Theory and Methodology

Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing

Randall S. Sexton [a,*], Robert E. Dorsey [b], John D. Johnson [c]

[a] *Department of Management, College of Business, Ball State University, Muncie, IN 47306, USA*
[b] *Department of Economics and Finance, College of Business, University of Mississippi, MS 38677, USA*
[c] *Department of Marketing and Management, College of Business, University of Mississippi, MS 38677, USA*

Abstract

The escalation of Neural Network research in Business has been brought about by the ability of neural networks, as a tool, to closely approximate unknown functions to any degree of desired accuracy. Although, gradient based search techniques such as back-propagation are currently the most widely used optimization techniques for training neural networks, it has been shown that these gradient techniques are severely limited in their ability to find global solutions. Global search techniques have been identified as a potential solution to this problem. In this paper we examine two well known global search techniques, Simulated Annealing and the Genetic Algorithm, and compare their performance. A Monte Carlo study was conducted in order to test the appropriateness of these global search techniques for optimizing neural networks. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Neural networks; Optimization; Genetic algorithm; Simulated annealing; Global solutions; Interpolation; Extrapolation

1. Introduction

The explosion of business applications of artificial neural networks (ANN) is evidenced by the more than 400 citations referencing articles that have applied neural networks to business applications, within the last two years. This trend shows no signs of slowing as more researchers discover and begin to utilize this approximation tool. The ability of the ANN to accurately approximate unknown functions and their derivatives [8,12], underlies this growing popularity. Because of its ease of use, an overwhelming majority of these applications have used some variation of the gradient technique, backpropagation [14,20,21,30] for optimizing the networks [22]. Although, backpropagation has unquestionably been a major factor for the success of past neural network applications, it is plagued with

___

* Corresponding author. Tel.: +1 765 285 5320; fax: +1 765 285 8024

inconsistent and unpredictable performances [1,13, 15,16,21,27–29,31].

Gradient search techniques such as backprop-agation are designed for local search. They typically achieve the best solution in the region of their starting point. Obtaining a global solution is often dependent on a fortuitous choice of starting values. Global search techniques are known to more consistently obtain the optimal solution. Work in this area is summarized in [25] and specific applications can be found in [7,19,26]. Recently, Sexton et al. [24] demonstrated that for a variety of complex functions the genetic algorithm (GA) was able to achieve superior solutions for neural network optimization than backpropagation and found that another global search heuristic, tabu search (TS), is able to systematically achieve superior solutions for optimizing the neural network than those achieved by backpropagation [23]. Although this literature demonstrates the superiority of global search to hill climbing techniques, the literature provides limited information about the relative performance of global search algorithms.

In addition to GA and TS another well known global search heuristic is simulated annealing (SA). SA has been shown to perform well for optimizing a wide variety of complex problems [10]. This paper compares the performance of these two well known global search algorithms, SA and GA, for ANN optimization.

The following section briefly describes the two global optimization techniques. The next section discusses the Monte Carlo experiment and results of the comparison. We then examine two real world problems to further show the benefits of global search techniques for neural network optimization and include a comparison with BP as well. The final section gives our conclusions.

2. The search algorithms

The two global search techniques used for this study are briefly described in the following two sections. More detailed descriptions of these algorithms can be found in [5] for GA and [10] for SA.

2.1. Genetic algorithm

GA is a global search procedure that searches from one population of points to another. As the algorithm continuously samples the parameter space, the search is directed toward the area of the best solution so far. This algorithm has been shown to perform exceedingly well in obtaining global solutions for difficult non-linear functions [4,5]. The application of the GA to one particularly complex non-linear function, the ANN, has also been shown to dominate other more commonly used search algorithms [6,7].

A formal description of the algorithm is provided in [4–7]. Basically, an objective function, such as minimization of the sum of squared errors or sum of absolute errors, is chosen for optimizing the network. The objective functions need not be differentiable or even continuous. Using the chosen objective function, each candidate point out of the initial population of randomly chosen starting points is used to evaluate the objective function. These values are then used in assigning probabilities for each of the points in the population. For minimization, as in the case of sum of squared errors, the highest probability is assigned to the point with the lowest objective function value. Once all points have been assigned a probability, a new population of points is drawn from the original population with replacement. The points are chosen randomly with the probability of selection equal to its assigned probability value. Thus, those points generating the lowest sum of squared errors are the most likely to be represented in the new population.

The points comprising this new population are then randomly paired for the crossover operation. Each point is a vector (string) of n parameters (weights). A position along the vectors is randomly selected for each pair of points and the preceding parameters are switched between the two points. This crossover operation results in each new point having parameters from both parent points.

