River Flow 2010 - Dittrich, Koll, Aberle & Geisenhainer (eds) - © 2010 Bundesanstalt für Wasserbau ISBN 978-3-939230-00-7

Uncertainty study of data-based models of pollutant transport in rivers

Adam P. Piotrowski, Pawel M. Rowinski & Jarosław J. Napiorkowski
Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, 01-452 Warsaw, Poland

ABSTRACT: This study concerns the problem of pollution transport in rivers in situations when data is scarce and no results from tracer tests are available. In such case approaches pertaining to numerous empirical formulae or data-based methods allowing estimation of the longitudinal dispersion coefficient based on hydraulic and morphometric characteristics are used. Different methods were proposed so far depending on number of available river characteristics like nonlinear regression, neural networks or transfer function. However, they are usually lacking the uncertainty analyzes. Moreover, the impact of the used optimization method on the results has not been studied in detail. In the present paper we investigate the uncertainty of the well-known regression based equation and neural networks models and assess performance of neural networks trained with different techniques. We restrict our considerations to well studied river reaches when most important river characteristics, including water velocity, shear velocity, bed slope, depth, width and sinuosity are available for each river reach of interest.

Keywords: Pollutant transport, Longitudinal dispersion, Uncertainty, Neural networks, Differential evolution

1 INTRODUCTION

The simplest and most popular approach of one dimensional pollutant transport modelling in engineering practice is that based on the advection–dispersion equation (Taylor, 1953; Fischer et al., 1979):

$$\frac{\partial \overline{C}}{\partial t} + U\frac{\partial \overline{C}}{\partial x} = \frac{1}{A}\frac{\partial}{\partial x}\left(AE_L \frac{\partial \overline{C}}{\partial x} \right) \tag{1}$$

where x is the longitudinal axis, t denotes time, $\overline{C}$ is the admixture concentration averaged over the cross-section A, E_L is the longitudinal dispersion coefficient and U is cross-sectional averaged velocity. Great effort has been performed to develop empirical methods allowing practitioners to estimate longitudinal dispersion E_L at particular river reach without performing time consuming and expensive tracer tests. Initially, mostly the regression equations were considered as a predictive tool – probably the most efficient ones were proposed by Seo and Cheong (1998)

$$E_L^{\,P} = HU^* 5.915\left(\frac{U}{U_*}\right)^{1.428}\left(\frac{B}{H}\right)^{0.62} \tag{2}$$

and Deng et al. (2001)

$$E_L^{\,P} = HU^* \frac{0.15}{8k}\left(\frac{B}{H}\right)^{5/3}\left(\frac{U}{U^*}\right)^2$$
$$k = 0.145 + \left(\frac{1}{3520}\right)\left(\frac{U}{U_*}\right)\left(\frac{B}{H}\right)^{1.38} \tag{3}$$

In both equations B is the channel width, H is the mean depth of flow, U^* is the shear velocity and $E_L^{\,P}$ is predicted dispersion coefficient. More detailed overview of the proposed equations may be found in Wallis and Manson (2004). Over the last 5 years neural networks have been given much attention for this task (Kashefipour et al. 2002; Rowinski et al., 2005; Tayfur and Singh, 2005; Piotrowski et al. 2006; Toprak and Cigizoglu, 2008; Riahi-Madvar et al. 2009). However, neural networks are usually applied somehow automatically as a predictive tool – without performing an uncertainty analysis, discussion on the proper optimization criteria or verification of the optimization algorithms used to find the proper weights of the network. These issues cannot be considered as technical details only (Rimer and Martinez, 2006). In the present paper three Evolu-

tionary Computation optimization methods belonging to so called Differential Evolution family are applied in order to optimize the simple multi-layer perceptron neural network with objective function designed specially to allow fitting the model to data which differ by a few orders of magnitude and consider input variables uncertainty into optimization process. This study concentrates on the impact of the optimization method on the results and shows the significant uncertainty connected with longitudinal dispersion estimation.

2 DATA

The data set used in the present study is composed of 81 experiments, 70 of them were collected by Deng et al. (2002) and 11 by Sukhodolov et al. (1997). Experiments were performed by various researchers, mostly in moderate climatic conditions of USA and Moldova. More detailed description of most of experiments performed in USA may be found in Nordin and Sabol (1974) and Godfrey and Frederick (1970).

It was verified that for almost every experiment the distance to the location where complete horizontal mixing over the river cross-section takes place (Jirka and Weitbrecht, 2005) is shorter than the distance to the first cross-sections where the measurements were taken.

To estimate the value of longitudinal dispersion, the five independent variables were used in this paper, namely B, H, U, U^* and river sinuosity (sin), hence $E_L^P = f(B,H,U,U^*,sin)$. All these variables were available for the considered river reaches. 50 out of 81 experiments were used as a training set to optimize neural networks and the remaining 31 were considered as testing set.

