Global Optimization for Artificial Neural Networks: A Tabu Search Application

Randall S. Sexton, Bahram Alidaee, Robert E. Dorsey, and John D. Johnson[12]

[1] Randall Sexton is an assistant professor in the College of Business, Ball State University. Bahram Alidaee, Robert Dorsey and John Johnson are Associate Professors in the College of Business, University of Mississippi. Dr's Johnson and Dorsey are supported in part by the Mississippi Alabama Sea Grant Consortium through NOAA.

[2]The authors would like to express their appreciation to two anonymous referees for comments that significantly improved this paper.

Global Optimization for Artificial Neural Networks: A Tabu Search Application

ABSTRACT

The ability of neural networks to closely approximate unknown functions to any degree of desired accuracy has generated considerable demand for Neural Network research in Business. The attractiveness of neural network research stems from researchers' need to approximate models within the business environment without having a priori knowledge about the true underlying function. Gradient techniques, such as backpropagation, are currently the most widely used methods for neural network optimization. Since these techniques search for local solutions, a global search algorithm is warranted. In this paper we examine a recently popularized optimization technique, Tabu Search, as a possible alternative to the problematic backpropagation.

A Monte Carlo study was conducted to test the appropriateness of this global search technique for optimizing neural networks. Holding the neural network architecture constant, 530 independent runs were conducted for each of seven test functions, including a production function that exhibits both increasing and diminishing marginal returns and the Mackey-Glass chaotic time series, were used for a comparison of Tabu Search and backpropagation optimized neural networks. Tabu Search derived significantly superior solutions for in-sample, interpolation, and extrapolation test data for all seven test functions. It was also shown that fewer function evaluations were needed to find these optimal values.

Introduction

Rising interest in applying artificial neural networks (ANN) to business applications is evident in both the academic and trade literature. This popularity is driven by the ability of the ANN to accurately approximate unknown functions and their derivatives (Funahashi, 1989; Hornik et al., 1989). Most applications of ANNs use some variation of the gradient technique, backpropagation (LeCun, 1986; Parker, 1985; Rumelhart, 1986) for optimizing the networks (Salchenberger, 1992; Werbos, 1993). The past success of backpropagation for optimizing neural networks is undeniable, but it has also been shown to be inconsistent in its application and unpredictable due to the local nature of its search (Archer & Wang, 1993; Hsiung et al., 1990; Lenard et al., 1995; Madey & Denton, 1988; Rumelhart et al., 1986; Subramanian & Hung, 1990; Wang, 1995; Watrous, 1987; White, 1987).

An ANN is a flexible form approximation method loosely patterned after the connectionist structure of the brain. The standard feedforward ANN examined in this paper, has a series of input nodes that comprise the input layer of the ANN. Each of these nodes is used for the explanatory variables. A second layer of nodes, the hidden layer, are each connected to all of the input nodes. Thus, the input to each of the hidden layer nodes is a weighted sum of the input nodes. The weights applied to the inputs are different for each hidden node. Each hidden node transforms its weighted sum by means of a nonlinear function. In this study, the nonlinear function used for each node is the standard sigmoid function:

$$h_{ij} = \frac{1}{1+e^{-\sum_k \gamma_{jk} x_{ki}}}$$

where h_{ij} is the ith output from the jth node, γ_{jk} is the weight if the jth node applied to the kth input and x_{ki} is the ith value of the kth input. The third layer of the ANN is the output layer and for this study it consists of a single node. This output node is the weighted sum of the values coming from the hidden nodes. The optimization problem is to search for the set of weights that allows the ANN to most accurately fit the data.

Since gradient techniques like backpropagation converge locally, they often become trapped at sub-optimal solutions depending upon the serendipity of the initial random starting point. Since obtaining an optimal

solution is the goal of ANN training, a global search technique seems more suitable for this difficult nonlinear optimization problem (Shang & Wah, 1996). Although, there have been many attempts to correct the limitations of backpropagation (Chen & Mars, 1990; Franzini, 1987; Holt & Semnani, 1990; Izui & Pentland, 1990; LeCun et al., 1989; Lee & Bien, 1991; Matsuoka & Yi, 1991; Plaut et al., 1988; Samad, 1990; Shoemaker et al., 1991; Sietsma & Dow, 1991; Solla et al., 1988; van Ooyen & Nienhuis, 1992; Weigend et al., 1990) the problems still exist.

The popularity of tabu search has grown significantly in the past few years as a global search technique. Although, most of the applications using this technique have been combinatorial problems, such as, the traveling salesmen problem, design optimization, and the quadratic assignment problem (Knox, 1989; Bland, 1991; Skorin-Kapov, 1990), there have been a few attempts to use it for the continuous problems (Bland, 1993; Bland, 1994). This paper extends this line of research by applying Tabu search to the difficult problem of neural network optimization. Here Tabu Search (TS) is compared with backpropagation for ANN optimization. The following section briefly describes the search algorithms, followed by a discussion of the Monte Carlo experiments and the conclusion.

Search Algorithms

Two Tabu Search algorithms were used for comparison with backpropagation trained networks. The first algorithm included only certain basic components of Tabu Search and was used as a baseline for comparison with the recent improvements that have been suggested in the literature. The second Tabu Search algorithm included many of these improvements, such as variable tabu list sizes, systematic diversification and intensification processes for neighborhood creation, and program termination based on the frequency of finding better solutions. Descriptions of the algorithms used in this study are provided in the following three sections.

Preliminary Tabu Search Algorithm

Glover initially introduced (1986) and later developed Tabu Search (TS) into a general framework (1988). Independently, Hansen (1987) proposed the Steepest Ascent, Mildest Descent (SAMD) algorithm which uses

similar ideas. Tabu Search can be thought of as an iterative descent method. An initial solution is randomly generated and a neighborhood around that solution is examined. If a new solution is found in the neighborhood that is preferred to the original, then the new solution replaces the old and the process repeats. If no new solution is found that improves upon the old function evaluation, then unlike a gradient descent procedure which would stop at that point, (a local minimum), the TS algorithm may continue by accepting a new value that is worse than the old value. Therefore, a collection of solutions in a given neighborhood is generated and the final solution would be based on the best solution found so far for that particular neighborhood. To keep from cycling, an additional step is included that prohibits solutions from recurring for a user defined number of solutions. This *tabu list* is generated by adding the last solution to the beginning of the list and discarding the oldest solution from the list. During this procedure the best solution found so far is retained. From the subset of acceptable neighborhoods the best solution is chosen. In a problem in which there is no known solution, a given value for the maximum number of iterations can be used to terminate the process. When this number of iterations has taken place with no improvement over the best solution, the algorithm will terminate.

Since the solution space for any ANN is complex, and the possibility of many local solutions is great, it appears reasonable that by increasing the number of neighborhoods (NH) searched, the probability of finding a global solution would increase. To evaluate this conjecture, three different levels for the number of neighborhoods (NH) were used (100, 200, 500). As implemented for this problem, the neighborhoods are simply randomly drawn points from a uniform distribution. The number of local searches (NS) within each neighborhood was also set at these same three levels. Local searches were conducted as random searches within the neighborhood. The neighborhood was a region restricted to ± 0.1% for each parameter value (weight) in the initial neighborhood point. No attempt was made to increase or decrease the size of the neighborhood search at this time. As mentioned earlier, this algorithm was used as a baseline for future development of the Tabu Search algorithm.

