BEYOND BACKPROPAGATION: USING SIMULATED ANNEALING FOR TRAINING NEURAL NETWORKS

Randall S. Sexton[1]
Department of Management
Ball State University
Muncie, Indiana 47306
Office: (765) 285-5320
Fax: (765) 285-8024
email: rssexton@mail.bsu.edu

Robert E. Dorsey
Department of Economics and Finance
University of Mississippi
University, Mississippi
Office: (601) 232-7575
email: dorsey@bus.olemiss.edu

John D. Johnson
Department of Marketing and Management
University of Mississippi
University, Mississippi
Office: (601) 232-5492
email: johnson@bus.olemiss.edu

[1]Randall Sexton is an assistant professor in the College of Business, Ball State University. 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.

BEYOND BACKPROPAGATION: USING SIMULATED ANNEALING FOR TRAINING NEURAL NETWORKS

ABSTRACT

The vast majority of neural network research relies on a gradient algorithm, typically a variation of backpropagation, to obtain the weights of the model. Because of the enigmatic nature of complex nonlinear optimization problems, such as training artificial neural networks, this technique has often produced inconsistent and unpredictable results. To go beyond backpropagation's typical selection of local solutions, simulated annealing is suggested as an alternative training technique that will search globally. In this research, backpropagation will be directly compared with this global search technique via an intensive Monte Carlo study on seven test functions

KEY WORDS: Neural networks; backpropagation; simulated annealing; global search algorithms

1. Introduction

Interest in business applications of artificial neural networks is growing rapidly. Evidence for this exists in both the academic and trade business literature. A search for articles applying artificial neural networks (ANN) to business applications, in the BUSI business index, returned 586 citations in journals from January 1988 to May 1997. The increase in popularity of ANNs is not surprising since neural networks can approximate unknown functions to any degree of desired accuracy (Funahashi, 1989; Hornik et al. 1989) and the underlying functions of many problems in business are generally unknown. ANNs are used for solving problems in business related areas such as production and operations, real estate, finance, economics, management and accounting. Simply by scanning journals in these areas a user can accumulate a wealth of evidence that the use of ANNs is increasing. Many other areas such as engineering, physics, chemistry and biology are also rapidly taking advantage of this new tool. With each new success, the popularity of ANNs grows, not only with researchers but with practitioners as well.

The vast majority of these studies rely on a gradient algorithm, typically a variation of backpropagation (LeCun, 1986; Parker, 1985; Rumelhart et al., 1986a; Werbos, 1993) to search for optimal weights of the model. The well-known limitations of gradient search techniques applied to complex nonlinear optimization problems such as training of artificial neural networks often result in inconsistent and unpredictable performance (Archer & Wang, 1993; Rumelhart et al., 1986a; White, 1987). Many modifications have been implemented in past ANN literature to overcome these problems, but have had limited success. In this paper we demonstrate that such modifications are unnecessary if a sufficiently complex initial architecture and an appropriate global search algorithm is used for network training. We further show that the simulated

annealing algorithm is an appropriate global search technique that consistently outperforms backpropagation for ANN training.

2. Statement of Problem

Backpropagation (BP) is the training method used by most ANN researchers (Salchenberger et al., 1992) and should be credited with much of the popularity of neural network research. Although, this technique has produced promising results, it has not fully realized ANN's ability to approximate any unknown function to any desired degree of accuracy, mainly because of its tendency to become trapped in local optima.

Researchers have used a variety of approaches to try to adjust for this characteristic of backpropagation. For example, parameters of the algorithm can be adjusted to affect the momentum or step size of the search so that the search will break out of locals and move toward the global solution. The correct value of these parameters, however, is not known *a priori* and is often problem specific. Therefore, for any given problem, many parameters must be tried to generate confidence that a global solution has been found. Since this algorithm is designed to converge on local solutions, a dependency occurs between the initial random starting points and the solutions found. Although, many initial points can be tried, there is no known method, beyond mere chance, for finding a global solution with BP.

A more effective approach might be to replace the gradient based search technique with a global search technique. A limited amount of research has been conducted that shows two global search techniques, the genetic algorithm (Sexton, Dorsey, & Johnson, 1998; Sexton & Dorsey, 1998; Dorsey, Johnson, & Mayer, 1994; Dorsey, Johnson, & Van Boening, 1994) and tabu search (Sexton, Dorsey, & Johnson, 1998), to consistently find superior solutions when compared with

BP for training ANNs. Our study examines simulated annealing (SA) as another alternative to BP for ANN training. The NULL hypothesis is:

H_0: Simulated annealing does not find significantly better solutions in training neural networks, compared with neural networks trained using backpropagation.

