

Particle swarm optimization feedforward neural network for modeling runoff

^{1*}K. K. Kuok; ¹S. Harun; ²S. M. Shamsuddin

¹Department of Hydraulics and Hydrology, University Technology Malaysia, Johor, Malaysia

²Department of Computer Graphics and Multimedia, University Technology Malaysia, Johor, Malaysia

Received 3 July 2009; revised 19 August 2009; accepted 21 August 2009; available online 1 December 2009

ABSTRACT: The rainfall-runoff relationship is one of the most complex hydrological phenomena. In recent years, hydrologists have successfully applied backpropagation neural network as a tool to model various nonlinear hydrological processes because of its ability to generalize patterns in imprecise or noisy and ambiguous input and output data sets. However, the backpropagation neural network convergence rate is relatively slow and solutions can be trapped at local minima. Hence, in this study, a new evolutionary algorithm, namely, particle swarm optimization is proposed to train the feedforward neural network. This particle swarm optimization feedforward neural network is applied to model the daily rainfall-runoff relationship in Sungai Bedup Basin, Sarawak, Malaysia. The model performance is measured using the coefficient of correlation and the Nash-Sutcliffe coefficient. The input data to the model are current rainfall, antecedent rainfall and antecedent runoff, while the output is current runoff. Particle swarm optimization feedforward neural network simulated the current runoff accurately with $R = 0.872$ and $E^2 = 0.775$ for the training data set and $R = 0.900$ and $E^2 = 0.807$ for testing data set. Thus, it can be concluded that the particle swarm optimization feedforward neural network method can be successfully used to model the rainfall-runoff relationship in Bedup Basin and it could be to be applied to other basins.

Keywords: Backpropagation neural network; Coefficient of correlation; Modeling runoff; Nash-Sutcliffe coefficient; Particle swarm optimization feedforward neural network

INTRODUCTION

Rainfall-runoff relationships are amongst the most complex hydrological phenomena to understand due to the tremendous spatial and temporal variability of catchment characteristics and rainfall patterns (Tokar and Johnson, 1999). The transformation of rainfall to runoff for streamflow forecasting remains important to hydrologists for water supply, flood control, irrigation, drainage, water quality, power generation, recreation and aquatic and wildlife protection issues. This transformation involves many highly complex components, including interception, depression storage, infiltration, overland flow, interflow, percolation, evaporation and meteorological conditions of the catchment. These data are usually hard to obtain and are not always available. All these non-stationary and usually non-linear phenomena make difficult the accurate estimation of runoff.

With the development of artificial intelligence (AI) in recent years, neural networks (NN) have been proposed to transpiration. Runoff also depends on catchment topography, river network topology, river cross-sections, soil characteristics and antecedent moisture content.

Moreover, antecedent moisture conditions are always changing and depend upon both present and past hydrological complex processes and large volumes of data. In particular, a backpropagation neural network (BPNN) is useful for handling real-time, non-stationary and non-linear natural phenomena (Nishimura and Kojiri, 1996). The natural behavior of rainfall-runoff systems is appropriate for the application of BPNN. The last decade has witnessed many applications of BPNN in water resources. These include modeling of the rainfall-runoff process (Elshorbagy *et al.*, 2000; Bessaih *et al.*, 2003); inflow estimation (Harun *et al.*, 1996); runoff analysis in a humid forest catchment (Gautam *et al.*, 2000); river flow prediction (Imrie *et al.*,

✉ *Corresponding Author Email: kkuok100@yahoo.com.sg
Tel./Fax: 6016 8544 298

2000; Dastorani and Wright, 2001); setting up stage-discharge relations (Jain and Chalisgaonkar, 2000); ungauged catchment flood prediction (Wright and Dastorani, 2001) and short term river flood forecasting (Garcia-Bartual, 2002), prediction of carbon monoxide as one of primary air pollutants (Abbaspour et al., 2005), forecasting the mean monthly total ozone concentration (Bandyopadhyay and Chattopadhyay, 2007) and evaluating performance of immobilized cell biofilter treating hydrogen sulphide vapors (Rene et al., 2008).

However, the major disadvantages of BPNN are its relatively slow convergence rate (Zweiri et al., 2003) and solutions being trapped at local minima. Basically, BPNN learning is a hill climbing technique. Hence, it runs the risk of being trapped in local minima, where every small change in synaptic weight increases the cost function. Sometimes, the network is stuck where there exists another set of synaptic weights for which the cost function is smaller than the local minimum in the weight space. This made termination of the learning process at local minima by BPNN is undesirable. Besides, there are many elements to be considered by NN modeler, such as the number of input, hidden and output nodes, learning rate, momentum rate, bias, minimum error and activation/transfer function. All these elements will also affect the convergence of BPNN learning. Many solutions are proposed by neural network researchers to overcome the slow convergence rate and solutions being trapped at local minima problems. Some powerful optimization algorithms, that based on simple gradient descent algorithm (Bishop, 1995) such as conjugate gradient decent, scaled conjugate gradient descent, quasi-Newton BFGS and Levenberg-Marquardt methods has been devised to improve the convergence rate. Another solution proposed by NN researcher is trying to guide the learning so that the converge speed become faster. The guidelines provided are including select better functions, learning rate, momentum rate and activation functions. Besides, new advance algorithms were developed to hybrid with NN. Genetic Algorithm (GA) is one of the latest algorithms proposed to determine the learning rate and momentum rate and will produce a set of weight that can be used for testing related data. However, the performance of GA is less competence compared with PSO. Lee et al. (2005) compared PSO and GA for excess return evaluation in stock market. Lee et al. (2005) proclaimed that PSO algorithm is better compared to GA, where particle swarm optimization (PSO) can reach the global optimum value with less

iteration, keep equilibrium versus GA and shows the possibility to solve the complicated problem using only basic equations. Meanwhile, Haza (2006) also proved that particle swarm optimization feedforward neural network (PSONN) is much more effective than genetic algorithm backpropagation neural networks (GANN) for solving the classification problems. Besides, Bong and Bryan (2006) also optimized hydraulic transient protection devices using GA and PSO approaches. Results also revealed that PSO has tended to discover a better solution than the GA approach.

Since the previous works revealed the performance of PSO is better than GA and PSO is able to solve a wide array of different optimization problems including most of the problems can be solved by GA (Van den Bergh, 2001), a novel method that hybrid the PSO with ANNs is developed for solving optimization problem especially in the field of hydrology. This hybrid method is called particle swarm optimization feedforward neural network. This PSONN is proposed to improve the convergence rate of NN and avoid solutions being trapped at local minima. According to Van den Bergh and Engelbrecht (1999), PSO is made up of particles, where each particle has a position and a velocity. The idea of PSO in NN is to get the best set of weight (or particle position) where several particles (problem solution) are trying to move to the best solution and this will avoid the solution trap at local minima. In this study, PSONN is applied for calibrating rainfall-runoff model to simulate current runoff accurately.