Finally, each weight has a small probability of being replaced with a value randomly chosen from the parameter space. This operation is referred to as mutation. Mutation enhances the GA by intermittently injecting a random point in order to

better search the entire parameter space. This allows the GA to possibly escape from local optima if the new point generated is a better solution than has previously been found, thus providing a more robust solution. This resulting set of points now becomes the new population, and the process repeats until convergence.

Since this method simultaneously searches in many directions, the probability of finding a global optimum greatly increases. The algorithm's similarity to natural selection inspires its name. As the GA progresses through generations, the parameters most favorable for optimizing the objective function will reproduce and thrive in future generations, while poorly performing parameters die out, as in "survival of the fittest". Research using the GA for optimization has demonstrated its strong potential for obtaining globally optimal solutions [3,11].

2.2. Simulated annealing

Annealing refers to the process which 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. In this process physical substances usually move from higher energy states to lower ones if the cooling process is sufficiently slow, so minimization naturally occurs. Due to natural variability, however, there is some probability at each stage of the cooling process 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 decreases.

A detailed description of the SA algorithm and its use for optimization can be found in [2]. In essence, SA draws an initial random point to start its search. From this point, the algorithm takes a step within a range predetermined by the user. This new point's objective function value is then compared to the initial point's value in order to determine if the new value is smaller. For the case of minimization, if the objective function value decreases it is automatically accepted and it becomes the point from which the search will continue. The algorithm will then proceed with another step. Higher values for the objective function may also be accepted with a probability determined by the Metropolis criteria (see [2] or [10]). By occasionally accepting points with higher values of the objective function, the SA algorithm is able to escape local optima. As the algorithm progresses, the length of the steps declines, closing in on the final solution. The Metropolis criteria uses the initial user defined parameters, T (temperature) and RT (temperature reduction factor) to determine the probability of accepting a value of the objective function that is higher. Paralleling the real annealing process, as T decreases the probability of accepting higher values decreases. T is reduced by the function $T_{i+1} = RT \times T_i$, where i is the ith iteration of the function evaluation after every NT iterations. NT is the preset parameter which establishes the number of iterations between temperature reductions. For this study these three parameters were set to values suggested in the literature [2,10]. Unlike the version of the GA used in this study in which all parameters are dynamically assigned by the algorithm, the performance of the SA algorithm depends on user defined parameters. Since the performance of the SA is significantly impacted by the choice of these parameters a range of values were used for three variables (T, RT, NT). We examine 12 different combinations in order to allow SA a full opportunity to obtain a global solution.

3. Monte Carlo study

The Monte Carlo comparison was conducted on the six test problems shown in Eq. (1). The functions for all problems are continuous and all are differentiable except number (3) which contains an absolute value. The GA was compared to twelve variations of the SA algorithm. For the first five test problems, 50 observations were used as the training set. The data was generated by randomly drawing the inputs from the sets $X_1 \in [-100, 100]$ and $X_2 \in [-10, 10]$. In order to test the networks, two additional data sets, consisting of 150 observations each were also constructed. The first test set was generated to test the ability of the optimized ANN to interpolate. The

interpolation test data was also drawn from the sets $X_1 \in [-100, 100]$ and $X_2 \in [-10, 10]$, but did not include any common observations with the training set. The second test set was generated to evaluate the ability of the optimized ANN to extrapolate outside of the range of the training set. This data was drawn from $X_1 \in [-200, -101]$ and $X_1 \in [101, 200]$ and from $X_2 \in [-20, -11]$ and $X_2 \in [11, 20]$. 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. 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 were a total of 25 weights. 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 [9,10]. This chaotic series is interesting because of its similarity to time 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 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 points that were generated from the randomly chosen point [−0.218357539, 0.05555536, 1.075291525, −1.169494128, 0.263368033].

Experiment problems

$$
\begin{aligned}
&(1)\ Y = X_1 + X_2, \\
&(2)\ Y = X_1 X_2, \\
&(3)\ Y = \frac{X_1}{|X_2| + 1}, \\
&(4)\ Y = X_1^2 - X_2^3, \\
&(5)\ Y = X_1^3 - X_1^2, \\
&(6)\ Y_t = Y_{t-1} + 10.5\left[\frac{0.2Y_{t-5}}{1 + (Y_{t-5})^{10}} - 0.1Y_{t-1}\right].
\end{aligned}
\tag{1}
$$

3.1. Training with simulated annealing

The version of SA used in this study came from [10]. This code, like the GA program was written in Fortran 77 and run on a Cray J-916 supercomputer. In training each problem using SA, there were three factors that were manipulated in an effort to find the best configuration for optimizing each problem. They included the temperature (T) at three levels, the temperature reduction factor (RT), and the number of function evaluations before temperature reduction (NT), each at two levels. The different combinations of these factors were used in an attempt to try and 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 [10]. Specifically, the number of cycles (NS) was set to 20. After $\text{NS} \times N$ function evaluations, each element of VM (step size) is adjusted so that approximately half of all function evaluations are accepted. The number of final function evaluations (NEPS) was set to 4. This is used as part of the termination process. The error tolerance for termination was set to 10^{-6} 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. [10] and Corana et al. [2], a more rigorous exploration of the parameters may have produced better performance for the SA. Table 1 illustrates the specific parameter settings for each combination. Since the optimal solutions are not known a