3. The Search Algorithms

The following sections provide a historical background of the algorithms as well as a general description of the simulated annealing algorithm used in this study. Comparisons of these algorithms are based upon their appropriateness for training ANNs, and the need for parameter adjustment for finding superior solutions.

3.1. Backpropagation

Although, Rumelhart et al. (1986a & 1986b) popularized the use of BP for training ANNs, they also made two important observations regarding the BP algorithm, which is its slowness in convergence and its inability to escape local optima. Many attempts have been made to improve upon the original BP algorithm. Most of these attempts focus upon one or more of the following strategies; differential scaling, changing the error metric, and modifying the transfer function. Several examples of such research are listed in Alpsan et al. (1995). Examples of some commonly known BP variations available to address such problems which are included in NeuralWare®, a popular commercial ANN software package, include Norm-Cum-Delta, Extended Delta-Bar-Delta, QuickProp, MaxProp, and Delta-Bar-Delta.

When using BP (Neural Works Professional II/Plus by NeuralWare®), user defined parameter adjustments such as step size, momentum value, learning rule (variation of BP), normalization technique, random seed, transfer function, and network architecture can all be

modified to find the best combination for solving the particular problem. Given the number of possible parameter combinations, the possibility of finding the correct combination of parameter settings for finding a global solution, given a random starting point, is unlikely and is based primarily on chance.

Since most of the problems with BP are due to its gradient nature, it seems plausible that many, if not all, of these problems could be eliminated by using global search methods that are not hill climbing algorithms.

3.2. Simulated Annealing

Annealing, refers to the process that occurs when physical substances, such as metals, are raised to a high energy level (melted) and then gradually cooled until some solid state is reached. The goal of this process is to reach the lowest energy state. Physical substances usually move from higher energy states to lower ones, so minimization naturally occurs, but in this process there is always some probability that a transition to a higher energy state will occur. As the energy state naturally declines, the probability of moving to a higher energy state will decrease.

Metropolis et al. (1953) proposed an algorithm to simulate the annealing process, modeling the transition of a solid from an elevated temperature to thermal equilibrium. The analogy between the slow cooling of a solid and minimizing the cost function of a combinatorial optimization problem was not realized until it was suggested by Kirkpatrick et al. (1982), and independently by Cerny (1985). Kirkpatrick et al. (1982) developed this “simulated annealing” algorithm by substituting cost for energy and by executing the Metropolis algorithm at a sequence of slowly decreasing temperature values. This method was reported to perform well even in the presence of a high number of variables. Modifications were proposed by Corana et al. (1987) and

Goffe (1994) that would allow this algorithm to be effectively used for functions defined in a continuous domain. However, these functions do not need to be smooth or even continuous.

Goffe et al. (1994) tested simulated annealing on four econometric problems and compared it with the simplex, conjugate gradient and quasi-newton algorithms for ANN training. Although, SA was found to out perform the other training algorithms, a comparison was not made with BP.

The simulated annealing (SA) implementation used in this study was taken from Goffe et al. (1994). While a complete description can be found there, a summary of this algorithm follows. Given the function *f(X)* to be minimized, several initial parameters must be set, including T_0, the starting temperature; *X*, the starting solution vector of weights; and *V*, the step length for *X*. Both *X* and *V* are vectors of length *n*, the number of weights in the ANN. Note that *x* and *v* are elements of vectors *X* and *V*. For example, x_1 is the first weight in *X* and v_1 is the first element in *V*. We define *f(X)* to be the sum-of-squared errors that is produced by the vector *X*. Taking the initial vector *X*, the weights are used for the ANN to calculate and save its *f* value as the current optimum and best optimum. A candidate X' is then chosen by varying each x_i by Equation 1, where *r* is a random number drawn from a uniform distribution between [-1,1].

$$x_i' = x_i + r * v_i \qquad (1)$$

From this new X' the function value f' is then computed. If f' is less than f, X' is accepted as the new X. If this new f is the smallest solution so far, f and its corresponding X are saved as the best solution. If f' is greater than or equal to f, the Metropolis criterion is used to test for acceptance. The value from Equation 2 is computed and compared with p', a random number drawn from a uniform distribution between [0, 1]. If p is greater than p', the new point is accepted and X is updated with X', otherwise, X' is rejected.

$$p = e^{\left(\frac{f - f'}{T}\right)} \qquad (2)$$

The process repeats for a user defined number of steps N_S through all elements of X. N_S is set to the recommended value of 20 for our study. During this process, the step length V is adjusted so that 50% of all moves are accepted. After N_T times through the above process, T is reduced. An examination of Equation 2, shows that a decrease in T will decrease the probability of accepting uphill moves. The reduction in T is calculated by the formula shown in Equation 3. The value r_T is also user defined and set between [0, 1].

$$T_1 = r_T * T_0 \qquad (3)$$

The first new point tried after a reduction in temperature *(T)* becomes the current optimum. A decrease in temperature causes the number of rejections to increase and the step lengths to decline. Termination of the algorithm occurs when the function value for the last temperature *(T)* reduction differs from the corresponding value at the current temperature by less than the error tolerance N_ϵ and the final function value at the current T differs from the current optimal function by less than $N_{\epsilon.}$ Escape from local solutions is possible by SA's acceptance of

uphill moves. As the algorithm iterates and the reduction in temperature occurs, acceptance of uphill moves and step lengths decrease.

