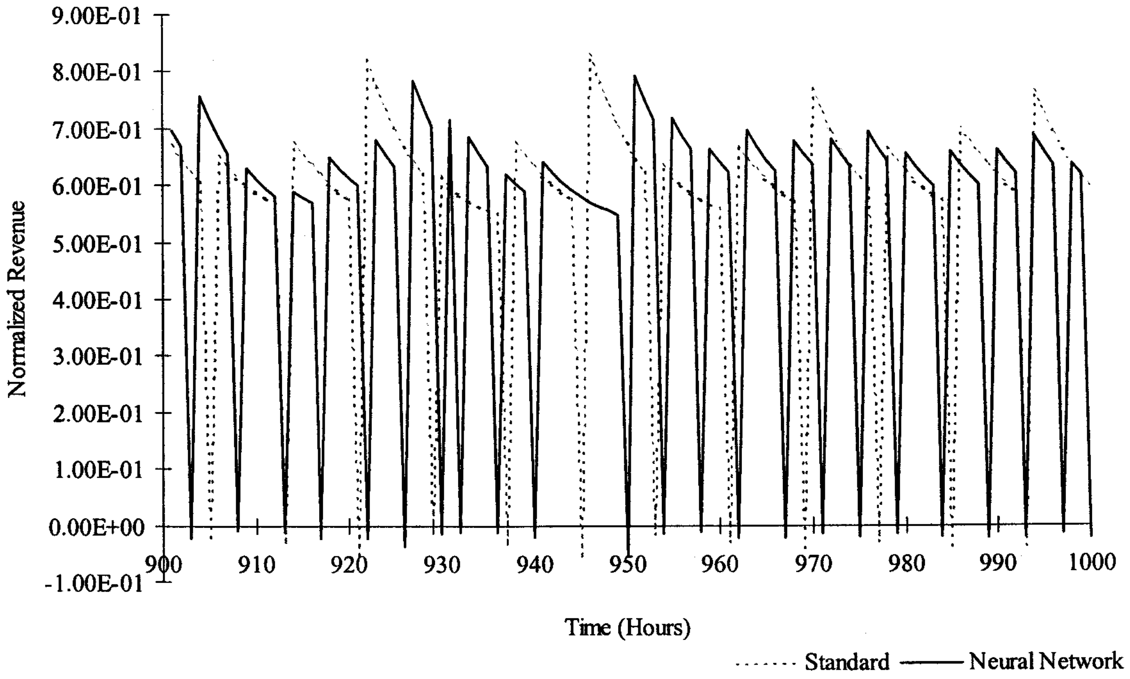
Closed Loop Test, 4 First Layer Neurons, 60 GA Vectors, Case 12



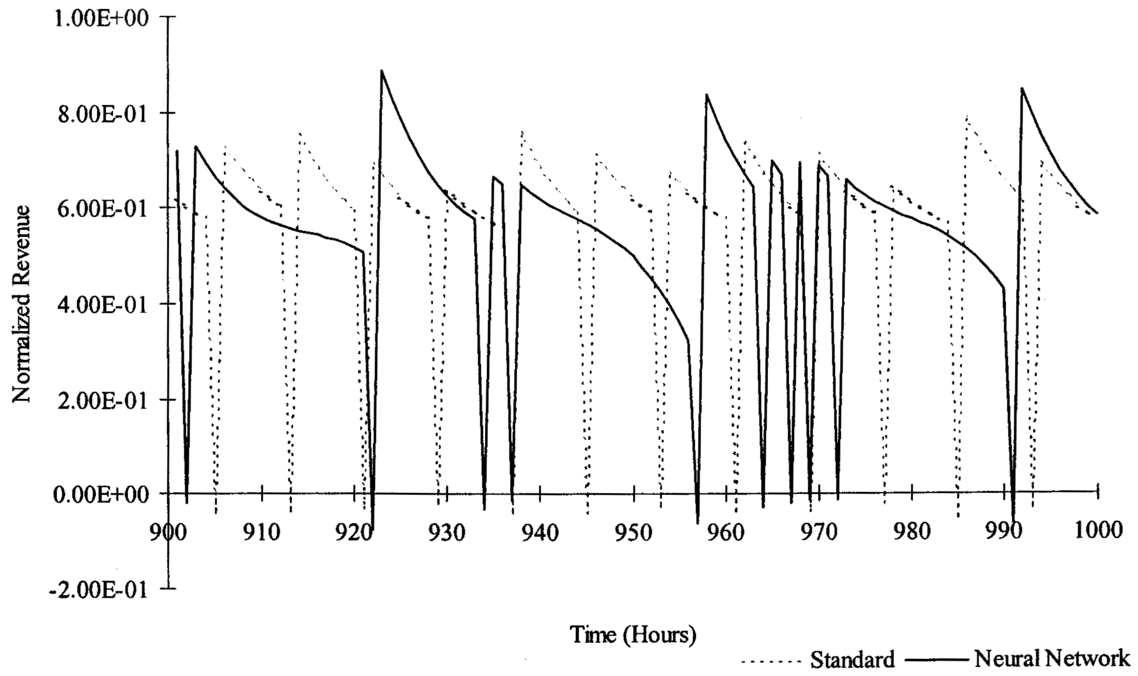
Closed Loop Test, 4 First Layer Neurons, 60 GA Vectors, Case 20