Testing examples were experiments number 3, 4, 6, 11, 13-15, 19, 20, 24, 26, 30, 37, 40, 41, 44, 46, 50, 51, 53, 54, 60-63, 67, 70 from Deng et al. (2002) and reach numbers 2, 3, 6 (with $Q = 0.465$ only) and 9 from Sukhodolov et al. (1997). Note that although there were 15 experiments in total mentioned in Sukhodolov et al. (1997), 4 of them were excluded from considerations in the present paper since they were performed in almost identical conditions at the same cross-sections. Sinuosity of river reaches were not reported in Sukhodolov et al. (1997) – these were measured by authors of the present paper based on available maps (see Rowinski et al. (2005).

3 MULTI-LAYER PERCEPTRON ARTIFICIAL NEURAL NETWORKS

From a number of types of neural networks, the supervised multi-layer perceptron feed-forward network (Haykin, 1999) is probably the most extensively applied to variety of problems. They approximate the values of output variable (y) based on the set of input variables $x_1, x_2, ..., x_K$. Usually all input and output variables are standardized to the [0,1] interval. To avoid the possibility of obtaining negative values of predicted dispersion, we set $y = ln(E_L)$ before standardization (see Kashefipour et al., 2002 and Rowinski et al., 2005). Artificial neural networks are comprised of nodes arranged usually in three layers – input, hidden and output one. The example of neural network topology is presented in Fig. 1. The number of input nodes is the same as the number of input variables, the number of hidden neurons should be found empirically for a

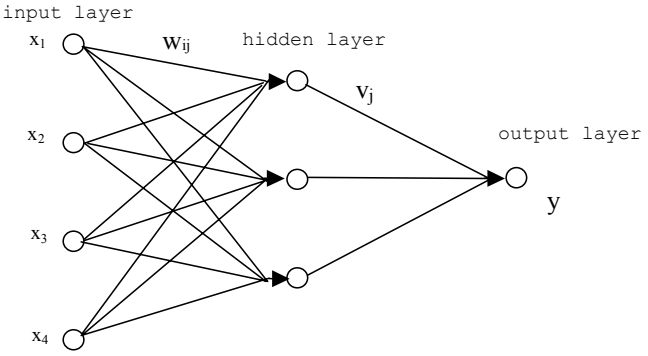

Figure. 1 Scheme of multi-layer perceptron neural network scheme.

given problem. The nodes are linked via weighted connections $\boldsymbol{w}$ and $\boldsymbol{v}$. The values of these connections are adaptively modified during the process of training the network. Each node performs a weighted sum of its inputs and filters it through a given, so-called activation function. Following a number of other authors, a sigmoidal function was used for this purpose in the hidden layer, i.e.:

$$f(z) = \frac{1}{1+\mathrm{e}^{-z}} \tag{4}$$

where z is the weighted sum

$$z_j = f\left(\sum_{i=1}^{K} w_{ji} x_i\right) \tag{5}$$

Afterwards the weighted signals z_j (multiplied by proper weights v_j), are transferred to the neuron of the third layer, where the new weighted sum is computed:

$$y^P = \left(\sum_{j=1}^{L} v_j f\left(\sum_{i=1}^{K} w_{ji} x_i\right)\right) \tag{6}$$

when L is the number of hidden nodes. Finally, after re-standardization of y^P, the predicted longitudinal dispersion value may be obtained as $E_L^P(\boldsymbol{w},\boldsymbol{v}) = \exp(y^P)$.

Because of small number of training data (50 experiments only) and 5 input variables we were forced to reduce the number of hidden nodes to 2. It resulted in neural network structure of 5 inputs, 2 hidden units and 1 output (5-2-1) – leaving 15 parameters for optimization process.

4 SELECTION OF OBJECTIVE FUNCTION

The parameters $\boldsymbol{v}$ and $\boldsymbol{w}$ have to be optimized based on the knowledge gained from training examples according to the assumed criterion. It is popular to use the objective function to be minimized such as the mean square error

$$J_1(\mathbf{w},\mathbf{v}) = \min_{w,v} \sum_{i=1}^{m} (E_{L\,i}^{P}(\mathbf{w},\mathbf{v}) - E_{Li})^2 \tag{7}$$

which is differentiable and has good statistical properties. Unfortunately it has a major drawback – the weights $\boldsymbol{w}$ and $\boldsymbol{v}$ would be fitted to properly reproduce the highest values of longitudinal dispersion in the training set. When dealing with dispersion coefficients which frequently differ by 4 orders of magnitude (see database in Deng et al. 2001) – it is very unfortunate behaviour. One may see that sometimes the models are really fitted to the highest examples only, what results in outstandingly poor estimation of all lower dispersion values (as in Tayfur, 2006). The objective function in the form of