Table 1
Simulated annealing parameters

Parameter	Values used
Temperature (T)	5 50,000 50 Million
Temperature reduction factor (RT)	0.50, 0.85
Number of function evaluations before temperature reduction	5, Max(100, $5 \times N$), N is the solution vector size

Table 2
GA and SA best and worst in-sample comparisons

Problems	GA		SA	
	Best	Worst	Best	Worst
1	4.16×10^{-7}	8.46×10^{-6}	1.05×10^{-5}	3.92×10^{-3}
2	1.27×10^{-2}	5.07×10^{-1}	3.17×10^{-2}	3.70
3	1.82×10^{-1}	1.29	1.77	4.10
4	4.09×10^{-2}	2.20	4.84×10^{-1}	4.58×10^{-1}
5	2.90×10^{-3}	1.09	4.39×10^{-1}	1.30×10^{-1}
6	2.53×10^{-2}	1.29×10^{-1}	1.02×10^{-2}	1.93×10^{-1}

priori, the size of the search space was set between −600 and 600.

Each of the 12 configurations was trained with 10 replications. Each replication was started with new random seeds, thus there were 120 SA trials for each problem. 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.

3.2. Training with the genetic algorithm

The particular GA used in this comparison was obtained from [7]. This program was also written in Fortran 77 and run on the CRAY J-916. This algorithm requires no preselection of parameters other than the number of hidden nodes, therefore only 10 replications were run to test the effectiveness of this algorithm. Each replication differed only in the random initialization of the network. Each training replication was trained for 5000 generations, which included a search of 20 points per generation. The same range for the parameter search [−600, 600] was used as was used with SA. Although 5000 generations did not allow full convergence for any of the six problems it was sufficient to demonstrate the effectiveness of this global search technique.

4. Results

Since the structure of the ANN was held constant there were no additional parameters to adjust with the GA. Therefore, the test problems were only trained 10 times each. The sole difference between each replication was the random initialization of the weights. The GA was terminated after 5000 generations as a base line for comparison with SA. A solution arbitrarily close to zero can be found given additional time for convergence. Although, the preset termination of the GA did not enable it to obtain the global solution, it provided a comparison point with the SA results for in-sample, interpolation and extrapolation data. In many cases, the GA's worst in-sample solutions dominated the best SA results. The Mackey–Glass equation, problem 6, was the only problem in which the best GA solution was not superior to the best SA in-sample results. The best in-sample results for the GA and SA are shown in Table 2.

Although, the 12 configurations of parameters for the SA included the recommended parameter settings of two previous studies [2,10], none of the six test problems found an optimal solution with this configuration. In order to determine the best configuration, each test problem was initialized at 10 different random points for all 12 configurations. Out of these 120 replications for each

Table 3
SA's best and worst configurations

Problems	Best			Worst		
	T	RT	NT	T	RT	NT
1	50,000	0.5	100	50,000	0.5	5
2	50 M	0.85	100	50 M	0.5	5
3	50,000	0.5	5	50,000	0.5	5
4	50,000	0.5	100	50,000	0.5	5
5	50 M	0.5	5	50 M	0.5	5
6	5	0.85	5	50,000	0.5	5

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, RT, and NT, we cannot conclude that our configuration is the optimal configuration. A difficulty we experienced with SA is that there are no rules for adjusting the configuration to produce better results. Heuristics for setting SA parameters are suggested in [2] and [10] and the suggestion to set NT = 5 from [10] and NT = Max[100, 5 × N] from [2] can be followed, as was done in this study, but there is no guarantee that an optimal configuration for the algorithm will be found. In calculating Max[100, 5 × N] for NT, only the number of input weights were used. SA only searched over these values, while the output weights were found with OLS. The SA configurations that achieved the best and worst solutions out of the 120 different replications are shown in Table 3.

The best solutions, or sets of weights, for the in-sample data were then applied to out-of-sample interpolation and extrapolation data for each corresponding problem. Each data set consisted of 150 observations drawn uniformly from the ranges described earlier. Although SA had many more function evaluations, the GA obtained superior in-sample solutions in every case except problem 6. A direct comparison of in-sample, interpolation and extrapolation results can be found in Table 4. The numbers shown in Table 4 are for the best solution of the 10 runs for the GA and the best solution of the 120 runs for SA. The ANN solution dominated those of SA in every case for interpolation and extrapolation.