Another parameter used in TS is the *tabu list* (TL). By increasing the size of the *tabu list* the possibility of rejecting new moves in increased and the algorithm is forced to find new solutions around the current point which have not recently been tested. The size of the Tabu list was also evaluated at three levels (1, 50, 100) and was maintained for both the objection function values and the corresponding solutions. This range is much larger

than is normal in the literature and was chosen primarily to observe the effect of such a wide range on the behavior of the algorithm. Since this research deals with real values, the likelihood of finding solutions that are identical is extremely small, so in order for a *tabu list* to be used, a proximity criterion was given. This *tabu criterion* (TC) relaxes the strict comparison of the new solution to the solutions in the *tabu list*. So, by implementing this TC a solution would be considered tabu if the new solution falls in a given range of the solutions in the list. For instance, if the TC were set to .01, the new solution of weights would be compared to the TL weights within ± .01% of the corresponding weights. To be rejected, all the weights for the new solution would need to be within this range for any of the solutions in the TL. The TC was set at two levels (.01, .03).

An aspiration level condition was also included which permits overriding the rejection of a new solution that meets the *tabu criteria*. As a solution to a problem converges to a minimum, there may be increasingly smaller decreases in error reduction. So, to allow for this possibility, a new solution that would normally be considered *tabu* would be accepted if this new solution happens to be better than the best solution found so far, no matter how small the improvement. A pseudo code description of this preliminary TS algorithm is shown in the Appendix.

Extended Tabu Search Algorithm

The preliminary TS algorithm above was used as a baseline in order to determine its potential for optimizing ANN's. Since the preliminary results using the preliminary TS algorithm were encouraging, an extended TS algorithm was constructed incorporating a number of recently suggested modifications. The extended TS algorithm developed for this research was based on modifications suggested in Glover (1990), Crainic, Gendreay, Soriano and Toulouse (1991), Glover and Laguna (1993), Kelly, Golden and Assad (1993) and Laguna, Kelly, González-Verlarde and Glover (1995). An explanation of this algorithm follows.

Consider a function *f* of *n* variables, x, where x is a vector of weights corresponding to the input layer of an ANN[3]. An initial solution x_0 is randomly drawn from a uniform distribution in the range of [-10,10]. The

[3]Search takes place only over the input layer parameters for Tabu search since the output weights can be obtained simply with ordinary least squares.

function value $f(x_0)$ (SSE of the ANN) is then calculated. Since x_0 is the current minimum solution and $f(x_0)$ is the minimum function value, both are used to set the best solution and function value parameters, $x_{best} = x_0$ and $f(x_{best}) = f(x_0)$, as well as entered into separate *tabu lists*.

The next solution x_1 is then randomly drawn from the same distribution and checked for acceptance by checking the aspiration level and tabu conditions. The aspiration level is checked by directly checking the $f(x_{new})$ function value with $f(x_{best})$. If $f(x_{new}) < f(x_{best})$ then the point is automatically accepted and both x_{best} and $f(x_{best})$ are updated, as well as entering both in their respective *tabu lists*. If $f(x_{new}) > f(x_{best})$ then the tabu conditions are tested for solution acceptance.

The tabu conditions are twofold, with the first condition predetermining the need for the second condition. If $f(x_{new})$ is within ± 0.01 of any value in the function value *tabu list*, the second tabu condition would then be applied, otherwise, the point is accepted and the respective *tabu lists* are updated. However, if the $f(x_{new})$ is within this tolerance for some point in the function value *tabu list* the new solution becomes suspect and a check for specific solution weights of x_{new} and the weights for the point which corresponds with the tabu list function value which caused suspicion is necessary. If all of the weights in x_{new} are within ± 0.01 from this tabu list point, the point is rejected and a new solution is randomly generated, otherwise the point is accepted and x_{new} and $f(x_{new})$ are entered into their respective *tabu lists*, dropping the oldest solution and function values from the list. This process continues for 1,000 iterations of accepted solutions.

The tabu list size is variable, depending upon the number of current tabu moves that have taken place during the search. As long as there are no tabu moves, the list size is set to 1,000. Once tabu moves are detected, the list decreases at the rate of (1,000 / # of tabu moves), to the nearest integer. If more than 1,000 tabu moves occur the list size is set to one.

Once 1,000 randomly drawn solutions have been accepted another 1,000 iterations begin, saving both tabu lists and the best solution. This process will continuously repeat if at least one $f(x_{new})$ is found to be the best function value so far, within a 1,000 iteration cycle. Once a 1,000 iteration cycle is finished without a reduction in $f(x_{best})$ the algorithm accepts this best solution as the initial point in which to start the neighborhood search.

The neighborhood search is based upon an alternating diversification and intensification processes. The intensification process attempts to zoom in on the best solution in order to converge upon the best solution found. Instead of randomly drawing new solutions, the focus is upon the x_{best} solution. The new points are drawn by modifying the x_{best} solution by a small *step* value. This *step* value is calculated by the following formula.

$$step = \frac{(0.1(x_{best(i)}) \& (0.2(x_{best(i)}(random)}{change}$$

The *random* variable is a random number drawn from a uniform distribution from the range of [0,1]. The *change* variable indicates the change from one process to another. Since the example shows the first process, the *change* variable is set to one. As the algorithm proceeds and processes change from intensification to diversification and back, the algorithm continues to decrease the size of changes to the best solution vector for the intensification process. This process continues as long as there is at least one reduction in the $f(x_{best})$ for a 1,000 iteration cycle up to a maximum of 20 repeats . Although, the algorithm would have continued to improve the solution, it was unnecessary to refine the solution to that degree for this demonstration. Once there is a 1,000 iteration cycle without a reduction in $f(x_{best})$ the diversification process begins, adding one to the *change* variable.

The diversification process attempts to expand the search area and only differs in the *step* formula. Instead of decreasing the size of weight changes of the x_{best} solution, the diversification process increases the values, thereby widening the search space around the best solution in order to escape local optima.

$$step = [(0.1(x_{best(i)}) \& (0.2(x_{best(i)}(random)](change$$

As the change process continues, the size of changes to the best solution vector increases. This process continues as long as there is at least one reduction in the $f(x_{best})$ for a 1,000 iteration cycle. Once there is a 1,000 iteration cycle without a reduction in $f(x_{best})$ the process turns back to a intensification phase, adding one to the loop variable.

These alternating intensification and diversification processes continue until there are five changes each, which terminates the neighborhood search around the x_{best} solution. At which time the whole process begins

again. The tabu lists are saved for comparison with the next search but the x_{best} and $f(x_{best})$ are reinitialized in order to search for new solutions in the weight space. The whole process repeats for 10 complete searches. Pseudo code for this algorithm is provided in the Appendix.

Backpropagation

Backpropagation was first introduced by Werbos (1974), Parker (1985), LeCun (1986), and Rumelhart, *et al* (1986a, 1986b) and later modified in various manners by numerous researchers in order to overcome its deficiencies. A backpropagation optimized neural network learns by using the generalized delta rule in a two-phase propagate-adapt cycle. The initial randomly drawn weights are applied to the NN in order to generate a starting point in which to begin the search. At this initial point the estimates are then compared to the desired output, and an error is computed for each of the given observations. The error for the first observation is simply calculated by subtracting the estimated value from the true value. The sum of squared errors is generally used as the objective function**.** In order to determine which direction to change the weights, the negative of the gradient, with respect to the weights, is calculated. The weight values are then adjusted by a magnitude proportional to the negative gradient. Thus, the objective function must be differentiable. This error is then transmitted backward from the output layer to each node in the hidden layer. For each hidden node only a portion of the error is received, based on the relative contribution of each hidden node to the given estimate. This process continues, layer by layer, until each node has received its error contribution. Once the errors have been assigned, the weights are adjusted in order to converge toward a solution.