$$J_2(\mathbf{w},\mathbf{v}) = \min_{w,v} \sum_{i=1}^{N} \left| E_{L\,i}^{P}(\mathbf{w},\mathbf{v}) - E_{Li} \right| \tag{8}$$

only partly mitigates this effect. One may of course set a criterion of

$$J_3(\mathbf{w},\mathbf{v}) = \min_{w,v} \sum_{i=1}^{N} (y^{P}{}_{i} - y_i)^2 \tag{9}$$

as for example in Kashefipour et al. (2002) and Rowinski et al. (2005), what makes the impact of different examples more even, but unfortunately, due to nonlinear relation between y and E_L, frequently $\boldsymbol{min\ J_3\ (w,v) \neq min\ J_1\ (w,v)}$ which means that we perform optimization of neural network not exactly to the task we are really interested in. This is why we propose yet another objective function J_4 which is defined as:

$$J_4(\mathbf{w},\mathbf{v}) = \sum_{i=1}^{N} \left(\exp \left| \ln \left(\frac{E_{L\,i}^{P}(\mathbf{w},\mathbf{v})}{E_{Li}} \right) \right| \right)^2 \tag{10}$$

$$E_{Li} > 0$$

which was especially designed for cases when all examples are of similar importance, no matter how many orders of magnitude they span. This expression takes into account the degree of over- or underestimation of the measured values – the predictions 10 and 0.1 when true value is 1, as well as 3000 and 30 when true value is 300 are similarly wrong. The second power is taken in order to increase the significance of over- or underestimation.

5 INCLUDING UNCERTAINTY INTO OPTIMIZATION PROCESS

Although a series of papers have dealt with the application of neural networks to estimate the longitudinal dispersion coefficients, to the best of our knowledge, none of them considered uncertainty analysis. Simple analysis of the sensitivity of the model results in 10% increase of each of independent input variables were performed for equation (3) in Deng et al (2001). Helton et al (2006) suggests to perform uncertainty analysis by generating a sample of input variables and propagating it through a model to estimate the model uncertainty. The sample generation should be, if possible, based on expert suggestions. Another options is the generation from an assumed distribution either randomly or by means of Latin Hypercube method. The second method is suggested for computationally demanding models to limit the number of the generated samples, which is not important in the case of simple neural networks.

It is difficult to define the uncertainty of E_L, B, H, U and *sin* – as it depends mostly on a quality of on site measurements, performed by different researchers in various conditions. Hence, in this paper it is assumed that these 5 variables are realizations from the normal distribution with $\mu_{ij} = u_{ij}$ and $\sigma_{ij} = 0.1u_{ij}$, where $u_{i1} = y_i$, $u_{i2} = x_{i1}, \ldots, u_{i5} = x_{i4}$; $i = 1, N$ represent each training example and $j = 1, 5$ represent input or output variables: E_L, B, H, U or *sin*. The coefficient of 0.1 used to define the variance was estimated empirically. The smaller σ_{ij} indicates that the variable is less uncertain. If this value was set as too small it would result in limiting the neural network robustness, whereas too high would make prediction very poor. Note that the true distribution of a particular variable at each experiment may differ from the Gaussian one and the observed u_{ij} may even be an outlier far from the true μ_{ij}. However, we cannot find proper distributions since the only information we have is single realization. Note that shear velocity U^* not considered above may be approximated by

$$U^* = \sqrt{gHs} \tag{11}$$

where s is the bed slope. It seems reasonable to assume that the bed slope – which is usually measured rather than U^* – is a realization from normal distribution with $\mu_{ij} = s_{ij}$ and $\sigma_{ij} = 0.1s_{ij}$.

In the present paper at each iteration, any variable for each individual (see section 6) and for every experiment was generated 100 times from pre-defined distributions. Together with the measured values it gave 101*50 training examples. All of them were used for the determination of value of objective function (Eq. (10)) of a particular individual at the given iteration. This process is similar to the approach that is called noise injection (jitter) in neural network literature (Bishop 1995; Reed et al. 1995; Zhang 2007) and allows for including the uncertainty of the measured training examples into optimization process improving the neural network robustness.

