

Joint Application of Artificial Neural Networks and Evolutionary Algorithms to Watershed Management

MISGANA K. MULETA and JOHN W. NICKLOW

Abstract. Artificial neural networks (ANNs) have become common data driven tools for modeling complex, nonlinear problems in science and engineering. Many previous applications have relied on gradient-based search techniques, such as the back propagation (BP) algorithm, for ANN training. Such techniques, however, are highly susceptible to premature convergence to local optima and require a trial-and-error process for effective design of ANN architecture and connection weights. This paper investigates the use of evolutionary programming (EP), a robust search technique, and a hybrid EP–BP training algorithm for improved ANN design. Application results indicate that the EP–BP algorithm may limit the drawbacks of using local search algorithms alone and that the hybrid performs better than EP from the perspective of both training accuracy and efficiency. In addition, the resulting ANN is used to replace the hydrologic simulation component of a previously developed multiobjective decision support model for watershed management. Due to the efficiency of the trained ANN with respect to the traditional simulation model, the replacement reduced the overall computational time required to generate preferred watershed management policies by 75%. The reduction is likely to improve the practical utility of the management model from a typical user perspective. Moreover, the results reveal the potential role of properly trained ANNs in addressing computational demands of various problems without sacrificing the accuracy of solutions.

Key words: evolutionary computation, multi-objective analysis, neural networks, watershed management

1. Introduction

Following the demonstration of a mathematically rigorous theoretical framework known as the back propagation (BP) algorithm to machine learning (Rumelhart *et al.*, 1986), artificial neural networks (ANNs) have become increasingly popular for modeling complex, nonlinear problems in science and engineering. Within water resources related disciplines, ANNs have recently been used as data driven models in rainfall–runoff prediction (Gupta *et al.*, 1997; Tokar and Johnson, 1999),

stream flow forecasting (Muttiah *et al.*, 1997), ground water simulation (Ranjithan *et al.*, 1993; Yang *et al.*, 1997), water quality modeling (Maier and Dandy, 1996; Rogers and Dowla, 1994), water demand forecasting (Jain *et al.*, 2001), reservoir operations (Cancelliere *et al.*, 2002) and other applications (ASCE, 2000b). The majority of these applications have relied on local search techniques, namely the BP algorithm, for ANN training (ASCE, 2000a). Like other gradient-based algorithms, however, BP often suffers from premature convergence to local optima. Particularly for complex problems, the success of training with BP depends on whether the modeler has sufficient knowledge of ANNs and of the problem at hand so as to design a compact and effective ANN. Yet such knowledge is often limited or unavailable for non-ANN experts facing realistic problems (Yao and Liu, 1998). As a consequence, a trial-and-error procedure is often applied to determine the best performing, and yet simple and compact, ANN architecture (ASCE, 2000a).

This study investigates the use of evolutionary programming (EP), a technique that belongs to a class of increasingly popular and robust search algorithms known as evolutionary algorithms (EAs), to reduce the shortcomings of gradient-based ANN training algorithms. However, similar to any other EA, if used independently, EP could be inefficient and ineffective in fine-tuning local searches for large problems. This lack of efficiency could be significantly improved by incorporating a local search procedure, such as BP, into the solution evolution. Specifically, this methodology would involve hybridizing EP's robust search ability with the fine-tuning ability of a gradient-based algorithm. Thus, EP could be used to determine the best region of the solution space, represented by ANN architecture and an initial weight surface, whereas BP could be used to determine optimal or near-optimal synaptic weights within this region.

The objective of this study is to compare the singular performance of EP with that of a hybrid EP-BP algorithm for training ANNs. BP is not included in the investigation since, if used independently, the algorithm ultimately involves highly subjective decisions and a trial-and-error approach for ANN construction, which are some of the same characteristics this study aims to reduce. Based on the comparison, the ANN associated with the superior training algorithm is subsequently used as a replacement for the simulation component of a previously developed multiobjective decision support model for watershed management. This unique application satisfies a secondary objective of demonstrating the efficiency of an ANN in comparison to traditional simulation tools when used within a decision support model.

2. Background Information

2.1. MOTIVATION BASED ON RELATED RESEARCH

Human interferences with the natural environment in the form of mechanized agriculture, large-scale construction, deforestation, and overgrazing have tremendously increased rates of erosion and sedimentation from watersheds. Large percentages of sediment from such mistreated watersheds ultimately enter water bodies, causing

environmental, economical, and social impacts. To assist in the assessment of the environmental impacts of erosion in large river basins, the U.S. Department of Agriculture (USDA) has developed a distributed hydrologic model known as the Soil and Water Assessment Tool (SWAT) (Arnold *et al.*, 1998; ASCE, 1999). The model operates on a daily time scale and on the spatial scale of a hydrologic response unit (HRU). Watersheds are typically subdivided into natural subbasins, which can be further divided to form HRUs depending on land-use and soil heterogeneity within the basin. SWAT can be used to simulate hydrologic processes such as surface runoff, percolation, lateral subsurface flow, ground water flow, potential evapotranspiration, snow melt, transmission losses, and sediment yield using simple and yet realistic techniques. Furthermore, it is capable of modeling crop growth and subsequent crop yield while accounting for stresses on plants due to water shortages and inadequate fertilizer. In addition to these capabilities, SWAT is interfaced with an ArcView® Geographic Information System (GIS), thus simplifying processes of data extraction from a digital elevation model (DEM), digital land-use maps and digital soil maps. As a result, simulation models such as SWAT are generally sufficient for estimating the impacts of erosion in response to a particular land-use policy or activity. Like most hydrologic models, however, SWAT does not allow for an assessment of the economical or social impacts associated with such a policy. Moreover, by themselves, such models are incapable of directly identifying the best policy among various alternatives to reduce this anthropogenic threat and its adverse impacts.

A better approach to address problems associated with erosion is the integrative and systematic planning and management of watershed activities. Integral to the success of this approach is a comprehensive simulation technique to solve the cause–effect relationships that influence erosion and sedimentation, thereby revealing the implication of management decisions on water quality and on aquatic ecology; a socioeconomic model to evaluate economic and social consequences of decisions on land owners in the watershed; and a systems approach that searches for the best decision among the many possible alternatives. Following this integrative philosophy, the authors have developed a watershed decision support model designed to aid in reducing the impacts of erosion while considering social and economic dynamics of the watershed (Muleta and Nicklow, 2001, 2002; Nicklow and Muleta, 2001; Muleta, 2003). Their model was based on coupling SWAT with an EA-based, multiobjective search method known as the Strength Pareto Evolutionary Algorithm (SPEA). The SWAT–SPEA model was designed to search for optimal or near-optimal watershed landscapes, defined as the combination of land uses and farm management practices (i.e., decision variable) on the spatial scale of a farm field that simultaneously minimize sediment yield and maximize net agricultural profit over a specified time horizon. The authors demonstrated the capabilities of the model using Big Creek watershed, a 130-km² drainage basin located in southern Illinois. Though their model was capable of solving this multiobjective problem, it was computationally intensive, requiring over 2.5 days of CPU time to

identify preferred landscapes. This computational demand was primarily the result of required repeated application of SWAT for evaluating sediment and crop yield for each of the numerous landscapes identified in the search process. Motivated by the potential impact of large computational times, namely the reduced practical utility of this multiobjective watershed decision support tool, the authors are now focused on the integration of an ANN within the model. Specifically, they aim to investigate the use of an efficient ANN training algorithm and explore the suitability of the resulting data driven model as a replacement for SWAT in efforts to reduce overall computational time.