To demonstrate this algorithm a simple example follows. Given a training data set, the initial weights of vector X are randomly drawn. These weights are plugged into the ANN and f or sum-of-squared errors is computed. For this example lets say the f value is 3.5. Using Equation 1, a candidate X' is generated and f' is computed that is equal to 3.25. Since, this value is less than f the candidate X' is accepted as the new X vector and the search continues from this point. A new candidate X' is then generated by Equation 1 and its f' function is computed to be 3.75. This function value is worse than the current optimum so the Metropolis criterion, Equation 2, is used to check if the point should be accepted. Given the initial temperature $T = 5$, the p value is computed to be .904837. A random number p' is then drawn from the range [0, 1]. If p' is less than p the point is accepted, otherwise the candidate vector is rejected. As the algorithm progresses with reductions in temperature, the chance of accepting uphill moves will decrease. For example, using the above scenario of f =3.25, f'= 3.75, and the new $T = 1$, the value p decreases to .606531. As can be seen, this value is smaller and the chance of drawing a random number p' that is larger than this value is increased, therefore, rejections of uphill moves increase as the temperature falls.

4. Problems and Data Generation

A Monte Carlo study was conducted on the seven test problems listed below to compare simulated annealing with BP. For the first five test problems, 50 observations were used as the training set and two sets of 150 observations were used for out-of-sample interpolation and extrapolation test samples. Interpolation data sets, for computer generated data,

1) $Y = X_1 + X_2$
2) $Y = X_1 X_2$
3) $Y = \dfrac{X_1}{|X_2| + 1}$
4) $Y = X_1^2 - X_2^3$
5) $Y = X_1^3 - X_1^2$
6) $Y_t = Y_{t-1} + 10.5\left[\dfrac{.2Y_{t-5}}{1 + (Y_{t-5})^{10}} - .1Y_{t-1}\right]$
7) $Y = \alpha_1 \arctan(x_1 - \beta_1) + \alpha_2 \arctan(x_2 - \beta_2) + \alpha_3 \arctan(x_3 - \beta_3) + \lambda$
   where $\lambda = -\alpha_1 \arctan(-\beta_1) - \alpha_2 \arctan(-\beta_2) - \alpha_3 \arctan(-\beta_3)$

are generated from the same ranges as the training data. The only difference between the interpolation data and training data is that there are no identical observations (combinations of input values) between the two. By using an interpolation data set for testing, we can test how well the ANN performs on observations that it has not seen during the training process but are within the same range as the training data. Extrapolation data sets are generated in a similar fashion only differing in the range from which input values are drawn. The input values are drawn from ranges that are outside the ranges of the training and interpolation data. This extrapolation test data demonstrates the ability of the ANN to generalize outside the range of the training data. For the first 5 problems, the training and interpolation data were generated by randomly drawing the inputs from the sets $X_1 \in$ [-100,100] and $X_2 \in$ [-10,10]. The extrapolation test sets were generated from $X_1 \in$ [-200, -101], [101, 200] and for $X_2 \in$ [-20, -11], [11, 20]. The training data was normalized from -1 to 1 for all algorithms in order to have identical training and output data for comparison with the BP trained ANN[2]. The same normalizing factor was used on the extrapolation data as was used for the training and interpolation data.

[2]Most BP software normalizes the data prior to training.

The sum of squared errors (SSE) was used as the objective function and the networks were compared on the basis of the Root Mean Squared (RMS) forecast error. Each network included 6 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 be better network architectures for the given problems, but since the goal of this research is training technique comparison there was no need to modify the ANN's 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 6 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 training set of 100 observations for problem six started from the initial point of [1.6, 0, 0, 0, 0]. The interpolation test data consisted of 150 observations generated from the randomly chosen point [-0.218357539, 0.05555536, 1.075291525, -1.169494128, 0.263368033]. An extrapolation data set was not generated for this problem because of its bounded nature.

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

$$\alpha_1 = 5 \quad \alpha_2 = 10 \quad \alpha_3 = 15$$
$$\beta_1 = 50 \quad \beta_2 = 100 \quad \beta_3 = 150$$

4.1. Training with Backpropagation

Several commercial neural network software packages were preliminarily evaluated to decide which package would provide the best solutions for the given problems. The packages 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, a PC-based neural network application, consistently outperformed the other programs and was chosen for the test series.

