JOURNAL OF
ENVIRONMENTAL HYDROLOGY

The Electronic Journal of the International Association for Environmental Hydrology
On the World Wide Web at http://www.hydroweb.com

VOLUME 16 2008

PERMEATE FLUX ESTIMATIONS OF CROSSFLOW MEMBRANE FILTRATION SYSTEMS USING GENETIC ALGORITHMS OPTIMIZED ARTIFICIAL NEURAL NETWORKS

Goloka Behari Sahoo[1]
Chittaranjan Ray[2]

[1]Department of Civil and Environmental Engineering
University of California at Davis, Davis, CA, USA
[2]Department of Civil and Environmental Engineering
University of Hawaii at Manoa, Honolulu, HI, USA

The geometry and modeling parameters of an artificial neural network (ANN) have significant effects on its predictive performance efficiency. The optimal geometry of an ANN is problem-dependent. Although some guidance is available in the literature for the choice of geometry and modeling parameters, most ANNs are calibrated using the trial-and-error approach. This paper presents the use of genetic algorithms (GAs) to search for the optimal geometry and values of modeling parameters of a multilayer feedforward back-propagation neural network (BPNN) and a radial basis function network (RBFN). The predictive performance efficiency of the GA-ANN combination is examined using an already published experimental dataset from a crossflow membrane filtration experiment. The data include the permeate flux decline under various operating conditions (e.g., transmembrane pressure and filtration time) with different physicochemical properties of feed water (e.g., different combinations of three particle diameters, three pH values, and four ionic strengths). It is illustrated that the GA-optimized ANN predicts the permeate flux decline more accurately than a network in which the ANN calibration is accomplished using a trial-and-error approach. It is shown that scaling the training dataset to the 0 to 1 range helps the modeler find the solution range of an RBFN using GA.

INTRODUCTION

Modeling techniques based on the direct analysis of experimental datasets (descriptive models) appear to be good alternatives to models that use phenomenological hypotheses (i.e. knowledge-based models) (Song 1998; Dormier et al. 1995; Razavi et al. 2003; Zhao et al. 2005). In recent years, artificial neural networks (ANNs) have been widely used in many fields such as chemical and environmental engineering (Shetty et al. 2003; Chakraborty et al. 2003; Chellam 2005; Zhao et al. 2005) and in hydrology and water resources applications (Maier and Dandy 1998, 2000; ASCE Task Committee 2000; Govindaraju and Rao 2000; Sahoo and Ray 2006), for optimum prediction of system parameters and variables. However, in most cases, parameters and system variables were forecasted employing suboptimal ANNs. This study presents an innovative approach for optimization of an ANN using a case study.

The crucial tasks in BPNN are to design a network with a specific number of layers each having a certain number of neurons and to train the network optimally, so that it can map the system's nonlinearity reasonably well. Maier and Dandy (2000), ASCE Task Committee (2000), and Birikundavyi et al. (2002) stressed that optimal neural network design is problem-dependent and is usually determined by trial and error, i.e. sensitivity analysis. Kingston et al. (2005) pointed out that a significant component of prediction uncertainty can be attributed to the suboptimal values of the parameters that govern the model function. Thus, it is necessary to determine the optimal network geometry and modeling parameters (e.g., spread and maximum number of neurons for an RBFN; and the number of neurons in first and second hidden layers, and the maximum epoch size for training a four-layer BPNN) of an ANN. Using the published permeate flux decline of a cross-flow membrane (CFM) experimental dataset of Faibish et al. (1998), it is shown in Table 1 that the spread and maximum number of neurons assigned in the RBFN have significant effects on the accuracy of prediction.

For RBFN, if the solution space for the spread and the maximum number of neurons range from 1 to 200 and 1 to 100, respectively, then there are 100 ′ 200 = 20,000 ways of arranging them. The spread is a real number; so, the number of ways of arranging the spread and the maximum number of neurons increases significantly. Searching for the optimal solution in this range using the trial-and-error approach is cumbersome and time consuming. In such cases, a genetic algorithm (GA)

Table 1. Effects of spread and maximum number of neurons of an RBFN on its predictive performance using unscaled small dataset.

Scenario	Spread	Neurons in the hidden layer	Performance efficiency					
			Training			Testing		
			R	ME	RMSE	R	ME	RMSE
1	10	70	0.9993	0.0000	0.0333	0.9603	0.0713	0.2069
2	50	70	0.9996	0.0000	0.0250	0.9911	-0.0428	0.0989
3	90	70	0.9998	0.0000	0.0172	0.9930	-0.0372	0.0872
4	110	70	0.9997	0.0000	0.0228	0.9931	-0.0367	0.0856
5	140	70	0.9995	-0.0001	0.0262	0.9919	-0.0361	0.0903
6	200	70	0.9991	0.0005	0.0376	0.0949	-2.4844	5.9507
7	80	10	0.9652	0.0000	0.2271	0.9502	-0.0469	0.2167
8	80	50	0.9988	0.0000	0.0430	0.9895	-0.0435	0.1042
9	80	70	0.9999	0.0000	0.0132	0.9930	-0.0372	0.0882
10	80	90	0.9999	0.0000	0.0043	0.0108	-75.113	141.80

can be used to search through the large space and determine the optimal arrangement based on an objective function (R in this study).

Bowden et al. (2002) reported that the input dataset for the ANN model should be scaled to the 0 to 1 range. The main reasons for scaling the dataset to this range are (1) to prevent inputs with much larger values from dominating in the training (Maier and Dandy, 2000) and (2) to enable penalty constraints to be included more easily (i.e. the maximum and minimum values can be identified by the GA as zeroes and ones). They also reported that size (i.e. the number of samples) of the training dataset has significant effect on the ANN's predictive performance efficiency.

In the recent past, ANNs were used extensively to predict chemical and environmental parameters and/or variables under different physicochemical conditions (e.g., Zhao et al. 2005; Chellam 2005; Aydiner et al. 2005) and to model water resources variables (e.g., Bowden et al. 2002; Akin 2005; Sahoo et al. 2005; Sahoo and Ray, 2006); however, all of these studies reported calibrating ANNs' geometry and modeling parameters using the trial-and-error approach. Therefore, the objectives of this study are (1) to develop methodologies for determining ANNs' geometry and modeling parameters using GA, (2) to evaluate the effect of scaling the input training dataset to the 0 to 1 range on ANN predictive performance efficiency, (3) to examine the effect of training dataset size — a small type (96 samples) or a large dataset (441 samples) — on ANN predictive performance efficiency, and (4) compare the GA optimized ANN model predictive efficiency with those of traditional multiple regression analysis. All analyses are performed on the published dataset of Faibish et al. (1998).

ARTIFICIAL NEURAL NETWORKS

Two types of ANN models, a four-layer (one input layer, two hidden layers, and one output layer), BPNN and an RBFN, were used in this study. The general structure of the RBFN and BPNN can be found in Sahoo et al. (2005) and Sahoo and Ray (2006), and the details can be found in Hagan et al. (1996), and Haykin (1999), and Sahoo and Ray (2006). The functions available in the Neural Network Toolbox of MATLAB (MathWorks Inc. 2002) were modified and used to create an RBFN and a BPNN.

Radial Basis Function Network (RBFN)

An RBFN starts with a minimal network of only one radial basis neuron. At each iteration, the global error (i.e. global mean square error) of the network is estimated and compared with the error goal (i.e. the threshold minimum mean square error set to 10^{-20} in this study) of the network. If the global error is above the threshold error goal, one radial basis neuron is added to the network. Otherwise, the training is stopped. The procedure is repeated until either the error goal is met or the maximum number of neurons is reached. The trained network is employed to predict flux decline using the testing dataset which is not used in the training process. The predictive performance efficiency of the network is estimated comparing the predicted values with measured values. The general form of the radial basis function F_i is

$$F_i(x) = \varphi\|x - c_i\| \tag{1}$$

where c_i is the center of the ith radial basis function (RBF) neuron, $\varphi(\cdot)$ is the radially symmetric basis activation function, x is the input vector, and $\|\cdot\|$ is the Euclidean distance between the center of the RBF neuron and input (Haykin 1999; Haddadnia et al. 2003; Sahoo and Ray 2006). Of the several RBFs, the most commonly used is the Gaussian RBF (ASCE Task Committee 2000; Govindaraju and Rao 2000; Haddadnia et al. 2003; Chang and Chen 2003). It is described by

$$F_i = -\exp\left(-\frac{\|x-c_i\|^2}{2\sigma_i^2}\right) \tag{2}$$

where σ_i is the spread or the radial distance from the center of the *i*th RBF neuron

The function value $\varphi(\cdot)$ is highest at the center, *c*, and drops off rapidly to zero as the argument, *x*, moves away from the center. Thus, the neurons in the RBFN have localized receptive fields (Hagan et al. 1996). The spread of a radial basis neuron determines the width of an area in the input space to which each neuron responds. Thus, the spread should be large enough that neurons respond strongly to the overlapping regions of the input space, although too large a spread can cause numerical instability (MathWorks 2002; Sahoo and Ray 2006). It is also important that the spread of each RBF neuron should not be so large that all the neurons respond essentially in the same manner. If the spread is large, the slope of the RBF surface becomes smooth, leading to a large area around the input vector. As a result, several neurons may respond to an input vector. On the other hand, if the spread is small, the RBF surface becomes steep so that neurons with weights closest to the input will have a larger output than other neurons (MathWorks 2002; Sahoo and Ray 2006). When input patterns fall in close proximity to each other, the RBF may have overlapping receptive fields. Therefore, it is important to determine the optimum value of the spread for an RBFN.