2.2. THEORETICAL FRAMEWORK OF ANNS

The multilayer feed forward network (FFNN), depicted in Figure 1, is a particular type of ANN that is used in this study. For brief illustration of a FFNN, consider node i on Figure 1, information from all four inputs (i.e., x_k , for $k = 1, \dots, 4$) is passed to this node through the connection links. The strength of this information transfer is measured by connection weights (i.e., w_{ki} , for $k = 1, \dots, 4$), and the output signal from the node, y_i , is obtained by evaluating the value of an activation function, f , given as

$$y_i = f \left(\sum_{k=1}^4 (x_k w_{ki}) - b_i \right) \quad (1)$$

where b_i is a nodal bias, or threshold value, which must be exceeded for the node to be activated. A commonly used activation function throughout neural network literature is the sigmoid function, which is a bounded, monotonic, strictly increasing and differentiable function expressed as

$$f(l) = \frac{1}{1 + e^{-l}} \quad (2)$$

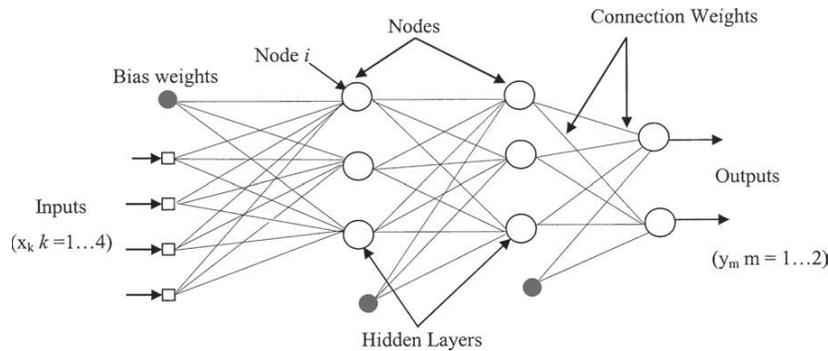


Figure 1. Feed forward multi-layer perceptron with two hidden layers.

where l represents the summation result from Equation (1). Sequentially, similar operations are applied to all nodes of the current and future layers until estimates are obtained from the output layer.

Analogous to the human brain, ANNs learn the system they model from examples presented to them. For a predefined FFNN architecture (i.e., number of hidden layers and number of nodes on each hidden layer), a training, or calibration, process is applied to determine weight matrices, W , and bias vectors, B , that minimize the difference between the predicted output vector, $Y = (y_1, \dots, y_m)$, and desired output vector, $D = (d_1, \dots, d_m)$. The suitability of parameters is evaluated using a network error function, such as the sum of squared errors between the predicted and desired output vector, or

$$V = \frac{1}{N \times m} \sum_{j=1}^N \sum_{l=1}^m (y_{jl} - d_{jl})^2 \quad (3)$$

where N is the number of training examples (i.e., input–output sets) provided for the learning scheme and m is the number of output nodes. Through training, it is hoped that the network learns, or generalizes, the nonlinear relationships that map inputs to outputs so that it can make reasonable estimates for an environment to which it was not exposed during the training process.

2.3. TRAINING WITH BACK PROPAGATION

BP is a gradient descent technique that can be used in efforts to minimize the network error function. When applied to ANN training, BP consists of two passes through the different layers of the network, a forward pass and a backward pass. In the forward pass, an input vector is applied to the sensory nodes of the network, and its effect propagates through the network according to the techniques described earlier. Finally, a set of outputs is produced and the value of the network error function is determined. During this stage of training, the synaptic weights are all fixed. In the backward pass, the error is propagated backward through the network against the direction of synaptic connections in such a way that it moves the actual response of the network closer to the desired response. During this backward pass, the synaptic weights are adjusted according to:

$$w_{ij}^m = w_{ij}^{m-1} + \Delta w_{ij}^m + \alpha \times \Delta w_{ij}^{m-1} \quad (4)$$

where

$$\Delta w_{ij} = \mu \times \frac{\partial V}{\partial w_{ij}} \quad (5)$$

and w_{ij}^m and w_{ij}^{m-1} are the weights between node i and j during the m th and $m-1$ th passes, respectively; V is the error function given in (3) and is implicitly dependent

on the weights through (1); and μ and α , both of which could take values ranging from 0 to 1 only, are the learning rate and momentum factor, respectively. The learning rate helps accelerate training in very flat regions of the error surface and helps prevent oscillations in connection weights. The momentum factor is used to reduce the probability of convergence to local minima. It should be noted, however, that there is still a high possibility for premature convergence to a non-global optima, in spite of the momentum factor. This statement is based on the fact that, as with any gradient-based technique, the quality of solutions obtained heavily depends on initially drawn solutions. Consequently, using BP for training involves a tedious and time-consuming trial-and-error process in which the effects of various initial weights and ANN architectures must be investigated. For further details regarding the BP algorithm, the reader is referred to Haykin (1999).

2.4. TRAINING WITH EVOLUTIONARY PROGRAMMING

A possible way to limit the drawbacks of gradient-based training techniques and improve the search for optimal or near-optimal solutions involves the adoption of a robust search algorithm. EAs refer to a class of population-based, stochastic search algorithms that are developed from the principles of natural selection. They include genetic algorithms (GAs), EP and evolutionary strategies (ES), genetic programming, and artificial life algorithms. EAs can generally handle large, complex, non-differentiable and multimodal spaces without requiring gradient information, making them a suitable candidate for evolving ANN connection weights and architecture.

Although many of the features of various EAs are similar, EP and ES are distinctly different from GAs in that they rely on mutation as a primary search operator. In contrast, GAs use crossover, or mating of alternative solutions (i.e., chromosomes), as a basic operator. The use of mutation-based EAs has recently been found to be superior to the use of crossover-based EAs for training ANNs (Yao, 1999). The reasoning associated with this finding involves the conception that crossover works best when *building blocks* (i.e., well-defined, low-order, and highly fit schema) exist. However, it is unclear what a building block might be in an ANN because they emphasize the distribution of knowledge among all of the weights. Recombining one part of an ANN with that of another ANN is therefore likely to destroy both parts (Yao, 1999). The use of mutation-based EAs, however, can reduce the disruptive features of the crossover operator's recombination process. As a result, EP is becoming more popular for training ANNs (Yao and Liu, 1997).

In EP, it is assumed that whatever genetic information transformations occur, the resulting change in each behavioral trait will follow a Gaussian distribution with a zero mean and standard deviation of unity (Fogel, 1994). When applied to real-valued function optimization, the EP methodology is implemented as follows:

1. An initial population of N individuals is generated at random from uniformly distributed numbers. Each individual is a pair of real-valued vectors, (p_{ij}, s_{ij}) ,

for $i = 1, \dots, n$ and $j = 1, \dots, N$, for all n parameters of N alternatives. Here, p_{ij} represents decision variable vectors to be optimized and s_{ij} represents self-adaptive variance vectors for Gaussian mutations.

2. Each individual (p_{ij}, s_{ij}) , creates a single offspring (p'_{ij}, s'_{ij}) using

$$\begin{aligned} s'_{ij} &= s_{ij} \exp(\tau' N(0, 1) + \tau N_j(0, 1)) \\ p'_{ij} &= p_{ij} + s'_{ij} N_j(0, 1), \end{aligned} \quad (6)$$

where $N(0,1)$ denotes a normally distributed, one-dimensional, random number with a mean of zero and variance of one. $N_j(0,1)$ indicates that the random number is newly generated for each value of j . The parameters τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively.

3. Determine the fitness of each individual, including both parents and offspring, based on the fitness measure (e.g., objective function to be optimized).
4. Conduct pair-wise comparison over the union of parents (p_{ij}, s_{ij}) and offspring (p'_{ij}, s'_{ij}) . For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is better than the opponent's, it receives a *win*. Select N individuals out of (p_{ij}, s_{ij}) and (p'_{ij}, s'_{ij}) that have the most wins to form the next generation. This is a technique known as tournament selection in EA literature.
5. Stop the algorithm if the halting criterion is satisfied, otherwise return to Step 2.

For additional details regarding EP, the reader is referred to Fogel (1999) and Porto (2000).

It should be noted that although EP is often useful in a robust evaluation of the best region of a solution space, it may be inefficient and ineffective in fine-tuning the local search within that region. The impact of this inability to fine-tune could possibly be limited by integrating a gradient-based algorithm in the late stages of training, thus taking advantage of one algorithm's strength to compensate for the other's weaknesses.