Monte Carlo Study

The Monte Carlo comparison was conducted on five separate data sets for each of the seven test problems listed below. Each of the data sets was independent and randomly drawn. For the first five test problems, 50 observations were used as the training set and 150 observation pairs were used for out of sample interpolation and extrapolation.

1) $Y = X_1 + X_2$
2) $Y = X_1 X_2$
3) $Y = \frac{X_1}{*X_2* + 1}$
4) $Y = X_1^2 - X_2^3$
5) $Y = X_1^3 - X_1^2$
6) $Y_t = Y_{t-1} + 10.5\left[\frac{.2Y_{t-5}}{1 + (Y_{t-5})^{10}} - .1Y_{t-1}\right]$
7) $Y = a_1\arctan(x_1 - ß_1) + a_2\arctan(x_2 - ß_2) + a_3\arctan(x_3 - ß_3) + ?$
   where $? = -a_1\arctan(-ß_1) - a_2\arctan(-ß_2) - a_3\arctan(-ß_3)$

The training and interpolation data were generated by randomly drawing the values from the sets X_1 e [-100,100] and X_2 e [-10,10]. The extrapolation test sets were generated from X_1 e [-200, -101] and X_1 e [101, 200] and for X_2 e [-20, -11] and X_2 e [11, 20]. The training data was normalized from -1 to 1 for both algorithms, in order to have identical training and output data for comparison. The same normalizing factor was used on the extrapolation data as was used for the training and interpolation data. The sum of squared errors (SSE) was used for optimization and the networks were compared on the basis of forecast error. Each network included six hidden nodes for all problems. A bias term was used on both the input and hidden layers so there was a total of 25 weights for problems one through four and 19 weights for problem five. There may well be better network architectures for the given problems, but since we are comparing the optimization methods, we chose a common architecture.

The sixth problem consisted of a discrete version of the Mackey-Glass equation that has previously been used in neural network literature (Gallant & White, 1992; Goffe et al., 1994). This chaotic series is interesting because of its similarity to economic and financial series found in financial markets. Because of its apparent randomness and many local optima, this function is a challenging test for global training algorithms. Again, we chose six hidden nodes for the neural network architecture. Five lagged values of the dependent variable were used as the input variables. Clearly, three of these were unnecessary but this information is often not known to the researcher and the ability of the ANN to ignore irrelevant variables is critical to its usefulness. Since there are five input variables, the total number of parameters included 36 weights connecting the five inputs and one bias term to

the hidden nodes and seven weights connecting the hidden nodes and bias term to the output node, totaling 43 parameters overall. The five training sets of 100 observations each for problem six were generated from five randomly selected starting points. The interpolation data sets each consisted of 150 points that were generated from five different randomly chosen points.

The seventh problem is a production function that exhibits both increasing and diminishing marginal returns (Harder, 1971). Three independent variables (X_1, X_2, X_3) were randomly drawn from a uniform distribution from the range (0, 200). We generated five training sets of 100 observations each and five interpolation sets of 150 observations from this same range. The five sets of 150 observations for extrapolation testing were generated from randomly drawn values from (201, 400). Six hidden nodes were again chosen for consistency, resulting in 31 total parameters. The equation's constants were set to the values below.

$$a_1 = 5 \quad a_2 = 10 \quad a_3 = 15$$
$$ß_1 = 50 \quad ß_2 = 100 \quad ß_3 = 150$$

Training with Backpropagation

Commercial neural net software as well as software downloaded from the Internet were preliminarily evaluated in order to determine which package would give the best estimates for the given problems. These included, Neural Works Professional II/Plus by NeuralWare®, Brain Maker by California Scientific, EXPO by Leading Markets Technologies and MATLAB by Math Works (using both the backpropagation and the Marquardt-Levenberg algorithms). Although the performance was similar for all programs, Neural Works Professional II/Plus by NeuralWare®, a PC-based neural network application seemed to give the best estimates and was chosen for the test series. In training each problem using BP, there were four factors that were manipulated in an effort to find the best configuration for optimizing each problem. They included the learning rate, momentum, epoch size, and the logicon algorithm. The different combinations of the learning rate and momentum are introduced in order to try and find the right combination that will allow the solution to escape local minima but not skip over the global solution. The epoch is defined as the number of iterations before evaluation and weight adjustments. Normally the delta rule only allows adjustment after every training pair, but recent modifications of

the algorithm allow for larger epoch sizes. The Logicon algorithm was introduced by Gregg Wilensky and Narbik Manukian and was developed to converge faster by combining the advantages of closed and open boundary networks, into a single network. Each of these factors was varied in an effort to reduce the chances of becoming trapped in local minima. While rules of thumb for setting these values have been suggested, there is no set standard upon which a researcher can draw for deriving optimal configurations for training with backpropagation. Guidelines suggested by the Neural Works manual were used in selecting the values used.

Backpropagation Training Factors

We examined 16 different configurations of these parameters and the NN was trained with each configuration on each data set. Ten replications were performed for each configuration with each data set and in every case the NN was trained for 20,000 iterations. These combinations are shown in Table 1. Each replication was started with a new random seed. Thus, there were 160 training attempts for each data set for each problem or a total of 800 training attempts for each problem. Out of this set of 160 for each data set, the best weights (lowest sum of squared errors) and the corresponding configuration (parameter settings) were used as the starting point for additional training.

Table 1: Back propagation Parameters

Parameter	Values Used
Learning Rate (Step Value)	.5, 1
Momentum	.3, .9
Epoch Size	1, 50
Logicon Algorithm	OFF, ON

Each problem was then trained until the reduction of out-of-sample interpolation error stopped. At that point the weights were again saved and these weights were next used as the starting point for training an additional 100,000[4] iterations up to a total of 1,000,000 iterations for each of the seven problems. During this last

[4] Depending on where the error stopped decreasing some problems were trained more than 100,000 iterations in order to reach 1,000,000. However, all were trained at least 100,000 beyond the last reduction in error.

sequence of training, the error was checked every 10,000 iterations for a reduction. The set of weights that achieved the smallest errors was then saved for the comparison with the TS.

Training with the Preliminary Tabu Search Algorithm

The preliminary Tabu Search program described above was constructed based upon portions of Glover's (1990) Tabu Search tutorial as well as on de Werra & Hertz (1989) and Hertz & de Werra (1990). This code, was written in Fortran 77 and run on a 83-MHZ Pentium PC. There were four parameter adjustments that were made in training with TS. These included the number of neighborhoods (NH), number of local searches around each neighborhood (NS), the length of the *tabu list* (TL), and the tabu criteria (TC) or percentage of tolerance for rejecting a new solution. These parameters were configured into 36 different configurations[5] that were tested ten times each with new random seeds on each problem with each data set. Table 2 illustrates the different combinations. This preliminary TS test included a total of 360 replications for each data set for each problem.