Table 5 shows the processing time and number of function evaluations for the search techniques. Even though the GA was terminated prematurely, the solutions typically dominated the SA results despite the fact that the SA search involved far more function evaluations. The numbers shown in Table 5 are for the best solution of the 10 runs for the GA and the best solution of the 120 runs for SA.

The SA estimates often diverge when estimating out-of-sample data. An examination of the solution weights revealed that in searching for a global solution, the SA algorithm typically reduced the complexity of the problems by often selecting negative and positive values that were very large in absolute terms. This caused the sigmoid function to output either 1 or 0 for many of the hidden nodes. For all six problems the SA algorithm chose weights for the in-sample data sets that caused at least three out of the six hidden nodes to output either 1's, 0's, or a combination of 1's and 0's. By converging on this type of local, the SA algorithm severely limited its out-of-sample estimation ability. When interpolation or extrapolation data was introduced that did not correspond with the given weights, it was unable to accurately forecast the data.

Table 4
Best RMS error comparison for in-sample, interpolation, and extrapolation

Problems	In-sample		Interpolation		Extrapolation	
	GA	SA	GA	SA	GA	SA
1	4.16×10^{-7}	1.05×10^{-5}	3.68×10^{-7}	1.56×10^{-5}	6.74×10^{-5}	1.48×10^{-4}
2	1.27×10^{-2}	3.17×10^{-2}	1.72×10^{-2}	1.85×10^{-1}	0.869	7.42
3	1.82×10^{-1}	1.76	1.38	5.53	7.85	8.06
4	4.09×10^{-2}	4.84×10^{-1}	4.53×10^{-2}	2.73	7.40	234.26
5	3.06×10^{-3}	4.39×10^{-1}	2.02×10^{-2}	1.36	104.99	1647.81
6	2.53×10^{-2}	1.02×10^{-2}	3.40×10^{-2}	2.69×10^{-1}	NA	NA

Table 5
Time and function evaluation comparison

Problems	Time on Cray (s)		Number of function evaluations	
	GA	SA	GA	SA
1	53	780	100,000	2,625,001
2	53	3786	100,000	12,625,001
3	122	70	100,000	112,501
4	122	1574	100,000	2,562,501
5	113	58	100,000	96,901
6	181	420	100,000	473,001

Figs. 1–3 are XY graphs showing the true values vs. forecasts from the ANNs trained with the two algorithms for problems 3, 4, and 6. The *x*-axis ranged from the 100th interpolation data point to the last data point in their respective interpolation data set. Both sets of forecasts tend to track the function for the interpolation data but the forecast from the SA trained ANN is very irregular. The GA trained ANN is relatively smooth thus allowing the derivative of the function to be more accurately computed and to better approximate the true function in general.
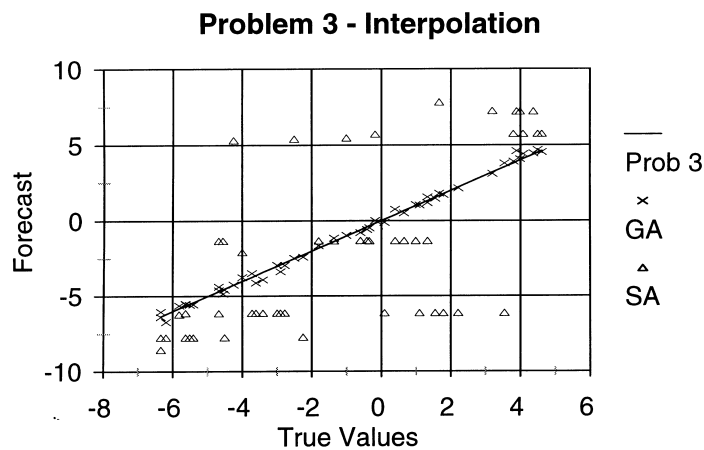
Problem 3 - Interpolation

Fig. 1. Problem 3 – Interpolation.