Currently, little works has been focused on using PSONN for solving optimization problem. Furthermore, the application of PSONN for solving hydrological optimization problem is really scarce. Therefore, there is a need to propose PSONN method for solving optimization problems particularly in calibration and optimization of rainfall-runoff model. However, PSONN has been successfully applied as an efficient tool in solving classification problems in different areas. Zhang et al. (2000) applied PSONN to solve the classification problems in the medical domain particularly in breast cancer and heart disease. Van den Bergh (2001) applied the PSO to train NN for classifying iris, cancer, wine, diabetes, hepatitis, Henan and cubic data sets. Haza (2006) proved that PSONN is much more effective than genetic algorithm backpropagation neural networks (GANN) for solving the classification problems using universal Exclusive Or (XOR), Cancer and Iris data set, where XOR is a logical operation on two operands that

results in a logical value of true if and only if one of the operands but not both has a value of true. In addition, Eberhart and Hu (1999) used PSO to evolve a NN to be used for analysis of human tremor, which distinguished between normal subjects and those with tremor including essential tremor and Parkinson’s disease. In this study, PSONN is employed to simulate current runoff accurately using only antecedent rainfall, current rainfall and runoff data. Section 2 will review the materials and methods used for model calibration that include basic concept of PSONN, the selected study area, model development, sensitivity of the PSONN performance to the length of the calibration data and related parameters using various data sets. Subsequently, the results obtained and discussion using various configuration of PSONN will be explained in section 3. Sungai Bedup Basin, sub-basin of Sadong Basin, Sarawak, Malaysia was selected for model calibration. The series data used for PSONN calibration are daily rainfall and runoff data ranging from 1997 to 1999.

MATERIALS AND METHODS

Particle swarm optimization feedforward neural network

Basic PSO procedure

Particle swarm optimization, a new branch of the soft computing paradigms called evolutionary algorithms (EA), was developed by Kennedy and Eberhart (1995). It is a group-based stochastic optimization technique for continuous nonlinear functions. It is also a simple concept adapted from natural decentralized and self-organized systems where all the particles move to get better results. According to Song and Gu (2004), researchers have paid more and more attention to PSO algorithm because of its convenience of realization and promising optimization ability in various problems. PSO is initialized with a group of random particles (trial solutions), which are assigned with random positions and velocities. The algorithm then searches for optima through a series of iterations where the particles are moved through the hyperspace searching for potential solutions. These particles “learn” over time in response to their own experience and those of other particles in their group (Ferguson, 2004). According to Eberhart and Shi (2001), each particle keeps track of its best fitness position in hyperspace that it has achieved so far. This best position value is called personal best or “pbest”. The overall best value obtained by any particle so far in the

population is called global best or “gbest”. During each iteration, every particle is accelerated towards its own “pbest” as well as in the direction of the “gbest” position. This is achieved by calculating a new velocity term for each particle based on the distance from its “pbest” as well as its distance from the “gbest” position. These two “pbest” and “gbest” velocities are then randomly weighted to produce the new velocity value for this particle, which will affect the next position of the particle in next iteration (Van den Bergh and Engelbrecht, 2000). The basic PSO procedure is shown in Fig. 1. The advantage of the PSO over many of the other optimization algorithms is its relative simplicity (Van den Bergh, 2001). According to Jones (2005), the only two equations used in PSO are the movement equation (Eq. 1) and velocity update equation (Eq. 2). The movement equation provides for the actual movement of the particles using their specific vector velocity while the velocity update equation provides for velocity vector adjustment given the two competing forces (“gbest” and “pbest”). The inertia weight (ω) was introduced by Shi and Eberhart (1998) to improve the convergence rate of PSO algorithm.

$$presLocation = prevLocation + V_i \Delta t \tag{1}$$

$$V_i = wV_{i-1} + c_1 * rand() * (pbest - presLocation) + c_2 * rand() * (gbest - presLocation) \tag{2}$$

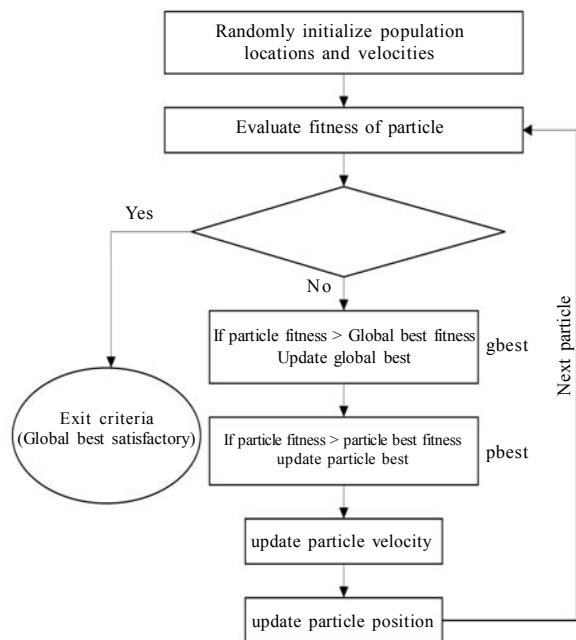


Fig. 1: The basic PSO procedure (Haza, 2006)

Where, V_i is the current velocity, Δt defines the discrete time interval over which the particle will move, ω is the inertia weight, $V_{i,j}$ is the previous velocity, $presLocation$ is the present location of the particle, $prevLocation$ is the previous location of the particle and $rand()$ is a random number between (0, 1), c_1 and c_2 are the acceleration constants for “gbest” and “pbest”, respectively. Particles’ velocities are limited by a user-specified value, maximum velocity V_{max} , to prevent the particles from moving too far from potential solution.

PSO has been successfully applied in various fields. Fukuyama *et al.* (1999) used PSO for reactive power and voltage control considering voltage stability. Gudise and Venayagamoorthy (2003) applied PSO to evolve digital circuits, to solve the problem of the human designs. Gies and Rahmat (2003) applied PSO for configurable phase-differentiated array designed. Besides, Lisa *et al.* (2003) presented an adaptive multimodel biometric fusion algorithm using PSO, which was a combination of Bayesian decision fusion and PSO. Meanwhile, PSO was used as route selection by Rajani and Lisa (2004) and optimal scheduler by Lisa and Kalyan (2004). Besides, Song

and Gu (2004) had detailed the potential performance of PSO on ANNs weights modification, as an alternative to Backpropagation method because of its convenience.

