A fixed functional set genetic algorithm (FFSGA) approach for function approximation

Mohammad Tufail and Lindell E. Ormsbee

ABSTRACT

This paper describes a simple mathematical technique that uses a genetic algorithm and least squares optimization to obtain a functional approximation (or computer program) for a given data set. Such an optimal functional form is derived from a pre-defined general functional formulation by selecting optimal coefficients, decision variable functions, and mathematical operators. In the past, functional approximations have routinely been obtained through the use of linear and non-linear regression analysis. More recent methods include the use of genetic algorithms and genetic programming. An example application based on a data set extracted from the commonly used Moody diagram has been used to demonstrate the utility of the proposed method. The purpose of the application was to determine an explicit expression for friction factor and to compare its performance to other available techniques. The example application results in the development of closed form expressions that can be used for evaluating the friction factor for turbulent pipe flow. These expressions compete well in accuracy with other known methods, validating the promise of the proposed method in identifying useful functions for physical processes in a very effective manner. The proposed method is simple to implement and has the ability to generate simple and compact explicit expressions for a given response function.

Key words | artificial neural networks, friction factor, functional approximation, genetic algorithms, genetic programming, turbulent pipe flow

Mohammad Tufail (corresponding author)
Lindell E. Ormsbee
Kentucky Water Resources Research Institute,
233 Mining and Minerals Building,
University of Kentucky, Lexington,
KY 40506-0107
USA
Tel.: +1 859 257 1299;
Fax: +1 859 323 1049;
E-mail: *mtufail@engr.uky.edu,*
lormsbee@engr.uky.edu

INTRODUCTION

The goal in most modeling is to find an optimal balance between model complexity and model applicability by applying basic principles of model parsimony. The principle of parsimony states that we employ the smallest possible number of parameters in a model (Box & Jenkins 1976). Process-based or deductive models can often be comprised of too many parameters that need to be calibrated and can lead to computational expense and added complexity. Inductive or data-driven models are becoming more and more popular due to their ease of use and simplicity as substitutes for more process-based models in a number of applications. For instance, inductive models may be preferred where (1) computational expense is a critical issue, (2) the process-based deductive model is over-parameterized and cannot be adequately calibrated, and (3) budgetary constraints do not allow for a complex deductive model. Inductive models are also more favorable in real time control of systems to assist in quick and effective management decisions needed to facilitate reliable and safe operation of such systems. For instance, in the real time control of combined sewer systems, such models can serve as an effective management tool for managers to make quick operational decisions during storm events. The use of deductive models in such scenarios can be very time-consuming and thus ineffective to make on the spot decisions. Examples of inductive models range from simple linear regression models to more complex nonlinear models based on artificial neural networks. Both linear and

doi: 10.2166/hydro.2006.021

non-linear inductive models can be used to fit a mathematical model to a given data set in order to represent a process. By definition, regression based models are restricted in the sense that the specific form of the function being sought has to be specified such as n-order polynomial, an exponential function, etc. In cases where the dominant functional relationships of the data sets cannot be precisely predetermined, other methods must be investigated.

More recently, inductive models derived using evolutionary and biological principles are becoming increasingly popular. Genetic Algorithms (GA) and Artificial Neural Networks (ANN) are two such evolutionary methods that have found numerous applications in the development and application of inductive models to real world engineering processes. GA represents a class of probabilistic search procedures that use computational methods based on natural evolutionary processes (Goldberg 1989). ANN-based models represent a digital model of the functional processes of the human brain (Zurada 1992). Artificial neural networks can also be thought to be evolutionary in the way that the training weights in the network evolve or are optimized to improve system performance. Each of these models has been found to be particularly powerful in those applications where a large number of solutions are to be evaluated over a shorter period of time.

Genetic programming (GP) is a branch of genetic algorithms (Koza 1992). It should be noted that GA is not a model-building tool and has been used traditionally for finding optimal values of parameters or decision variables of existing models. Thus while GA use a string of numbers to represent the solution, GP has the capability to create computer programs or models that can turn inputs to outputs from specified building blocks such as mathematical operations and variables. The output from a GP is an empirical model used for approximation whereas the output from a GA is the optimal values of the parameters or decision variables of a known empirical model (Alvarez *et al.* 2000). While GP has been successfully used in model building of response functions, they often result in complex expressions for the function sought that are not often simple and easy to use. Also, such expressions can be difficult to interpret and could lead to overfitting problems (Giustolisi & Savic 2004). The fact that empirical models resulting from GP are of variable size and shape as model structures continuously undergo adaptations from a class of parse trees explains the variability in the resulting optimal model. This is evident in the explicit polynomial approximation model for friction factor in turbulent pipe flow using GP in the work done by Davidson *et al.* (1999).

ANN-based models, such as the popular multi-layer feed-forward networks, have also been used to approximate the response of a particular system by training with available data. Such models require considerable data for training and are not favorable for applications where the objective is to obtain a simple, easy to use, and functionally compact approximation. As the number of hidden layers and number of neurons in each hidden layer increases, the functional form extracted from these so-called black-box models can turn out to be a long expression (a linear and non-linear combination of sigmoidal functions) with numerous terms. Other disadvantages of the ANN-based models as pointed out by Giustolisi & Savic (2004) include parameter estimation and overfitting.

