Using Grammatical Evolution to Predicting Streamflow during Drought

Wai-Yi Leung1, Li Chen2*, Tai-Sheng Wang2, Po-Kuan Chiang3
1 Geotechnical Engineering Research Center, Sinotech Engineering Consultants, Taipei, Taiwan 110, R.O.C.
2 Department of Civil Engineering and Engineering Informatics, Chung Hua University, Hsin Chu, Taiwan 300, R.O.C.
3 Department of Civil Engineering, Katholieke Universiteit Leuven, Hydraulics Laboratory, Kasteelpark Arenberg 40, Heverlee 3001, Belgium.
m09404041@chu.edu.tw, lichen@chu.edu.tw, d09404008@chu.edu.tw, victory66@mail2000.com.tw

Abstract

This paper explores the feasibility of applying a grammatical evolution (GE) system and combines it with the genetic algorithm (GA) to establish the inflow predicting model of De-Chi Reservoir in central Taiwan. First, a GE is an evolutionary automatic programming type system, which can discover relationships among observed data and express them mathematically. Further, a GA was used with this GE to optimize the appropriate function type automatically. We apply this GE model to fit to the inflow data series on dry year. Experimental results are presented to demonstrate the applicability of GE for forecasting long-term time series, and the results are found to be better compared with the traditional multi-regressive (MR) method.

1. Introduction

Forecasting a historical time series has been of the most complicated tasks owing to the wide range of data. Modeling the inflow process of a reservoir is however inherently complex, highly nonlinear and temporally and spatially non-uniform. Several kinds of data mining techniques are developed, such as statistics, memory-based reasoning, artificial neural networks (ANNs), and decision trees. Evolutionary algorithms have been used with much success for the automatic generation of programs. It has an advantage over traditional statistical methods because it is distribution free, i.e., no prior knowledge is needed about the statistical distribution of the data. However, constructing the data structure of dynamic tree of the genetic programming could be a difficult task while programming [3].

The inflow forecasting model of De-Chi Reservoir in central Taiwan was constructed and discussed in this paper. This data structure of binary string is convenient to combine with the genetic algorithm (GA), which can optimize the functions generated by GE automatically. In the case study, multi-regressive (MR) method and GE approach were used to model reservoir inflows and the results of both were compared through four criteria and dry hydrological conditions.

2. Grammatical Evolution

Grammar evolution (GE) has been applied to all manner of automatic programming problems, from symbolic regression, to C programs, or generation of graphical objects. The common view of GE is that, given a particular problem statement, a program that satisfied the fitness function is to be generated. GE is an evolutionary automatic programming type system that uses a combination of a variable length binary string genome and a BNF (Backus-Naur Form) grammar to evolve interesting structures. It presents a unique way of using grammars in the process of automatic programming. Variable-length binary string genomes are used with each codon representing an integer value where codons are consecutive groups of 8 bits. The integer values are used in a mapping function to select an appropriate production rule from the BNF definition, the numbers generated always representing one of the rules that can be used at that time. This technique draws inspiration from the overlapping genes phenomenon exhibited by many bacteria, viruses, and mitochondria that enables them to reuse the same genetic material in the expression of different genes [3].

2.1. Backus-Naur Form

BNF is a notation for expressing the grammar of a language in the form of production rules. BNF grammars consist of terminals, which are items that can appear in the language, e.g., +, -, etc., and nonterminals, which can be expanded into one or more terminals and nonterminals.
A grammar can be represented by the tuple \( \{ N \cdot T \cdot P \cdot S \} \), where \( N \) is the set of nonterminals, \( T \) the set of terminals, \( P \) a set of production rules that maps the elements of \( N \) to \( T \), and \( S \) is a start symbol that is a member of \( N \). When there are a number of productions that can be applied to one particular \( N \), the choice is delimited with the ‘|’ symbol.

Below is an example BNF, where

\[
N = \{ \text{expr, op, pre_op} \}
\]

\[
T = \{ \text{Sin, Cos, +, -, *, /, Variable, Constant} \}
\]

\[
S = \text{expr}
\]

And \( P \) can be represented as

(1) \( \text{expr} :: = \text{expr}\langle\text{op}\rangle\text{expr} \ldots \ldots \text{rule 0} \)

\( | \langle\text{expr}\langle\text{op}\rangle\text{expr}\rangle \ldots \ldots \text{rule 1} \)

\( | \langle\text{pre-op}\rangle \text{expr} \ldots \ldots \text{rule 2} \)

\( | \text{var} \ldots \ldots \ldots \ldots \ldots \ldots \text{rule 3} \)

(2) \( \text{op} :: = + \ldots \ldots \ldots \ldots \ldots \text{rule 0} \)

\( | - \ldots \ldots \ldots \ldots \text{rule 1} \)