PSO NN Algorithm

In this study, PSO is applied to train a feedforward neural network for enhancing the convergence rate and learning process. The basic element of a NN is a neuron. Each neuron is linked with its neighbors with an associated weight that represents information used by the net to solve a problem. The learning process involves finding a set of weights that minimizes the learning error.

According to Al-kazemi and Mohan (2002), the position of each particle in a PSO NN represents a set of weights for the current iteration. The dimension of each particle is the number of weights associated with the network. The learning error of this network is computed using the mean squared error (MSE) between the observed and simulated runoff. The particle will move within the weight space attempting to minimize learning error.

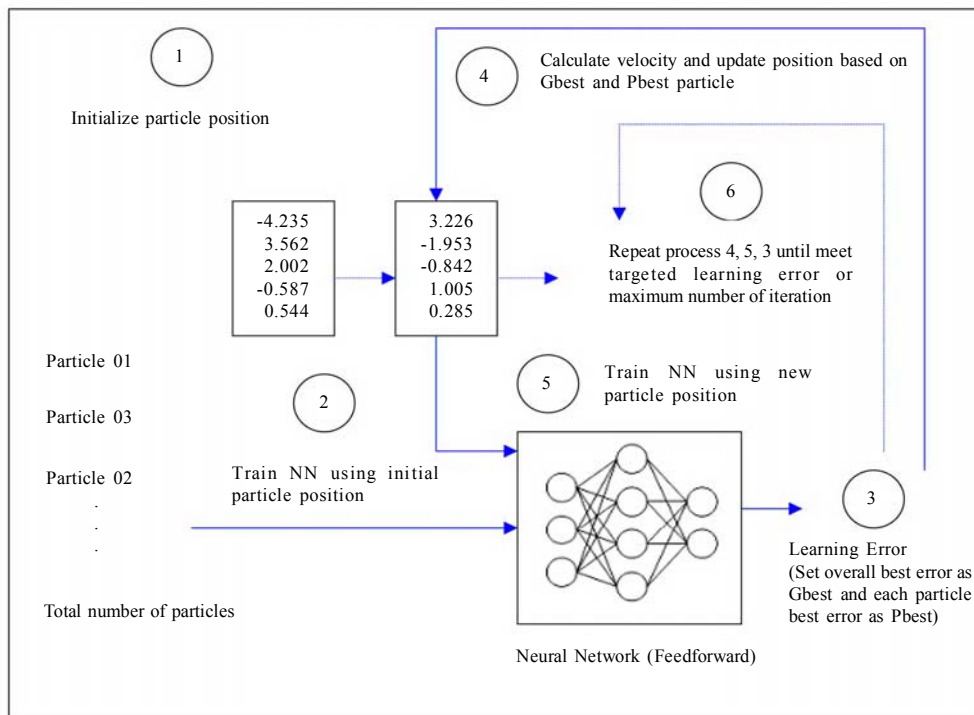


Fig. 2: PSO NN learning process (Van den Bergh, 2001)

Fig. 2 shows the learning process of PSONN. The learning process of PSONN is initialized with a group of random particles (step 1), which are assigned random PSO positions (weight and bias). The PSONN is trained using the initial particles position (step 2). Then, the feedforward NN in PSONN will produce the learning error (particle fitness) based on an initial weight and bias (step 3). The learning error at the current epoch or iteration will be reduced by changing the particles position, which will update the weight and bias of the network.

The “pbest” value (each particle’s lowest learning error so far) and “gbest” value (lowest learning error found in entire learning process so far) are applied to the velocity update equation (Eq. 2) to produce a value for position adjustment to the best solutions or targeted learning error (step 4). The new sets of positions (NN weight and bias) are produced by adding the calculated velocity value to the current position value using the movement equation (Eq. 1). Then, the new sets of positions are used to produce new learning errors for the feedforward NN (step 5). This process is repeated until the stopping conditions, either minimum learning error or maximum number of iteration are met (step 6). The optimization output, which is the solution for the optimization problem, was based on the gbest position value. In this study, PSONN program was developed based on Sombrero function optimization. The PSO particle positions are represented in two-dimensional (2D) vector of x and y values in Sombrero function. The objective is to reach the value of 1 based on value of x and y in Sombrero equation (Eq. 3) and the goal for the PSO (z) is to maximize the function.

$$z = 6 * \cos\left(\frac{\sqrt{x^*x + y^*y}}{x^*x + y^*y + 6}\right) \quad (3)$$

Where, x is value in x -axis, y is value in y -axis, z is value in z -axis. Further, explanation about Sembrero function can be referred to Ashlock (2006).

Study Area

The selected study area is Sungai Bedup Basin, a sub-basin of Sadong Basin, Sarawak, Malaysia. This basin is located approximately 80 km from Kuching City. Model calibration used data series of daily rainfall and observed runoff from year 1997 to year 1999.

The locality plan of Sungai Bedup Basin was presented in Fig. 3. Moreover, Fig. 3a shows the location

of Sadong Basin, which is one of the 22 river basins in Sarawak, Malaysia. Main boundary of the Sadong Basin, rainfall and river stage gauging stations within Sadong Basin, are shown in Fig. 3b. The Bedup Basin is upstream of Batang Sadong, where it is a non-tidal influence river basin. Fig. 3c shows the 5 rainfall gauging stations available in Sungai Bedup Basin, namely, Bukit Matuh (BM), Semuja Nonok (SN), Sungai Busit (SB), Sungai Merang (SM) and Sungai Teb (ST) and one river stage gauging station at Sungai Bedup located at the outlet of the basin.

The basin area is approximately 47.5km² and the elevation varies from 8 m to 686 m above mean sea level (JUPEM, 1975). The vegetation cover is mainly shrub, low plant and forest. Sungai Bedup Basin has a dendritic type channel system. The maximum stream length for the basin is approximately 10 km, which is measured from the most remote point on the stream network to the basin outlet.

The input data used are daily rainfall data from the 5 rainfall stations. Observed daily mean runoff data are converted from water level data through a rating curve given by Eq. (4) (DID, 2004). (4)

$$Q = 9.19(H)^{1.9}$$

Where, Q is the discharge (m³/s) and H is the stage discharge (m). These observed runoff data were used to compare the model runoff.