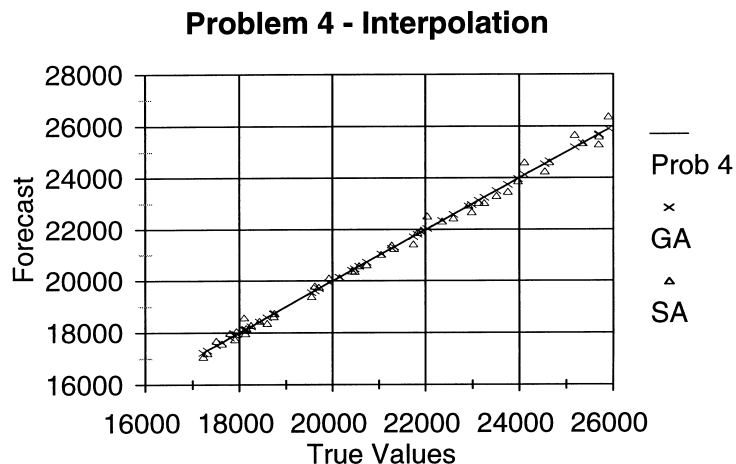### Problem 4 - Interpolation

Fig. 2. Problem 4 – Interpolation.
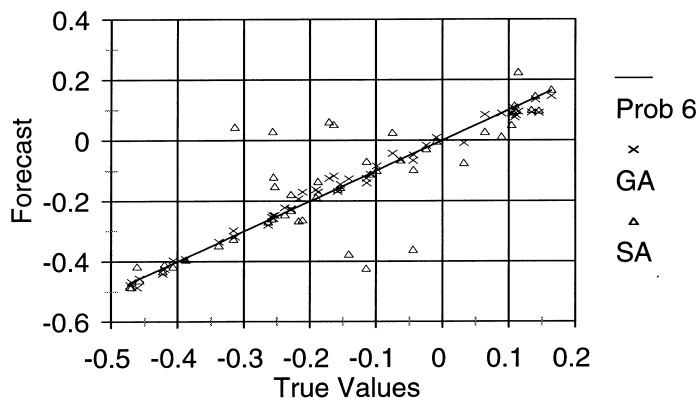
Problem 6 - Interpolation

Fig. 3. Problem 6 – Interpolation.

Since the SA example obtained one run which found an in-sample solution superior to the one found by GA for problem 6 further exploration of the performance of the two algorithms on this problem seemed warranted. In addition to the architecture described above four other architectures were examined. These contained 2, 4, 8 and 10 hidden nodes. As before, for each case, 10 replications were run, each starting from a different random seed. The genetic algorithm was trained for 15,000 generations and then terminated while the SA was allowed to run until it self terminated. The results are shown in Tables 6–9. As before, the average in-sample solution obtained by the GA was substantially better than that obtained by SA and the variation was also much smaller. The out-of-sample performance by the GA trained ANN dominated the performance of the SA trained ANN in every case. The one case reported earlier for the 6 hidden node SA trained ANN is still the best in-sample solution found but the best solutions found for the 8 and 10 hidden node cases by the GA are within 0.002 of that solution for the in-sample RMSE. The best out-of-sample results were again all obtained by the GA for all architectures.

Table 6
Average in-sample results for problem 6

Hidden nodes	GA RMSE	SA RMSE	GA StDev	SA StDev
2	1.33×10^{-1}	2.68×10^{-1}	2.48×10^{-9}	7.31×10^{-2}
4	5.37×10^{-2}	1.34×10^{-1}	9.08×10^{-7}	4.53×10^{-3}
6	3.38×10^{-2}	9.55×10^{-2}	6.46×10^{-3}	3.04×10^{-2}
8	2.37×10^{-2}	7.93×10^{-2}	1.06×10^{-2}	1.16×10^{-2}
10	2.29×10^{-2}	5.15×10^{-2}	1.10×10^{-2}	7.66×10^{-3}

Table 7
Average out-of-sample results for problem 6

Hidden nodes	GA RMSE	SA RMSE	GA StDev	SA StDev
2	1.66×10^{-1}	3.41×10^{-1}	7.33×10^{-8}	9.23×10^{-2}
4	6.89×10^{-2}	2.19×10^{-1}	2.24×10^{-6}	1.93×10^{-2}
6	4.32×10^{-2}	4.04×10^{-1}	6.37×10^{-3}	1.18×10^{-1}
8	3.19×10^{-2}	5.12×10^{-1}	1.16×10^{-2}	1.04×10^{-1}
10	3.14×10^{-2}	6.45×10^{-1}	1.26×10^{-2}	8.74×10^{-2}

Table 8
Best solutions for problem 6

Hidden nodes	In-sample		Out-of-sample	
	GA RMSE	SA RMSE	GA StDev	SA StDev
2	1.33×10^{-1}	1.58×10^{-1}	1.66×10^{-1}	2.00×10^{-1}
4	5.37×10^{-2}	1.26×10^{-1}	6.89×10^{-2}	2.05×10^{-1}
6	1.75×10^{-2}	1.02×10^{-2}	2.54×10^{-2}	2.69×10^{-1}
8	1.19×10^{-2}	6.12×10^{-2}	1.66×10^{-2}	4.41×10^{-1}
10	1.16×10^{-2}	3.87×10^{-2}	1.67×10^{-2}	6.42×10^{-1}

Table 9
Average CPU time and function evaluations (FE) for problem 6

Hidden nodes	GA Time	SA Time	GA FE	SA FE
2	193	132	300,000	171,841
4	437	622	300,000	463,201
6	681	1058 [a]	300,000	516,381
8	905	1951	300,000	744,481
10	1161	3006	300,000	928,801

[a] This time was estimated PC seconds calculated from Cray CPU time.