There are limited applications of the use of GA for function approximation. Some of these include the work done by Rogers & Hopfinger (1994) and Shi *et al.* (1998). These include model building as a combination of linear polynomials as well as polynomials of higher order to represent biological activity using physicochemical properties of a series of compounds. More recently, a new technique called Evolutionary Polynomial Regression (EPR) was developed which integrates numerical and symbolic regression to search for an explicit functional approximation of the system being modeled. The disadvantages of GP and ANN mentioned above was one of the key motivations for this new GA-based polynomial functional approximation. EPR uses polynomial structures to formulate functional forms and allows a GA search engine to obtain optimal exponents of such expressions (Giustolisi & Savic 2004; Giustolisi *et al.* 2004). Considering the limitations and disadvantages of GP and ANN for use in developing models for a given response function, there is motivation to develop data-based inductive models that are simple to implement and produce compact and easy to use explicit expressions.

The work described in this paper is a result of such a motivation, and it describes an evolutionary method based

on GA for functional approximation of response functions from a given data set. It is referred to as Fixed Functional Set Genetic Algorithm (FFSGA). The method starts with a general pre-defined functional form, and searches for the optimal (best) computer model (empirical expression) by using a GA to search from a fixed set of functions (of decision variables or model inputs) and mathematical operators (Tufail *et al.* 2004). In addition, the structure can include numeric coefficients to provide greater flexibility and accuracy to the resulting model. The basic GA operators used in the search process include the operations of *reproduction, crossover*, and *mutation*. FFSGA is different from EPR in that it does not use a polynomial structure for functional approximation, and allows the user to formulate any pre-defined form comprised of functions of model inputs (or combination of such functions), coefficients, and operators. In FFSGA, the user has the ability to control the complexity of the structure by evaluating simple (fewer terms) to complex (greater number of terms) in the formulation. FFSGA also offers the flexibility and diversity to include linear or highly non-linear elementary functions in the library of internal functions provided for the GA search process. FFSGA thus allows a breadth first and depth next search of optimal functions as the user continues to augment the available library of functions according to the performance of the search process. One can argue that the requirement of starting from a general (pre-defined) form is a significant limitation of the proposed approach. However, the fact that formulating such a starting functional form does not require prior knowledge of the response function and can be achieved with relative ease favors the proposed method. This is demonstrated in the example application discussed later. While the fixed functional framework of the proposed technique may limit the accuracy of the resulting models, such a compromise may be offset by a final model that is simple, compact, and easy to use (Tufail *et al.* 2004).

Inductive models that result in a functional approximation of the response function tend to provide the added benefit of model parsimony. However, it should be realized that not all inductive models (including the ones generated by the proposed approach) are parsimonious in that they (1) improve function interpretability, (2) contain less parameters and/or variables, and (3) provide better generalization and interpolation capabilities. The proposed method (FFSGA) allows the user the flexibility to control the complexity of the functional structure and parameters used. Depending on the choice of the internal functions of the model inputs, the resulting expressions from the FFSGA model vary in their parameter and functional complexity. Model complexity comes with improved accuracy, but it renders the model to be less parsimonious and functionally interpretable.

FUNCTIONAL APPROXIMATION USING GENETIC PROGRAMMING

Evolutionary methods have been successfully used to develop inductive models that fit available data to provide a closed form approximation of the response function. The most successful of these applications have been found in the use of Genetic Programming (GP) which evolves symbolic expressions resulting in a formula for the given data set (Babovic *et al.* 2001). GP can be classified as a machine-learning method that induces a population of computer programs or models that improve automatically as they experience the data on which they are trained (Banzhaf *et al.* 1998). The most frequently used GP method is so-called symbolic regression proposed by Koza (Giustolisi & Savic 2004; Koza 1992). Given a set of variables where some variables are dependent on others, GP helps to develop functions or models that relate the dependent and independent variables. GP evolves tree-like solutions in finding the optimal function (or computer program) that best fits the given data set. Each potential function is evaluated with the given data set and is assigned a fitness value based on how well the model fits into the data set. The main distinctive feature of GP is thus its ability to search for a solution to the given problem by changing model structures (tree-like) rather than by finding better values of model parameters or decision variables. An example tree representation for the expression $X+(Y\times Z)$ is given in Figure 1.