Models developments

The training and testing processes of the daily rainfall-runoff model are investigated using the PSONN model. An optimal configuration of PSONN is determined. According to Shi (2004), PSONN architecture with a well-selected parameter set can have good performance. This means the combination of parameters will greatly affect the optimization results of PSONN. Hence, various parameters that affect PSONN performance are investigated for searching the best model configuration of PSONN for simulating daily runoff at Sungai Bedup. The performance of PSONN was firstly investigated for four basic parameters in PSO (Jones, 2005). These four basic parameters are important to determine the optimal configuration of SONN and these are:

- a) Acceleration constants for “gbest” ($c1$)
- b) Acceleration constants for “pbest” ($c2$)
- c) The time interval (Δt) constant
- d) The number of particles

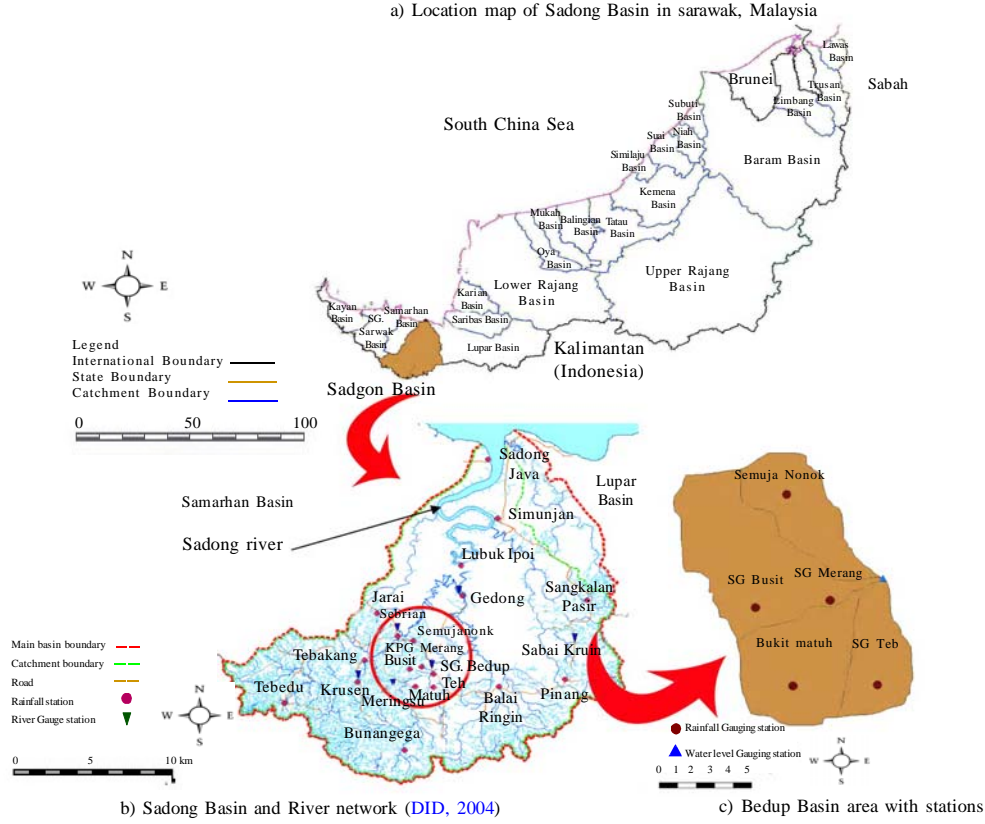


Fig. 3: Locality map of Bedup Basin, Sub-basin of Sadong Basin, Sarawak, Malaysia

Besides, these basic parameters, the effect of other parameters to the optimization results of PSOINN was also investigated for model calibration. These parameters include particle dimension, maximum iteration (stopping condition), number of hidden neurons in the hidden layer, length of input data and number of antecedent days. In PSOINN, the number of dimensions refers to the number of weight and the bias that depends on the training dataset and PSOINN architecture. The particle dimension relies on the number of input neuron, number of hidden neuron in hidden layer and number of output neuron. The dimension of PSOINN is calculated using Eq. 5.

$$\begin{aligned} \text{Dimension} = & (\text{input} * \text{hidden input}) + \\ & (\text{hidden} * \text{output hidden}) \\ & + \text{hidden}_{\text{bias}} + \text{output}_{\text{bias}} \end{aligned} \quad (5)$$

Five models were developed for investigating the effect of the number of antecedent days on the performance of PSOINN. These five different models

are labeled as PSOINN1, PSOINN2, PSOINN3, PSOINN4 and PSOINN5. The input data of the models consists of antecedent rainfall, $P(t-1), P(t-2), \dots, P(t-n)$, antecedent runoff, $Q(t-1), Q(t-2), \dots, Q(t-n)$ and current rainfall, $P(t)$. Whereas, the output is the runoff for the current day, $Q(t)$. The configurations of five models with different number of antecedent days are listed below:

$$\begin{aligned} \text{PSOINN1 model} \\ Q(t) = f[P(t), P(t-1), Q(t-1)] \end{aligned} \quad (6)$$

$$\begin{aligned} \text{PSOINN2 model} \\ Q(t) = f[P(t), P(t-1), P(t-2), Q(t-1), Q(t-2)] \end{aligned} \quad (7)$$

$$\begin{aligned} \text{PSOINN3 model} \\ Q(t) = f[P(t), P(t-1), P(t-2), P(t-3), Q(t-1), Q(t-2), Q(t-3)] \end{aligned} \quad (8)$$

$$\begin{aligned} \text{PSOINN4 model} \\ Q(t) = f[P(t), P(t-1), P(t-2), P(t-3), P(t-4), \\ Q(t-1), Q(t-2), Q(t-3), Q(t-4)] \end{aligned} \quad (9)$$

PSONND5 model

$$Q(t)=f[P(t),P(t-1),P(t-2),P(t-3),P(t-4),P(t-5), Q(t-1),Q(t-2),Q(t-3),Q(t-4),Q(t-5)] \quad (10)$$

Where, t = time (days), P = precipitation (mm), Q = discharge (m^3/s). Eqs. 6, 7, 8, 9 and 10 represent operations to forecast discharge at current day with 1, 2, 3, 4 and 5 days of antecedent data, respectively. The inputs were arranged sequentially as time is one of the important factors in the model.

The rainfall and runoff data are normalized before the PSONN computation is carried out. Normalization will transform the original rainfall and runoff data into the range of 0.001 to 0.999. The equation of normalization method is by dividing all data by the highest data value in the group as shown in Eq. 11.

$$y = \frac{x}{max-x} \quad (11)$$

Where, y is the transform series and x is the original series of rainfall or runoff data. The objective function used is mean squared error. This optimization objective will ensure that MSE or learning error is getting lesser with the increase of number of iteration as the simulated runoff is getting closer to observed runoff. The performance of the PSONN is measured by the ‘coefficient of correlation’ (R) and ‘Nash-Sutcliffe coefficient’ (E^2). These two criterions will measure the overall differences between the simulated and observed runoff. R and E^2 values of 1.0 implies a perfect fit. The formulas of these two coefficients are given in Table 1.