5. Additional comparison using a classification and time series problem

Two additional real world problems were examined using GA, SA and BP. Since, the BP algorithm used in this comparison is PC based, the GA and SA programs were ported and optimized for a PC based 200 MHz Pentium Pro using the NT 4.0 operating system.

5.1. Data set descriptions

The first problem is a two group classification problem for predicting breast cancer [18]. This data set included nine inputs that were constructed based on visual inspection by an experienced oncologist of possible cancerous cells. These values include, clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleioli, and mitoses. More detailed information on this data set can be found in [17,18]. The output value classified individuals as whether or not the patient's breast lump was malignant and were set to the discrete values of −1 for non-malignant and 1 for malignant. The original research classified this data using LP techniques for net-

work optimization. This training data set included 369 records, each consisting of a nine dimensional input vector and corresponding output. The testing set included an additional 334 records. The data was downloaded from ftp.cs.wisc.edu in the directory /math-prog/cpo-dataset/machine-learn/.

Since the training data set included more negative outputs than positive, the cutoff value for classification was adjusted based on the frequency in the training set, which moved the value from 0.0 to −0.112 on a [−1, 1] scale. The network architecture was held constant at six hidden nodes for the GA, SA, and BP. Comparisons were made of total classification rate as well as type 1 and 2 errors. Type 1 error is defined as the event of assigning an observation to group 1 that should be in group 2, while the type 2 error involves assigning an observation to group 2 that should be in group 1. In the current comparison we will refer to negative outcomes (non-malignant) to group 1 and positive outcomes (malignant) to group 2.

The second problem is a financial time series used to predict the following day's S&P 500 index closing price. The eight inputs included the previous daily percentage change of the close for the Dow Jones Industrial Average, close for the Dow Jones Transportation Average, total volume for the New York Stock Exchange, close for the Dow

Jones 20 Bond Average, close for the Dow Jones Utility Average, high for the S&P 500 index, low for the S&P 500 index, and the close for the S&P 500 index. The observations were drawn from 4 January 1993 to 13 December 1996, using the first 800 observations for training and the last 200 observations for testing. The network architecture was held constant at six nodes in the hidden layer. This architecture and selection of input variables are not proposed as being optimal for solving this problem but are used solely for comparisons of how well each algorithm achieved a minimal solution given the same data. Comparisons were made based on the 200 out-of-sample observations. The data for this problem was downloaded from the *Journal of Computational Intelligence in Finance* site at http://ourworld.compuserve.com/homepages/ftpub/dd.htm.

5.2. Training with Backpropagation

Commercial neural network software was evaluated in order to determine which package would provide the best estimates for the two 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, a PC-based neural network application seemed to give the best estimates and was chosen for the test series. When training with BP, there are many parameters which can be manipulated. For this comparison the initial step size and momentum was set to 1 and 0.9, respectively. In order to better converge upon a solution during the training process the learning coefficient (Lcoef) parameter was set to the recommended value of 0.5. This parameter decreases the step size and momentum term by 0.5 for every 10,000 epochs in order to eliminate osculation. An epoch can be defined as one pass through the training set. The specific BP algorithm used was the Norm-Cum-Delta. The BP networks were trained for 1,000,000 epochs, at which time for

every 10,000 additional epochs the error was checked to determine if a new solution was found that was smaller than the last. Termination of training occurred once there were 10 consecutive checks were there were no improvements. For each problem the BP networks were replicated 10 times starting each replication with a different random seed.

5.3. Training with simulated annealing

The code for SA was ported and optimized for a PC-based machine so that results could be compared directly with the PC-based BP software. The recommended parameter values were used for both problems. Similar to BP, there were 10 replications, for each problem, using different random seeds for network initialization. The SA networks terminated once the training error tolerance reached 10^{-6} or maximum evaluations exceeded 10^9.

5.4. Training with the genetic algorithm

The code for the GA was ported and optimized for a PC-Based machine to compare results directly with the BP and SA PC-based software. Ten replications were run, only changing the random seed for each replication. The number of generations for the GA was limited to 5000 or in other words 100,000 function evaluations for each problem.

5.5. Classification results

Each of the algorithms was evaluated based on the total classification rate, as well as type 1 and type 2 classification errors. This rate is the average correct classifications for all 10 replications for each of the algorithms. In addition CPU time and the number of function evaluations for each algorithm are shown for each technique. Table 10 provides the data for all algorithms for both problems. As can be seen, the GA found superior classification rates. While, the GA found these

Table 10
Average algorithm comparisons for the breast cancer problem

Algorithm	GA	SA	BP
Total classification rate	98.5%	97.2%	97.2%
Type 1 error rate	1.5%	3.1%	2.3%
Type 2 error rate	1.4%	1.6%	4.1%
Number of function evaluations	100,000	493,635	1,174,000
CPU time in seconds [a]	618	4,812	497

[a] CPU times were based on a 200 MHz Pentium-Pro machine.

solutions in a shorter time than SA, it took on average 22% longer to converge than BP. Although, BP converged on a solution faster, it converged upon inferior solutions.