The weighted sum of the inputs at the output layer is transformed to the network output using a linear activation function. The output *y* of the RBFN is computed using the following equation (Haykin 1999; Birikundavyi et al. 2002; Haddadnia 2003; Chang and Chen 2003):

$$y = \sum_{i=1}^{N_0} w_i F_i(x) + w_0 \tag{3}$$

where w_0 is the bias, w_i is the connection weight between the hidden neuron and the output neuron, and N_0 is the total number of RBF centers. The first term is the weighted sum of all inputs. Since each RBF center must respond to at least one input pattern, N_0 is always less than or equal to the total number of input patterns (*N*). Thus, setting a large value for N_0 (i.e. close to *N*) does not mean that the network will produce a good result. In such a case, the global error during training may be low because the network may be overtrained. As shown in the Table 1, setting an appropriate number of radial basis neurons in the network is important.

Back-propagation Neural Network (BPNN)

Sahoo and Ray (2006) demonstrated that a BPNN with two hidden layers outperforms a BPNN with one hidden layer. Flood and Kartam (1994) reported that many functions are difficult to approximate with one hidden layer. Flood and Kartam (1994) and Maier and Dandy (1998) noted that the use of more than one hidden layer provides greater flexibility and enables the approximation of complex functions with fewer neurons. Therefore, a two-hidden-layer BPNN is considered for this study. Moreover, Maier and Dandy (1998), and Sahoo and Ray (2006) showed that the optimal ANN predictive performance efficiency depends on the ratio of first hidden layer neurons (h_1) to second hidden layer neurons (h_2). Conversely, the predictive performance efficiency of a network is undermined by either overtraining (i.e. the number of epochs is higher than optimum, E_0) or undertraining (i.e. the number of epochs is lower than optimum, E_0) of a specified network (Principe et al. 1999; Maier and Dandy 2000). Thus, the network structure (i.e. h_1 and h_2) and epoch size should be optimized. Other uncertainties, such as generation of initial weight vector and stopping criteria of ANN training, are enumerated in Sahoo and Ray (2006).

The Levenberg-Marquardt training algorithm (LM) was selected from among the BP training algorithms, because it was reported to have faster convergence ability (see the Appendix) than the gradient-descent method. Additionally, LM does not use a learning rate and momentum factor like the steepest-descent or gradient-descent algorithms do (see Appendix). In the steepest-descent or gradient-descent algorithms, the momentum factor can speed up the ANN training in flat regions of the error surface and can help prevent oscillations in the weight, while a learning rate is used to increase the chance of avoiding the training process being trapped in local minima instead of global minima (Hagan et al. 1996; ASCE Task Committee, 2000). Hagan et al. (1996) and Maier and Dandy (1998, 2000) reported that optimization of these values is highly problem-dependent and should be fixed so that oscillation in the error surface can be avoided. Sometimes, the steepest-descent or gradient-descent algorithms fail to converge on problems for which optimized learning rate and momentum factor find a solution (Hagan et al. 1996). To avoid these two problems, the LM training algorithm was used in this study. The LM algorithm searches in the direction of convergence with minimum error at each iteration by introducing the Hessian matrix containing the second-order derivatives of the quadratic error function and its convergence is faster than that of other algorithms (Hagan et al. 1996; see the Appendix).

Maier and Dandy (1998) and Ray and Klindworth (2000) reported that a tangent sigmoid (‘tansig’) activation function (between -1 and 1) produces better network performance in terms of predictive performance efficiency (e.g., RMSE), convergence, and central processing unit time than the linear (between -1 and 1) and bipolar logarithmic sigmoid (‘logsig’) functions (between 0 and 1). The present study examined the BPNN predictive performance efficiency in terms of R using the ‘tansig’ and ‘logsig’ functions. R-values were found to be 0.9955 and 0.9965 for the ‘tansig’ and ‘logsig’ functions respectively using a large dataset (see section Domain Discussion and Data Used). While using a small dataset, the R-values were found to be 0.9936 and 0.9934 for ‘tansig’ and ‘logsig’, respectively. The R-values were found to be close to each other for both functions and for both datasets. Thus, a hyperbolic tangent sigmoid activation function (between -1 and 1) was chosen for the hidden layers. However, the output layer was provided with a linear (‘purelin’) activation function (see Appendix for equations), so that the output range was between $-\infty$ and ∞ and by doing so avoided re-mapping of the outputs.

GENETIC ALGORITHMS

The genetic algorithm (GA) begins with a set of solutions to the problem under investigation. This set of solutions (represented by chromosomes in GA) is called the population. Each chromosome in the population is evaluated on its performance with respect to the fitness function (R in this study). The three basic operators: selection, crossover, and mutation are applied to select parents for mating on the basis of R, to exchange genetic information between the two parents (i.e. chromosomes) to form offspring and to introduce diversity from one solution to other, respectively. The parents are then killed and replaced in the population by the offspring to keep the population size stable. When generating populations from only two parents, it is possible that the best chromosome from the last population may be lost. To prevent this, “elitism” is used. GA rapidly eliminates chromosomes with poor measures until all the chromosomes in the population are 95% similar.

MULTIPLE REGRESSION ANALYSIS (MRA)

Multiple regression analysis establishes the relationship between several independent or predictor variables (i.e. transmembrane pressure (*P*), ionic strength (*IS*), *pH* value, particle size (*PS*), and time (*T*)) and a dependent or criterion variable (i.e. permeate flux decline (*F*)). The general computational problem that needs to be solved in multiple regression analysis is to fit a straight line to a number of points so that the squared deviations of the observed points from that line are minimized. Thus, this general procedure is sometimes referred to as least squares estimation. Mathematically,

$$F = a_0 + a_1P + a_2IS + a_3pH + a_4PS + a_5T \tag{4}$$

where a_0, a_1, a_2, a_3, a_4, and a_5 are unknown coefficients and the multiple regression model solves by performing the least square method.

DOMAIN DISCUSSION AND DATA USED

Crossflow membrane filtration (CMF) is an established process for the removal of colloidal particles, proteins, macromolecules, and biological particles from fluids (Figure 1). However, significant permeate flux decline due to the hydraulic resistance created by particle accumulation on membrane surfaces cannot be avoided. Permeate flux decline during CMF of colloidal suspensions is governed by concentration polarization and cake formation on the membrane surfaces (Fane and Fell 1987; Belfort et al. 1994; Song 1998). These processes, in turn, are controlled by hydrodynamic and physicochemical operating conditions, such as transmembrane pressure, particle size, crossflow velocity, ionic strength, and solution pH. Predictive performance of membrane filtration using phenomenological models is hampered significantly due to the lack of suitable descriptions of these physicochemical phenomena. This has led many design procedures to be empirical and system-specific (Bowen and Jenner 1995). Thus, analysis of an experimental dataset using an empirical method such as ANN is often more appropriate for prediction purposes.