Learning mechanism

The PSONN is composed of three layers, namely, the input layer, the hidden layer and the output layer. The model is investigated with:

- a) 1, 2, 3, 4 and 5 number of antecedent days.
- b) Different $c1$ and $c2$ values ranging from 1.2 to 2.2.

- c) Different time interval constant ranging from 0.0025 to 0.0300.
- d) 16, 18, 20 and 22 number of particles.
- e) Different length of training data from 11 months to 23 months and tested with 4 to 7 months of testing data.
- f) Different max iteration ranging from 300 to 600.
- g) Different number of hidden neuron ranging from 70 to 150.

The daily rainfall and runoff data were used for model calibration. Daily rainfall and runoff data were used for model validation. The model was initially trained with 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 and 23 months of data taken from the period October 1997 to December 1999. The calibrated model was then tested with new data taken January 1996 to September 1997, which is about 1/3 of length of training data.

RESULTS AND DISCUSSION

Many computations were conducted to find the optimal configuration of PSONN. It was found the parameters, including acceleration constants for “gbest” ($c1$), acceleration constants for “pbest” ($c2$), time interval (Δt), number of particle, length of training and testing data, numbers of maximum iterations, number of antecedent days, number of neurons in hidden layer and number of antecedent days to the performance of PSONN are interrelated. The effect of each parameter to PSONN are presented below.

Number of antecedent days

The aim of investigating the effect of the number of antecedent days is to determine the best time series to produce the best network. As shown in Table 2, the performance of PSONND3 consistently yields the highest correlation and efficiency coefficients compared to other PSONN models. This indicates that PSONND3 is the best model for simulating daily runoff in Bedup Basin. The number of antecedent days determines the number of input neurons in input layer.

Table 1: Statistics for model comparison

Coefficient	Symbol	Formula
Coefficient of correlation	R	$\frac{\sum (obs - \bar{obs})(pred - \bar{pred})}{\sqrt{\sum (obs - \bar{obs})^2 \sum (pred - \bar{pred})^2}}$
Nash-Sutcliffe coefficient	E^2	$E^2 = 1 - \frac{\sum_i (obs - pred)^2}{\sum_i (obs - \bar{obs})^2}$

Where, obs = observed value, $pred$ = predicted value, \bar{obs} = mean observed values, \bar{pred} = mean predicted values and j = number of values.

Table 2: The performance of PSONN with different number of antecedent days

Different antecedent days	Training		Testing	
	R	E ²	R	E ²
PSO _{NN} D1	0.680	0.6439	0.797	0.6516
PSO _{NN} D2	0.826	0.7358	0.819	0.7590
PSO _{NN} D3	0.872	0.7754	0.900	0.8067
PSO _{NN} D4	0.773	0.6838	0.795	0.6656
PSO _{NN} D5	0.687	0.7448	0.681	0.6588

Note: PSONN calibrated with $c1$ and $c2$ values of 2.0, 20 number of particles, Δt of 0.01, 21 months of training data, 7 months of testing data, 500 numbers of max iteration and 100 numbers of hidden neurons

The network may not have sufficient degrees of freedom to learn the process correctly when the number of input neurons is small (PSO_{NN}D1 and PSO_{NN}D2). If the number is too high, the network will take a long time to get trained as there are too many parameters to be estimated and may sometimes over fit the data (PSO_{NN}D4 and PSO_{NN}D5).

Acceleration constants $c1$ and $c2$

According to Eberhart and Shi (2001), the acceleration constants $c1$ and $c2$ represent the stochastic acceleration that pulls each particle towards “pbest” and “gbest” positions. The $c1$ constant affects the influence of the global best solution over the particle, whereas the $c2$ constant affects how much influence of the personal best solution has over the particle. The performance of PSONN improved as $c1$ and $c2$ values increased from 1.2 to 2.0. This is because the particles are attracted towards “pbest” and “gbest” with higher acceleration and abrupt movements when $c1$ and $c2$ values increased from 1.2 to 2.0 since $c1$ and $c2$ parameters are determining the particles acceleration. At $c1$ and $c2$ values of 2.2, PSONN was unable to converge. The best $c1$ and $c2$ values for PSONN are 2.0 where the values of R and E² were 0.872 and 0.7754 for training, 0.900 and 0.8067 for testing, respectively. The results for PSONN trained with different $c1$ and $c2$ values are tabulated in Table 3.

Time interval (Δt) constant

The Δt parameter defines the time interval over which movement takes place in the solution space. Decreasing this parameter provides higher granularity movement within the solution space and a higher Δt value performs lower granularity movement (greater distance achieved in less time). The performance of PSONN is increased as Δt increases from 0.0025 to

Table 3: Performance of PSO_{NN}D3 according to different $c1$ and $c2$ Values

$c1$ and $c2$ values	Training		Testing	
	R	E ²	R	E ²
1.2	0.700	0.5871	0.771	0.6638
1.4	0.665	0.7100	0.718	0.6618
1.6	0.626	0.7070	0.827	0.6212
1.8	0.778	0.7088	0.704	0.6454
2.0	0.872	0.7754	0.900	0.8067
2.2	0.646	0.6244	0.756	0.7161

Note: PSO_{NN}D3 trained with 21 months of training data, 7 months of testing data, Δt of 0.01, 20 numbers of particles, 500 max iteration and 100 numbers of hidden neurons

Table 4: Performance of PSO_{NN}D3 according to different time interval (Δt) constant

Δt constant values	Training		Testing	
	R	E ²	R	E ²
0.0025	0.775	0.6340	0.841	0.5894
0.0050	0.814	0.8164	0.781	0.6861
0.0100	0.872	0.7754	0.900	0.8067
0.0150	0.744	0.6780	0.816	0.6891
0.0200	0.639	0.7713	0.859	0.6644
0.0250	0.749	0.6983	0.646	0.6544
0.0300	0.774	0.6284	0.729	0.7357

Note: PSO_{NN}D3 trained with $c1$ and $c2$ values of 2.0, 21 months of training data, 7 months of testing data, 20 numbers of particles, 500 max iteration and 100 numbers of hidden neurons

0.01. PSONN is unable to converge at Δt of 0.0025 due to the high granularity movement within the solution space. Then, the PSONN performance is decreased as Δt decreases from 0.01 to 0.03 due to the low granularity movement within solution space. Results show that the optimal Δt for PSONN in this study is consistently given as 0.01. Table 4 represents the performance of PSONN when trained and tested with different Δt values.