Table 2 Tabu Search Parameters

Parameter	Values Used
Number of Neighborhoods (NH)	100, 200, 500
Local Searches around each Neighborhood (NS)	100, 200, 500
Tabu List Size	1, 50, 100
Tabu Criteria	.01, .03

Results with the Preliminary Tabu Search

Although BP is currently the most popular technique for ANN optimization, the preliminary TS algorithm found superior solutions for four out of the seven test problems, which included problems 1, 2, 4, & 5, for in-sample, interpolation and extrapolation. It is also interesting to note, that for all of the test problems, BP solution

[5]There are 54 different combination of these parameter values. Since this was a preliminary evaluation of performance, the following combinations of NH and NS were not examined, (100,100), (200,200) and (500,500).

variance was significantly higher, demonstrating that the TS algorithm was less likely to get stuck in bad local solutions. In comparing the time and function evaluations for found solutions (Table 3), TS was more efficient than BP. The comparisons were run on a Pentium 83 MHZ PC. The compiler used for the TS algorithm was Absoft Inc.'s 32 bit based Fortran 77 software. All TS solutions converged with significantly fewer function evaluations than the BP algorithm. The corresponding processor times also demonstrate the advantage of searching with TC. No pattern was found that would dictate the optimal configuration for the preliminary TS[6]. However, since this rudimentary algorithm demonstrated such promise relative to the most popular method of ANN optimization, further modifications of TS to incorporate additional features were proposed and tested.

Table 3 - Time and Function Evaluation Comparison (Data Set 1)

	PROCESSOR TIME ON PC 83 MHZ (sec)		FUNCTION EVALUATIONS	
PROBLEMS	Basic TS	BP	Basic TS	BP
1	141	2,017	100,00	1,000,000
2	141	2,017	100,000	1,000,000
3	145	2,017	103,532	1,000,000
4	28	2,017	20,000	1,000,000
5	67	1,987	50,000	1,000,000
6	190	2,402	100,011	1,000,000
7	28	2,211	20,000	1,000,000

Training with the Extended Tabu Search Algorithm

Since the *tabu list* was variable and program termination was determined by solution reduction frequency, the extended Tabu Search algorithm was conducted for 10 replications for each data set on each problem, changing only the random seed for each initialization. These 10 replications for each data set for this new algorithm compare to 160 replications for backpropagation on each data set and 360 replications for each data set with the preliminary Tabu Search algorithms.

[6]Complete results for the preliminary TS runs are available form the authors.

RESULTS

In all seven problems the best epoch size for BP was one and momentum was .9. Also, for these problems the Logicon algorithm generated sizeably inferior estimates.[7] The only factor that appeared to be problem and data set specific was the learning rate. Table 4 shows the factors and levels which achieved the best results across the 160 different training runs for each of the five data sets for each problem.

Each of the seven problems converged within 1,000,000 iterations using the BP algorithm. The weights were saved at the last point of error reduction and training continued until a total of 1,000,000 iterations was achieved. The network out-of-sample interpolation error was checked every 10,000 iterations for a reduction in error for 1,000,000 iterations at which point the search was abandoned. Although, the BP algorithm only trained on the

Table 4 - Optimal Configuration Factors for Back Propagation for all five Data sets

PROBLEM	LEARNING RATE		MOMENTUM		EPOCH SIZE		LOGICON	
	.5	1	0.3	0.9	1	50	On	Off
1	0	5	0	5	5	0	0	5
2	0	5	0	5	5	0	0	5
3	0	5	0	5	5	0	0	5
4	4	1	0	5	5	0	0	5
5	0	5	0	5	5	0	0	5
6	5	0	0	5	5	0	0	5
7	0	5	0	5	5	0	0	5

in-sample data, by using the interpolation data as a criterion for saving the weights, BP had an unfair advantage over TS, since TS did not use this information. However, even with this advantage, BP found inferior solutions compared to the extended version of TS. The in-sample, interpolation, and extrapolation Root Mean Squared

[7] Recommendations for much lower learning and momentum rates for the Logicon algorithm were tried, but there was no significant difference in effect.

(RMS) error comparisons are tabulated in Table 5. Table 5 shows the means of the best solutions for each data set and Table 6 shows the standard deviations of these solutions across the data sets..

Table 5 - The Mean Best RMS Error Comparisons (Data Sets 1-5)

Best RMSE	IN-SAMPLE		INTERPOLATION		EXTRAPOLATION	
Problem	TS	BP	TS	BP	TS	BP
1	1.42E-06	5.23E-01	2.79E-06	1.76E+00	1.69E-04	3.79E+01
2	8.38E-02	1.10E+01	1.96E-01	9.11E+01	1.70E+01	4.38E+02
3	2.45E-01	8.43E+00	1.69E+00	2.34E+01	7.36E+00	1.39E+01
4	4.32E-01	1.88E+02	1.15E+00	4.31E+02	2.51E+02	4.75E+03
5	2.56E-02	8.57E+03	5.68E-02	5.28E+04	1.81E+03	3.06E+05
6	5.35E-02	1.55E-01	6.75E-02	1.95E-01	N/A	N/A
7	1.22E+00	6.58E+00	2.18E+00	8.48E+00	2.94E+00	2.86+01

Table 6 - The Standard Deviation of the Best RMS Error Comparisons (Data Sets 1-5)

Best RMSE	IN-SAMPLE		INTERPOLATION		EXTRAPOLATION	
Problem	TS	BP	TS	BP	TS	BP
1	2.58E-06	1.45E-01	5.54E-06	1.26E+00	3.12E-04	6.43E+01
2	6.62E-02	1.42E+00	1.54E-01	5.40E+01	1.69E+01	2.33E+02
3	1.17E-01	4.28E+00	1.86E+00	1.25E+01	5.05E+00	7.90E+00
4	6.84E-01	1.48E+01	2.02E+00	1.51E+02	4.59E+02	9.99E+02
5	2.01E-02	1.51E+03	2.68E-02	3.70E+04	2.82E+03	1.53E+05
6	2.13E-02	5.34E-02	2.10E-02	6.15E-02	N/A	N/A
7	4.92E-01	2.43E+00	6.97E-01	2.59E+00	3.52E+00	1.09E+01

Not only did TS find superior solutions for all test problems, it did so with fewer function evaluations. Table 7 illustrates the savings in time and function evaluations TS had over BP in converging on found solutions on the first data set. Since the ability of ANN's to forecast out-of-sample is often of most interest, Figures 1-6 are provided to graphically illustrate the differences in TS and BP for the interpolation forecasts on the first data sets.

Tables 8, 9 an 10 provide a statistical comparison of the in-sample, interpolation, and extrapolation results for the seven problems. All comparisons were found to be highly significant, with TS finding superior solutions for all problems.