In training the ANNs with BP, four factors were manipulated to find the best configuration for training. They included the learning rate, momentum, epoch size, and the logicon algorithm. The different combinations of the learning rate and momentum are introduced to try to 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 function evaluation and weight adjustments. Normally, BP makes adjustments after every training pair, but recent modifications of the algorithm allow for larger epoch sizes (Norm-Cum- Delta). The logicon algorithm was introduced by Gregg Wilensky and Narbik Manukian and was developed to help backpropagation to converge faster by combining the advantages of closed and open boundary networks, into a single network (Wilensky and Manukian, 1992).

Each of these four factors was varied to reduce the chances of becoming trapped in local minima. While rules of thumb exist for setting these values, there are no set standards for deriving optimum configurations for BP networks. Guidelines suggested by the Neural Works manual were used in selecting the values used.

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

Table 1 - Backpropagation Parameters

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

The best solution for each of the seven problems was then trained at least 100,000 iterations, and up to a maximum of 1,000,000 iterations. During this last sequence of network training, the error was checked every 10,000 iterations for a reduction, terminating when 10 consecutive checks resulted in no reduction in error. The sets of weights that achieved the smallest errors were then saved for the comparison.

Since simulated annealing was implemented on the CRAY J-916, precision and operation time could affect the comparison with the PC-based Neural Works, so a BP algorithm written and optimized for the CRAY was acquired[3]. Using this program on the CRAY J-916, 10 additional replications were conducted for each problem. The same data was used for both PC and CRAY replications. Since the speed of the CRAY J-916 was much greater than the PC used for this study many more iterations were possible without affecting the time for each run. Each replication was trained for 1,000,000 initial iterations for all problems, differing only in the random seed. The best replication for each problem was then trained for 50,000,000 additional iterations. The ANN that had the smallest error out of all 170 different replications for each problem across both platforms was then selected for comparison.

4.2. Training with Simulated Annealing

The version of simulated annealing used in this study came from Goffe et al. (1994). This code was written in Fortran 77 and implemented on a CRAY J-916 supercomputer. In training the ANNs with SA, three factors were manipulated to find the best configuration for training each problem. They included the temperature T at three levels, the temperature reduction factor r_T,

---

[3] This software was provided by Roger W. Meier at the US Army Engineering Waterways Experimentation Station, Vicksburg MS.

and the number of function evaluations before temperature reduction N_T, each at two levels. The different combinations of these factors were used in an attempt to try to find the best combination for SA to obtain the global solution. Other parameters that could have been adjusted were set to the recommended values in Goffe et al (1994). Specifically, the number of cycles N_S was set to 20. After N_S*N function evaluations, each element of V (step length) is adjusted so that approximately half of all function evaluations are accepted. The number of final function evaluations N_{EPS} was set to 4. This is used as part of the termination process. The error tolerance for termination was set to 10E-06 and the upper limit for the total number of function evaluations *MAXEVL* was set to 10^9.

Although this Monte Carlo study included the parameter settings recommended by both Goffe et al. (1994) and Corana et al. (1987), a more rigorous exploration of the parameters may produce better results for SA. Table 2 illustrates the specific parameter settings for each combination. Since the optimal solutions are not known *a priori*, the size of the search space was set between -600 and 600.

Table 2 - Simulated Annealing Parameters

Parameter	Values Used
Temperature (T)	5, 50 Thousand, 50 Million
Temperature Reduction Factor (RT)	.50, .85
Number of Function Evaluations Before Temperature Reduction (NT)	5, 125

A total of 120 SA networks for each problem were trained that consisted of these 12 configurations with 10 replications for each configuration. Each replication was started with a new random seed The maximum number of function evaluations was set arbitrarily high

(MAXEVL = 10^9) in order for all problems to terminate with the internal stopping rule based on the predefined error tolerance.

5. Results

The first section describes the results obtained from the 170 replications for each problem generated by the BP ANNs. The second section describes the SA results obtained from the 120 replications for each of the seven problems. Finally, a comparison of these results is discussed in the last section.

5.1. Backpropagation Results

Sixteen network configurations[4] were initially tested to determine the optimal combination of parameter levels for each of the seven test problems. Out of the four chosen parameter adjustments (step value, momentum, epoch size, and the logicon algorithm) used to find an optimal backpropagation configuration, the momentum, epoch size, and logicon algorithm were found not to vary when determining the optimum configuration for each of the seven problems.[5] The only factor that appeared to be problem specific was the learning rate. Table 3 shows the factors and levels that achieved the best results across the 160 different training runs for each problem. Since the BP program used in this study was PC-based and the SA algorithm was implemented on the CRAY J-916, there was a need to test these PC solutions with another BP algorithm optimized for use on the CRAY J-916. Using the same training data, for each problem, the best configuration of parameters was taken from the PC results and 10 new replications, each

---

[4] Configurations included combinations of the four parameter settings with two levels each for the learning rate, momentum, epoch size, and logicon algorithm. All the network architectures included six hidden nodes.