3. Training Data

As the ANN developed herein is designed to reproduce SWAT estimates of sediment yield and agricultural profit, the data used for training must be generated using the same methodologies, constraints and assumptions implemented within the SWAT-based decision support model. Within this watershed management model, decision variables are represented as cropping and tillage practice combinations for a particular HRU. It is assumed that each HRU represents a particular farm field that is singularly or commonly owned, thus implying that a landowner's decisions regarding their own property will have no influence on decisions made by neighboring landowners. This formulation allows each landowner within the watershed to make independent decisions, yet all decisions contribute toward the overall goal of reducing sediment yield from their watershed. This approach supports the

Illinois Environmental Protection Agency's (ILEPA's) recognition that watershed planning and management begins with the shared responsibility of farmers and other landowners who have ownership rights within the watershed.

Farm management decisions are also made under consideration of multi-year criteria, such as crop rotation, rather than strictly single-year concerns. Accordingly, it is assumed here that a decision policy dictates the seasonal sequence of crops to be grown on an individual farm field for a 3-year period. Furthermore, in the search process, only field crops are considered, and a maximum of two crops per year are permitted to grow. A crop year commences in January, and the second crop of the year can be planted only after the preceding crop is harvested. Therefore, within a 3-year rotation, a maximum of five crops can be grown. The first crop planted in the 3-year period is a warm season crop and is harvested in late September. A winter crop is then planted in early October and is harvested in June. Next, using a double cropping system, warm season crops, such as soybean, that can grow following harvest of winter crops are planted. The fourth crop is a warm season crop that is planted in March or April, and finally the fifth and the last crop of the sequence is a winter crop. In addition, once planted, perennial crops such as hay and pasture are allowed to remain in the field until the end of the three-year plan. These criteria together represent crop management constraints. Satisfaction of these constraints is checked for each sequence of decision variables through use of systematically assigned crop codes (see Table I). For additional detail regarding formulation of the watershed management problem, the reader is referred to Muleta and Nicklow (2002).

To be consistent with the SWAT-based decision support model, the ANN must be trained on a spatial scale of an HRU (i.e., farm field). Thus, for each HRU, a number of five-season sequences of crop types and management practice combinations are randomly generated according to the previously described assumptions and constraints and are used as inputs for the ANN. A typical example of these five land use and management practice sequences (i.e. inputs of the ANN) may be CRNT, WWNT, SYWC, WWFT, SRST, with their associated codes (Table I) of 4, 17, 10, 19, and 8, respectively. The corresponding sediment and crop yield for each

Table I. Examples of codes defining crop types and tillage practice

Crop	Tillage practice	Acronym	Integer code
Soybean	No tillage	SYNT	1
Corn	No tillage	CRNT	4
Sorghum	Conservation tillage	SGCT	8
Wheat	Fall tillage	WWFT	19
Wheat	No tillage	WWNT	17
Soybean after wheat	Conservation tillage	SYWC	10
Alfalfa	No tillage	AFNT	12
Pasture	No tillage	PSNT	14

sequence were estimated by SWAT, which in turn represent the two desired outputs in the training process. Specifically, 500 potential decision policies (i.e., inputs) were generated for each HRU and their corresponding average annual sediment yields and average annual net profits (i.e., outputs) were estimated. Of the 500, 300 input—output sets were used as training data for determining optimal or near-optimal connection weights and ANN architecture that would bring ANN estimates sufficiently close to the desired outputs. However, a cross-training procedure is usually recommended to limit the potential for overtraining, or overfitting. Overfitting occurs when the ANN starts to memorize the individual training examples rather than generalize the trends within the entire dataset. The objective in cross-training is to stop the learning process when the network starts to overfit. To do so, the network is allowed to determine values of the network error function using cross-training datasets, rather than training data, after one complete presentation of training data, or epoch. During the early stages of the search, errors for both the training and the cross-training dataset decrease. After a number of search iterations, however, the training data error may continue to be reduced while the cross-training data starts to increase, which is indicative of overfitting. This is a suitable point to stop training and consider the current weights and architectures as final solutions. One-hundred datasets that were different from the alternatives used during the training process were preserved for cross-training. In addition, performance of an ANN can be best evaluated by subjecting the trained ANN to new patterns that it has not seen during training or cross-training, a process known as verification. Another 100 datasets that were different from policies used in the training and cross-training activity were used to verify the trained ANNs.

Haykin (1999) describes the advantages of input normalization, or standardization, and recommends that it can be undertaken to accelerate the learning process, especially if BP is used in training. According to LeCun (1993) this can be achieved if each input variable is preprocessed so that its mean value, over the entire training set, is close to zero or is small compared to its standard deviation. In addition, it is important that values of the desired outputs of the system be standardized so that they lie within the range of the activation function used in training. There should be an offset by some amount away from the limiting values which otherwise tend to drive the parameters such as synaptic weights and bias to large values, and thereby slow the learning process. In this respect, the hidden nodes are essentially driven into saturation (Haykin, 1999). Accordingly, for each input to be used in training, cross-training and verification in this study, the mean of the 500 input sets was evaluated and subtracted. The resulting inputs were further standardized in such a way that all inputs lie within a range of ± 0.95 . Also, because the activation function used in training is the sigmoid function, which is bounded between 0 and 1, the output datasets were standardized so that they lie within the range of 0.05 to 0.95, allowing an offset of 0.05 from both extremes. Once the data was generated and preprocessed, training was conducted, first using EP alone, and then by hybridizing the EP and BP algorithms.

4. Methodology

4.1. STAGE I—EP TRAINING

The logistical framework of the complete training process is depicted in Figure 2. For application of EP, the authors adopted a population size of 1000 solutions, a maximum of 100 generations, a maximum of six hidden layer and a minimum of

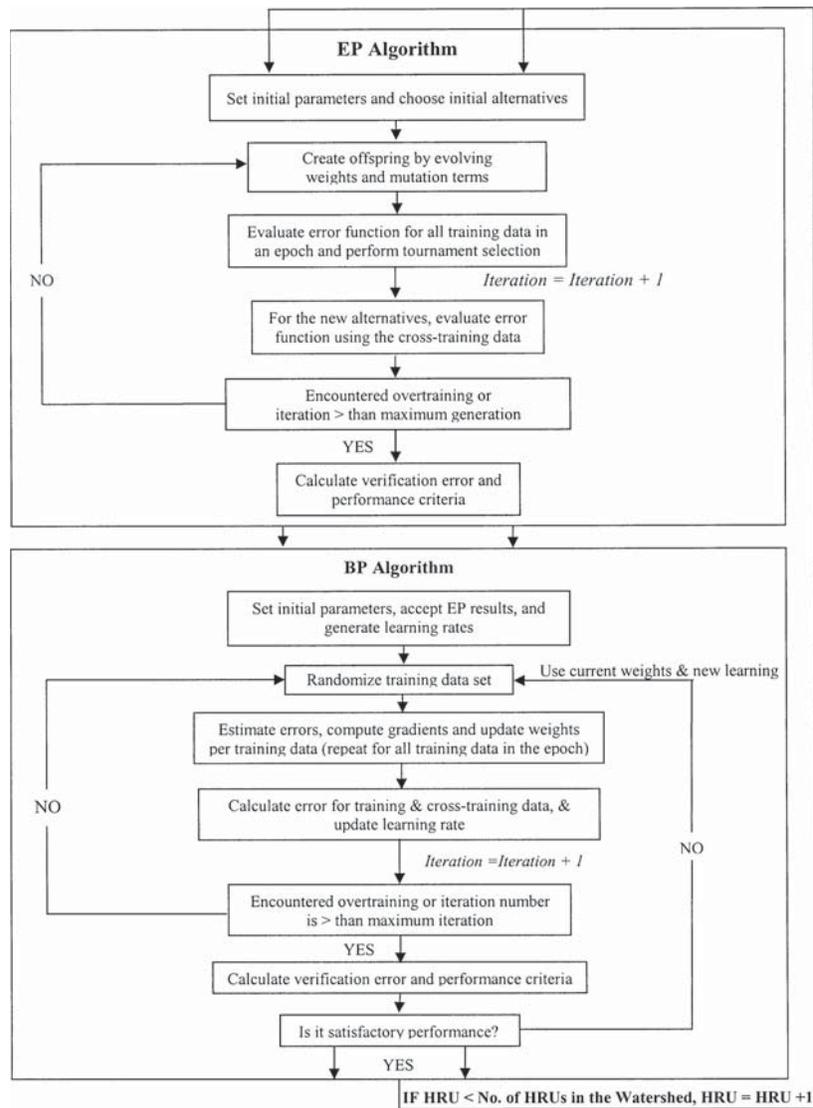


Figure 2. Logistic framework of an EP–BP algorithm.