Table 7 - Total Time and Function Evaluation for all replications for modified TS and BP (Data set 1)

	PROCESSOR TIME ON PC 83 MHZ (sec)		FUNCTION EVALUATIONS	
PROBLEMS	TS	BP	TS	BP
1	1,190	8,360	365,027	4,180,000
2	1,033	8,360	317,106	4,180,000
3	1,725	8,360	529,577	4,180,000
4	1,937	8,360	594,426	4,180,000
5	619	8,360	190,021	4,180,000
6	3,024	10,032	928,061	4,180,000
7	2,522	9,196	774,083	4,180,000

Table 8 Wilcoxon Matched Pairs Signed Ranks Test {2-T- P Significance} Insample

Prob	Data Set 1			Data Set 2			Data Set 3			Data Set 4			Data Set 5		
	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P
1	50	0	.00	50	0	.00	50	0	.00	50	0	.00	50	0	.00
2	50	0	.00	50	0	.00	50	0	.00	50	0	.00	50	0	.00
3	50	0	.00	50	0	.00	50	0	.00	50	0	.00	50	0	.00
4	50	0	.00	50	0	.00	50	0	.00	50	0	.00	50	0	.00
5	50	0	.00	50	0	.00	50	0	.00	50	0	.00	50	0	.00
6	70	30	.00	89	11	.00	95	5	.00	79	21	.00	100	0	.00
7	100	0	.00	100	0	.00	99	1	.00	100	0	.00	91	9	.00

Table 9 Wilcoxon Matched Pairs Signed Ranks Test {2-T- P Significance} Interpolation

Prob	Data Set 1			Data Set 2			Data Set 3			Data Set 4			Data Set 5		
	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P
1	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
2	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
3	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
4	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
5	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
6	107	43	.00	135	15	.00	143	7	.00	121	29	.00	150	0	.00
7	149	1	.00	150	0	.00	149	1	.00	139	11	.00	137	13	.00

Table 10 Wilcoxon Matched Pairs Signed Ranks Test {2-T- P Significance} Extrapolation

Prob	Data Set 1			Data Set 2			Data Set 3			Data Set 4			Data Set 5		
	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P	TS	BP	P
1	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
2	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
3	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
4	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
5	150	0	.00	150	0	.00	150	0	.00	150	0	.00	150	0	.00
6															
7	145	5	.00	150	0	.00	150	0	.00	147	3	.00	150	0	.00