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

with a new random seed, were run on the CRAY J-916. These replications were trained to 50,000,000 iterations to ensure that they had converged. For all seven problems, the PC-based ANN software found superior solutions over the CRAY version.

Table 3 - Optimal Configuration Factors for Backpropagation

Problem	1	2	3	4	5	6	7
Learning Rate	1	1	1	.5	1	.5	1
Momentum	.9	.9	.9	.9	.9	.9	.9
Epoch Size	1	1	1	1	1	1	1
Logicon	OFF	OFF	OFF	OFF	OFF	OFF	OFF

5.2. Simulated Annealing Results

Although, the twelve configurations included the recommended parameter settings of two previous studies (Corana et al., 1987; Goffe et al., 1994), none of the seven test problems found an optimal solution with these configurations. To determine the best configuration, each test problem was initialized at ten different random points for all twelve configurations. Out of these 120 replications for each problem, the best solution so far was then chosen as the optimal configuration of the SA parameters for the particular test problem. However, since there were several other variables that could have been adjusted, as well as the infinite combination of possible levels for T, r_T, and N_T, an optimal configuration cannot be determined by this study.

A problem facing a researcher who wishes to use SA is the lack of rules for adjusting the configuration to produce better results. Heuristics for setting SA parameters are suggested in Corana (1987) and Goffe (1994). The suggestion to set N_T = Max[100, 5*N] can be followed, as was done in this study, but there is no guarantee that an optimal configuration for the algorithm

would be found. The optimal configurations for the seven problems, given the limited levels of parameters, are shown in table 4, as well as the number of function evaluations for each solution.

Each of the 120 trials was terminated with an error tolerance of 10E-06 used in conjunction with a user defined value N_{EPS} of 4. For example, if no reduction in the best function evaluation occurs that is greater than the error tolerance 10E-06 for N_{EPS} (4) iterations, then the process would be terminated.

Table 4 - Optimal Configuration Parameters for SA

Problem	T	r_T	N_T	Function Evaluations
1	50,000	.50	125	2,625,001
2	50,000,000	.85	125	12,625,001
3	50,000	.50	5	112,501
4	50,000	.50	125	2,562,501
5	50,000,000	.50	5	96,901
6	5	.85	5	473,001
7	5	.85	125	8,680,001

5.3. Comparison Results

After selecting the best solutions for each problem and algorithm, these weights were then applied to the interpolation and extrapolation data sets. A comparison for in-sample, interpolation, and extrapolation RMS error for each algorithm is given in table 5. As shown in this table, SA found solutions for the seven problems that had superior RMS errors for in-sample, interpolation, and extrapolation estimates.

Table 5 - Root Mean Squared error (RMSE) Comparison

	In-Sample		Interpolation		Extrapolation	
Problem	BP	SA	BP	SA	BP	SA
1	0.36	0.00	0.43	0.00	17.53	0.00
2	11.41	0.03	23.79	0.14	259.21	7.42
3	3.57	1.77	7.81	5.89	26.32	8.06
4	164.83	0.48	213.11	2.53	5199.60	234.26
5	6518.58	0.43	19233.13	1.27	149673.44	1647.81
6	0.12	0.01	0.47	0.19	N/A	N/A
7	3.51	0.06	6.15	1.11	29.95	0.37

To show statistical differences for these algorithms, each data set was compared using the Wilcoxon Matched Pairs Rank test, as shown in table 6. SA was significantly superior, at the .01 level, for 19 out of 20 comparisons. The in-sample comparison for problem 3 was at the .0518 level of significance. Although the in-sample results were not as strong for problem three the SA found significantly superior solutions for both interpolation and extrapolation test sets, indicating a better estimate of the underlying function. Out of the 2,400 estimates, SA found 2,281 solutions that were closer to the actual output value than those estimated by the BP networks. To illustrate out-of-sample differences graphically, figures 1-4 show the estimates for both algorithms for problem 4. As can be seen from these figures, the SA trained ANN is much closer to finding the true functional form.

Table 6 - Wilcoxon Matched Pairs Signed Ranks Test {2 - Tailed P Significance}

Problem	In-Sample			Interpolation			Extrapolation		
	# Superior		2-T-P	# Superior		2-T-P	# Superior		2-T-P
	BP	SA		BP	SA		BP	SA	
1	0	50	.0000*	0	150	.0000*	0	150	.0000*
2	0	50	.0000*	0	150	.0000*	1	149	.0000*
3	19	31	.0518	49	101	.0019*	0	150	.0000*
4	0	50	.0000*	0	150	.0000*	2	148	.0000*
5	0	50	.0000*	0	150	.0000*	0	150	.0000*
6	4	96	.0000*	22	128	.0001*	N/A	N/A	N/A
7	1	99	.0000*	9	141	.0000*	12	138	.0000*
* Represents significant differences at the 0.01 level between the algorithm estimates.									

