Advances in Engineering Software 41 (2010) 939–945

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

A machine code-based genetic programming for suspended sediment concentration estimation

Ozgur Kisi [a,*], Aytac Guven [b]

[a] Erciyes University, Civil Engineering Department, Hydraulics Division, 38039 Kayseri, Turkey
[b] University of Gaziantep, Civil Engineering Department, Hydraulics Division, 27310 Gaziantep, Turkey

ARTICLE INFO

Article history:
Received 25 March 2010
Received in revised form 1 June 2010
Accepted 8 June 2010

Keywords:
Suspended sediment concentration
Modeling
Linear genetic programming
Neuro-fuzzy
Neural networks
Rating curve

ABSTRACT

Correct estimation of suspended sediment concentration carried by a river is very important for many water resources projects. The application of linear genetic programming (LGP), which is an extension to genetic programming (GP) technique, for suspended sediment concentration estimation is proposed in this paper. The LGP is compared with those of the adaptive neuro-fuzzy, neural networks and rating curve models. The daily streamflow and suspended sediment concentration data from two stations, Rio Valenciano Station and Quebrada Blanca Station, operated by the US Geological Survey (USGS) are used as case studies. The root mean square errors (RMSE) and determination coefficient (R^2) statistics are used for evaluating the accuracy of the models. Comparison of the results indicated that the LGP performs better than the neuro-fuzzy, neural networks and rating curve models. For the Rio Valenciano and Quebrada Blanca Stations, it is found that the LGP models with RMSE = 44.4 mg/l, R^2 = 0.910 and RMSE = 13.9 mg/l, R^2 = 0.952 in test period is superior in estimating daily suspended sediment concentrations than the best accurate neuro-fuzzy model with RMSE = 52.0 mg/l, R^2 = 0.876 and RMSE = 17.9 mg/l, R^2 = 0.929, respectively.

1. Introduction

Accurately estimation of suspended sediment concentration carried by a river is important with respect to channel navigability, reservoir filling, hydroelectric-equipment longevity, river aesthetics, fish habitat and scientific interests. All surface water reservoirs are designed to a volume known as "the dead storage" to accommodate the sediment income that will accumulate over a specified period called the economic life. The under-estimation of sediment yield results in insufficient reservoir capacity. To acquire an appropriate reservoir design and operation it is mandatory to determine sediment yield accurately. In environmental engineering, if the particles also transport pollutants, the estimation of river sediment load has an additional significance.

The popularity of neuro-fuzzy (NF) and artificial neural network (ANN) techniques increase in various areas because their capability for solving complex problems that might otherwise not have a tractable solution. According to the recent experiments, the NF and ANN offer promising results in the field of water resources and hydrology, such as rainfall–runoff modeling [13,25], streamflow estimation [12,24,26,29–31], reservoir inflow forecasting [6,35], predicting hydromechanic parameters [19,20], and suspended sediment estimation [14,27,28,42]. However, there are some disadvantages of the ANN and NF methods. The ANN is a black box model and network structure is hard to determine. It is usually determined using a trial-and-error approach, i.e. sensitivity analysis [2,26,27]. Its training algorithm has the danger of getting stuck into local minima, etc. The NF models combine the linguistic representation of a fuzzy system with the learning ability of the ANN. Therefore, they can be trained to perform an input/output mapping, just as with an ANN, but with the additional benefit of being able to provide the set of rules on which the model is based. However, the NF is also a black box modeling approach, where the fuzzy rules have been added. Because the knowledge of the models' structure was known but the equation of the phenomenon, which is actually transparent was not known. In this study, the proposed LGP models are explicit mathematical formulations.

GP can be successively applied to areas where the interrelationships among the relevant variables are poorly understood; finding the size and shape of the ultimate solution is hard and a major part of the problem; conventional mathematical analysis does not, or cannot, provide analytical solutions; an approximate solution is acceptable; small improvements in performance are routinely measured (or easily measurable) and highly prized; there is a large amount of data, in computer readable form, that requires examination, classification, and integration [9].

GP has been successfully applied and verified in the field of water resources engineering [8,15,17,20–22,39,41]. However, there are only a few records of LGP applications. More recently,

* Corresponding author. Tel.: +90 352 4374901; fax: +90 352 4375784.
E-mail addresses: kisi@erciyes.edu.tr, ozgurkisi@hotmail.com (O. Kisi).

0965-9978/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2010.06.001

Guven [23] modeled the time series of daily flow rate in rivers using LGP, and Guven et al. [18] presented LGP as an alternative tool in the prediction of scour depth around a circular pile due to waves in medium dense silt and sand bed. Only three studies observed for sediment modeling using GP approach; Babovic [7] used experimental flume data of others and expressed a new GP-based formulation for bed concentration of suspended sediment. Kizhisseri et al. [32] used GP methodology to explore a better correlation between the temporal pattern of fluid field and sediment transport by utilizing two datasets; one from numerical model results and other from Sandy Duck field data. Aytek and Kisi [3] obtained an explicit formulation of the daily suspended sediment–discharge relationship by applying the GP technique on the daily streamflow and suspended sediment data from two stations on Tongue River in Montana. Azamathulla et al. [5] used genetic programming approach for predicting sediment concentrations of Malaysian rivers. To the knowledge of the authors, LGP has not yet been applied in estimation of suspended sediment concentration.