one hidden layer, a maximum of 15 nodes for each hidden layer and a minimum of one node, and a maximum and minimum weight of 2 and -2 , respectively. These values are established on the basis of complexity of the problem and are guided by limits used and recommended in the literature. To begin, an initial population that consists of random architecture (i.e., random number of hidden layers and random number of nodes on each hidden layer), random weights, bias values, the EP strategic parameters, are generated. Hidden layer assignment is equitably distributed among the initial population. For example, if the maximum number of hidden layers is five and the minimum is one, and if the size of the initial population is 1000, there will be 200 alternatives from each of the possible number of hidden layers. Numbers of nodes for each hidden layer, connection weights and EP strategic parameters within nodes of successive layers are randomly generated from uniformly distributed values. Following the EP methodology, each alternative solution is permitted to yield offspring by evolving its weight and self-adaptive mutation parameter according to Equation (6). For every parent and offspring, and after using the entire training dataset in an epoch, the network error function is evaluated. Tournament selection is then applied to choose solutions that become part of the next generation. The concept of elitism is also applied, which insures against losing the best performing alternative from previous generations. For the best solution of every generation, the error function is evaluated for overfitting tendencies by using the cross-training dataset. If error from the cross-training data increases for five subsequent generations while error from training dataset decreases, the search procedure is stopped and the final architecture and weights are those corresponding to the iteration immediately before that in which cross-training error began to grow. Finally, for the final solution obtained, the verification dataset is applied and the generalization ability of the model to a new environment is tested.

4.2. STAGE II—BP TRAINING

Unless the performance of the EP training is fully satisfactory, for relaxed constraints on weight limits, the BP algorithm is subsequently applied. The weight vectors obtained from EP are used as the initial weight surface from where the BP algorithm commences the fine-tuning process. Similar to EP, training, cross-training and verification are essential to the application of the algorithm. For BP, there are two common modes of training, a sequential mode and a batch mode. In sequential training, weights are updated after the presentation of each training example; whereas in the batch mode of training, they are updated only after all training datasets in an epoch have been presented. For large, difficult problems that have highly redundant training datasets, such as the watershed management problem, the sequential mode of BP learning is computationally faster than the batch mode since it takes advantage of the redundancy as the examples are presented. In addition, randomization of the order in which the training examples are presented from one epoch to the next makes the search region stochastic. This stochastic characteristic

in turn makes BP less likely to be trapped in local optima (Haykin, 1999). Therefore, sequential training and a randomized presentation order are adopted in this study.

In application, the forward pass of BP algorithm is performed until outputs are estimated. Network error is then evaluated according to Equation (3), except that N now assumes a value of unity for the sequential mode of training. This error is propagated backward through the layers of the network by refining weights according to Equation (4), and the forward—backward evaluation is repeated for the next training dataset. Following each epoch, the order of training data presentation is randomized, and the cross-training data is applied and corresponding error computed. Similar to EP, if the cross-training errors continue to increase for five subsequent iterations while the training error decreases, the search is halted and the final weight vectors are those corresponding to the iteration immediately before that in which cross-training error began to increase.

To accelerate convergence of BP, Haykin (1999) recommends using a different learning-rate for every adjustable network parameter (i.e., weights and bias values) and altering this rate from one iteration to the next. This suggestion has been implemented by assigning different learning rates for all connection weights and by updating those weights at a linearly decreasing rate from iteration to iteration. Specifically, weights are updated according to

$$\mu_{ij}^{(m+1)} = \mu_{ij}^{(m)} + \frac{\mu_{ij}^M - \mu_{ij}^1}{M} \quad (7)$$

where $\mu_{ij}^1, \mu_{ij}^M, \mu_{ij}^m, \mu_{ij}^{m+1}$ are learning rates between node i and node j at the initial iteration, final (M th) iteration, iteration m , and iteration $m + 1$, respectively. Furthermore, because layers near the output layer generally have larger gradients than those at the front end of the network, smaller learning rate values are assigned to end layers (i.e., μ_{ij}^1 and μ_{ij}^M) so that all nodes in the network learn at similar rates. The momentum factor was, however, set to a constant of 0.2.

When the convergence criterion, defined here as a maximum of 1000 iterations, is satisfied, resulting weights are applied to the verification dataset and performance of the model is tested. If its performance is not satisfactory, the search will restart by assigning a new learning rate (μ_{ij}^1), and the BP search will be repeated for that particular HRU until the convergence criterion is satisfied. After some preliminary testing, a maximum of five such repetitions are allowed in this study. After five attempts, the best result among the original and five additional attempts is accepted.

4.3. PERFORMANCE CRITERIA

There are various methods available for evaluating model performance, including graphical and numerical indicators. In this study, the authors use a scatter plot

of simulated and desired outputs for calibration and verification of datasets. In addition, two numerical measures are used, namely the root mean square error (RMSE) and the Nash and Sutcliffe (1970) R^2 efficiency. These metrics can be expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (d_i - y_i)^2}{N}} \quad (8)$$

and

$$R^2 = \frac{F_0 - F}{F_0} \quad (9)$$

where

$$F_0 = \sum_{i=1}^N (d_i - \bar{y})^2 \quad (10)$$

and

$$F = \sum_{i=1}^N (d_i - y_i)^2 \quad (11)$$

Here, N is the total number of datasets, training or verification; d_i is a desired output (i.e., SWAT estimate) for the i th dataset; y_i is the ANN output; and \bar{y} is the mean value of the desired output for the training data. An ideal value of RMSE is zero, in which case the R^2 efficiency index assumes a value of unity.

5. Results and Discussion

Big Creek watershed, located in the Cache River basin of southern Illinois and shown in Figure 3, is used to test the hybrid training algorithm and evaluate the suitability of the resulting ANN as a replacement for SWAT in the watershed decision support model. Because of its high sediment yield and influence on the Lower Cache River, multiple government agencies and private organizations have identified the Big Creek watershed as a priority area for targeted remediation. The area is undergoing extensive study as part of the Illinois' Pilot Watershed Program, through cooperation among the Illinois Department of Natural Resources (IDNR), the Illinois Department of Agriculture, ILEPA, and the U.S. Natural Resources Conservation Service (IDNR, 1998).