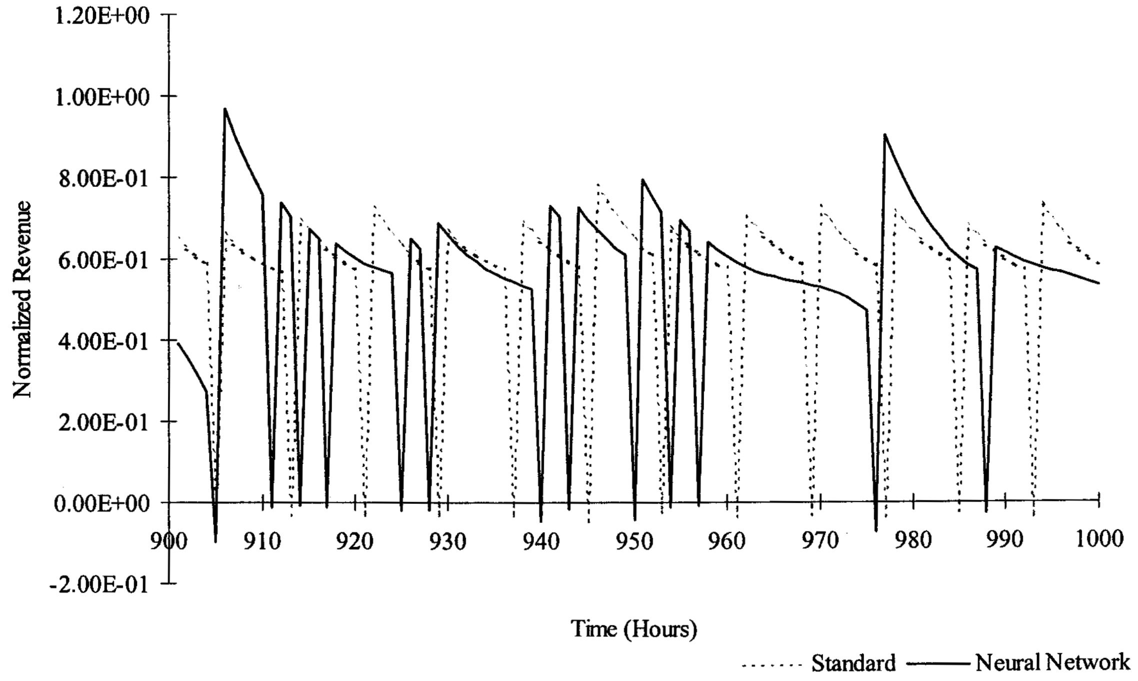
Closed Loop Test, 4 First Layer Neurons, 240 GA Vectors, Case 4



Closed Loop Test, 4 First Layer Neurons, 240 GA Vectors, Case 6



Closed Loop Test, 4 First Layer Neurons, 240 GA Vectors, Case 17



Closed Loop Test, 4 Neurons, 240 GA Test Vectors, Case 20