In comparing SA with BP for these two problems, SA's total cumulative classification rate was the same as BP. BP found superior classification rates for type 1 errors, while SA found superior classification rate for type 2. These results do not indicate that BP is a superior algorithm for finding global solutions over SA. Since the only differences between the replications was the initial random draw of weights it is apparent that SA is greatly affected by the starting point of the search.

5.6. Time series results

The second comparison uses financial data to compare the three algorithms on their ability to estimate a time series problem. Specifically to estimate the S&P 500 index one day in the future. The mean RMSE, standard deviation of the RMSE, best solution RMSE, worst solution RMSE, number of function evaluations, and CPU time for the three algorithms are shown in Table 11. In every replication the GA found better solutions than BP. As shown in Table 11, the GA's

worst solution out of the 10 replications was better than BP's best solution. It can also be seen by looking at the standard deviations of the GA and BP, that both of these algorithms were consistent in obtaining consistent results across different replications. In the comparison between the GA and SA, the GA's average RMSE was superior to SA's average RMSE. In fact, the GA's average RMSE was the same as the best SA's RMSE. It is interesting to note for SA that the standard deviation was significantly higher than the other two algorithms. This may be due in part to a dependency on the initial random draw of weights, and may be in part be due to the selection of parameter settings for the specific problem being estimated. Although, SA found two solutions that were better than the best BP solutions it failed to out perform BP on average.

The Wilcoxon Matched Pairs Signed Ranks test was conducted to show the differences between each of the 10 replications for each algorithm. Table 12 illustrates the number of replications for each comparison where an algorithm found significantly superior solutions, at the 0.01 level of significance, over another algorithm. As can be seen in this table the GA found superior solutions over BP in all 10 replications. However, SA only found two solutions out of the 10 replications that

Table 11
Algorithm comparisons for the time series problem

	GA	SA	BP
Mean RMSE (10 runs)	7.11×10^{-3}	9.49×10^{-2}	7.83×10^{-3}
Standard deviation (10 runs)	3.57×10^{-3}	6.58×10^{-2}	1.56×10^{-3}
Best solution	7.06×10^{-3}	7.11×10^{-3}	7.80×10^{-3}
Worst solution	7.16×10^{-3}	2.05×10^{-1}	7.85×10^{-3}
Number of function evaluations	100,000	580,348	1,228,000
CPU time in seconds	1330	9396	635

Table 12
Statistical comparisons

Comparisons	0.01	level	Above 0.01
GA and SA	GA 8	SA 0	2
GA and BP	GA 10	BP 0	0
SA and BP	SA 2	BP 8	0

were significantly superior to BP. In comparing the GA with SA, the GA found 8 of 10 solutions that were significantly better than SA, while the other two comparisons were not significantly different at the 0.10 level.

6. Conclusions

In order for researchers to utilize artificial neural networks as a highly versatile flexible form estimation tool, methods of global optimization must be found. The most frequently used algorithm for optimizing neural networks today is backpropagation. This gradient search technique is likely to obtain local solutions. Sexton et al. [24] have recently demonstrated that the local solutions obtained by backpropagation often perform poorly on even simple problems when forecasting out-of-sample. Simulated annealing, a global search algorithm, performs better than backpropagation, but also uses a point to point search. Both BP and SA have multiple user determined parameters which may significantly impact the solution. Since there are no established rules for selecting these parameters, solution outcome is based on chance.

The genetic algorithm appears to be able to systematically obtain superior solutions to simulated annealing for optimizing neural networks. These solutions provide the researcher with superior estimates of interpolation data. The genetic algorithm's process of moving from one population of points to another enables it to discard potential local solutions and also to achieve the superior solutions in a computationally more efficient manner.

References

[1] N. Archer, S. Wang, Application of the back propagation neural network algorithm with monotonicity constraints for two-group classification problems, Decision Sciences 24 (1) (1993) 60–75.

[2] A. Corana, M. Marchesi, C. Martini, S. Ridella, Minimizing multimodal functions of continuous variables with the simulated annealing algorithm, ACM Transactions on Mathematical Software 13 (1987) 262–280.

[3] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Michigan, 1975.

[4] R.E. Dorsey, W.J. Mayer, Optimization using genetic algorithms, in: J.D. Johnson, A.B. Whinston (Eds.), Advances in Artificial Intelligence in Economics, Finance, and Management, vol. 1, JAI Press, Greenwich, CT, pp. 69–91.