\( | / \ldots \ldots \ldots \ldots \text{rule 2} \)

\( | * \ldots \ldots \ldots \ldots \text{rule 3} \)

(3) \( \text{pre-op} :: = \text{Sin} \ldots \ldots \ldots \ldots \text{rule 0} \)

\( | \text{Cos} \ldots \ldots \ldots \ldots \text{rule 1} \)

\( | \text{Log} \ldots \ldots \ldots \ldots \text{rule 2} \)

(4) \( \text{var} :: = \text{X} \ldots \ldots \ldots \ldots \text{rule 0} \)

\( | 1.0 \ldots \ldots \ldots \ldots \text{rule 1} \)

In GE, the BNF definition is used to describe the output language to be produced by the system, i.e., the compilable code produced will consist of elements of the terminal set \( T \). As the BNF is a plug-in component of the system, it means that GE can produce code in any language thereby giving the system a unique flexibility. The C language is used for plugging in to GE easily in this study.

### 2.2. Mapping Process

The genotype is used to map the start symbol onto terminals by reading codons of 8 bits to generate a corresponding integer value from which an appropriate production rule is selected by using the following mapping function:

\[
\text{Rule} = (\text{codon integer value}) \ MOD \ (\text{number of rules for the current nonterminal})
\]

(1)

Considering the following rule, i.e., given the nonterminal \( \text{op} \), there are four production rules to select from:

(2) \( \text{op} :: = + \ldots \ldots \ldots \ldots \text{rule 0} \)

\( | - \ldots \ldots \ldots \ldots \text{rule 1} \)

If we assume the codon being read produces the integer 6, then

6 MOD 4 = 2

would select \( \langle\text{op}\rangle \) as rule 2: \( / \). Each time a production rule has to be selected to map from a nonterminal, another codon is read. In this way, the system traverses the genome.

For example, consider the individual in Table 1. There are fourteen 8-bit binary codons in one string. The decoding process is described as follows.

(1) First, concentrating on the start symbol \( \langle\text{expr}\rangle \), we can see that there are four productions to choose from. To make this choice, we read the first codon from the chromosome “11001000” and use it to generate a number “200”. Because the standard decode of the binary 11001000 is

\[
1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 2 \times 2^4 + 1 \times 2^3 + 2 \times 2^2 + 0 \times 2^1 + 2 \times 2^0
\]

which equals to 200. This number will then be used to decide which production rule to use according to Eq. (1) in BNF. Thus, we have 200 MOD 4 = 0, meaning we must take the zeroth production, rule (0), so that \( \langle\text{expr}\rangle \) is now replace with

\( \langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle \).

(2) Continuing with the first \( \langle\text{expr}\rangle \), i.e., always starting from the leftmost nonterminal, a similar choice must be made by reading the next codon value 160 and again using the given formula we get 160 MOD 4 = 0, i.e., rule 0. The leftmost \( \langle\text{expr}\rangle \) will now be replaced with \( \langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle \) to give

\( \langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle \).

(3) Again, we have the same choice for the first \( \langle\text{expr}\rangle \) by reading the next codon value 206, the result being the application of rule 2 to give

\( \langle\text{pre-op}\rangle\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle \).

(4) Now, the leftmost \( \langle\text{pre-op}\rangle \) will be determined by the codon value 96 that gives us rule 0, which is \( \langle\text{pre-op}\rangle \) becomes \( \text{Sin} \). We have the following:

\[
\text{Sin}(\langle\text{expr}\rangle)\langle\text{op}\rangle\langle\text{expr}\rangle\langle\text{op}\rangle\langle\text{expr}\rangle
\]

\[
\cdot
\]

\[
\cdot
\]

(14) The mapping continues until eventually we are left with the following expression:

\[
\text{Sin}(X) \ast \text{Cos}(X) + 1.0
\]
### Table 1. Example of each codon converted into corresponding BNF grammar