6 OPTIMIZATION ALGORITHMS BASED ON DIFFERENTIAL EVOLUTION

In most of the applications the gradient-based methods are used to train the neural network – probably the most popular are simple gradients, conjugant gradients (Haykinn, 1999) and the Levenberg-Marquardt algorithms (Press et al. 1990). Such methods were also used in previous papers applying neural networks to dispersion assessment. However, they are only local search methods which require many independent runs. Different runs usually provide very different results depending on the initial parameter values – many of them are outstandingly poor when the algorithm sticks in a shallow local optimum. They also require a differentiable objective function, what is very restrictive in many cases. Most evolutionary computation (EC) techniques do not suffer from such drawbacks and, as long as the dimensionality of the problem is not too high, almost always provide reasonable results. Examples of EC methods recently used for neural network training may be found in Sedeki et al. (2009), Lin et al. (2009) and Aliev et al. (2009). Among a variety of methods we chose three algorithms belonging to Differential Evolution (DE) family – the basic DE method (Storn and Price, 1995), Self Adaptive DE (SaDE, Qin et al., 2009) and Grouped Multi-Strategy DE (GMS, Piotrowski and Napiorkowski, 2010). The basic DE method has been applied to neural network training several times, for example by Illonen et al. (2003), Yu and He (2006), Jordanov and Georgieva (2007), Rowinski and Piotrowski (2008). The remaining two algorithms are recent modifications which, until now have been tested only on artificial test functions.

In all DE-based methods (in fact – most EC ones) to find the optimal solution, the population of P individuals $\boldsymbol{p}_i$ (represented by vectors of M parameters which are to be optimized) is searching for the optimum in the M-dimensional space. There are different opinions about the relation between P and M. Storn and Price (1995) suggested $P = 10M$, but later various researchers, including Weber et al. (2009) suggested smaller populations. Since the number of parameters in 5-2-1 neural network is 15, we set P to 60 for all 3 algorithms (DE, SaDE and GMS). Usually EC techniques need some stopping criteria – it may be the maximum number of the objective function calls, lack of sufficient improvement during the predefined period or more complex stopping criteria (used in GMS). In this paper maximum number of function calls was used and set to a relatively high value of 600000. In GMS, apart from maximum number of iterations, the stopping technique proposed by Piotrowski and Napiorkowski (2010) was used to cease the algorithm when the population structure does not justify further search.

In basic DE, at each iteration, for each of $i = 1, \ldots, P$ individuals ($\boldsymbol{p}_i$), three other distinct points ($\boldsymbol{p}_a$, $\boldsymbol{p}_b$, and $\boldsymbol{p}_c$) are randomly chosen. The new point ($\boldsymbol{p}_{\mathbf{new}}$) is generated, according to the strategy described as a weighted (w) perturbation of the location of point $\boldsymbol{p}_c$:

$$\mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{c}} + F(\mathbf{p}_{\mathbf{a}} - \mathbf{p}_{\mathbf{b}}) \tag{12}$$

The value of F is the parameter of the algorithm, suggested by Storn and Price (1995) to be 0.8. Then, in the basic DE, for each of $j = 1, \ldots, M$ elements separately, the crossover of vectors $\boldsymbol{p}_i$ and $\boldsymbol{p}_{\mathbf{new}}$ is performed, yielding the final offspring $\boldsymbol{p}_{\mathbf{off}}$. The probability of crossover (CR) – in other words probability of adopting the location for element j from $\boldsymbol{p}_{new}$ point – may vary depending on the problem, but 0.5 is an average, usually successful choice. In DE terminology the strategy (12) with such binominal crossover is called DE/rand/1/bin. The objective function value of $\boldsymbol{p}_{\mathbf{off}}$ is compared with the objective function value of $\boldsymbol{p}_i$. Only the better one is moved to the next iteration.

In SaDE, instead of only one strategy, four different strategies are used together, each one with adaptively changed probability. They include:

1. the basic strategy DE/rand/1/bin,
2. DE/target-to-best/2/bin, where $\boldsymbol{p}_{\mathbf{BEST}}$ is the best individual in population

$$\begin{aligned}\mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{i}} + F\cdot(\mathbf{p}_{\mathbf{BEST}} - \mathbf{p}_{\mathbf{i}}) + \\ F\cdot(\mathbf{p}_{\mathbf{a}} - \mathbf{p}_{\mathbf{b}}) + F\cdot(\mathbf{p}_{\mathbf{c}} - \mathbf{p}_{\mathbf{d}})\end{aligned} \tag{13}$$

3. DE/rand/2/bin, where $\boldsymbol{p}_d$, and $\boldsymbol{p}_e$ are two another distinct individuals from population

$$\begin{aligned}\mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{a}} + F\cdot(\mathbf{p}_{\mathbf{b}} - \mathbf{p}_{\mathbf{c}}) + \\ F\cdot(\mathbf{p}_{\mathbf{d}} - \mathbf{p}_{\mathbf{e}})\end{aligned} \tag{14}$$

4. and DE/current-to-rand/1/none (no crossover), where $F2$ is an additional scaling factor, each time

generated anew from standardized normal distribution

$$\mathbf{p}_{\mathbf{off}} = \mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{i}} + F2 \cdot (\mathbf{p}_{\mathbf{a}} - \mathbf{p}_{\mathbf{i}}) + F \cdot (\mathbf{p}_{\mathbf{b}} - \mathbf{p}_{\mathbf{c}}) \quad (15)$$

SaDE allows for adaptation of probability of choosing each strategy by particular individual at the given iteration according to the success rates obtained during a number of previous generations. Similarly, also *CR* parameter is adapted according to the developed procedure and *F* values are generated from normal distribution with 0.5 mean and 0.3 standard deviation. For more details we refer the reader to Qui et al. (2009). In our application we followed the parameter setting and adaptation methods developed in that paper.