The experimental permeate flux decline dataset presented in Faibish et al. (1998) was extracted for use in this study. The different physicochemical operating conditions investigated by Faibish

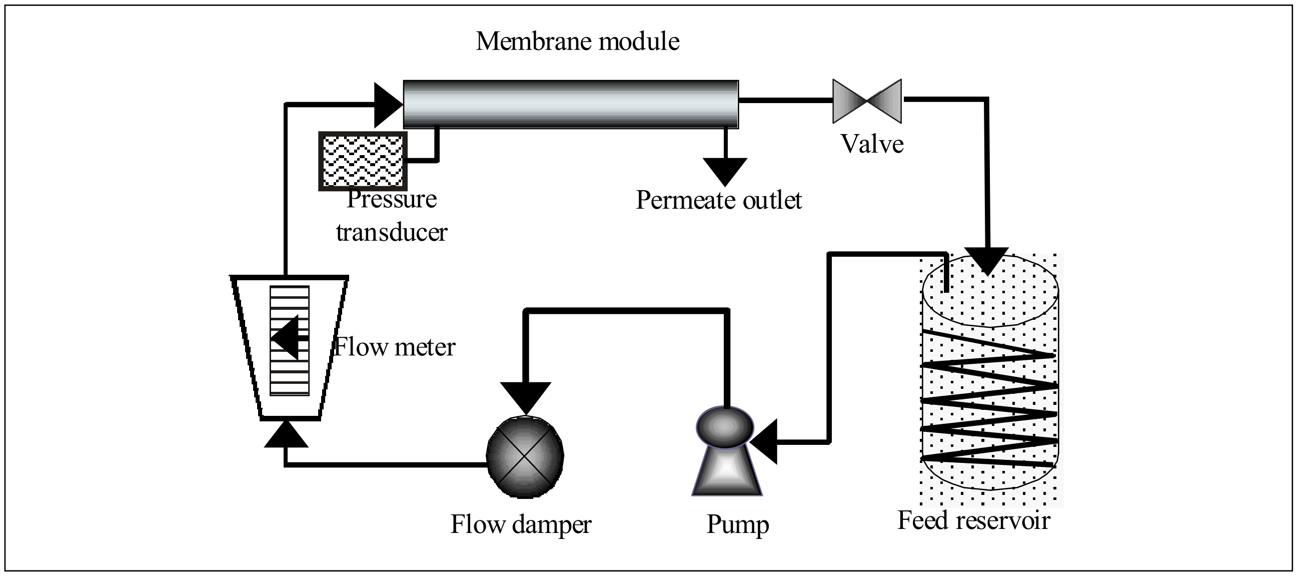

Figure 1. Schematic of crossflow membrane filtration test unit (redrawn after Faibish et al., 1998).

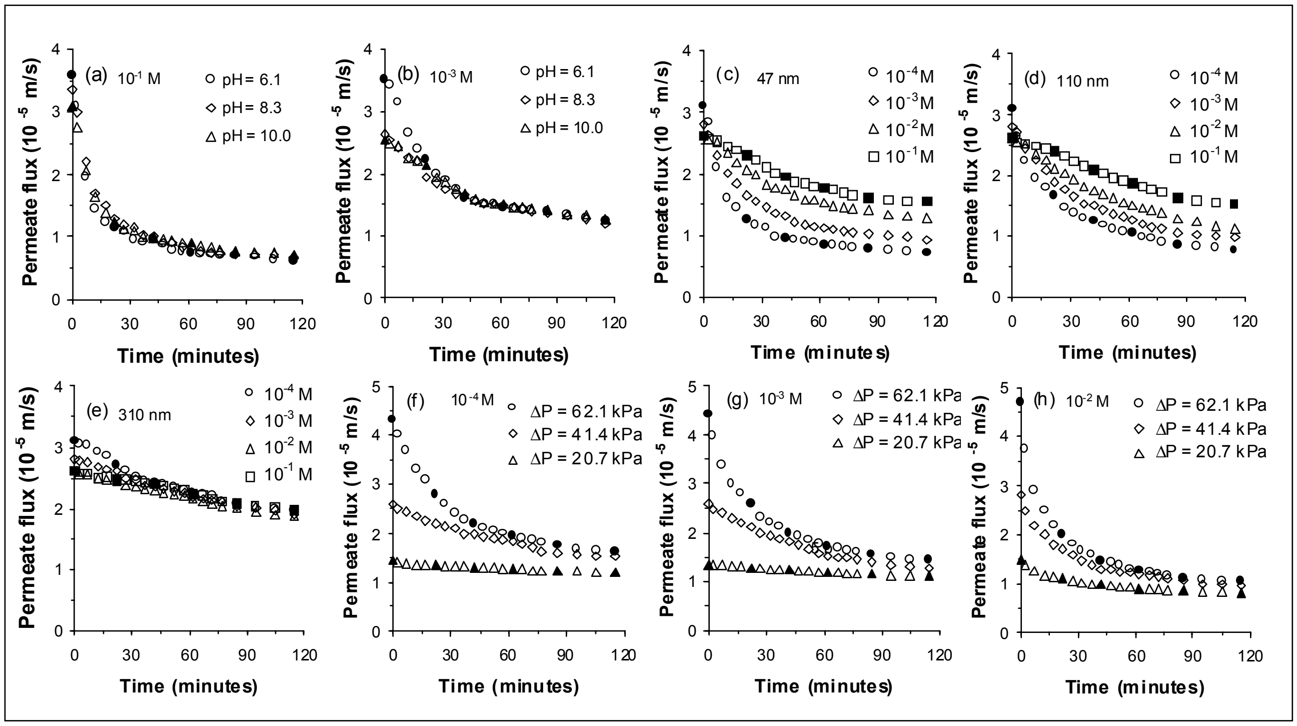

Figure 2. Effect of solution pH on permeate flux decline at (a) high (10^{-1} M KCL) and (b) low (10^{-3} M KCL) ionic strength for filtration conditions of particle size 47 nm and transmembrane pressure 41.4 kPa. Effect of ionic strength on permeate flux decline for filtration condition of (c) 47 nm, (d) 110 nm, and (e) 310 nm diameter particles; pH 10; and transmembrane pressure 41.4 kPa. Effect of transmembrane pressure on permeate flux decline at ionic strength of (f) 10^{-4} M KCL, (g) 10^{-3} M KCL, and (h) 10^{-2} M KCL for filtration condition of particle size 47 nm; and pH 10.0. Crossflow velocity, particle volume concentration, and temperature were 0.246 m/s, 0.01%, and, 20°C, respectively for all cases. All data were extracted from Faibish et al. (1998). Shown are the training (solid symbols) and testing data (hollow symbols) for small dataset.

were (a) three monodisperse silica suspensions with particle diameters of 47, 110, and 310 nm; (b) three pH values of 6.1, 8.3, and 10; (c) four ionic strengths of 10^{-1}, 10^{-2}, 10^{-3}, and 10^{-4} M KCL; and (d) three transmembrane pressures of 20.7 kPa, 41.4 kPa, and 62.1 kPa (see Figure 2). Permeate fluxes for different combinations were measured over a period of 120 minutes. The crossflow velocity 0.246 m s^{-1} corresponds to a shear rate of 280 s^{-1} and a Reynolds number of 1720 (based on the crossflow velocity and inner membrane diameter). The reader is referred to Faibish et al. (1998) for the detailed effects of physicochemical filtration conditions on the permeate flux decline.

Two datasets: a small training dataset (96 of 567 samples) (see Figure 2) and a large training dataset (441 of 567 samples) were prepared from the same population for the ANN and to examine the effect size of training dataset on ANN predictive performance. In the case of the large training dataset, the testing dataset included data points from each middle line of Figure 2(a) and (b) (i.e. data points of pH = 8.3); each third line from top of Figure 2(c), (d) and (e) (i.e. data points of ionic strength =10^{-3} M KCL); and only the middle line of Figure 2(g) (i.e. DP = 41.4 kPa); other data points were selected for the training dataset. Thus, the remaining samples available for testing small and large datasets were 471 and 126, respectively. The ASCE Task Committee (2000) reported that ANNs (BPNN and RBFN) are not good predictors when the prediction dataset is far wide from the training range. Thus, care was taken to include the two input patterns having extreme output values (lowest as well as highest) in small and large datasets. Also, the ASCE Task Committee (2000) and Bowden et al. (2002) recognized that training and testing datasets must be representative of the same population. The training dataset was made large to ensure that the training samples included all information in all dimensions. As shown in Figure 2, the small dataset for training was selected to ensure that it included the highest and lowest values as well as another four equally spaced data points of each scenario of the experimental data so that the training dataset included all ranges of information that the testing dataset included. Since the ionic strengths differ significantly from each other, their reported form in the input dataset may affect the predictive efficiency of the BPNN and RBFN. Therefore, the negative logarithms of these values were used as the input parameter to the ANN models.