<table>
<thead>
<tr>
<th>No.</th>
<th>8-bit binary codon</th>
<th>Integer value</th>
<th>Mapping function</th>
<th>BNF grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11001000</td>
<td>200</td>
<td>200 MOD 4 = 0</td>
<td>&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>2</td>
<td>10100000</td>
<td>160</td>
<td>160 MOD 4 = 0</td>
<td>&lt;expr&gt;&lt;op&gt;&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>3</td>
<td>11001110</td>
<td>206</td>
<td>206 MOD 4 = 2</td>
<td>&lt;pre-op&gt;(&lt;expr&gt;)&lt;op&gt;&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>4</td>
<td>01100000</td>
<td>96</td>
<td>96 MOD 3 = 0</td>
<td>Sin(&lt;expr&gt;)&lt;op&gt;&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>5</td>
<td>00011011</td>
<td>27</td>
<td>27 MOD 4 = 3</td>
<td>Sin(&lt;var&gt;)&lt;op&gt;&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>6</td>
<td>01001000</td>
<td>72</td>
<td>72 MOD 2 = 0</td>
<td>Sin(X)&lt;op&gt;&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>7</td>
<td>01101011</td>
<td>107</td>
<td>107 MOD 4 = 3</td>
<td>Sin(X)*&lt;expr&gt;&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>8</td>
<td>00111110</td>
<td>62</td>
<td>62 MOD 4 = 2</td>
<td>Sin(X)*&lt;pre-op&gt;(&lt;expr&gt;)&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>9</td>
<td>00010110</td>
<td>22</td>
<td>22 MOD 3 = 1</td>
<td>Sin(X)*Cos(&lt;expr&gt;)&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>10</td>
<td>00110111</td>
<td>55</td>
<td>55 MOD 4 = 3</td>
<td>Sin(X)*Cos(&lt;var&gt;)&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>11</td>
<td>01011000</td>
<td>88</td>
<td>88 MOD 2 = 0</td>
<td>Sin(X)*Cos(X)&lt;op&gt;&lt;expr&gt;</td>
</tr>
<tr>
<td>12</td>
<td>01100100</td>
<td>100</td>
<td>100 MOD 4 = 0</td>
<td>Sin(X)*Cos(X)&lt;expr&gt;</td>
</tr>
<tr>
<td>13</td>
<td>11001011</td>
<td>203</td>
<td>203 MOD 4 = 3</td>
<td>Sin(X)*Cos(X)+&lt;var&gt;</td>
</tr>
<tr>
<td>14</td>
<td>00101001</td>
<td>41</td>
<td>41 MOD 2 = 1</td>
<td>Sin(X)*Cos(X)+1.0</td>
</tr>
</tbody>
</table>

3. GE Combined with Genetic Algorithm

#### 3.1. Genetic Algorithm

The genetic algorithm (GA) is an iterative procedure, which maintains a population of individuals that are candidate solutions to specific domain. During each generation, the individuals in the current population are rated for their effective evaluations, and a new population of candidate solutions is formed using specific genetic operators such as reproduction, crossover, and mutation. These steps are repeated until the convergence criterion is satisfied or a predetermined number of generations are reached. Reproduction is a process in which individual trees are set according to their fitness function values. The reproduction (selection) operator may be implemented in algorithmic form in a number of ways, such as proportional, rank, tournament selection.

#### 3.2. GEGA

First, a GE was employed to transfer the binary string through BNF grammars to mathematical function which maps the input onto output. Further, a GA was incorporated with this GE to optimize the objective value of those functions. In other words, the GA was used as a search strategy to determine the most proper relationship among the observed data.

4. A Case Study of Inflow Prediction in De-Chi Reservoir

In this paper, the proposed GE coupled with traditional MR is applied to the De-Chi Reservoir for inflow forecasting.

#### 4.1. System Description

Located in central Taiwan the Da-Chia river is about 140 kilometers long with an average channel slope of 1/39 and a total watershed size of 1,236 square kilometers. De-Chi Reservoir, which was completed in 1974 and has an efficient storage capacity of $169 \times 10^6$ m$^3$, is one of the major storage reservoirs in Da-Chia River Basin, shown in Fig. 1. The primary water use in the basin is hydroelectric power generation, because it is the steepest channel in Taiwan.

![Fig. 1. The De-Chi Reservoir and the basin of Da-Chia River in Taiwan](image)
Historical ten-day (the traditional time period of reservoir operation in Taiwan) inflows to the reservoir for a period of 40 years (1959-1998), excluding three years were used for modeling. Case of inflow were chosen as the testing cases - the dry condition (year 1964).

4.2. Forecast through GE Modeling

This system identification problem may be viewed as a search for a proper function (and its parameters) which maps input values onto an output value. According to the statistical correlation analyses, the first (t-1), second (t-2), third (t-3), and 36th (t-36) ahead ten-day inflow were chosen as the input variables to predict the inflow at time t.