Since BP and SA were implemented on two different platforms, direct time comparisons of each algorithm are provided as well as an estimated time for a PC based SA algorithm. These estimates were based on using the same SA code, which was recompiled and run on an 83-MHz PC, using the Windows 95 operating system, which was also the system used for the PC-based BP algorithm. Although, SA took longer to converge on 4 out 7 of these test problems (shown in Table 7), the comparison shows that these times are reasonable considering that SA is finding far superior solutions. Further research needs to be conducted, beyond the scope of this paper, to identify ways in which SA can be modified to decrease convergence time.

Table 7 - Time Comparisons in Seconds

Number of processing seconds used to find the best solutions			
Problem	BP	SA PCγ	SA Cray
1	1,250	4,591	1,633
2	1,250	22,084	7,855
3	1,250	196	70
4	1,250	4,425	1,574
5	1,100	163	58
6	1,350	1,180	420
7	1,250	11,313	4,024
γ=Estimated time calculated by actual PC version of SA			

6. Conclusions

The increased popularity of artificial neural networks in business is brought about by its ability to serve as a flexible form estimator. Although, these networks can estimate any unknown function to any desired degree of accuracy, the most popular methods of network optimization, variations of backpropagation, have not achieved this end. Seemingly, global search techniques, such as simulated annealing are essential for finding solutions that will perform well not only for in-sample observations but for observations that have not been used in the training process.

Although, backpropagation is the most common algorithm chosen for neural network optimization, it is shown in this study that a global search algorithm, such as simulated annealing, may indeed be a superior search alternative. Hopefully, the results of this study will encourage researchers and practitioners to move beyond the limitations of backpropagation in order to find superior global training techniques.