METHODOLOGY

Data scaling

The training and testing datasets were scaled to the range of 0 to 1 using the equation $x_{ni} = (x_i - x_{\min})/ (x_{\max} - x_{\min})$, where x_i is the input value, x_{ni} is the scaled input value of the input value x_i, and $x_{\max}$ and $x_{\min}$ are the respective maximum and minimum values of the unscaled measured data. The network-estimated output values, which are in the range of 0 to 1, are converted to real-world values using the equation $x_i = x_{ni} (x_{\max} - x_{\min}) + x_{\min}$.

Estimation of performance efficiency of ANN

The performance efficiency of the network was estimated using the measured and ANN-estimated values. The ANN performance measures used in this study are R, mean error (ME), root mean square error (RMSE), and mean square error (MSE). The mathematical expressions of R, ME, RMSE, and MSE can be found in any statistics book. In brief, ANN predictions are optimum if R, ME, RMSE, and MSE are found to be close to 1, 0, 0, and 0, respectively. In the present study, MSE is used only for the network training, whereas R, ME, and RMSE are used to measure the predictive performance of ANN on the testing dataset.

GA–ANN combination

The procedure for searching for an optimal network structure using GA involves the sharing of information from both GA and ANN. In the beginning, GA generates one set of solutions (i.e. spread and maximum number of neurons for RBFN and the first and second layers of neurons and epoch size for BPNN). The solution set is passed on to ANN. Using the solution set of GA, the ANN geometry is created and ANN is trained using the training dataset and defined modeling parameters. The performance efficiency of the trained network is estimated using the testing dataset, and the fitness value (R in this case) is passed on to GA. Based on the R-value of the testing dataset, GA creates another set of solutions for ANN. The solution set of GA and the fitness value, R, estimated by ANN using the testing dataset are exchanged in search of the optimal solution. Because the primary objective of the GA–ANN combination is to minimize the difference between the measured and ANN-predicted values, the ANN predictive performance efficiency in terms of correlation coefficient, R, should be high. So, the objective function of the GA–ANN combination is

$$\max \quad R = \left(\sum_{i=1}^{N} m_i p_i \right) \Bigg/ \left(\sqrt{\sum_{i=1}^{N} m_i^2} \sqrt{\sum_{i=1}^{N} p_i^2} \right) \tag{5}$$

where N is the number of samples, $m_i = M_i - \overline{M}$, $p_i = P_i - \overline{P}$, and M_i and P_i are the measured and predicted values for $i = 1, \ldots, N$ and $\overline{M}$ and $\overline{P}$ are the mean values of the measured and predicted datasets, respectively.

Equation 4 is subjected to the following two constraints:

$$\text{For} \quad \text{RBFN}\begin{cases} \sigma_{\min} \le \sigma_i \le \sigma_{\max} \\ 1 \le N_0 \le N \end{cases} \tag{6}$$

$$\text{For} \quad \text{BPNN}\begin{cases} 1 < h_1 \le h_{1,\max} \\ 1 \le h_2 \le h_{2,\max} \\ 1 \le E_0 \le E_{\max} \end{cases} \tag{7}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the minimum and maximum spreads of an RBFN, respectively; and $h_{1,\max}$, $h_{2,\max}$, and $E_{\max}$ are maximum neurons of the first hidden layer, maximum neurons of the second hidden layer, and maximum epoch number of a BPNN, respectively. Depending on the specific problem, the minimum value of N_0, h_1, h_2, and E_0 can be set at higher values than 1.

Setting solution range and internal parameters for GA

When solving a problem, we usually look for the best among all solutions. The space of all feasible solutions (the set of solutions among which the desired solution resides) is called a search space (also state space). Each point in the search space represents one possible solution, each of which can be marked by its value (or fitness) for the problem. With GA, we look for the best solution among a number of possible solutions. From preliminary examination (see Table 1), it is found that the solution spaces for the spread and the maximum number of neurons for the small dataset are in the range of 10 to 150 and 10 to 90, respectively. However, for the large dataset, the solution spaces for the spread and the maximum number of neurons are in the range of 10 to 150 and 10 to 400, respectively. Thus, GA is used to search for the optimal solution in these ranges for the RBFN and unscaled input dataset. For scaled datasets, the solution space for the spread will change but that for the maximum number of neurons will not. Therefore, for both cases, the spread is searched in the range of 0.1 to 2.0. For BPNN, the solution ranges of first-layer neurons, second-layer neurons, and epoch size are 2 to 20, 2 to 20, and 100 to 800, respectively.

Goldberg and Deb (1991) indicated that of the various selection schemes, the tournament selection imposes a more controlled selection pressure and has a faster convergence characteristic. Therefore, the binary tournament selection method was used in this study. In the binary tournament selection, two chromosomes selected from the population compete for a position in the mating pool based on their fitness values. The chromosome with the higher fitness value enters into the mating pool and the other dies off. Crossover and mutation operators are applied to the selected chromosomes. Petridis et al. (1998) and Damousis et al. (2003) reported that the crossover (*Pc*) operator leads to faster convergence and the mutation operator (*Pm*) helps to maintain population diversity. However, premature convergence and excessive diversity may occur when setting high values of crossover and mutation, respectively. In either case, the search becomes inefficient. Damousis et al. (2003) reported that GA is robust for *Pc* and *Pm* in the ranges 0.4 to 1.0 and 0.0 to 0.01, respectively; while Cieniawski et al. (1995) reported that these ranges are 0.7 to 0.9 and 0.025 to 0.05, respectively. Thus it is evident that *Pc* and *Pm* are problem specific as reported by

Wardlaw and Sharif (1999). In this study, GA was examined for combinations of *Pc* and *Pm* (i.e. 0.5 and 0.01, 0.5 and 0.02, 0.5 and 0.03, 0.9 and 0.01, 0.9 and 0.02, and 0.9 and 0.03) using a scaled large dataset (see section data used) and RBFN. In all cases, the RBFN performance efficiency was found to be identical (i.e. *R* equal to 0.9937). Therefore this study used a *Pc* of 0.5 and a *Pm* of 0.02 for all simulations. For each case, 10 generations were used to create 10 populations each, for a total of 100 different populations (i.e. solutions). The best fitness values were recorded after each generation.

RESULTS AND DISCUSSIONS

The GA-generated 100 sets of first-layer neurons and second-layer neurons and the corresponding *R*-values of the four-layer BPNN for the unscaled large and the small training datasets are plotted in Figure 3. The wire-mesh surface that shows the variation of *R*-values based on the first- and second-layer neurons was created by the linear interpolation of *R*-values. This shows how ANN performance varies for the combinations of first- and second-layer neurons. Although epoch size is an important parameter of BPNN training (see Table 2), for the sake of clarity it is not shown in the Figure 3. Table 2 presents the highest *R*-value among all 100 solutions using unscaled testing datasets. In Figure 3, the valley portion shows that the performance efficiency was low for the combination of first- and second-layer neurons around it, whereas the hilltop shape shows that ANN performance efficiency was highest for the combination of first- and second-layer neurons. It is evident in Figure 3 that the numbers of neurons in first- and second-layer neurons have a

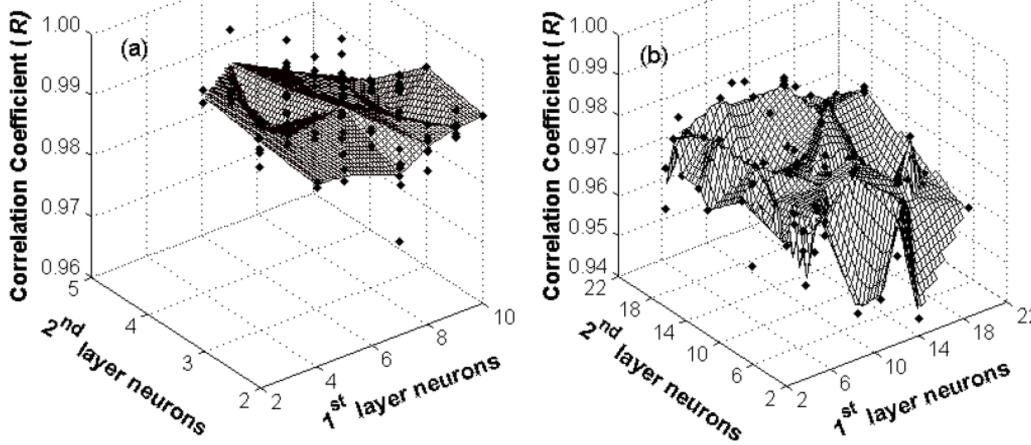

Figure 3. Linearly interpolated topography of *R*-values of 100 populations for the combination of first- and second-layer neurons using unscaled (a) large and (b) small training datasets and a four-layer BPNN. The solid diamond symbols represent actual *R*-values.