Number of particles

The number of particles in the simulation or swarm represents the amount of space that is covered in the problem. The performance of PSONN is increased with the increase of number of particles from 16 to 20. PSONN is unable to simulate well with 16 particles because the space covered in the problem is not sufficient. Then, at 22 numbers of particles, the PSONN performance starts decreasing (Table 5) due to the space covered in solving the problem being too wide. The best number of particles was found to be 20. It was observed that the optimization period was getting longer with the increase of number of particles. This is because when more particles are presented, the

Table 5: Performance of PSNN according to numbers of particles

Number of Particles	Training		Testing	
	R	E ²	R	E ²
16	0.711	0.6501	0.757	0.7105
18	0.848	0.7325	0.843	0.7229
20	0.872	0.7754	0.900	0.8067
22	0.690	0.6529	0.802	0.6613

Note: PSNN trained with *c1* and *c2* values of 2.0, 21 months of training data, 7 months of testing data, Δt of 0.01, 500 max iteration and 100 numbers of hidden neurons

Table 6: Performance of PSNN according to length of training and testing data investigated

Length of training data	Training		Length of testing data	Testing	
	R	E ²		R	E ²
11 months	0.815	0.6971	3 months	0.820	0.8156
12 months	0.640	0.6711	4 months	0.650	0.6741
13 months	0.687	0.7734	4 months	0.838	0.8010
14 months	0.839	0.7150	4 months	0.851	0.7076
15 months	0.802	0.6078	5 months	0.722	0.6291
16 months	0.779	0.7734	5 months	0.814	0.7928
17 months	0.762	0.6963	5 months	0.772	0.7213
18 months	0.859	0.7575	6 months	0.814	0.7865
19 months	0.811	0.7655	6 months	0.878	0.8639
20 months	0.808	0.7221	6 months	0.831	0.7245
21 months	0.872	0.7754	7 months	0.900	0.8067
23 months	0.800	0.7654	7 months	0.855	0.7652

Note: PSNN calibrated with *c1* and *c2* values of 2.0, 20 number of particles, Δt of 0.01, 500 max iteration and 100 numbers of hidden neurons

Length of training and testing data

The performance of PSNN is increased as the length of training is increased. It was found that a minimum of 21 months of training data was required to achieve best accuracy and yielded R = 0.872 and E² = 0.7754 values for the training data set and R = 0.900 and E² = 0.8067 for validation data set. Table 5 shows that when more data are used, PSNN may performs better (produces higher coefficient results), because a more accurate determination of the synaptic weights is made by the PSNN. However, when the calibration using 23 months of training data, the accuracy of results obtained was not improved compared to 21 months of training data. Therefore, 23 months of training data is not selected in this study. The results of PSNN calibration with different length of training and testing data are tabulated in Table 6.

Number of maximum iteration

The performance of PSNN was also investigated using different numbers of maximum iterations ranging

from 300 to 600. When R is used as the criterion of performance, the best maximum number of iterations obtained was 500 and yielded R=0.872 and R=0.900 for training and testing, respectively. When E² is used as the criterion, the best maximum iteration is 550 with E² = 0.8433 and E² = 0.8222 for training and testing, respectively. The best maximum number of iterations adopted was 500 to avoid over training and over fitting of model. Once over trained, PSNN will only simulate accurately for trained data, but unable simulate for different sets of input data accurately. The number of maximum iterations will determine if a network is well or efficient trained, under trained or over trained. If the maximum number of iterations is not sufficient (from 300 to 450), the network is under trained and unable to approximate any continuous function to any degree of accuracy. In contrast, when too many iterations are used (at maximum iteration of 550 and 600), PSNN may be over trained and sometimes may over fit the data. Table 7 compares the effects of various maximum iterations to PSNN.

Table 7: The performance of PSNN with different maximum iteration

Maximum Iteration	Training		Testing	
	R	E ²	R	E ²
300	0.811	0.6366	0.785	0.7606
350	0.770	0.6777	0.654	0.6176
400	0.872	0.7375	0.858	0.7702
450	0.845	0.7713	0.893	0.8028
500	0.872	0.7754	0.900	0.8067
550	0.822	0.8433	0.840	0.8222
600	0.809	0.7855	0.801	0.7988

Note: PSNN calibrated with *c1* and *c2* values of 2.0, 20 number of particles, Δt of 0.01, 21 months of training data, 7 months of testing data and 100 numbers of hidden neurons.

Table 8: Performance of PSNN according to different number of hidden neurons investigated

Different Hidden Neurons	Training		Testing	
	R	E ²	R	E ²
70	0.776	0.6111	0.826	0.7359
80	0.673	0.6482	0.826	0.6571
90	0.779	0.6940	0.831	0.7960
100	0.872	0.7754	0.900	0.8067
110	0.790	0.7315	0.784	0.7910
120	0.832	0.7309	0.827	0.7281
130	0.800	0.7177	0.797	0.6909
140	0.770	0.7194	0.709	0.6776
150	0.721	0.6653	0.687	0.6653

Note: PSNN calibrated with *c1* and *c2* values of 2.0, 20 number of particles, Δt of 0.01, 21 months of training data, 7 months of testing data and 500 numbers of max iteration

Number of hidden neurons

The investigation initially was started from 10 hidden neurons. However, it was observed that 10 to 60 hidden neurons are unable to produce accurate results. Thus, the simulations reported here have been carried out with 70 to 150 hidden neurons in the hidden layer. Increasing the number of hidden neurons from 70 to 100 increases the accuracy of simulation results as shown in Table 8. However, the performance of PSONND3 is decreased from 125 hidden neurons to 150 hidden neurons. This is because when the number of hidden neurons is small, the network may not have sufficient degrees of freedom to learn the process correctly. In contrast, if the number is too high, the network will take a long time to get trained and may sometimes over fit the data. Moreover, if the hidden layer has too many neurons, then there are too many

parameters to be estimated and this can cause network convergence problems. With sufficient hidden neurons, PSONN is able to approximate any continuous function to any degree of accuracy by performing efficient training. In this study, the optimal PSONN model uses 100 hidden neurons.