A 30-meter resolution U.S. Geological Survey (USGS) DEM, an IDNR land-use map, and a soils map were obtained for the region of study. The Big Creek watershed was delineated from the DEM and subdivided into 73 subbasins, and

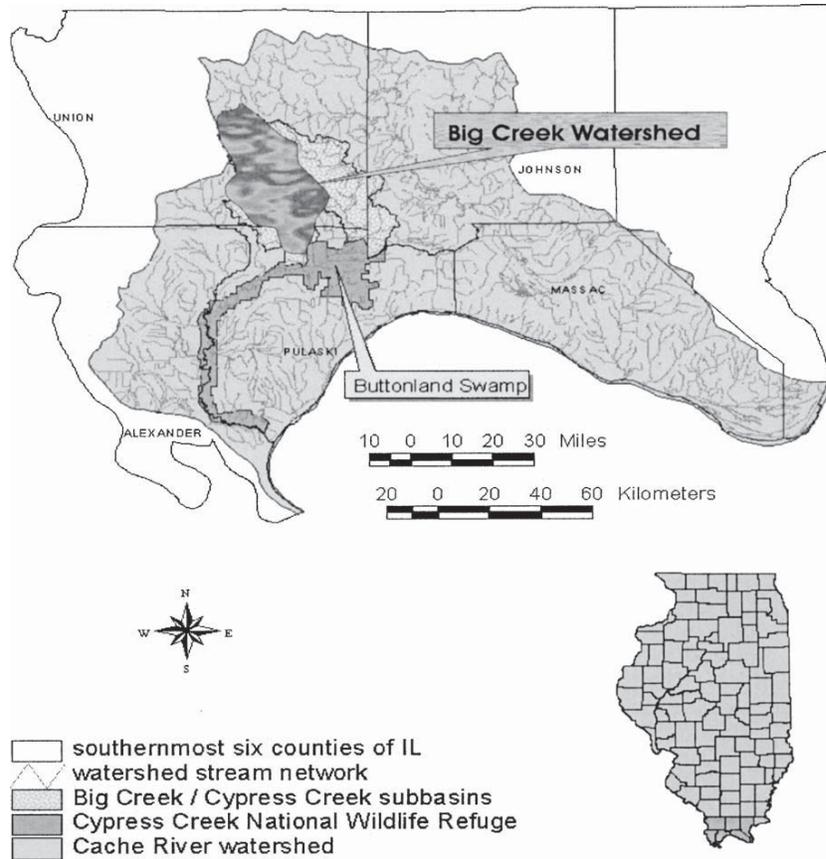


Figure 3. Location of Big Creek watershed, Southern Illinois.

the land use and soils maps were superimposed over the subdivided watershed to identify HRUs. For this application, dominant soils types and land uses from each subbasin were used in establishing HRUs, a statement that implies that each farm field consists of a single soil type and land cover during any one season and that there are the same number of HRUs as subbasins. Observed data related to daily precipitation, daily maximum temperature and daily minimum temperature were obtained from the National Weather Service for Anna, IL, a nearby weather station.

A database of 19 suitable cropping and tillage practice combinations was prepared for the application. This database contains miscellaneous information on planting dates, harvesting dates, dates to apply tillage, fertilizer and pesticide types, application dates and dosages, heat units required for plant maturity, and runoff curve numbers. Information for the management database was collected from the Illinois Agronomy Handbook (UIUC, 2000) and from the National Agricultural Statistics Service (USDA, 2000). Additionally, an economic database was

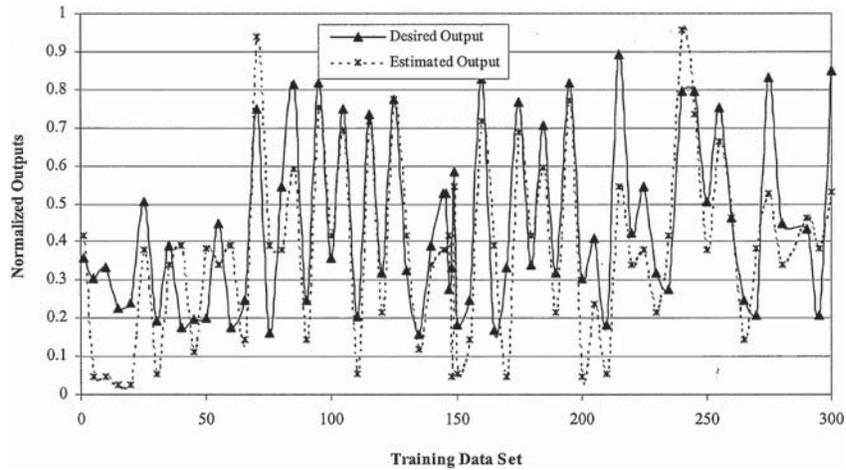
prepared that provides information on production expenses and selling prices for associated crop types. The production expenses were broadly classified as variable costs and fixed costs. Variable costs include expenses for seed, chemical, insurance and interest for machinery, labor and trucking. Fixed costs are related to the cost of owning land and machinery and were not used in the search process. Ten-year (i.e., 1990–1999) averages of production expenses and selling price data for the study area were collected from various sources and were subsequently used in estimating the net economic benefit of implementing a potential decision policy. The major sources used in preparing the economic database were the University of Illinois at Urbana-Champaign (UIUC) Farm and Resource Management Laboratory (FaRM Lab) (UIUC, 1999), the Illinois Census of Agriculture (USDA, 1997a), and the Cost and Returns Estimator model (CARE) farm budget for Southern Illinois (USDA, 1997b).

Using the operational management and economic databases, along with model inputs, SWAT was used to generate datasets for training, cross-training and verification for the 73 HRUs. These datasets were based on randomly generated decisions, or combinations of land uses and management operations. The model inputs and outputs were then standardized, and datasets used for the cross-training and verification were checked for uniqueness so that the generalization ability of the trained ANN to a new environment could be tested.

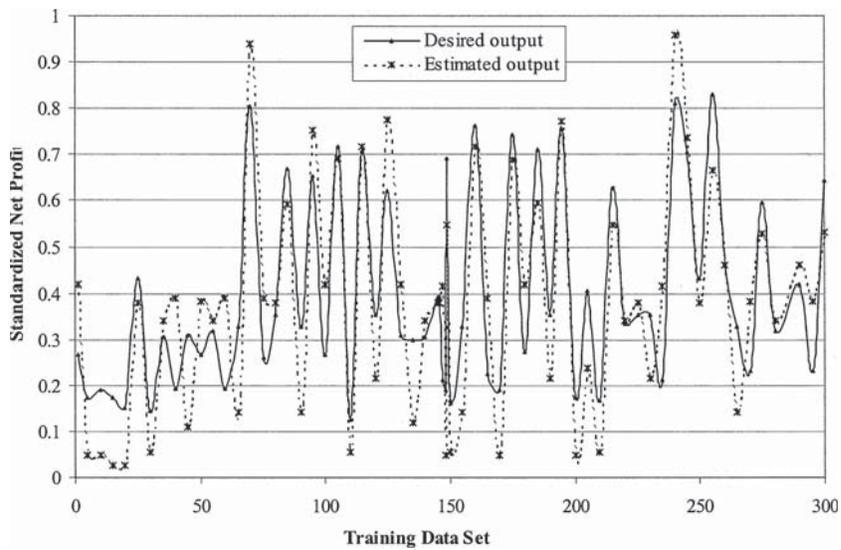
5.1. PERFORMANCE OF INDEPENDENT EP TRAINING

In assessing the performance of EP alone, the limiting values on synaptic connections and strategy parameters were relaxed to ± 20 and ± 10 , respectively. The search was allowed to continue for 500 generations, with an initial population of 1000 solutions. For a typical HRU, a graphical comparison of desired outputs, provided by SWAT, and ANN outputs (also referred to as estimated outputs) are given in Figure 4a and Figure 4b for sediment yield and net profit, respectively. Similar plots are provided in Figure 5a and 5b for the verification data. Performance was also evaluated for both training and verification using the R^2 efficiency and RMSE; rather than reporting these indices for all HRUs, however, summarized statistics are presented in Table II. This summary table provides the worst, best, mean and standard deviation of indices from all HRUs for both training and verification. It should be noted that, the extreme (i.e., worst and best) values do not necessarily correspond to a single HRU. The worst R^2 value for sediment yield in the training category may be for one HRU, while the same parameter for net profit may be for another HRU.