Table 2. The best among 100 possible solutions for BPNN hidden layer geometry (neurons at first and second hidden layers) and epoch size in terms of network predictive performance efficiency (*R*-value) using the testing dataset.

Training data type	Hidden layer neurons		Epoch size	Performance efficiency		
	1st layer	2nd layer		R	RMSE	ME
Large unscaled	7	4	354	0.9965	0.0762	-0.0160
Small unscaled	5	5	280	0.9915	0.1001	-0.0042
Large scaled	5	3	445	0.9952	0.0775	0.0146
Small scaled	6	2	587	0.9936	0.0870	-0.0250

significant effect on ANN predictive performance efficiency. Figure 3(b) shows deeper valleys (i.e. lower *R*-values) than Figure 3(a). This means that for all combinations of first- and second-layer neurons and epoch size, the ANN performance trained with the large dataset fluctuated less than that trained with the small dataset. Another important feature of Figure 3 is that BPNN needs more neurons in the first and second hidden layers to fit the data using a small training dataset, on the other hand, BPNN needs fewer neurons if the training dataset is large. This means that one could easily end up with a large BPNN geometry in search of the optimum*R*-value using the trial and error approach for the case of the small training dataset. This illustrates that GA can identify the optimal BPNN geometry for a specific training dataset.

Figure 4 presents the GA-generated 100 sets of maximum number of neurons and spreads, and the corresponding *R*-values of the RBFN, for large and small unscaled training datasets. Table 3 shows the highest *R*-value among all 100 GA-generated solutions using unscaled testing datasets and RBFN. In the case of the large training dataset, the performance efficiency of the RBFN drops off sharply for the maximum number of neurons and spread greater than 350 and 100, respectively (Figure 4a). Similarly, for the case of the small training dataset, the performance efficiency of the RBFN drops off sharply when the maximum number of neurons is greater than or equal to 90 (Figure 4b). As for the spread, it is evident in Tables 1 and 3 that the spread for the small training dataset is less than 90.

It can be inferred from the above results and analysis that the solution domain for the large

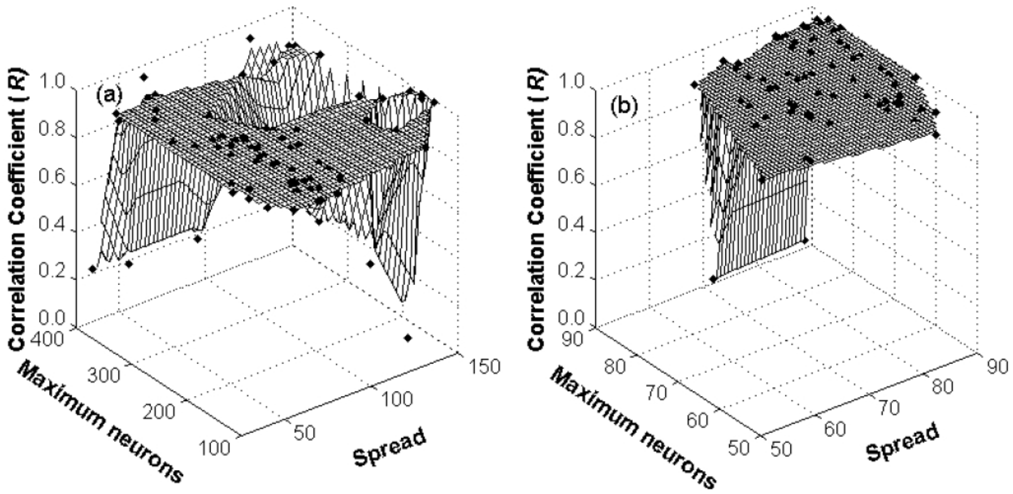

Figure 4. Linearly interpolated topography of*R*-values of 100 populations for the combination of maximum number of neurons and spread using unscaled (a) large and (b) small training datasets and an RBFN. The solid diamond symbols represent actual*R*-values.

Table 3. The best among 100 possible solutions for RBFN hidden layer geometry (number of neurons in hidden layer) and spread in terms of network predictive performance efficiency (*R*-value) using the testing dataset.

Training data type	Spread	Number of neurons in hidden layer	Performance efficiency		
			R	RMSE	ME
Large unscaled	74.6032	211	0.9916	0.0828	-0.0023
Small unscaled	88.7097	83	0.9935	0.0835	-0.0361
Large scaled	0.5000	122	0.9937	0.0706	0.0171
Small scaled	0.5000	71	0.9924	0.0900	-0.0224

training dataset is greater than that for the small training dataset. This means it is harder for the modeler to derive the optimum or close to optimum ANN parameters by a trial-and-error approach using a small training dataset.

In both of the cases above, the training datasets were not scaled to the range of 0 to 1, so the spread could vary from 50 to 200 (Tables 1 and 3). In general, the optimal spread varies according to the training dataset. Thus, it is often difficult to present the solution range to GA, if there is no preliminary knowledge regarding how the RBFN performance efficiency responds to the training dataset. However, scaling the training dataset to the range 0 to 1; the solution range for spread lies in the range of 0.1 to 1.0.

Table 2 presents the highest R-value among 100 GA-generated solutions of a four-layer BPNN for the combination of first- and second-layer neurons and epoch size using scaled (i.e. 0 to 1) large and small testing datasets. Examining Table 2, it is evident that the BPNN, using a scaled small training dataset, improves the predictive performance efficiency relative to using an unscaled small training dataset (i.e. for unscaled training dataset, $R = 0.9915$ and RMSE = 0.1001; for scaled training dataset, $R = 0.9935$ and RMSE = 0.0835). However, for the case of a large training dataset the performance efficiencies of the BPNN using unscaled and scaled datasets are close to each other (i.e. for unscaled training dataset, $R = 0.9965$ and RMSE = 0.0762; for scaled training dataset, $R = 0.9952$ and RMSE = 0.0775).

Figure 5 presents the GA-generated 100 solution sets for BPNN's hidden layers geometry and epoch size; and R-values of corresponding solution, using scaled testing datasets. It is interesting to see that the BPNN trained using scaled datasets needs fewer neurons in the second hidden layer than the BPNN trained using unscaled datasets (see Table 2 and Figures 3 and 5). Moreover, A BPNN trained using a scaled small dataset produces fewer and shallower valleys than a BPNN trained using an unscaled small dataset (see Figure 3(b) and 5(b)). This means that for all combinations of first and second-layer neurons and epoch size, the performance efficiency of the BPNN trained using scaled datasets fluctuates less than that of the BPNN trained using unscaled

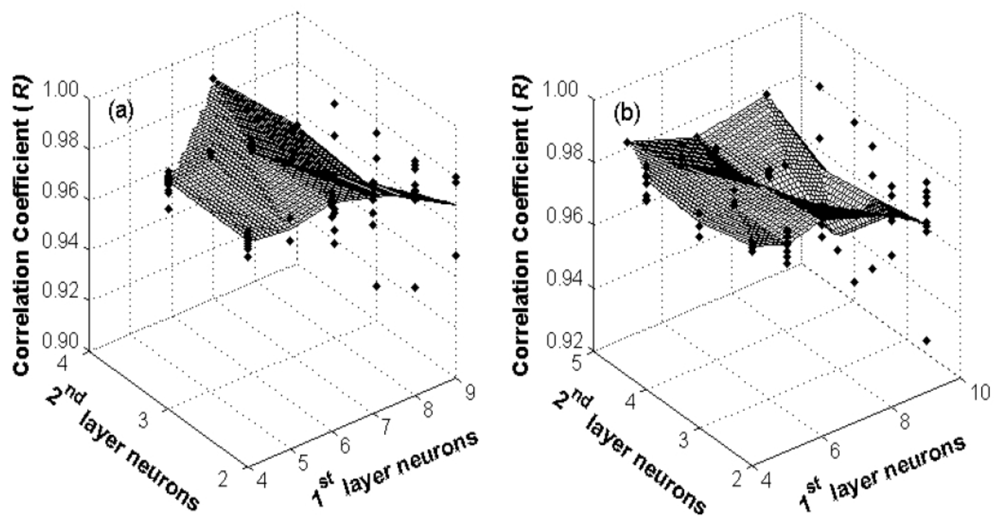

Figure 5. Linearly interpolated topography of R-values of 100 populations for the combination of first- and second-layer neurons using scaled (a) large and (b) small training datasets and a four-layer BPNN. The solid diamond symbols represent actual R-values.

datasets.