An outline of GMS algorithm is as follows. It also uses three strategies:
1. basic DE/rand/1/bin,
2. DE/rand/1/exponential – which is basic DE strategy but with exponential crossover (for detailed definition see Price et al., 2005 and Piotrowski and Napiorkowski, 2010) and
3. DE/rand/1/Mod-either-or:

$$p_{off}^{j} = \begin{cases} p_{new}^{j} \quad from \quad Eq.(1.12) \quad if \quad rand[0,1]_j \le CR \\ p_{new}^{j} = p_i^j + (p_b^j + p_c^j - 2 \cdot p_i^j) \cdot RN(0,1) \quad otherwise \end{cases} \quad (16)$$

where $RN(0,1)$ is random value generated from standardized normal distribution and $rand[0,1]_j$ is random number from uniform [0,1] distribution generated for each parameter j ($j = 1, M$) separately. In this strategy the parent $\boldsymbol{p_i}$ does not take part in the creation of an offspring. The parameters *F* and *CR* are both equal 0.5, independent of the strategy.

In GMS algorithm population is divided into 4 groups, 3 of them are most of the computing time searching for the optimum separately (what means that the individuals $\boldsymbol{p_a}$, $\boldsymbol{p_b}$, and $\boldsymbol{p_c}$ are chosen only from individuals belonging to the same group as a parent $\boldsymbol{p_i}$) when the fourth one gains the knowledge from the entire population. The specific conditions are defined when individuals from one of 3 separate groups may learn from the whole population, and the idea of "freezing" some of the best individuals is introduced ("frozen" individuals are not considered as parents – hence they cannot be lost from population but may be chosen as $\boldsymbol{p_a}$, $\boldsymbol{p_b}$, and $\boldsymbol{p_c}$). For the detailed description and pseudocode of the algorithm, see Piotrowski and Napiorkowski (2010).

To perform a fair comparison, 20 neural networks were optimized by each algorithm, starting from the randomly selected weight values within the [-1,1] interval. The bound constrains for optimized weights were set to [-1000,1000], rebounding procedure was applied when an algorithm proposed a solution outside the bound constrains (Piotrowski and Napiorkowski, 2010).

7 RESULTS

The best (MIN), average (AVE) and worst (MAX) values of objective function (10) obtained from 20 neural networks trained by each algorithm are presented in Table 1. One may easily conclude that networks are better trained by GMS and SaDE algorithms than by the basic DE method. Difference between GMS and SaDE algorithm is rather small, especially comparing to the average results. However, GMS is slightly better and faster. While SaDE always performed 600000 iterations, GMS stopped after 150000-575000 iterations, with average of 285500.

Table 1. The performance of neural networks trained by each algorithm: TR – training set, TE – testing set.

Obj	DE		SaDE		GMS	
funct	TR	TE	TR	TE	TR	TE
MIN	377.7	828.6	298.7	757.0	293.3	672.7
AVE	425.7	1138.9	314.1	867.9	309.1	837.2
MAX	515.1	1719.4	390.6	1148.2	332.1	1063.5

It is clear that the criterion values for testing set are about 3 times higher than those obtained for training examples. The reason may be seen in Fig 2, which presents uncertain longitudinal coefficient values predicted (101 black cases) and measured (101 grey cases) for 31 testing data by the example of moderately performed (objective function value J_4 for: training set – 304, testing set – 792) network trained by GMS algorithm. These plots would differ for different neural networks trained by GMS or SaDE algorithms and the relation between the uncertain predicted and the measured dispersion values would be poorer for basic DE method, but general appearance would remain the same.

It seems that 5 major issues may be noted from the plots 1) predicted uncertain values are almost always more scattered than the measured ones, 2) in most cases substantial part of the measured values are within an interval covered by the predicted ones but 3) for 6 cases all the predicted values are over- or underestimated, 4) in 2 cases (number 9 and 11) the under- or over estimation is very significant, and finally 5) the quality of longitudinal dispersion estimation does not depend on the order of magnitude of longitudinal dispersion.