Figure 1

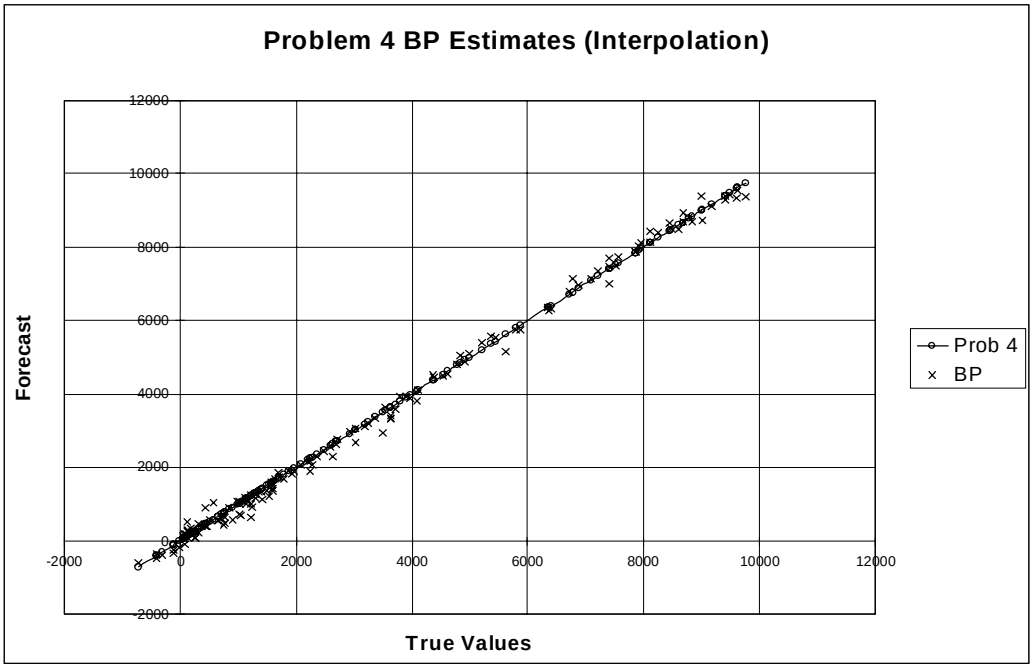


Figure 2

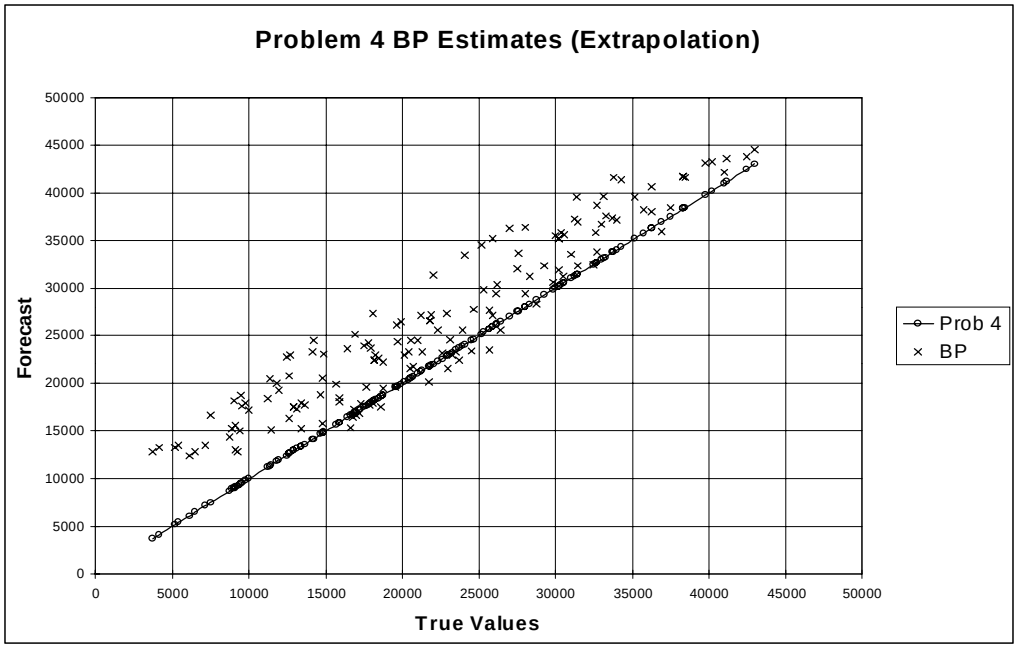


Figure 3

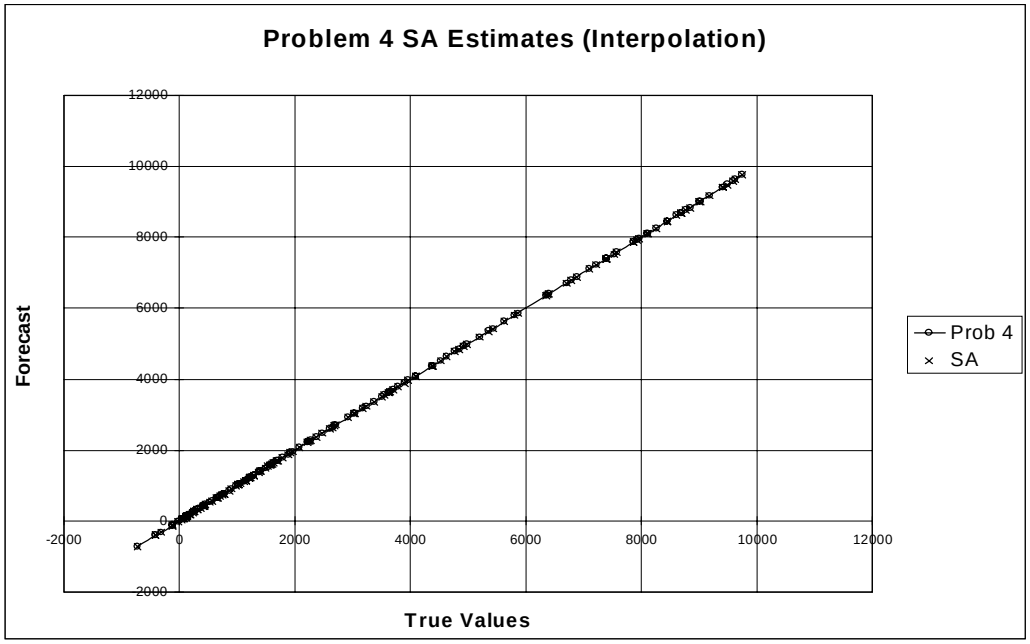
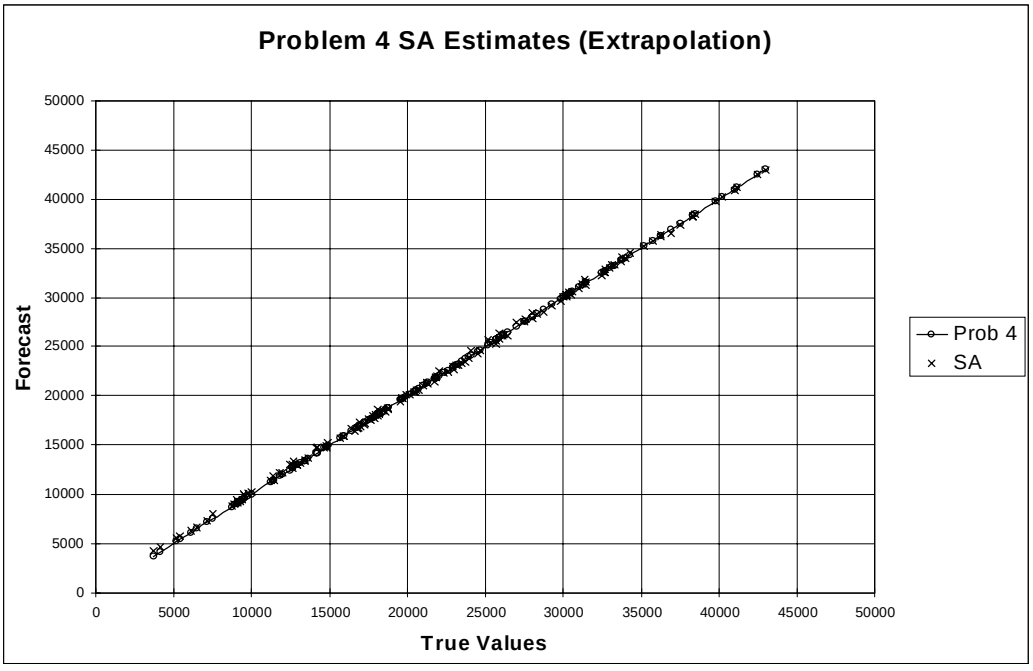