[5] R.E. Dorsey, W.J. Mayer, Genetic algorithms for estimation problems with multiple optima, non-differentiability, and other irregular features, Journal of Business and Economic Statistics 13 (1) (1995) 53–66.

[6] R.E. Dorsey, J.D. Johnson, W.J. Mayer, The genetic adaptive neural network training (GANNT) for generic feedforward artificial neural systems, School of Business Administration Working Paper, available at http://www.bus.olemiss.edu/johnson/compress/compute.htm.

[7] R.E. Dorsey, J.D. Johnson, W.J. Mayer, A genetic algorithm for the training of feedforward neural networks, in: J.D. Johnson, A.B. Whinston (Eds.), Advances in Artificial Intelligence in Economics, Finance and Management, vol. 1, JAI Press, Greenwich, CT, 1994, pp. 93–111.

[8] K. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (3) (1989) 183–192.

[9] R.A. Gallant, H. White, On learning the derivatives of an unknown mapping with multilayer feedforward networks, in: Artificial Neural Networks, Blackwell, Cambridge, MA, 1992, pp. 206–223.

[10] W.L. Goffe, G.D. Ferrier, J. Rogers, Global optimization of statistical functions with simulated annealing, Journal of Econometrics 60 (1994) 65–99.

[11] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison–Wesley, Reading, MA, 1989.

[12] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.

[13] J.T. Hsiung, W. Suewatanakul, D.M. Himmelblau, Should backpropagation be replaced by more effective optimization algorithms? in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), vol. 7, 1990, pp. 353–356.

[14] Y. LeCun, Learning processes in an asymmetric threshold network, in: Disordered Systems and Biological Organization, Springer, Berlin, 1986, pp. 233–240.

[15] M. Lenard, P. Alam, G. Madey, The applications of neural networks and a qualitative response model to the auditors going concern uncertainty decision, Decision Sciences 26 (2) (1995) 209–227.

[16] G.R. Madey, J. Denton, Credit evaluation with missing data fields, in: Proceedings of the INNS, Boston, 1988, p. 456.

[17] O.L. Mangasarian, Mathematical programming in neural networks, ORSA Journal on Computing 5 (4) (1993) 349–360.

[18] O.L. Mangasarian, W.N. Street, W.H. Wolberg, Breast cancer diagnosis and prognosis via linear programming, Operations Research 43 (4) (1995) 570–577.

[19] D.J. Montana, L. Davis, Training feedforward neural networks using genetic algorithms, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1989, pp. 379–384.

[20] D. Parker, Learning logic, Technical Report TR-87, Center for Computational Research in Economics and Management Science, MIT Press, Cambridge, MA, 1985.

[21] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: D. Rumelhart, J. McCleeland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, Cambridge, MA, 1986.

[22] L.M. Salchenberger, E.M. Cinar, N.A. Lash, Neural networks: A new tool for predicting thrift failures, Decision Sciences 23 (1992) 899–916.

[23] R. Sexton, B. Allidae, R. Dorsey, J. Johnson, Global optimization for artificial neural networks: A tabu search application, European Journal of Operational Research 106 (2–3) (1998) 570–584.

[24] R. Sexton, R. Dorsey, J. Johnson, Toward a global optimum for neural networks: A comparison of the genetic algorithm and backpropagation, Decision Support Systems 22 (2) (1998) 171–186.

[25] J.D. Schaffer, D. Whitley, L.J. Eshelman, Combinations of genetic algorithms and neural networks: A survey of the state of the art, in: COGANN-92 Combinations of Genetic Algorithms and Neural Networks, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 1–37.

[26] J.D. Schaffer, Combinations of genetic algorithms with neural networks or fuzzy systems, in: J.M. Zurada, R.J. Marks, C.J. Robinson (Eds.), Computational Intelligence: Imitating Life, IEEE Press, 1994, pp. 371–382.

[27] V. Subramanian, M.S. Hung, A GRG@-based system for training neural networks: Design and computational experience, ORSA Journal on Computing 5 (4) (1990) 386–394.

[28] S. Wang, The unpredictability of standard back propagation neural networks in classification applications, Management Science 41 (3) (1995) 555–559.

[29] R.L. Watrous, Learning algorithms for connections and networks: Applied gradient methods of nonlinear optimization, in: Proceedings of the IEEE Conference on Neural Networks, vol. 2, San Diego, IEEE, 1987, pp. 619–627.

[30] P. Werbos, The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting, Wiley, New York, 1993.

[31] H. White, Some asymptotic results for back-propagation, in: Proceedings of the IEEE Conference on Neural Networks, vol. 3, San Diego, IEEE, 1987, pp. 261–266.