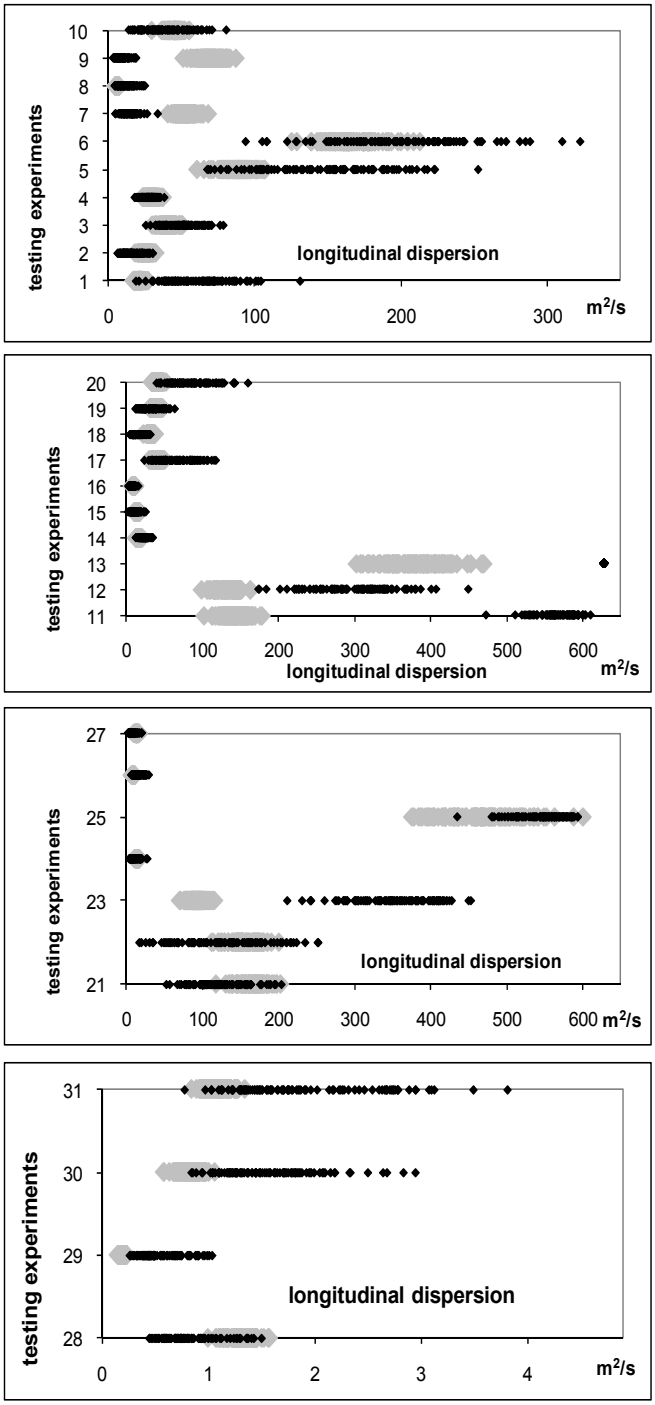

Figure 2. Uncertain longitudinal dispersion coefficient values predicted by neural network (black cases) and measured (grey cases) for testing data collected in 31 rivers.

The mentioned 2 very poorly predicted testing cases (nr 9 and 11) contribute significantly to the poorer objective criterion for testing data. Of course the precise number of poor cases may differ depending on the used neural network and the training algorithm, but always some testing examples are over- or underestimated.

Comparing the neural network results with the one obtained from the regression equations (2) and