Figure 6 presents the GA-generated 100 solution sets for RBFN geometry and spread and corresponding R-values of each solution using scaled testing datasets. The best R-values among 100 solutions are presented in Table 3. Similar to BPNN, the maximum number of neurons required for a RBFN trained using scaled datasets to produce the best R-value among 100 solutions are found to be lower than that of a RBFN trained using unscaled datasets (see Figures 4 and 6, and Table 3). Moreover, the spread corresponding to best the R-value among 100 solutions using scaled datasets is found to be 0.5. This means it is certain that the solution domain of the spread using scaled datasets is in the range of 0.1 to 1. Because the maximum number of neurons cannot exceed the number of samples in the dataset, a practical range can be easily set. However, it is hard to put a correct solution range for spread into GA for an unscaled training dataset if the modeler has no preliminary knowledge.

Table 4 presents the predictive performance efficiency of multiple regression analysis (MRA) for large and small training datasets. Shown are the regression coefficients (i.e. a_0, a_1, a_2, a_3, a_4, and a_5) estimated using least square method. MRA with large training dataset outperform MRA with small training dataset. However, for both training datasets, ionic strength has significant effect (i.e. values of a_2 are 0.1663 and 0.1949 for small and large dataset, respectively) on the permeate fluxes estimation compared to other variables. This is consistent with the findings of Faibish et al. (1998). The weight matrices of input layer to first hidden layer of BPNN for large training dataset (Table 5) shows identical results i.e. ionic strength has greater effect than others on permeate fluxes. MRA predictive performance efficiencies for both large and small training dataset were found to be much less than those of GA optimized BPNN and RBFN.

The BPNN- and RBFN-predicted results with the highest R-values (see Tables 2 and 3) among the 100 solutions using the scaled large and small training datasets are presented in Figure 7 and Figure 8, respectively. Predicted permeate flux decline values are scattered around the 1:1 line. It is hard to say which ANN model (RBFN or BPNN) is better, because both can predict the permeate flux decline of CMF with R above 0.99, given that the ANNs' geometry and modeling

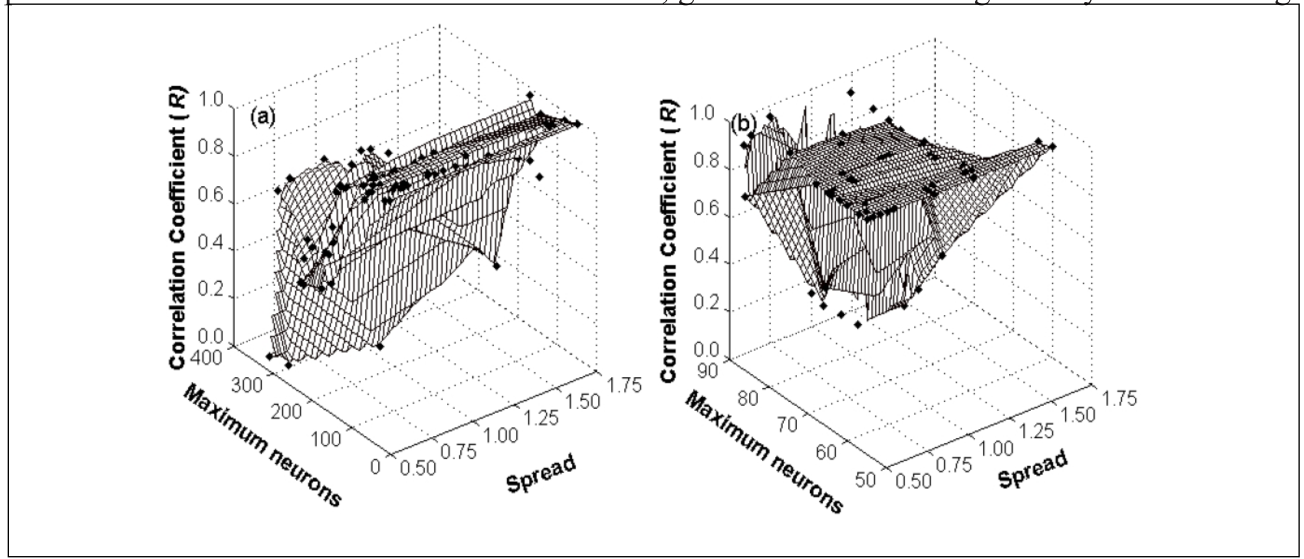

Figure 6. Linearly interpolated topography of R-values of 100 populations for the combination of maximum number of neurons and spread using scaled (a) large and (b) small training datasets and an RBFN. The solid diamond symbols represent actual R-values.

Table 4. Performance efficiency of multiple regression analysis and coefficient of the equation $F = a_0 + a_1P + a_2IS + a_3pH + a_4PS + a_5T$.

		Small training dataset	Large training dataset
Performance efficiency	R	0.8692	0.8933
	ME	-0.1078	0.0208
	RMSE	0.3321	0.2823
Coefficients of the multiple regressions analysis equation	a_0	0.9426	0.7123
	a_1	0.0250	0.0235
	a_2	0.1663	0.1949
	a_3	-0.0192	-0.0108
	a_4	0.0028	0.0027
	a_5	-0.0133	-0.0122

Table 5. Weight matrix of the BPNN input layer for large dataset.

Neuron at 1st hidden layer	Weights from input variables of input layer to 1st hidden layer				
	P	IS	pH	PS	T
1	0.178	0.402	-0.214	0.088	0.327
2	-0.033	-0.683	0.274	-0.002	-0.026
3	-0.114	1.462	0.321	0.403	0.008
4	1.903	0.404	-0.420	-0.381	0.012
5	0.033	0.662	-0.320	0.002	0.033
6	0.014	-0.742	0.043	-0.004	-0.061
7	0.345	1.853	0.016	0.016	-0.035
Total weight	2.325	3.359	-0.299	0.122	0.257

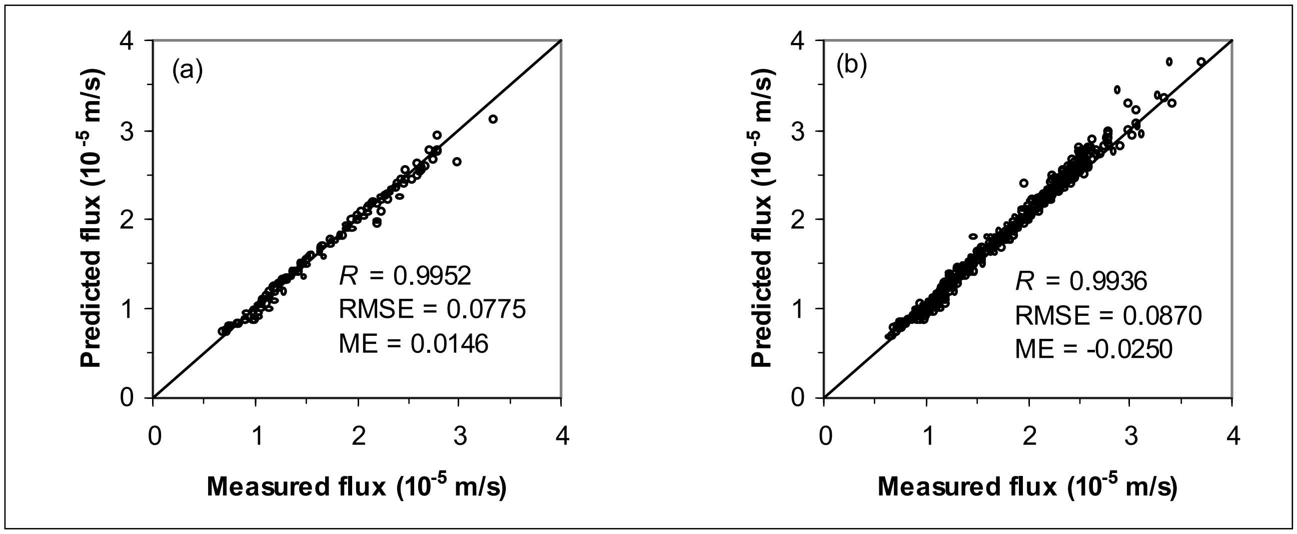

Figure 7. Comparison of predictive performance efficiency of a four-layer BPNN using the scaled (a) large and (b) small training datasets.