Figure 4



REFERENCES

Alpsan, D., Towsey, M., Ozdamar, O., Tsoi, A. C., & Ghista, D. N. (1995) "Efficacy of modified backpropagation and optimisation methods on a real-wold medical problem," Neural Networks, 8(6), 945-962.

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.

Cerny, V. (1985). "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," J. Opt. Theory Appl., 45, 41-51.

Corana, A., Marchesi, M., Martini, C., & Ridella, S., (1987). "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," ACM Transactions on Mathematical Software, 13, 262-80.

Dorsey, R. E., Johnson, J. D., & Mayer, W. J. (1994). "A genetic algorithm for the training of feedforward neural networks," Advances in Artificial Intelligence in Economics, Finance, and Management (J. D. Johnson and A. B. Whinston, eds.) Greenwich, CT: JAI Press Inc., 1, 93-111

Dorsey, R. E., Johnson, J. D., & Van Boening, M. V. (1994). "The use of artificial neural networks for estimation of decision surfaces in first price sealed bid auctions," In W. W. Cooper and A. B. Whinston (eds.), New Directions in Computational Economics, Netherlands: Kluwer Academic Publishers, 19-40.

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.

Goffe, W. L., Ferrier, G. D., & Rogers, J., (1994). "Global optimization of statistical functions with simulated annealing," Journal of Econometrics, 60, 65-99.

Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer feed-forward networks are universal approximators," Neural Networks, 2(5), 359-366.

Kirkpatrick, S., Gelatt Jr. C. D. & Vecchi, M. P. (1982). "Optimization by simulated annealing," IBM Research Report RC 9355.

LeCun, Y. (1986). "Learning processes in an asymmetric threshold network," Disordered Systems and Biological Organization, Berlin: Springer Verlag, 233-240

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). "Equation of state calculations by fast computing machines," Journal of Chemistry and Physics, 21, 1087-1090.

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

Rumelhart, D. E., Hinton, G. G., & Williams, R. J. (1986a). "Learning internal representations by error propagation," Parallel Distributed Processing: Exploration in the Microstructure of Cognition (pp. 318-362). Cambridge MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back propagating errors," Nature 323: 533-536.

Salchenberger, L. M., Cinar, E. M., & Lash, N. A., "Neural networks: A new tool for predicting thrift failures," Decision Sciences, 23, 899-916.

Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1998). "Toward a global optimum for neural networks: A comparison of the genetic algorithm and backpropagation. Decision Support Systems, 22(2), 171-186.

Sexton, R. S., Alidaee, B., Dorsey, R. E., & Johnson, J. D. (1998). "Global optimization for artificial neural networks: A tabu search application. European Journal of Operational Research, 106/2(3), 570-584.

Sexton, R. S. & Dorsey, R. E. (1998). "The use of parsimonious neural networks for forecasting financial time series," Journal of Computational Intelligence in Finance, 6(1), 24-31.

Werbos, P. (1993). "The roots of backpropagation: From ordered derivatives to neural networks and political forecasting," New York, NY: John Wiley & Sons, Inc.

White, H, (1987). "Some asymptotic results for back-propagation," Proceedings of the IEEE Conference on Neural Networks 3, San Diego, IEEE, 261-266.

Wilensky, G. & Manukian, N. (1992). "The projection neural network," International Joint Conference on Neural Networks II, 358-367.