Figure 1 - Problem 2

Interpolation

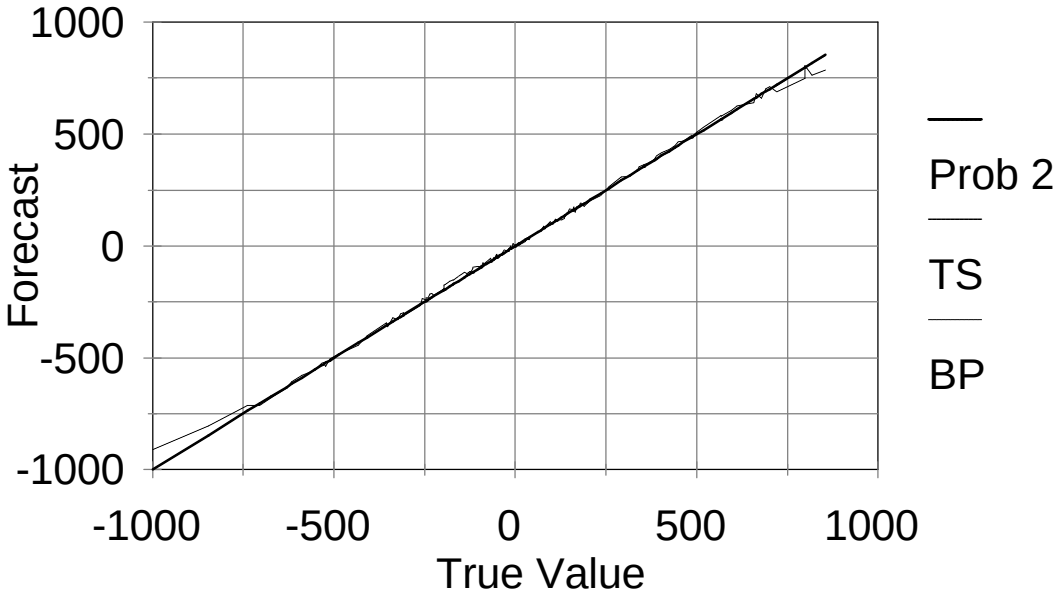


Figure 2 - Problem 3

Interpolation

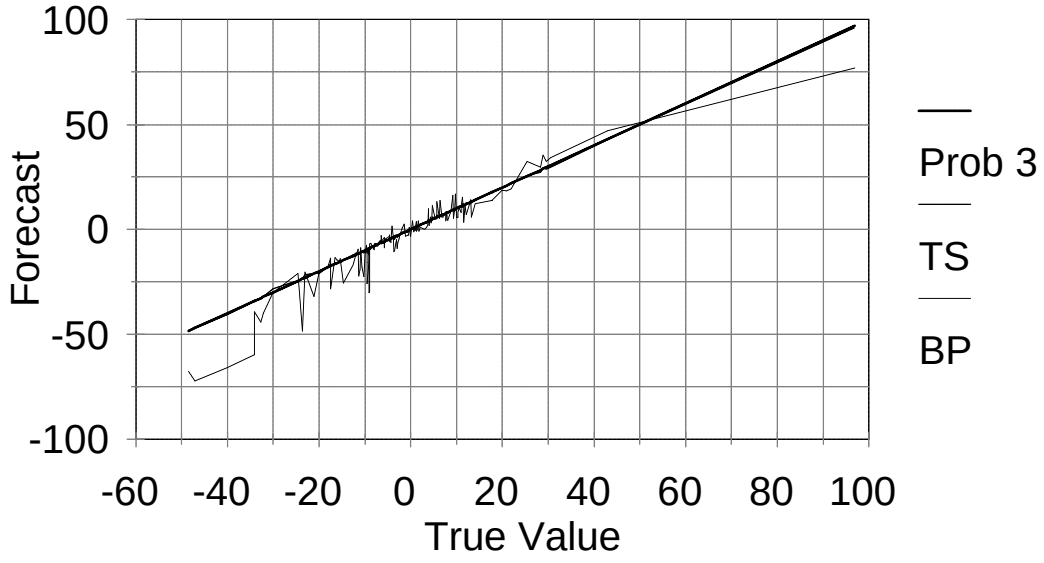


Figure 3 - Problem 4

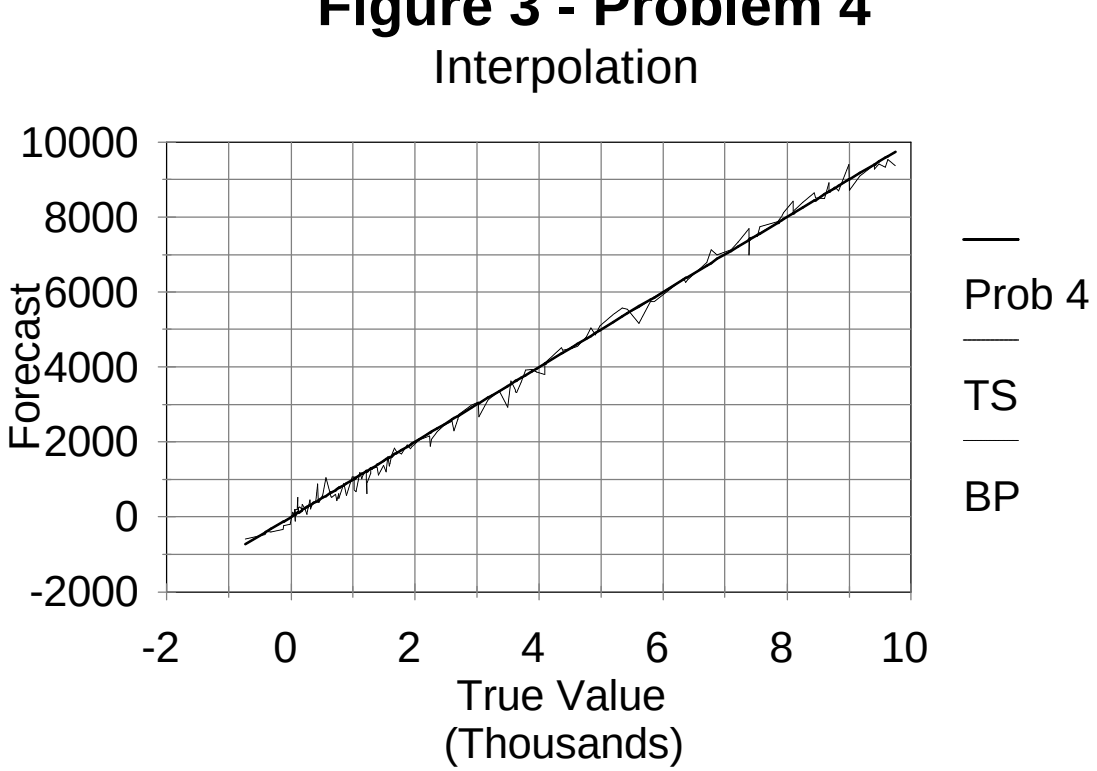


Figure 4 - Problem 5

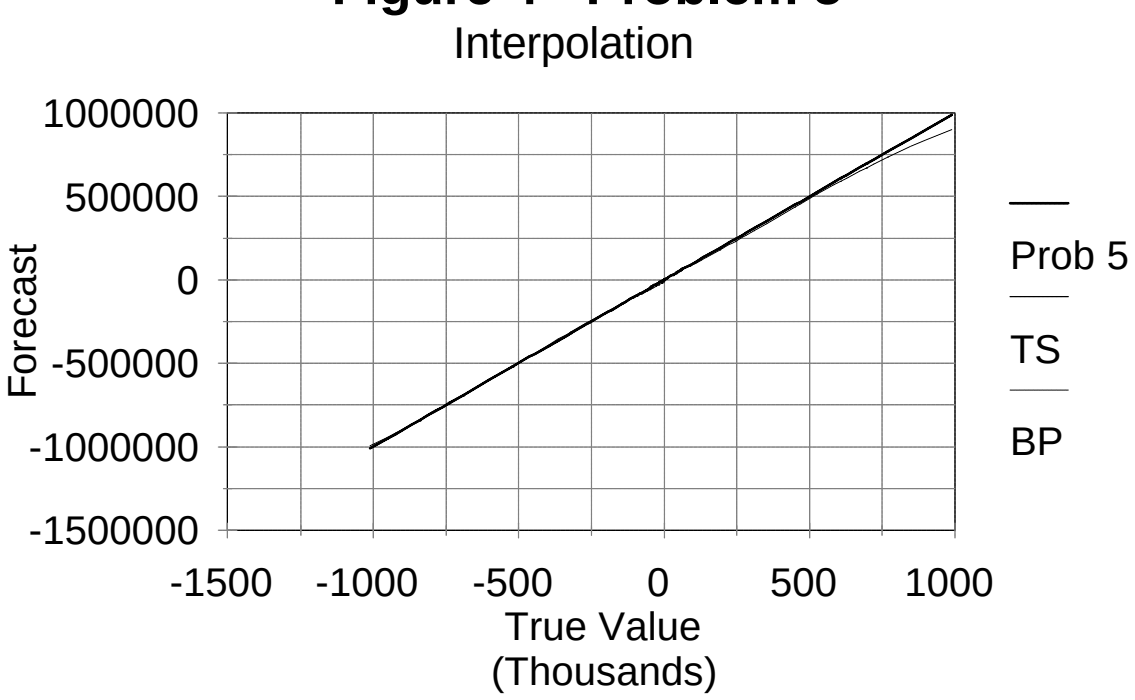


Figure 5 - Problem 6

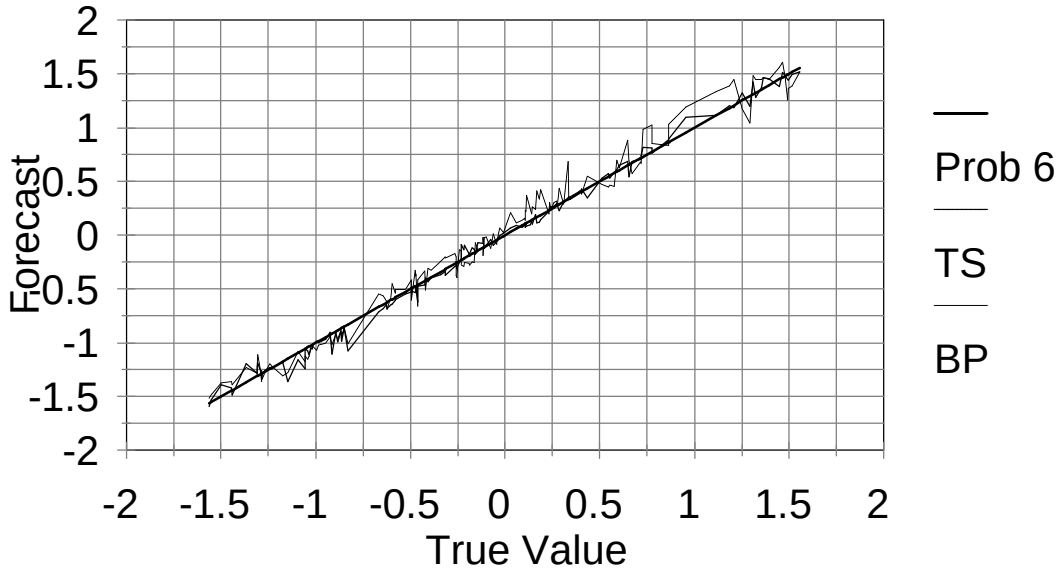
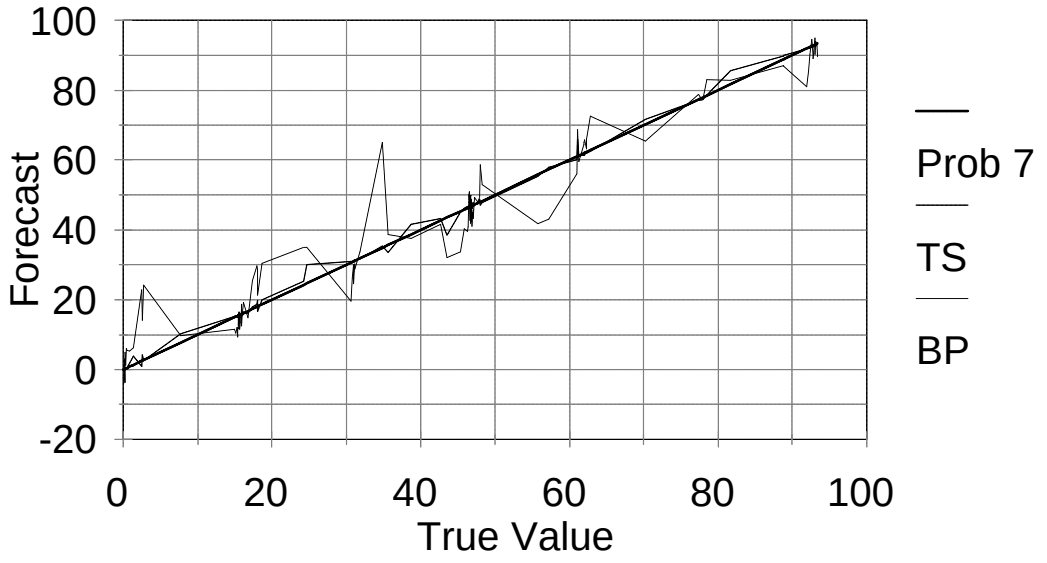


Figure 6 - Problem 7



Conclusions

Business research using artificial neural networks has been increasing because of their ability to serve as flexible form estimators. ANN's have been shown to be able to estimate any unknown function to any desired degree of accuracy, but in order to achieve this end, a global solution is essential. Many researchers are currently using algorithms, such as backpropagation, that converge locally, and thus often perform poorly on even simple problems when forecasting out-of-sample. A possible solution to this local convergence dilemma is the Tabu Search algorithm.