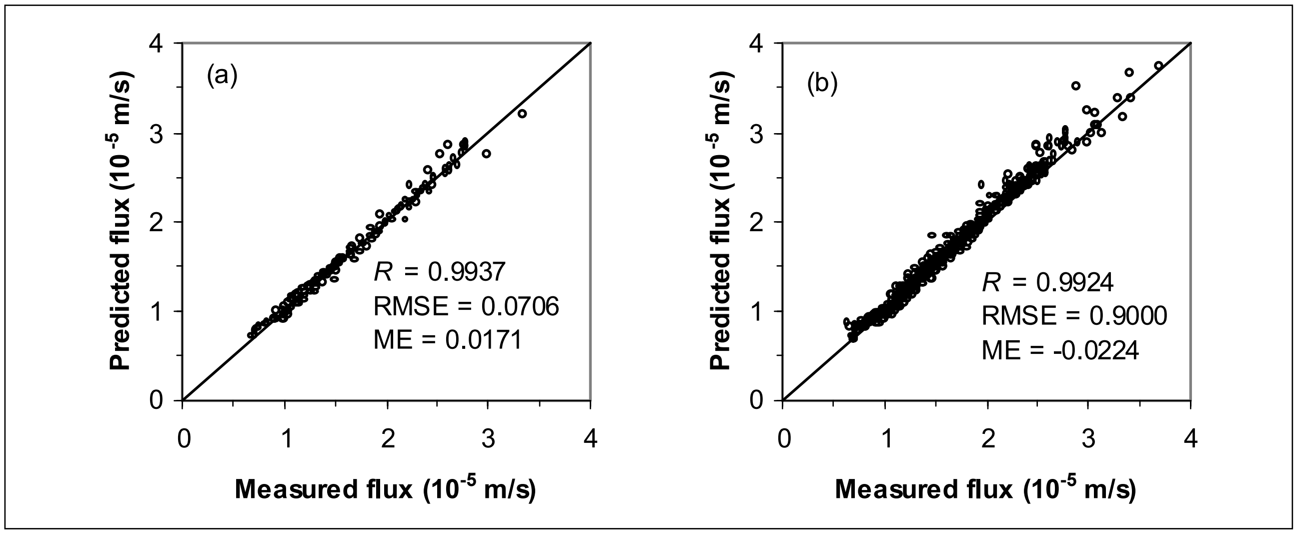

Figure 8. Comparison of predictive performance efficiency of an RBFN using the scaled (a) large and (b) small training datasets.

parameters are optimized and ANN is optimally trained.

SUMMARY AND CONCLUSIONS

BPNN and RBFN are increasingly being used as an alternative to descriptive models to predict water resources and environmental variables. However, one of the most difficult tasks in their respective design is choosing proper ANN's modeling parameters and appropriate network geometry. Although some guidance is available in the literature, the choice of values and network geometry is generally determined by trial and error. This paper demonstrates the effect of network geometry and modeling parameters on the predictive performance efficiency of ANN models, and presents a method to determine the optimal design network using GAs. The predictive performance efficiency of the GA–ANN combination is estimated using an already published CMF experimental dataset. Comparing predictive performance efficiency, it was found that GA-optimized ANNs significantly outperform trial and error-calibrated ANNs. It was also found that scaling the training dataset to the range of 0 to 1 helps determine the solution domain of spread for an RBFN (i.e. the solution domain of an RBFN lies in the range of 0.1 to 1.0). Moreover, the performance efficiency of a BPNN trained using scaled datasets fluctuates less than that of a BPNN trained using unscaled datasets (see Figures 3 and 5).

The ASCE Task Committee (2000) and Bowden et al. (2002) reported that the size and number of samples included in the training dataset have a significant effect on the predictive performance efficiency of ANN models. It is illustrated that even a small training dataset can predict with *R* greater than 0.99, given that the training and testing datasets are representatives of same population and the network is optimized. However, it is demonstrated that for the case of the small dataset, the optimal solution for ANN geometry and modeling parameters lies within a narrow range (see Figure 3). Thus, performance efficiency using a small dataset deviates largely if the model is not optimized and well trained. GA is found to be a good alternative to the trial-and-error approach to determine the optimal ANN geometry and modeling parameters quickly and efficiently.

It is demonstrated here that ANN-GA model can predict the permeate flux of a CMF with *R* greater than 0.99. Thus, it may not be necessary to carry out an entire series of expensive pilot or full scale tests to collect data for verification of a CMF. Although, the research findings presented herein are based on an experimental dataset, the procedure can be served as the basis for application to other datasets.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Mr. Philip Moravcik, Technology Transfer Specialist at the Water Resources Research Center, University of Hawaii at Manoa, Honolulu, Hawaii, USA; Dr. Ramesh Gautam, Hydraulic Modeler at Earth Tech Inc. Long Beach, California, USA; Dr. Sivakumar Bellie, Department of Land, Air and Water, University of California Davis, One Shields Avenue, Davis, California, USA; who critically reviewed the manuscripts and provided valuable suggestions and remarks for the improvement of quality of the manuscript. Partial funding for this research was provided by the US Department of Agriculture, National Research Institute/ Competitive Grants Program, Grant Number 99-35102-8551.

REFERENCES

Akin, S. 2005. Tracer model identification using artificial neural networks. Water Resour. Res., 41: W10421, doi:10.1029/2004WR003838.

Alp, M., and H.K. Cigizoglu. 2007. Suspended sediment load simulation by two artificial neural network methods using hydrometeorological data. Environ. Modell. Softw., Vol. 22(1), pp. 2–13.

ASCE Task Committee. 2000. Artificial neural network in hydrology. J. Hydrol. Eng., Vol. 5, pp. 124–144.

Aydiner, C., I. Demir, and E. Yildiz. 2005. Modeling flux decline in crossflow microfiltration using neural networks: The case of phosphate removal. J. Membr. Sci., Vol. 248, pp. 53–62.

Belfort, G., R.H. Davis, and A.L. Zydney. 1994. The behavior of suspensions and macromolecular solutions in crossflow microfiltration. J. Membr. Sci., Vol. 96, pp. 1–58.

Birikundavyi, S., R. Labib, H.T. Trung, and J. Rousselle. 2002. Performance of neural networks in daily streamflow forecasting. J. Hydrol. Eng., Vol. 7, pp. 392–398.

Bowden, G.J., H.R. Maier, and G.C. Dandy. 2002. Optimal division of data for neural networks models in water resources applications. Water Resour. Res., Vol. 38, doi:10.1029/ 2001WR000266.

Bowen, W.R., and F. Jenner. 1995. Theoretical descriptions of membrane filtration of colloids and fine particles: An assessment and review. Adv. Colloid. Interfac. Sci., Vol. 56, pp. 141–200.

Chakraborty, M., C. Bhattacharya, and S. Dutta. 2003. Studies on the applicability of artificial neural network (ANN) in emulsion liquid membranes. J. Membr. Sci., Vol. 220, pp. 155–164.

Chang, F., and Y. Chen. 2003. Estuary water-stage forecasting by using radial basis function neural network. J. Hydrol., Vol. 270, pp. 158–166.

Chellam, S. 2005. Artificial neural network model for transient crossflow microfiltration of polydispersed suspensions. J. Membr. Sci., Vol. 258, pp. 35–42.

Cieniawski, S.E., J.W. Eheart, and S. Ranjithan. 1995. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. Water Resour. Res., Vol. 31(2), pp. 399–409.

Damousis, I.G., A.G. Bakirtzis, and P.S. Dokopoulos. 2003. Network-constrained economic dispatch using real-coded genetic algorithms. IEEE T. Power Syst., Vol. 18(1), pp. 198–205.

Dormier, M., M. Decloux, G. Trystram, and A. Lebert. 1995. Dynamic modeling of crossflow microfiltration using neural networks. J. Membr. Sci., Vol. 98, pp. 263–273.

Faibish, R.S., M. Elimelech, and Y. Cohen. 1998. Effect of interparticle electrostatic double layer interactions on permeate flux decline in crossflow membrane filtration of colloidal suspensions: An experimental investigation. J. Colloid. Interfac. Sci., Vol. 204, pp. 77–86.

Fane, A.G., and C.J.D. Fell. 1987. A review of fouling and fouling control in ultrafiltration. Desalination, Vol. (62), pp. 117–136.

Flood, I., and N. Kartam. 1994. Neural networks in civil engineering I: Principles and understanding. J. Comput. Civil Eng., Vol. 8(2), pp. 131–148.

Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison–Wesley–Longman, Reading, Mass, USA.