Best configuration

PSONN have shown encouraging and promising results in terms of simulating daily runoff. The optimal configuration of PSONN for modeling daily rainfall-runoff relationship was found to be:

- a) 3 antecedent days
- b) $c1$ and $c2$ values of 2.0
- c) time interval of 0.0100
- d) 20 number of particles
- e) 21 months of training data and 7 months of testing data

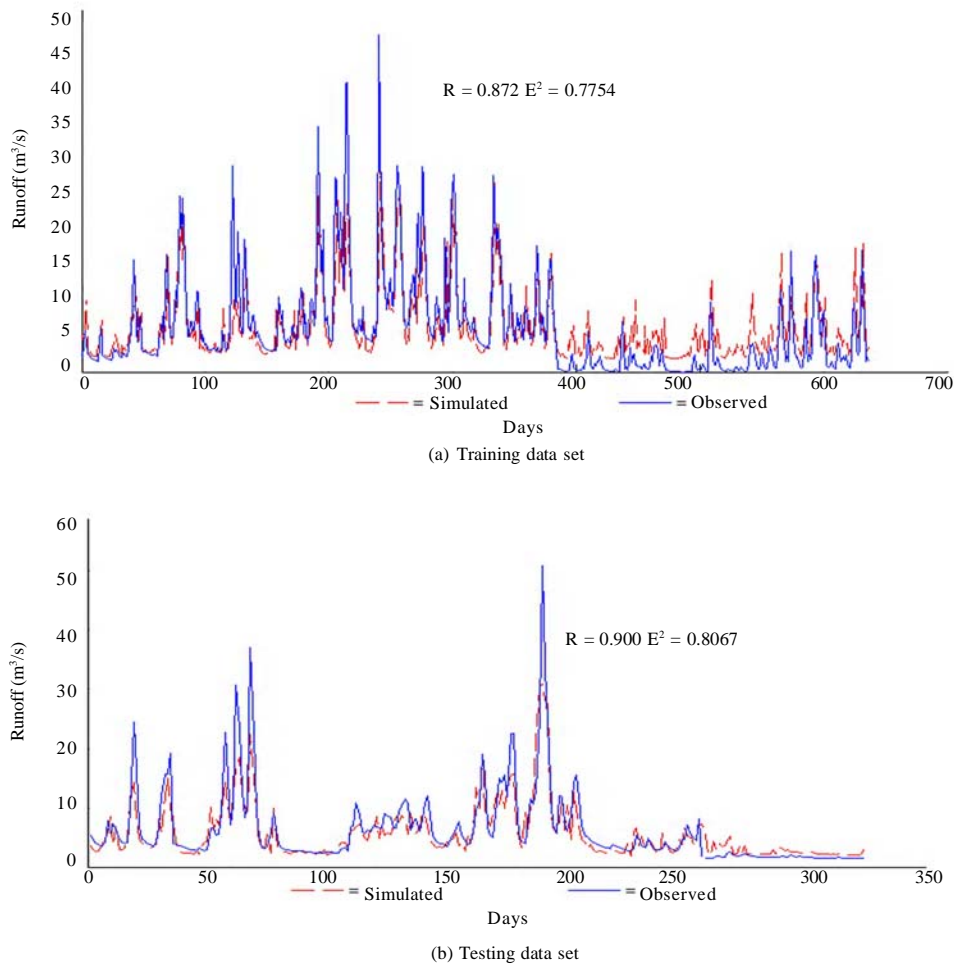


Fig. 4: Comparison between simulated and measured runoff for the optimum PSONN investigation

- f) 500 max iteration
- g) 100 hidden neurons

The comparison between simulated and measured runoff for the optimum PSONN is presented in Fig. 4. The results show the simulated runoff generated by optimum PSONN is successful in approaching the highest peak of measured runoff for both training and testing data. However, the simulated runoff generated by PSONN is less successful in calculating low runoff especially the runoff that are approaching 0 m³/s.

CONCLUSION

A new type of neural network algorithm (PSONN) is successfully developed for solving the optimization problems particularly calibrating the rainfall-runoff model relationship for Bedup Basin, Malaysia. This was revealed by the results where PSONN displays a promising ability to simulate daily runoff accurately. The best model is found to be PSONND3 with the configuration of $c_1=2.0$, $c_2=2.0$, time interval of 0.0100, 20 number of particles, 21 months of training data and 7 months of testing data, 500 max iteration and 100 hidden neurons. The optimal results obtained are $R=0.872$ and $E^2=0.7754$ for training and $R=0.900$ and $E^2=0.8067$ for testing data.

This study shows that it is not necessary to include the lag time as input. The PSONN tested did demonstrate the ability to adapt to the respective lag time of each gauge through training. For catchment in tropical region, rainfall and runoff are sufficient as inputs to develop rainfall-runoff model. Inclusion of more parameters such as temperature, moisture content, evaporation will make the PSONN more complex, the learning time very long and this may decrease the performance of PSONN.

The research for PSONN in hydrology is still in the nascent stage. Basic PSO algorithm was adapted to hybrid with NN in this study. The future research directions of PSONN would be adapting newly developed PSO techniques to hybrid with NN, such as modified dynamic neighborhood PSO algorithm, guarantee convergence PSO algorithm, multi-objective PSO optimization algorithm to enhance the convergence speed and obtain better accuracy simulation results.

REFERENCES

Abbaspour, M.; Rahmani, A. M.; Teshnehlab, M., (2005). Carbon monoxide prediction using novel intelligent network.