Clearly the enhancements suggested in recent research are important. The extended Tabu search, as applied in this research, found superior solutions for all seven problems despite BP's advantages of having 160 replications for each data set compared to TS's 10 replications, more function evaluations for each search, and using the out-of-sample interpolation data as a check point for saving solutions. The TS algorithm used in this research explores many possible points before selecting a starting point thus enhancing the possibility of finding better solutions. Once the initial point is found, the utilization of the intensification and diversification phases are used for zooming in on the good solution while at the same time escaping local optima. It should be noted that many other techniques could be used to enhance this search algorithm for solving feedforward neural networks. Potential enhancements include, initializing a backpropagation network with the best weights found from a Tabu Search, and combining Tabu Search with other search techniques like the Genetic Algorithm or Simulated Annealing for optimizing ANN's. This research demonstrates the importance of using global search algorithms for optimizing neural networks and demonstrates that the extended Tabu search algorithm should be considered as a possible answer to the local optima problem.

Appendix

Psuedo Code for Preliminary Tabu Search

```
Input   NH {Number of Neighborhoods}
        NS {Number of local searches}
        TL {Length of Tabu List}
        TC {Tabu Criterion}
        Z {an mxk matrix of m observations of k explanatory variables}
        Y {an mx1 vector of m observations of the dependent variable}
        Maxitt {maximum number of iterations}

While i # NH
Choose X0 { a vector of weights, each component randomly drawn from (-10,10)}
Evaluate NN(Z,Y,X0) {The neural net evaluated with data Z and Y and weights XO}
Set     NNBEST = NN(Z,Y,X0)
        XBEST = X0
        TABUNN(1) = NN(Z,Y,X0)
        TABUX(1) = X0
        Numitt=1
   While j # NS
   Choose XN { a vector of weights, each component randomly drawn from X0 ± 1%}
   Evaluate NN(Z,Y,XN)
   If NN(Z,Y,XN) < NNBEST THEN
        NNBEST = NN(Z,Y,XN)
        XBEST=XN
        Move each element of Both Tabu vectors one position forward
         TABUNN(1) = NN(Z,Y,XN)
        TABUX(1) = XN
        Numitt =  1
   ELSEIF XN is  not within ±  TC of  every element of  the TABUX Matrix  AND NN(Z,Y,XN) is not
                within ±  TC of  every element of  the TABUNN vector
        Move each element of Both Tabu vectors one position forward
         TABUNN(1) = NN(Z,Y,XN)
        TABUX(1) = XN
        Numitt = Numitt + 1
   ENDIF
   Next j
If Numitt > Maxitt then go to output
Next i
Output NNBEST, XBEST
```

Psuedo Code for Extended Tabu Search

```
Input   NH {Number of Neighborhoods}
        NS {Number of local searches}
        TL {Length of Tabu List}
        TC {Tabu Criterion}
        Z {an mxk matrix of m observations of k explanatory variables}
        Y {an mx1 vector of m observations of the dependent variable}
        Maxitt {maximum number of iterations}

search = 1
While search # NH
Numitt=1
TABUCNT=0
first=1
count = 1
    While i # 1000
        Choose XN { a vector of weights, each component randomly drawn from (-10,10)}
        Evaluate NN(Z,Y,XN)
        If NN(Z,Y,XN) < NNBEST  or if i = 1THEN
                NNBEST = NN(Z,Y,XN)
                XBEST=XN
                Move each element of Both Tabu vectors one position forward
                TABUNN(1) = NN(Z,Y,XN)
                TABUX(1) = XN
        ELSEIF XN is  not within ±  TC of  every element of  the TABUX Matrix  AND NN(Z,Y,XN) is not
               within ±  TC of  every element of  the TABUNN vector
                Move each element of Both Tabu vectors one position forward
                TABUNN(1) = NN(Z,Y,XN)
                TABUX(1) = XN
        ELSE
                TABUCNT =TABUCNT+1
                TL= Max{1 or Nearest Integer to (1000/TABUCNT)
                 i=i-1
        ENDIF
        If first = 1
                OLDBEST = NNBEST
                 i=0
                 first = 1
         ENDIF
        If  i = 1000 and first > 1 and count < 20
                IF NNBEST < OLDBEST
                i=0
                count=count + 1
        ENDIF
    i=i+1
    END While Loop (i)

    Change = 1
    While m # 5
        While p # 2
          While j  # 1000
              If p = 1 Then
                  XN = (.1*XBEST -.2*XBEST*Random)/Change
```

```
            elseif p = 2 then
                XN = (.1*XBEST -.2*XBEST*Random)*Change
            ENDIF
            Evaluate NN(Z,Y,XN)
            If NN(Z,Y,XN) < NNBEST  or THEN
                NNBEST = NN(Z,Y,XN)
                XBEST=XN
                Move each element of Both Tabu vectors one position forward
                TABUNN(1) = NN(Z,Y,XN)
                TABUX(1) = XN
                j=0
            ELSEIF XN is  not within ±  TC of  every element of  the TABUX Matrix  AND NN(Z,Y,XN) is
                not within ±  TC of  every element of  the TABUNN vector
                Move each element of Both Tabu vectors one position forward
                TABUNN(1) = NN(Z,Y,XN)
                TABUX(1) = XN
            ENDIF
          j=j+1
          End While loop (j)
      Change = change +1
      p = p+1
      End While loop (p)
   m=m+1
   End While Loop (m)
search = search +1
End While loop (search)
Output NNBEST, XBEST
```

REFERENCES

Archer, N., & Wang, S (1993). "Application of the Back Propagation Neural Network Algorithm with Monotonicity Constraints for Two-Group Classification Problems," *Decision Sciences*, 24(1), 60-75.

Bland, J. A., & Dawson, G. P. (1991). "Tabu Search and Design Optimisation", *Computer-Aided Design*, 23, 195-201.

Bland, J. A. (1993). "Nonlinear Optimization of Constrained Functions Using Tabu Search", *Int. J. Math. Educ. Sci. Technol*., 24(5), 741-747.

Bland, J. A. (1994). "A Derivative-Free Exploratory Tool for Function Minimisation Based on Tabu Search", *Advances in Engineering Software*, 91-96.