Goldberg, D.E., and K. Deb. 1991. A comparative analysis of selection schemes used in genetic algorithms, in Foundations of Genetic Algorithms. In Rawlins JE (eds) Morgan-Kaufmann, Burlington, Mass., pp. 69–93.

Govindaraju, R.S., and A.R. Rao. 2000. Artificial Neural Networks in Hydrology. Kluwer Academic Publishers, Dordecht, The Netherlands.

Haddadnia, J., K. Faez, and M. Ahmadi. 2003. A fuzzy hybrid learning algorithm for radial basis function neural network with application in human face recognition. Pattern Recogn., Vol. 36, pp. 1187–1202.

Hagan, M.T., H.P. Demuth, and M. Beale. 1996. Neural Network Design. PWS Publishing, Boston, Mass., USA.

Haykin, S. 1999. Neural Networks: A Comprehensive Foundation. Macmillan, New York, N.Y., USA.

Kingston, G.B., M.F. Lambert, and H.R. Maier. 2005 Bayesian training of artificial neural networks used for water resources modeling. Water Resour. Res., Vol. 41, W12409, doi:10.1029/2005WR004152.

Maier, H.R., and G.C. Dandy. 1998. The effect of internal parameters and geometry on the performance of back-propagation neural networks: an empirical study. Environ. Modell. Softw., Vol. 13(2), pp. 193–209.

Maier, H.R., and G.C. Dandy. 2000. Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications. Environ. Modell. Softw., Vol. 15, pp. 101–124.

Marquardt, D. 1963. An algorithm for least squares estimation of non-linear parameters. J. Soc. Ind. Appl. Math., pp. 431–441.

MathWorks Inc. 2002. MATLAB. 3 Apple Hill Drive, Natick, Mass., USA.

Petridis, V., S. Kazarlis, and A. Bakirtzis. 1998. Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problems. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 28(5), pp. 629–640.

Principe, J.C., N.R. Euliano, and W.C. Lefebvre. 1999. Neural and Adaptive Systems: Fundamentals Through Simulations. John Wiley & Sons, New York.

Ray, C., and K.K. Klindworth. 2000. Neural networks for agrichemical vulnerability assessment of rural private wells. J. Hydrol. Eng., Vol. 5(2), pp. 162–171.

Razavi, M.A., A. Mortazavi, and M. Mousavi. 2003. Dynamic modeling of milk ultrafiltration by artificial neural network. J. Membr. Sci., Vol. 220, pp. 47–58.

Sahoo, G.B., and C. Ray. 2006. Flow forecasting for a Hawaiian stream using rating curves and neural networks. J. Hydrol., Vol. 317, pp. 63–80.

Sahoo, G.B., C. Ray, and H.F. Wade. 2005. Pesticide prediction in ground water in North Carolina domestic wells using artificial neural network. J. Ecol. Modell., Vol. 183, pp. 29–46.

Shetty, G.R., H. Malki, and S. Chellam. 2003. Predicting contaminant removal during municipal drinking water nanofiltration using artificial neural networks. J. Membr. Sci., Vol. 212, pp. 99–112.

Song, L. 1998. Flux decline in crossflow microfiltration and ultrafiltration: Mechanisms and modeling of membrane fouling. J. Membr. Sci., Vol. 139, pp. 183–200.

Wardlaw, R., and M. Sharif. 1999. Evaluation of genetic algorithms for optimal reservoir system operation. J. Water Res. Pl., Vol. 125, pp. 25–33.

Zhao, Y., J.S. Taylor, and S. Chellam. 2005. Predicting RO/NF water quality by modified solution diffusion model and artificial neural networks. J. Membr. Sci., Vol. 263, pp. 38–46.

APPENDIX

Gradient descent algorithms

The error function $F(w)$ with respect to w_{xy} is defined by (Hagan et al. 1996; Principe et al. 1999):

$$F(w) = \sum_{y=1}^{N} (M_y - P_y)^2 = \sum_{y=1}^{N} e_y^2(w) = e^T(w)\, e(w) \tag{A-1}$$

where w_{xy} is the connection weight matrix for neuron x to yth neuron and $e(w)$ is the error between measured (M_y) and predicted (P_y) values at the current step (t) weight matrix. The objective of back propagation is to minimize $F(w)$. To achieve this, w_{xy} is computed repeatedly using the gradient-descent method as follows:

$$w_{xy}(t+1) = w_{xy}(t) - \gamma \nabla F(w) \tag{A-2}$$

where $\nabla F(w) = \partial F(w)/\partial w_{xy}$ and γ is the learning rate which scales the gradient. If γ is too small, many steps are needed to reach an acceptable solution; on the other hand, a large γ will possibly lead to oscillation (Hagan et al. 1996). $\partial F(w)/\partial w_{xy}$ is computed using the chain rule:

$$\frac{\partial F(w)}{\partial w_{xy}} = \frac{\partial F(w)}{\partial M_y} \frac{\partial M_y}{\partial net_y} \frac{\partial net_y}{\partial w_{xy}} \tag{A-3}$$

where net_y is the weighted sum of the inputs of neuron y. To eliminate the oscillations due to the learning rate, a momentum is generally introduced:

$$w_{xy}(t+1) = w_{xy}(t) - \gamma \nabla F(w) + \xi \Delta w(t-1) \tag{A-4}$$

where $\Delta w\,(t - 1) = w_{xy}\,(t) - w_{xy}\,(t - 1)$ and ξ is the momentum term. Here, ξ scales the influence of the previous step (t - 1) on the current step (t). The momentum term has the benefit of preventing the learning process from terminating in a shallow local minimum on the error surface (Haykin 1999). Maier and Dandy (1998, 2000) and Alp and Cigizoglu (2007) reported that the optimal value of y and ξ are problem-dependent and are determined by trial and error.

Levenberg–Marquardt algorithm

The Levenberg–Marquardt algorithm, an approximation to Newton's method (Marquardt 1963), is

$$\Delta w = -\left[\nabla^2 F(w)\right]^{-1} \nabla F(w) \tag{A-5}$$

where $\nabla^2 F(w)$ is the Hessian matrix and $\nabla F(w)$ is the gradient. $\nabla^2 F(w)$ and $\nabla F(w)$ can be shown that

$$\nabla F(w) = J^T(w) e(w) \tag{A-6}$$

$$\nabla^2 F(w) = J^T(w)J(w) + S(w) \tag{A-7}$$

where $J(w)$ is a Jacobian matrix and

$$S(w) = \sum_{i=1}^{N} e_i \nabla^2 e_i(w) \tag{A-8}$$

For the Gauss–Newton method it is assumed that $S(w)$ » 0, and Equation (A-5) becomes

$$\Delta w = -\left[J^T(w)J(w)\right]^{-1} J^T(w)e(w) \tag{A-9}$$

The Levenberg–Marquardt modification to the Gauss–Newton method is

$$\Delta w = -\left[J^T(w)J(w) + \mu I\right]^{-1} J^T(w)e(w) \tag{A-10}$$

where I is the unit matrix and μ is a scalar value. Equation (A-10) can be written as (Hagan et al. 1996; Haykin 1999; Principe et al. 1999)

$$w(t+1) = w(t) - \left[J^T\{w(t)\}J\{w(t)\} + \mu I\right]^{-1} J^T\{w(t)\}e\{w(t)\} \tag{A-11}$$

where $w(t)$ is the weight matrix of current step (i.e. iteration) t.

When the scalar μ is zero, equation (A-11) is just the Gauss–Newton's method; on the other hand when μ is large, equation (A-11) becomes the gradient descent with step size $1/\mu$. Gauss–Newton's method is faster and more accurate than gradient-descent near an error minimum, so the aim is to shift toward Gauss–Newton's method as quickly as possible. The steepest-descent method, on the other hand, has a slow asymptotic convergence rate.

Activation functions

The hyperbolic tangent sigmoid, 'tansig' (between -1 to 1); logarithmic sigmoid, 'logsig' (between 0 to 1); and linear, 'purelin' (between -∞ to ∞), activation functions are defined, respectively, as follows:

$$\varphi(z) = \frac{2}{1+e^{-2z}} - 1 \tag{A-12}$$

$$\varphi(z) = \frac{1}{1+e^{-z}} \tag{A-13}$$

$$\varphi(z) = z \tag{A-14}$$

where z is the argument.

ADDRESS FOR CORRESPONDENCE
Goloka Behari Sahoo
Department of Civil and Environmental Engineering
University of California at Davis
One Shield Avenue
Davis, CA, 95616
USA

Email: gbsahoo@ucdavis.edu