- Int. J. Environ. Sci. Tech., 1 (4), 257-264 (8 pages).
- Al-kazemi, B.; Mohan, C. K., (2002). Training feedforward neural network using multi-phase particle swarm optimization. Proceedings of the 9th. International Conference on Neural Information Processing, New York.
- Ashlock, D., (2006). The evolutionary computation for modeling and optimization. Springer. Germany.
- Bandyopadhyay, G.; Chattopadhyay, S., (2007). Single hidden layer artificial neural network models versus multiple linear regression model in forecasting the time series of total ozone. Int. J. Environ. Sci. Tech., 4 (1), 141-150 (10 pages).
- Bessaih, N.; Mah, Y. S.; Muhammad, S. M.; Kuok, K. K.; Rosmina, A. B., (2003). Artificial neural networks for daily runoff simulation. Faculty of Engineering, University Malaysia Sarawak, Charles River Media Inc.
- Bishop, C. M., (1995). Neural networks for pattern recognition. Oxford University Press. Chapter 7, 253-294.
- Bong, S. K.; Bryan, W. K., (2006). Hydraulic optimization of transient protection devices using GA and PSO approaches. J. Water Res. PL-ASCE., 132 (1), 44-52 (10 pages).
- Dastorani, M. T.; Wright, N. G., (2001). Artificial neural network based real-time river flow prediction. School of Civil Engineering, University of Nottingham, Nottingham NG7 2RD, UK.
- DID, (2004). Hydrological Year Book. Department of Drainage and Irrigation Sarawak, Malaysia.
- Eberhart, R.; Hu, X., (1999). Human tremor analysis using particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation, CEC. Washington.
- Eberhart, R.; Shi, Y., (2001). Particle swarm optimization: Developments, application and resources. IEEE., 1, 81-86 (6 pages).
- Elshorbagy, A.; Simonovic, S. P.; Panu, U. S., (2000). Performance evaluation of artificial neural networks for runoff prediction. J. Hydrologic Eng., 5 (4), 424-427 (4 pages).
- Ferguson, D., (2004). Particle swarm. University of Victoria, Canada.
- Fukumaya, Y.; Takamaya, S.; Nakanishi, Y.; Yoshida, H., (1999). A particle swarm optimization for reactive power and voltage control in electric power systems. Proceedings of the Genetic and Evolutionary Computation Conference Orlando, Florida, USA.
- Garcia-Bartual, R., (2002). Short term river flood forecasting with neural networks. Universidad Politecnica de Valencia, Spain, 160-165.
- Gautam, M. R.; Watanabe, K.; Saegusa, H., (2000). Runoff analysis in humid forest catchment with artificial neural networks. J. Hydrol., 235 (1-2), 117-136 (20 pages).
- Gies, D.; Rahmat-Samii, Y., (2003). Particle swarm optimization for reconfigurable phase-differentiated array design. Micro. Opt. Tech. Lett., 38 (3), 168-175 (8 pages).
- Gudise, V. G.; Venayagamoorthy, G. K., (2003). Evolving digital circuits using particle swarm. Proceedings of the International Joint Conference on Neural Networks.
- Harun, S.; Kassim, A. H.; Van, T. N., (1996). Inflow estimation with neural networks. 10th. Congress of the Asia and Pacific Division of the International Association for Hydraulic Research, 150-155.



- Haza, N., (2006). Particle swarm optimization for neural network learning enhancement. M.Sc. Thesis, University Technology of Malaysia.
- Imrie, C. E.; Durucan, S.; Korre, A., (2000). River flow prediction using artificial neural networks: Generalization beyond the calibration range. *J. Hydrol.*, 233, 138-153 (16 pages).
- Jain, S. K.; Chalisgaonkar, C., (2000). Setting up stage-discharge relations using ANN. *J. Hydro. Eng.*, 5 (4), 424-433 (10 pages).
- Jones M. T., (2005). AI application programming. 2nd. Ed. Hingham, Massachusetts.
- JUPEM (1975). Jabatan ukur dan pemetaan Malaysia. Scale 1, 50,000.
- Kennedy, J.; Eberhart, R. C., (1995). Particle swarm optimization. Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE Press. 1942-1948.
- Lee, J. S.; Lee, S.; Chang, S.; Ahn, B. H., (2005). A comparison of GA and PSO for excess return evaluation in stock markets. Springer-Verlag, 221-230.
- Lisa, A. O.; Kaylan, V., (2004). Optimal scheduling in sensor network using swarm intelligence. CISS, Princeton, New Jersey.
- Lisa, A. O.; Veeramachaneni, K.; Varshney, P., (2003). Adaptive multimodel biometric fusion algorithm using particle swarm. Proceedings of SPIE Vol. 5099.
- Nishimura, S.; Kojiri, T., (1996). Real-time rainfall prediction using neural network and genetic algorithm with weather radar data, 10th Congress of the Asia and Pacific Division of the International Association for Hydraulic Research, 204-211 (8 pages).
- Rajani, M.; Lisa, A. O., (2004). Decision making in a building access system using sensor using swarm intelligence and POSets. CISS, Princeton, New Jersey.
- Rene, E. R.; Kim, J. H.; Park, H. S., (2008). An intelligent neural network model for evaluating performance of immobilized cell biofilter treating hydrogen sulphide vapors. *Int. J. Environ. Sci. Tech.*, 5 (3), 287-296 (9 pages).
- Shi, Y., (2004). Particle swarm optimization. IEEE Neural Network Society: 8-13.
- Shi, Y.; Eberhart, R. C., (1998). A modified particle swarm optimizer. Proceedings of the 105 IEEE Congress on Evolutionary Computation, 69-73.
- Song, M. P.; Gu, G. H., (2004). Research on particle swarm optimization: A Review. Proceedings of the 3rd. International Conference on Machine Learning and Cybernetics. Shanghai, China.
- Tokar, A. S.; Johnson, P. A., (1999). Rainfall-runoff modeling using artificial neural networks. *J. Hydro. Eng.*, 4 (3), 223-239 (17 pages).
- Van den Bergh, F.; Engelbrecht, A. P., (1999). Particle swarm weight initialization in multi-layer perceptron artificial neural networks. ICAI. Durban, South Africa.
- Van den Bergh, F.; Engelbrecht, A. P., (2000). Cooperative learning in neural networks using particle swarm optimizers. *S. Afr. Comput. J.*, 26, 84-90 (7 pages).
- Van den Bergh, F., (2001). An analysis of particle swarm optimizers. Ph.D dissertation, University of Pretoria. South Africa.
- Wright, N. G.; Dastorani, M. T., (2001). Effects of river basin classification on artificial neural networks based ungauged catchment flood prediction, Proceedings of the 2001 International Symposium on Environmental Hydraulics.
- Zhang, C.; Shao, H.; Li, Y., (2000). Particle swarm optimization for evolving artificial neural network. IEEE, 2487-2490 (4 pages).
- Zweiri, Y. H.; Whidborne, J. F.; Sceviratne, L. D., (2003). A three-term backpropagation algorithm. *Neurocomputing*, 50, 305-318 (14 pages).

AUTHOR (S) BIOSKETCHES

Kuok, K. K., Ph.D. Candidate, Department of Hydraulics and Hydrology, Faculty of Civil Engineering, University Technology Malaysia, 81310 UTM, Johor, Malaysia. Email: kkuok100@yahoo.com.sg

Harun, S., Ph.D., Associate Professor, Department of Hydraulics and Hydrology, Faculty of Civil Engineering, University Technology Malaysia, 81310 UTM, Johor, Malaysia. Email: sobriharun@gmail.com

Shamsuddin, S. M., Ph.D., Full Professor, Department of Computer Graphics and Multimedia, Faculty of Computer Science and Information System, University Technology Malaysia, 81310 UTM, Johor, Malaysia. Email: mariyam@utm.my

How to cite this article: (Harvard style)

Kuok, K. K.; Harun, S.; Shamsuddin, S. M., (2009). Particle swarm optimization feedforward neural network for modeling runoff. Int. J. Environ. Sci. Tech., 7 (1), 67-78.