A review of these results reveals that the EP-based training algorithm lacks a capability to fine-tune the search, even though it has a good generalization tendency. Its search result is not sufficient, especially for the verification data, where it yielded R^2 efficiencies as high as -0.8721 . The negative R^2 efficiencies indicates that the model being tested is yielding verification outputs that are worse estimates than the



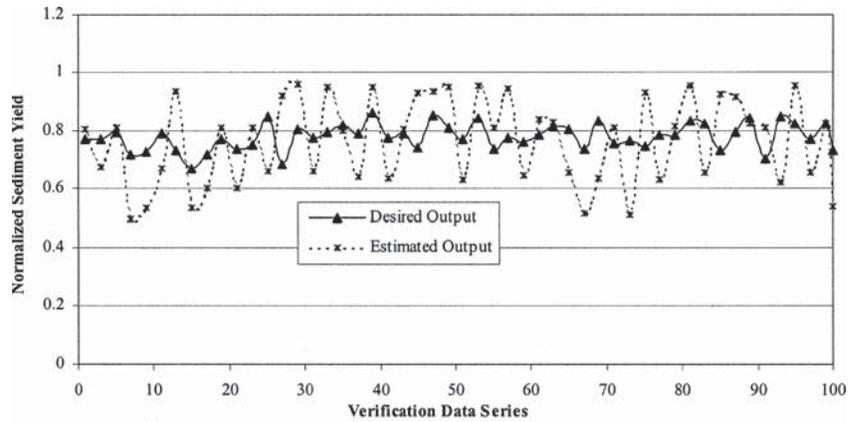
(a)



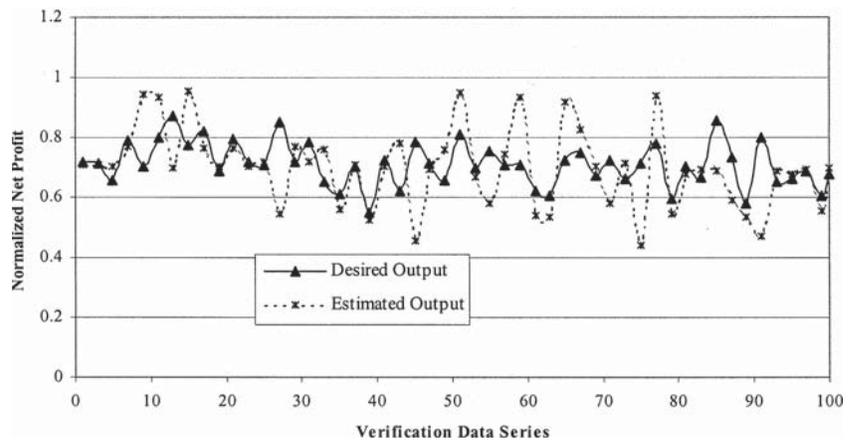
(b)

Figure 4. (a) Comparison of estimated and desired sediment yield for training data using the EP algorithm. (b) Comparison of estimated and desired net profit for training data using the EP algorithm.

mean of the outputs used in training. For this particular HRU, the search did not improve after approximately 175 generations, as shown in the error convergence plot for training and cross-training in Figure 6. Furthermore, the computational demand required for this application makes the independent performance EP even less



(a)



(b)

Figure 5. (a) Comparison of estimated and desired sediment yield for verification data using the EP algorithm. (b) Comparison of estimated and desired net profit for verification data using the EP algorithm.

tolerable. The operation required about 42.2 hrs on a 1.3 GHz, Pentium IV processor. Therefore, even though the literature indicates that EP is generally computationally faster than a GA (Yao, 1999), it was not sufficiently fast for the problem considered in this study.

5.2. HYBRID EP–BP PERFORMANCE

Dissatisfied by the capability of EP to adequately train the ANN, the performance of the hybrid EP–BP algorithm was subsequently evaluated. Once EP located a

Table II. Statistical summary of ANN performance using the EP algorithm

Statistics	R^2 efficiency				RMSE			
	Training		Verification		Training		Verification	
	Sed. Yield	Net Profit	Sed. Yield	Net Profit	Sed. Yield	Net Profit	Sed. Yield	Net Profit
Worst	0.4181	0.3709	0.3135	-0.8721	0.2400	0.2283	0.2108	0.3054
Best	0.9416	0.9493	0.9702	0.9634	0.0660	0.0597	0.0476	0.0513
Mean	0.7389	0.7840	0.7545	0.5531	0.1411	0.1264	0.1221	0.1575
S.D	0.1307	0.1165	0.1449	0.3561	0.0445	0.0367	0.0338	0.0577

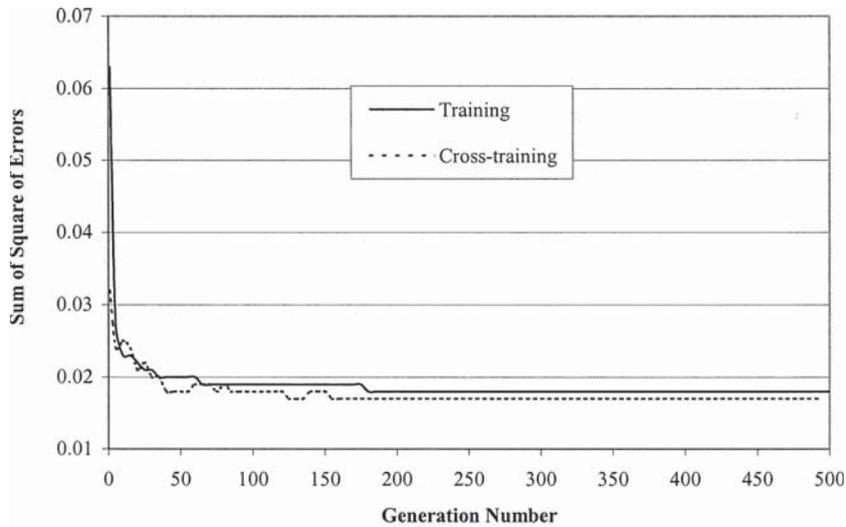
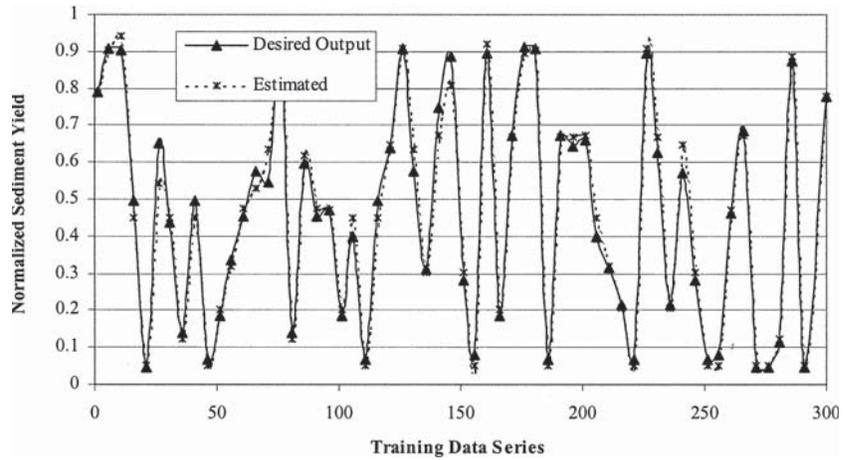
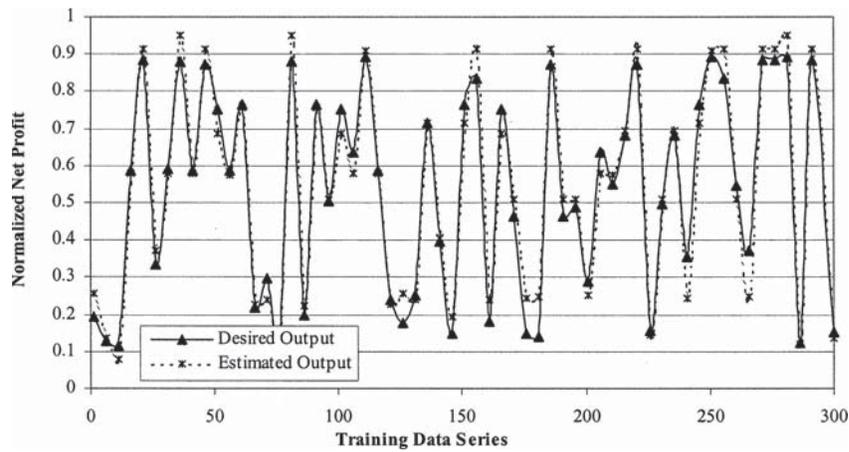


Figure 6. Error convergence plot for training and cross-training data using the EP algorithm.