The main aim of this study is to develop an explicit formulation based on LGP for accurately estimation of suspended sediment concentration. The accuracy of LGP is compared with those of the adaptive neuro-fuzzy, neural networks and rating curve models employed in the previous work of Kisi [28]. This is the first study that compares the accuracy of the LGP with those of the neuro-fuzzy and neural network models in the hydrological context.

2. Linear genetic programming (LGP)

Genetic programming (GP) automatically creates computer programs to perform a selected task using Darwinian natural selection. GP is a robust, dynamic and fast growing discipline, and it has been applied to diverse and difficult problems with great success [16,33]. In this study, we use LGP variant of GP, which operates directly on machine code. The main characteristic of LGP in comparison to tree-based GP is that expressions of a functional programming language (like LISP) are substituted by programs of an imperative language (like C) [10]. Nordin's idea of using machine code for evolution was the most radical "down-to-bones" approach [10,36] in this context. The methodology was subsequently expanded [37] and led to the automatic induction of machine code by genetic programming (AIMGP) system [9,10,38]. In AIMGP, individuals are manipulated directly as binary machine code in memory and are executed directly without passing an interpreter during fitness calculation. This results in a significant speedup compared with interpreting systems. Because of their dependence on specific processor architectures, however, AIMGP systems are restricted in portability [9–11,16]. Present LGP technique utilizes individual programs, which are represented as variable-length strings composed of simple C instructions. An extract from a linear genetic program is illustrated as follows:

$$
\begin{aligned}
&r[0] -= -1.45;\\
&r[0] *= v[0]\\
&r[1] += r[0];\\
&r[0] /= v[2];\\
&r[0] *= v[1];\\
&r[2] = -r[0];\\
&r[0] /= r[2];
\end{aligned}
$$

where $v[i]$ represents the input and output variables used in LGP modeling, $r[i]$ are the temporary computation registers in the programs LGP creates. The output of these registers is the value remaining in $r[0]$ after the program executes [23].

While LGP programs are apparently very simple it is actually possible evolve very complex functions using only simple arithmetic functions on a register machine [16]. The *function (instruction) set* of LGP system is composed of arithmetic operations (+, −, /, *), conditional branches (*if, then*), and function calls (*power, trigonometric, logarithmic,* etc.). In present study, the performance of different function sets was evaluated based on fitness of testing set, and the function set listed in Table 1 was recorded to give the best results. Each element of instructon set implicitly includes an assignment to a variable (*destination variable*). This facilitates the use of multiple program outputs in LGP, whereas in tree-based GP those side effects need to be incorporated explicitly [10].

Instructions either operate on two variables (*operand variables*) or on one variable and one integer constant. Variables and constants form the "terminal set" of LGP. Each instruction is encoded into a four-dimensional vector that holds the instruction identifier, indexes of all participating variables, and a constant value (optionally). Because each vector component uses one byte of memory only, the maximum number of variables is restricted to 256 and constants range from 0 to 255 at maximum. This representation allows an efficient recombination of the programs as well as an efficient interpretation [10].

The evolutionary algorithm (EA) of present LGP applies *tournament selection* and puts the lowest selection pressure on the individuals by allowing only two individuals to participate in a tournament. The loser of each tournament is replaced by a copy of the winner. In such a *steady-state* EA, the population size is always constant and determines the number of individuals created in one *generation*. A segment of random position and random length is selected in each of the two parents and exchanged. If one of the resulting children would exceed the maximum length, crossover is aborted and restarted with exchanging equally sized segments. The crossover points only occur *between* instructions. *Inside* instructions, the *mutation* operation randomly replaces the instruction identifier, a variable, or the constant (if existent) by equivalents from valid ranges. Constants are modified through a certain standard deviation (*mutation step size*) from the current value. Exchanging a variable, however, can have an enormous effect on the program flow that might be the reason why in LGP, high mutation rates have been experienced to produce better results [10].

In GP, the maximum size of the program is usually restricted to prevent programs from growing without bound [10]. In present LGP, the maximum number of instructions allowed per program

Table 1
Parameters of the LGP models.

Parameter	Description of parameter	Setting of parameter
p_1	Function set	+, −, *, /, √, power
p_2	Population size	500
p_3	Mutation frequency (%)	95
p_4	Crossover frequency (%)	50
p_5	Number of replication	10
p_6	Block mutation rate (%)	30
p_7	Instruction mutation rate (%)	30
p_8	Instruction data mutation rate (%)	40
p_9	Homologous crossover (%)	95
p_{10}	Program size	Initial 64, maximum 256 (input combination 1) initial 80, maximum 512 (other combinations)