The Objective Function

The main consideration of objective function of the inflow prediction model is to minimizing the mean absolute error (MAE). The criterion of MAE is suitable for measuring the accuracy of the whole flow data [3]. It for the ten-day periods is defined as follows:

\[
MAE = \frac{1}{N} \sum_{t=1}^{N} |Q_t - \hat{Q}_t| \quad \text{………………(2)}
\]

Where

- \(Q_t\): the actual inflow at time t
- \(\hat{Q}_t\): the predicted inflow at time t
- \(N\): the total number of time steps
- \(t\): time steps (ten-day)

4-3. Multi-Regressive Analysis

To compare with traditional multiple regression (MR), the same input variables were used to construct the model, shown as follows.

\[
t-1 t-2 t-3 t-36 \ Q_1 \cdot \cos(\sin(Q_2)) + \log(Q_3) + \log(Q_4) \quad \text{………………(3)}
\]

Where

- \(\hat{Q}_1\): the predicted inflow at time step t
- \(Q_1\): the actual inflow at time step t-1
- \(Q_2\): the actual inflow at time step t-2
- \(Q_3\): the actual inflow at time step t-3
- \(Q_4\): the actual inflow at time step t-36
- \(b_0 \cdot b_1 \cdot b_2 \cdot b_3 \cdot b_4 \cdot b_5\): the coefficients

4-4. Error Measure Indices

In order to compare the predicting ability of these two models, four statistical and hydrological indexes were proposed. The variables are defined as follows.

- \(Q_t\): the actual inflow at time t
- \(\hat{Q}_t\): the predicted inflow at time t
- \(\bar{Q}\): the average of predicted inflow
- \(N\): the total number of time steps
- \(t\): time steps (ten-day)

(1) Mean Absolute Error—MAE (defined as equ. (2))

(2) Root Mean Squared Error—RMSE

\[
RMSE = \sqrt{\frac{1}{N} \sum_{t=1}^{N} (Q_t - \hat{Q}_t)^2} \quad \text{………………(4)}
\]

MAE and RMSE are two indexes for calculating the deviations between the actual and predicted value. The smaller value represents the better results. The RMSE is influenced more by higher deviations, whereas the MAE is an unbiased interpreter of the forecast performance [4].

(3) Percentage Absolute Error—PAE

\[
PAE = 1 - \frac{MAE}{\bar{Q}} \quad \text{………………………….(5)}
\]

(4) Coefficient of Correlation—CC

\[
CC = \frac{\hat{\sum} (Q-Q) \cdot \hat{\sum} (Q-Q)}{\sqrt{\hat{\sum} (Q-Q)^2} \cdot \sqrt{\hat{\sum} (Q-Q)^2}} \quad \text{………………………….(6)}
\]

When the value of PAE and CC approach 1, means the prediction is more accurate.

4-5. Results and Discussion

The multi-regressive (MR) method is presented for comparison with the newly developed GE. The linear function obtained from MR is shown as the following equation.

\[
35.840+0.470*Q_1+0.633*Q_2+0.072*Q_3-0.0122*Q_4 \quad \text{………………………….(7)}
\]

With population size =100 and through 800 generations, the final optimal equation obtained from GE is shown as equ. (8). It is indicated that the equation is complex non-linear form, including the terms of Sin, Cos and Log defined in <pre-op>.

\[
t-1 t-2 t-36 t-1 Q_1 \cdot \cos(\sin(Q_2)) + \log(Q_3) + \log(Q_4) \quad \text{…….……….……(8)}
\]

In the case of dry year 1964, the MAE of GE is 34.512; and that of MR is 105.774. Obviously, the result of GE is much better than that of the traditional MR method. The actual and predicted inflows generated by GE and MR in the year 1964 are shown in Fig. 2. It is shown that the significant over-estimations occur at the 4th, 24th, 26th and 27th time steps obtained by MR result in the large amount of errors of prediction including mean absolute error (MAE) and root mean squared error (RMSE). On the
other hand, the predicted results of GE can fit the trend of curve and the peaks of actual inflow properly, so the coefficient of correlation (CC) of GE is higher than that of MR. These four criteria are compared between these two models on this case, shown in Table 2. It is indicated that the performances of GE are significantly better than those of MR in all criteria on the case of low flow.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>GE</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>34.512</td>
<td>105.774</td>
</tr>
<tr>
<td>RMSE</td>
<td>60.868</td>
<td>143.498</td>
</tr>
<tr>
<td>CC</td>
<td>0.923</td>
<td>0.724</td>
</tr>
<tr>
<td>PAE</td>
<td>0.859</td>
<td>0.568</td>
</tr>
</tbody>
</table>

Table 2. The criteria of GE and MR on dry year (1964)

Fig. 2. The inflows predicted by GE and MR on dry year (1964)

5. Conclusions

The grammatical evolution (GE) has been presented to predict the inflow of a reservoir and compared with the conventional multi-regressive (MR) models. Because of the complexities of function types, the capability of fine tuning is better using GE in the case study. It can deal easily with nonlinear transfer problems among several input and output data.

Results reported here have shown that GE outperforms traditional multi-regression (MR) on all of the criteria. The results of comparison also indicate that the GE is a powerful tool for input-output mapping and can be effectively used for reservoir inflow forecasting. Further researches can be improved to use the real-coded expression of GE and GA. The coefficients of equation can be more precise and accurate.

References