near-optimal ANN architecture and starting values of connection weights, the BP algorithm was used to fine-tune the search. Weights and Gaussian mutation terms for the EP algorithm were limited in a way that they do not affect the BP algorithm's performance. For example, the limits on weights were assigned primarily on the basis of the fact that the final weights obtained by EP are those that would be supplied as initial values for BP, which is often recommended to be in the range of ± 1 . Otherwise, the BP algorithm may suffer from saturation of hidden layers and subsequent slowing of the search process (Haykin, 1999). The number of search iterations for EP was also reduced to 100 to minimize computational demand. Search results for the hybrid algorithm are presented both graphically and numerically. Figure 7a shows comparison plots of desired and estimated outputs for sediment yield as



(a)

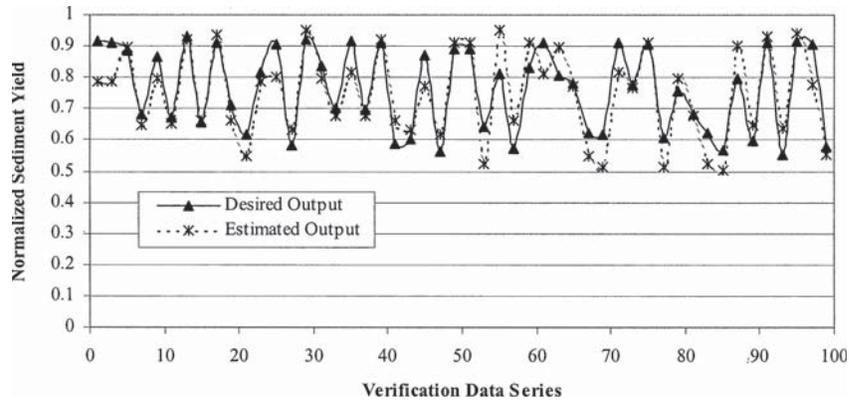


(b)

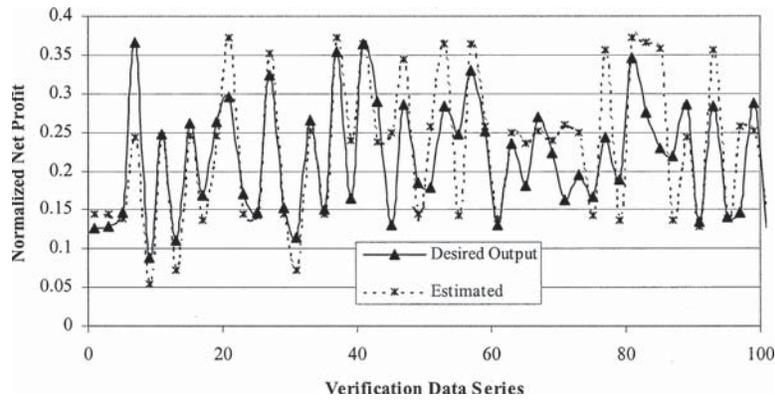
Figure 7. (a) Comparison of estimated and desired sediment yield for training data using the BP-EP algorithm. (b) Comparison of estimated and desired net profit for training data using the BP-EP algorithm.

evaluated using the training data, whereas Figure 7b shows a similar plot for net profit. Figures 8a and 8b present similar graphical measures for verification data. A statistical summary of R^2 and RMSE indices over all HRUs of the watershed is given in Table III, and an error convergence plot for the training and cross-training datasets is given in Figure 9.

Among all HRU's, the largest number of hidden layers identified by EP as the favored architecture was two, with nine nodes on layer one and two nodes on the



(a)



(b)

Figure 8. (a) Comparison of estimated and desired sediment yield for verification data using the BP-EP algorithm. (b) Comparison of estimated and desired net profit for verification data using the BP-EP algorithm.

second hidden layer, thus representing a compact architecture. As indicated in both graphical and numerical results, the hybrid algorithm has significantly enhanced the overall search capability. Unlike the independent EP performance where R^2 efficiency as low as -0.8721 was obtained for verification of the net benefit, the worst R^2 efficiency obtained using the hybrid algorithm was 0.3518 . Moreover, the average R^2 and RMSE are promising and low standard deviations are obtained, thus demonstrating the robustness of the hybrid algorithm. However, a brief comparison of Figure 6 and Figure 9 reveals that, for this particular HRU, the sum of square of errors obtained by the EP algorithm is lower than that obtained by the EP-BP algorithm; whereas a comparison of Figures 4, 5, 7, and 8, which are also obtained

Table III. Statistical summary of the ANN performance using the hybrid EP–BP algorithm

Statistics	R^2 efficiency				RMSE			
	Training		Verification		Training		Verification	
	Sed. Yield	Net Profit	Sed. Yield	Net Profit	Sed. Yield	Net Profit	Sed. Yield	Net Profit
Worst	0.3518	0.4959	0.5292	0.4004	0.2313	0.2000	0.1685	0.1900
Best	0.9915	0.9848	0.9949	0.9925	0.0239	0.0365	0.0197	0.0245
Mean	0.8894	0.8751	0.8928	0.7725	0.0774	0.0862	0.0706	0.1030
S.D.	0.1198	0.1188	0.0971	0.1824	0.0449	0.0407	0.0330	0.0454

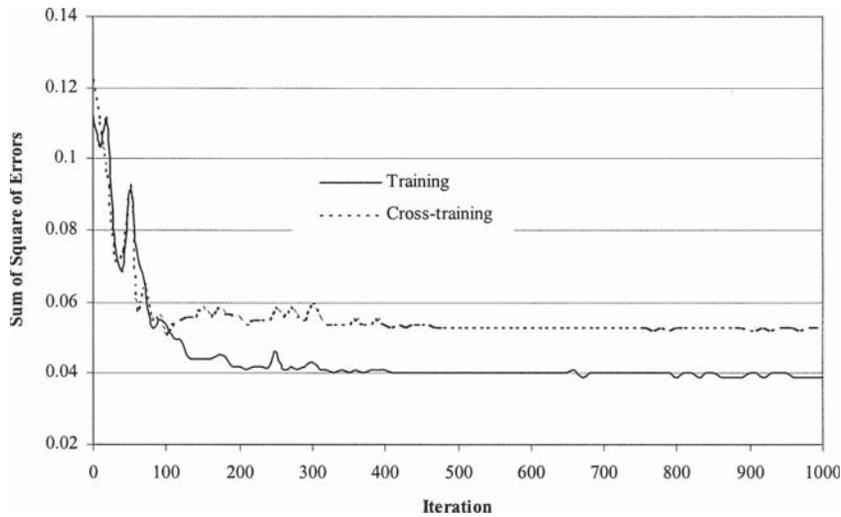


Figure 9. Error Convergence plot for training and cross-training data using the BP–EP algorithm.

for the same HRU, proves otherwise. This raises an interesting question regarding suitability of using single criteria such as sum of square of errors, as a goodness-of-fit for models.

5.3. EFFICIENCY OF THE ANN-BASED DECISION SUPPORT MODEL

Impressed by performance of the EP–BP algorithm, the trained ANN was used as a replacement for SWAT in the multiobjective watershed management model (Muleta and Nicklow, 2002). For the Big Creek watershed, using a population of 100 solutions and a maximum of 100 generations, the SWAT-based decision support model required 63.25 hrs of computational time on a 1.3 GHz, Pentium IV processor. The ANN-based decision support model, however, finished its execution in just

4.5 min, representing an extraordinary improvement. The solutions derived by the ANN-based and the SWAT-based multiobjective models are the same for 90% of the HRUs in the watershed, witnessing the capability of the developed ANN to replace SWAT. In examining execution times for the entire process including data generation (6.3 hrs), training (9.6 hrs), and the actual search process (4.5 min), the ANN-based model required just 16 hrs to locate optimal or near-optimal land use and management patterns. The latter represents approximately a 75% reduction in computational time when compared to the SWAT-based model. It should be noted that the data generation and training processes need only be performed once for the same watershed and simulation period, assuming other environmental variables are not changed. Therefore, following the initial execution, repeated searches could be performed in a matter of minutes. Considering the average user, this reduction in computational time could potentially improve the practical utility of the decision support model. After all, one of the many criteria often used by those in practice to select a particular model, whether in water resources engineering or other disciplines, is the feasibility of computational time required for execution. In addition, the results can be further generalized to studies that target performance comparisons between various search algorithms and search operators.