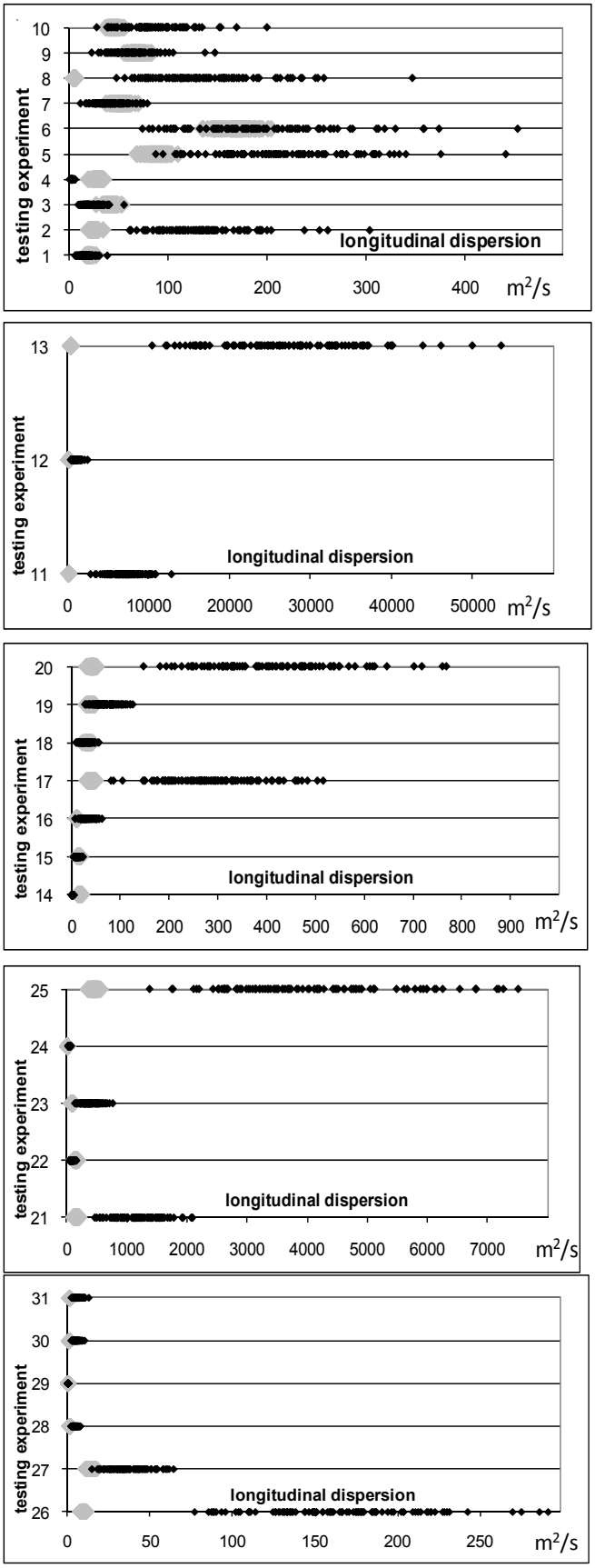

Figure 3. Uncertain longitudinal dispersion coefficient values predicted from Deng et al (2001) equation (black cases) and measured (grey cases) for testing data collected in 31 rivers.

(3) shown in Table 2, one may see some improvement. Both equations make constant error of overestimation of the longitudinal dispersion. Figure 2 presents the uncertain longitudinal disper-

sion coefficients estimated from Deng et al. (2001) equation. The significant difference with the neural network prediction trained for uncertain data is obvious.

Table 2. The performance of empirical equations with dividion into two sets like in Table 1: TR – training set, TE – testing set.

	Seo Cheong		Deng	
	TR	TE	TR	TE
Obj funct	1368292.4	135865.5	88039.2	33665.6

8 CONCLUDING REMARKS

The final results seem to restrained optimistic. Although properly trained neural networks may significantly outperform the well known regression equations from the literature, the confidence in their results remains limited. It is an open question if it is due to very limited number of hidden nodes. We performed the experiments with 3 hidden nodes and they did not provide reduction of error for testing data.

REFERENCES

Aliev, R.A., Guirimov, B.G. Fazlollahi, B., Aliev, R.R. 2009. Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: Recurrent fuzzy neural networks. Fuzzy sets and systems 160, 2553-2566.

Bishop, C.M. 1995. Training with Noise is Equivalent to Tikhonov Regularization. Neural Computation 7(1), 108-116

Deng, Z.Q., Singh, V.P., Bengtsson, L. 2001. Longitudinal dispersion coefficient in straight rivers. J. Hydraul. Engng. ASCE 127(11), 919-927.

Deng, Z.Q., Bengtsson, L., Singh, V.P. 2002. Longitudinal dispersion coefficient in single-channel streams. J. Hydraul. Engng. ASCE 128(10), 901–916.

Fischer, H.B., List, E.J., Koh, R.C.Y., Imberger, J., Brooks, N.H. 1979. Mixing in inland and coastal waters. Academic, New York, N.Y., 104-138

Godfrey, R.G., and Frederick, B.J. 1970. Stream dispersion at selected sites. U.S.Geological Survey Professional Paper, 433–K.

Haykin, S. 1999. Neural Networks, a Comprehensive Foundation. Macmillan College Publishing Co., New York, USA.

Helton, J.C., Johnson, J.D., Sallaberry, C.J., Storlie, C.B. 2006. Survey of sampling-based methods for uncertainty and sensitivity analysis. Reliability Engineering and System Safety 91, 1175-1209.

Ilonen, J., Kamarainen, J. K., Lampinen, J. 2003. Differential Evolution training algorithm for feed-forward neural networks. Neural Processing Letters 17, 93–105.

Jirka, G.H., Weitbrecht, V. 2005. Mixing Models for Water Quality Management in Rivers: Continuous and Instantaneous Pollutant Releases. In: Water Quality Hazards.