Chen, J. R., & Mars, P. (1990). "Stepsize Variation Methods for Accelerating the Back Propagation Algorithm", *Proceedings of The International Joint Conference on Neural Networks*, Washington, DC, 1, 601-604.

Crainic, T., Gendreay, M., Soriano, P. and Toulouse, M., (1991), "A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements," Publication #797, Centre de Recherche sur les Transports, Université de Montreal

de Werra, D. & Hertz, A. (1989). "Tabu Search Techniques: A Tutorial and an Application to Neural Networks", *OR Spekrum*, 11, 131-141.

Franzini, M. A. (1987). "Speech Recognition with Back Propagation", *Proceedings of the IEEE/Ninth Annual Conference of the Engineering in Medicine and Biology Society*, Boston, MA, 9, 1702-1703.

Funahashi, K. (1989). "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, 2(3), 183-192.

Gallant, R. A. & White, H. (1992). "On Learning the Derivatives of an Unknown Mapping with Multilayer Feedforward Networks." *Artificial Neural Networks*. Cambridge, MA: Blackwell Publishers, 206-223.

Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computer Operations Research*, 13, 533-549.

Glover, F. (1988). "Tabu Search", CAAI Report 88-3. University of Colorado, Boulder.

Glover, F. (1990). "Tabu Search: A Tutorial", *Interfaces*, 20, 74-94.

Glover, F., and Laguna, M. (1993), "Bandwidth Packing: A Tabu Search Approach," *Management Science*, 39/4, pp. 492-500.

Goffe, W. L., Ferrier, G. D., & Rogers, J. (1994). "Global Optimization of Statistical Functions with Simulated Annealing," *Journal of Econometrics*, 60, 65-99.

Hadar, J (1971). "Mathematical Theory of Economic Behavior", Addison-Wesley Publishing Company Inc, Philippines, 31-33.

Hansen, P & Jaumard, B. (1987). "Algorithms for the Maximum Satisfiability Problem", RUTCOR Research Report 43-87. Rutgers University, New Brunswick, NJ.

Hertz, A. & de Werra, D. (1990). "The Tabu Search Metaheuristic: How we used it", *Annals of Mathematics and Artificial Intelligence*, 1, 111-121.

Holt, M. J., & Semnani, S. (1990). "Convergence of Back Propagation in Neural Networks using a Log-Likelihood Cost Function", *Electron Letters*, 26, 1964-1965.

Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer Feed-Forward Networks are Universal Approximators," *Neural Networks*, 2(5), 359-66.

Hsiung, J.T., Suewatanakul, W., & Himmelblau, D. M. (1990) "Should Backpropagation be Replaced by more Effective Optimization Algorithms? *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 7, 353-356.

Izui, Y., & Pentland, A. (1990). "Analysis of Neural Networks with Redundancy", *Neural Computation*, 2, 226-238.

Kelly, J., Golden, B., and Assad, A., (1993) "Large-scale Controlled Rounding using Tabu Search with Strategic Oscillation," *Annals of Operation Research*, 41, pp. 69-84.

Knox, J. E. (1989). "The Application of Tabu Search to the Symmetric Traveling Salesman Problems", Ph.D. Thesis, University of Colorado, USA.

Laguna, M., Kelly, J., González-Verlarde, and Glover, F., (1995) "Tabu Search for the Multilevel Generalized Assignment Problem," *European Journal of Operational Research, 82, pp. 176-189.*

LeCun, Y. (1986) "Learning Processes in an Asymmetric Threshold Network," *Disordered Systems and Biological Organization*, 233-40. Berlin: Springer Verlag.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jsckel, L. D. (1989). "Backpropagation applied to handwritten zip code recognition", *Neural Computation*, 1, pp. 541-551.

Lee, J., & Bien, Z. (1991). "Improvement on Function Approximation Capability of Backpropagation Neural Networks", *Proceedings of the International Joint Conference on Neural Computation*, 1, 541-551.

Lenard, M., Alam, P., & Madey, G. (1995). "The Applications of Neural Networks and a Qualitative Response Model to the Auditor's Going Concern Uncertainty Decision," *Decision Sciences*, 26(2), 209-227.

Madey, G. R., & Denton, J. (1988). "Credit Evaluation with Missing Data Fields," *Proceedings of the INNS*, Boston, 456.

Matsuoka, K., & Yi, J. (1991). "Backpropagation based on the Logarithmic Error Function and Elimination of Local Minima", *Proceedings of the International Joint Conference on Neural Networks*, Singapore, 2, 1117-1122.

Parker, D. (1985). " Learning Logic (Technical Report TR-87)", Cambridge, MA: Center for Computational Research in Economics and Management Science, MIT.

Plaut, D. C., Nowlan, S. J., & Hinton, G. E. (1988). "Experiments on Learning by Back Propagation (CMU-CS-86-126), Department of Computer Science, Pittsburgh, PA: Carnegie-Mellon University.

Rumelhart, D., Hinton, G., & Williams, R., (1986). "Learning Internal Representations by Error Propagation," In D. Rumelhart & J. McCleeland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Salchenberger, L. M., Cinar, E. M., & Lash, N. A. (1992). "Neural Networks: A New Tool for Predicting Thrift Failures," *Decision Sciences*, 23, 899-916.

Samad, T. (1990). "Backpropagation Improvements Based Heuristic Arguments", *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, Vol. I, pp. 565-568.

Shang Y. & Wah B. W., "Global Optimization for Neural Network Training"," *Computer*, March 1996, 45-54.

Shoemaker, P. A., Carlin, M. J., & Shimabukuro, R. L. (1991). "Back propagation learning with trinary quantization of weight updates," *Neural Networks*, 4, 231-241.

Sietsma, J., & Dow, R. J. F. (1991). "Creating Artificial Neural Networks that Generalize", *Neural Networks*, 2, pp. 67-79.

Skorin-Kapov, J. (1990). "Tabu Search Applied to the Quadratic assignment Problem", *ORSA J. Comp*., 2, 33-45.
Solla, S. A., Levin, E., & Fleisher, M. (1988). "Accelerated Learning in Layered Neural Networks", *Complex Systems*, 2, pp. 39-34.

Subramanian, V., & Hung, M.S. (1990). "A GRG@-based System for Training Neural Networks: Design and Computational Experience," *ORSA Journal on Computing*, 5(4), 386-394.

van Ooyen, A., & Nienhuis, B. (1992). Improving the convergence of the back propagation algorithm," *Neural Networks*, 5, 465-471.

Wang, S. (1995). "The Unpredictability of Standard Back Propagation Neural Networks in Classification Applications," *Management Science*, 41(3), 555-59.

Watrous, R. L. (1987). "Learning Algorithms for Connections and Networks: Applied Gradient Methods of Nonlinear Optimization," *Proceedings of the IEEE Conference on Neural Networks 2*, San Diego, IEEE, 619-27.

Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). "Back Propagation, weight-elimination and Time Series Prediction", In D. S. Touretzky, J. L. Elman, T. J. Sejnowski & G. E. Hinton (Eds,), *Connectionist Models. Proceedings of the 1990 Connectionist Models Summer School* (105-116). San Mateo, CA: Morgan Kaufmann.

Werbos, P., "The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting," New York, NY: John Wiley & Sons, Inc., 1993.

White, H, "Some Asymptotic Results for Back-Propagation," *Proceedings of the IEEE Conference on Neural Networks 3*, San Diego, IEEE, 1987, 261-66.