6. Conclusions

The EP–BP hybrid training algorithm adopted in this study is effective for calibrating an ANN to the highly nonlinear and complex processes of watershed erosion and sedimentation as a function of land use and management combinations. The sediment yields and the net economic benefits generated by the trained ANN and those generated by SWAT model, as a result of implementing a sequence of the land use and management practices given in Table I over span of five cropping seasons, are in excellent agreement (Table III, and Figures 7 and 8). The hybrid algorithm limits some of the common drawbacks inherent to BP and other gradient-based algorithms. These include a heavy dependence on the skills of the modeler, convergence to local optima, and the typical trial-and-error procedure required for designing compact, effective ANN architectures. In addition, as demonstrated herein, the hybrid outperforms the independent application of EP for training. The EP–BP algorithm could be useful for solving a variety of complex problems apart from development of ANN-based hydrologic simulation models.

The replacement of SWAT by an ANN within the watershed decision support model has resulted in a significant reduction in computational time. The resulting impact may be the improved practical utility of the overall model (Muleta and Nicklow, 2002) for solving erosion and sedimentation problems. Moreover, this study represents an example of the potential role of ANNs in addressing computational demands of various problems without sacrificing accuracy of rigorous models. For example, the use of optimization–simulation tools that are based on theoretically justified techniques, such as distributed hydrologic models, are often

impractical due to their large computational times, an issue that plagues many engineering systems applications. The application of ANNs, along with effective and efficient training algorithms such as the EP–BP hybrid algorithm, can potentially alleviate this problem by providing quick and reasonable estimates of the theoretically inspired models.

Acknowledgements

The authors wish to thank the Illinois Council for Food and Agricultural Research (CFAR) and Southern Illinois University at Carbondale for their support of this ongoing research effort and the anonymous reviewers for their valuable comments.

References

- Arnold, J. G., Srinivasan, R., Muttiah, R. S., and Williams, J. R., 1998, 'Large area hydrologic modeling and assessment part I: Model development', *J. American Water Resour. Assoc.* **34**(1), 73–89.
- ASCE, 1999, GIS Modules and Distributed Models of Watersheds, ASCE, Reston, VA.
- ASCE, 2000a, 'Task committee on application of artificial neural networks in hydrology, artificial neural networks in hydrology I: Preliminary concepts', *J. Hydrologic Eng.* **5**(2), 115–123.
- ASCE, 2000b, 'Task committee on application of artificial neural networks in hydrology, artificial neural networks in hydrology II: Hydrologic applications', *J. Hydrologic Eng.* **5**(2), 124–137.
- Cancelliere, A., Giuliano, G., Ancarani, A. and Rossi, G., 2002, 'A neural network approach for deriving irrigation reservoir operation rules', *Water Resour. Manage.* **16**, 71–88.
- Fogel, D. B., 1994, 'An introduction to simulated evolutionary computation', *IEEE Trans. on Neural Netw.* **5**(1), 3–14.
- Fogel, L. J., 1999, *Intelligence Through Simulated Evolution*, Wiley, New York.
- Gupta, H. V., Ksu, K. and Sorooshian, S., 1997, 'Superior Training of Artificial Neural Networks Using Weight–Space Partitioning', *Proceedings of the IEEE International Conference on Neural Networks*, Institute of Electrical and Electronics Engineers, New York.
- Haykin, S., 1999, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ.
- IDNR, 1998, *The Pilot Watershed Program: Watershed Management, Monitoring and Assessment*, Illinois Department of Natural Resources, Springfield, IL.
- Jain, A., Varshney, A. K. and Joshi, V. C., 2001, 'Short-term water demand forecast modeling at IIT Kanpur using artificial neural networks', *Water Resour. Manage.* **15**, 299–321.
- LeCun, Y., 1993, *Efficient Learning and Second-Order Methods*, *A Tutorial at NIPS 93*, Denver, CO.
- Maier, H. R. and Dandy, G. C., 1996, 'The use of artificial neural networks for the prediction of water quality parameters', *Water Resour. Res.* **32**(4), 1013–1022.
- Muleta, M. K., 2003, 'A Decision Support System for the Management of Non-Point Source Pollution from Watersheds', *Ph.D. Thesis*, College of Engineering, Southern Illinois University at Carbondale, Carbondale, IL.
- Muleta, M. K. and Nicklow, J. W., 2001, 'Using Genetic Algorithms and SWAT to Minimize Sediment Yield from an Agriculturally Dominated Watershed', *Proceedings of the 2001 World Congress of the Environmental and Water Resources Institute (CD-ROM)*, ASCE, Reston, VA.
- Muleta, M. K. and Nicklow, J. W., 2002, 'Evolutionary algorithms for multiobjective evaluation of watershed management decisions', *J. Hydroinformatics* **4**(2), 83–97.
- Muttiah, R. S., Srinivasan, R. and Allen, P. M., 1997, 'Prediction of two-year peak stream discharges using neural networks', *J. Am. Water Resour. Assoc.* **33**(3), 625–630.

- Nash, J. E. and Sutcliffe, J. V., 1970, 'River flow forecasting through conceptual models: Part I—A discussion of principles', *J. Hydrol.* **125**, 277–291.
- Nicklow, J. W. and Muleta, M. K., 2001, 'Watershed management techniques to control sediment yield in agriculturally dominated areas', *Water Int.* **26**(3), 435–443.
- Porto, V. W., 2000, Evolutionary programming, *Evolutionary Computation I: Basic Algorithms and Operators*, in T. Back, D. B. Fogel and Z. Michalewicz (eds.), Institute of Physics, Philadelphia, PA, 89–102.
- Ranjithan, S., Eheart, J. W. and Rarret, J. H., Jr., 1993, 'Neural network screening for groundwater reclamation under uncertainty', *Water Resour. Res.* **29**(3), 563–574.
- Rogers, L. L. and Dowla, F. U., 1994, 'Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling', *Water Resour. Res.* **30**(2), 457–481.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986, Learning internal representations by error propagation, in D. E. Rumelhart, J. L. McClelland and the PDP Research Group (eds.), *Parallel distributed processing: Vol. I*, MIT Press, Cambridge, MA, pp. 318–362.
- Tokar, A. S. and Johnson, P. A., 1999, 'Rainfall-runoff modeling using artificial neural networks', *J. Hydrologic Eng.* **4**(3), 232–239.
- UIUC, 1999, Farm and Resource Management Laboratory, <http://w3.aces.uiuc.edu/ACE/farmlab/crop_budget/> (April 20, 2001).
- UIUC, 2000, Illinois Handbook of Agronomy, <<http://web.aces.uiuc.edu/aim/IAH/>> (Januray 25, 2001).
- USDA, 1997a, Illinois Census of Agriculture, <<http://www.census.gov/prod/ac97/ac97a-13.pdf>> (April 10, 2001).
- USDA, 1997b, Cost and Return Estimator (CARE) Model, <<http://waterhome.brc.tamus.edu/care/index.html>> (April 15, 2001).
- USDA, 2000, National Agricultural Statistics Service, <<http://www.usda.gov/nass/>> (January 25, 2001).
- Yang, C. C., Prasher, S. O., Lacroix, R., Sreekanth, S., Patni, N. K. and Masse, L., 1997, 'Artificial neural network model for subsurface-drained farmland', *J. Irrig. Drain. Eng.* **123**(4), 285–292.
- Yao, X., 1999, 'Evolving artificial neural networks', *Proc. IEEE* **87**(9), 1423–1447.
- Yao, X. and Liu, Y., 1997, 'Fast evolution strategies', *Control Cybern.* **26**(3), 467–496.
- Yao, X. and Liu, Y., 1998, 'Towards designing artificial neural networks by evolution', *Appl. Math. Comput.* **91**, 83–90.