Jordanov, I., Georgieva, A. 2007. Neural network learning with global heuristic search. IEEE Transactions on Neural Networks 18(3), 937-942.

Kashefipour Kashefipour, S. M., Falconer, R. A., Lin, B. 2002. Modeling longitudinal dispersion in natural channel flows using ANNs. In: River Flow 2002 (ed. By D. Bousmar & Y. Zech), 111–116. A.A. Balkema/Swets & Zeitlinger, Lisse, The Netherlands.

Lin, S.W., Chen, S.S., Wu, W.J., Chen, C.H. 2009. Parameter determination and feature selection for back-propagation network by particle swarm optimization. Knowl. Inf. Syst. 21, 249-266.

Nordin, C.F., Sabol, G.V. 1974. Empirical data on longitudinal dispersion. US Geol. Survey Water Resour. Investigations 20–74.

Piotrowski, A., Rowiński, P.M., Napiórkowski, J.J. 2006. Assessment of longitudinal dispersion coefficient by means of different neural networks. In: 7th Int. Conf. on Hydroinformatics, HIC 2006, Nice, France, (ed by P. Gourbesville, J. Cunge, V. Guinot & S. Y. Liong), Research Publishing.

Piotrowski, A.P. Napiorkowski, J.J. 2010. Grouping differential evolution algorithm for multi-dimensional optimization problems. Control and Cybernetics – accepted for publication.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. 1990. Numerical Recipes in C. The Art of Scientific Computing. Cambridge University Press, Cambridge, UK.

Qin, A.K. Huang, V.L., Suganthan, P.N. 2009. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. IEEE Transactions on Evolutionary Computation 13(2), 398-417.

Reed, R., Marks, R.J. and Oh, S. 1995. Similarities of Error Regularization, Sigmoid Gain Scaling, Target Smoothing and Training with Jitter. IEEE Transactions on Neural Networks 6(3), 529-538.

Riahi-Madvar, H., Ayyoubzadeh, S.A., Khadangi, E., Ebadzadeh, M.M. 2009. An expert system for predicting longitudinal dispersion coefficient in natural streams by using ANFIS. Expert Ststems with Applications 36(4), 8589-8596.

Rimer, M. Martinez, T. 2006. Classification based objective functions. Machine learning 63(2), 183-2005.

Rowinski, P.M., Piotrowski, A., Napiórkowski, J.J. 2005. Are artificial neural network techniques relevant for the estimation of longitudinal dispersion coefficient in rivers?. Hydrol. Sci. J., 50(1), 175–187.

Rowinski, P.M., Piotrowski, A. 2008. Estimation of parameters of transient storage model by means of multi-layer perceptron neural networks. Hydrol. Sci. J., 53(1), 165-178.

Sedeki, A., Ouazar, D., El Mazoudi, E. 2009. Evolving neural network using real coded genetic algorithm for daily rainfall-runoff forecasting. Expert systems with Applications 36, 4523-4527.

Seo, I.W., Cheong, T.S. 1998. Predicting longitudinal dispersion coefficient in natural streams. J. Hydraul. Engng. ASCE, 124(1), 25-32.

Storn, R. Price, K.V. 1995. Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Report TR-95-012, International Computer Sciences Institute, Berkeley, California, USA.

Sukhodolov, A. N., Nikora, V. I., Rowiński, P. M., Czernuszenko, W. 1997. A case study of longitudinal dispersion in small lowland rivers. Water Environ. Res. 69(7), 1246–1253.

Tayfur, G. and Singh, V.P. 2005. Predicting longitudinal dispersion coefficient in natural streams by artificial neural network. Journal of Hydraul. Engng. ASCE 131(11), 991-1000.

Tayfur, G. 2006. Fuzzy, ANN and regression models to predict longitudinal dispersion coefficient in natural streams. Nord. Hydrol., 37(2), 143-164.

Taylor, G.I. 1953. Dispersion of soluble matter in solvent flowing slowly through a tube. Proc. R. Soc. London. Ser. A, 219, 186-203.

Toprak, Z.F., Cigizoglu, H.K. 2008. Predicting longitudinal dispersion coefficient in natural streams by artificial intelligence methods. Hydrol. Process., 22, 4106---4129.

Wallis, S.G., Manson, J.R. 2004. Methods for predicting dispersion coefficients in rivers. Water Manage., 157(WM3), 131–141.

Weber, M., Neri, F., Tirronen, V. 2009. Distributed differential evolution with exlplorative-exploitative population families. Genet. Program. Evolvable Mach. 10, 343-371.

Yu, B., He, X. 2006. Training radial basis function networks with differential evolution. Transactions on Engineering, Computing and Technology V11, 157–160.

Zhang, G.P. 2007. A neural network ensemble method with jittered training data for time series forecasting. Information Sciences 177, 5329-5346.