These so-called parse trees represent a node–link structure whose nodes are procedures, functions, variables, and constants. In Figure 1, the variables X, Y, and Z are leaves in the parse tree and belong to the so-called terminal set, while the mathematical operations $+$ and $*$ are

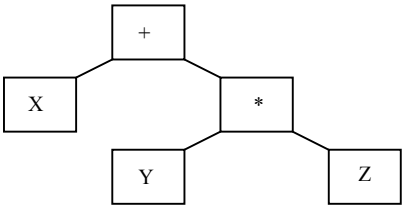

Figure 1 | Tree-like structure in GP for the expression $X + (Y^*Z)$.

functions and are members of the so-called functional set. Like genetic algorithms, the genetic operations of crossover and mutation take place in the same manner in GP. Crossover can be achieved by replacing one or more nodes from one individual with those from another, while mutation can be performed by changing a node's argument or operator function. The result of crossover and mutation operations is the production of two new individuals in which they inherit some characteristics of the parents. The process is continued until the fitness of the entire population increases and converges to finding the near-optimal solution set. As in most evolutionary algorithms, the models that produce the best fit to the given data set have the greatest opportunity to become parents and produce children. The better models produce the smallest errors, or differences between the calculated output and the observed output. While GP may generate a satisfactory function that reproduces the desired output vector $\{Y\}$ for a given input vector $\{X\}$, there is no guarantee that the resulting model structure obtained by GP will give an insight into the actual working of the system. The general idea of GP can be illustrated as given in Figure 2. A training data set is fit to evolving computer programs generated by

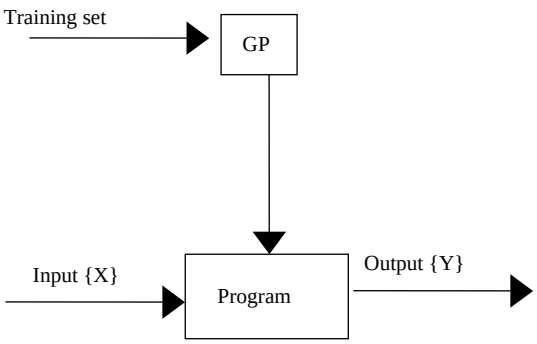

Figure 2 | General representation of GP.

GP and the resulting optimal program is then used to generate output from given inputs.

Babovic (1996) introduced the idea of GP in the area of water resources and since then a number of researchers have used the technique to analyze water management problems. Interested readers are particularly referred to Babovic & Abbott (1997), Babovic & Keijzer (2000), Drecourt (1999), Davidson *et al.* (1999), and Savic *et al.* (1999).

For cases where a more simple and compact response function is sought, the proposed FFSGA approach can provide promising results. In the proposed method, a pre-defined general form is formulated comprising of coefficients, sub-functions of decision variables or model inputs, and mathematical operators. A GA algorithm is then used to search for the best combination of sub-functions of decision variables or model inputs (logarithmic, exponential, sine, cosine, etc.) and mathematical operators ($+$, $-$, $\times$, /, ^) that will fit the pre-defined functional form in order to minimize the difference between observed and predicted outputs. The coefficients in such a functional expression are then sought using least squares optimization by starting from a pre-evolved starting point. Such a method is particularly suited for applications where some relationships between the input and output data sets may be known or at least hypothesized (e.g. logarithmic or exponential relationship). The distinct feature of this method that distinguishes it from standard GP is that the general form of the approximation function sought is defined prior to the evolutionary search process. In the absence of prior knowledge about the response function sought, it is still possible with considerable ease to formulate "a very general" form via some logical arrangement of coefficients, decision variables (model inputs), and mathematical operators. Such a starting formulation determines the number of parameters and elementary functions to be used in the functional form sought and thus allow the user to control the complexity of the expression. Once such a formulation is assumed, the goal of the technique is to select the optimal components of the pre-defined form that would minimize the mean square error between the observed and predicted outputs. The resulting expression becomes the functional approximation of the response function under study. The process of getting the optimal functional form is achieved in

two steps; first the GA algorithm determines the optimal sub-functions and mathematical operators, and then the coefficients (if any) assigned to the functional form are obtained by using least squares optimization. Such an approach helps in eliminating any potential convergence problems associated with searching for numeric coefficients (constants) and functional form (structure) of the optimal expression sought at the same time. For a response function sought, more than one general formulation can be evaluated by varying the structure in different ways as deemed appropriate by the user. This is demonstrated in the example application below in which five different formulations are evaluated.

The proposed fixed functional set genetic algorithm (FFSGA) approach is used to derive a closed form expression for the friction factor in turbulent pipe flow and compare it with an explicit polynomial expression derived previously by Davidson *et al.* (1999) using GP coupled with polynomial regression. The equations derived using five different pre-defined functional forms are also compared to an explicit expression for friction factor, known as the Swamee and Jain equation (Equation (3)), which is a well known expression used in most fluid mechanics text books. The proposed method competes well when compared to these expressions and results in expressions that are simple, compact, and easy to use. This added benefit can be at the expense of reduced accuracy in model performance expressed in terms of the mean square error (MSE) for the data set analyzed.

FIXED FUNCTIONAL SET GENETIC ALGORITHM (FFSGA)

The FFSGA approach starts with a pre-defined functional form which is a combination of numeric coefficients, functions of decision variables (model inputs), and mathematical operators. In the first step, the GA searches for the optimal functions of the decision variables and mathematical operators to obtain the optimal functional components that will constitute the structure of the desired functional form. In the second step, the coefficients of the functional form are obtained by least squares optimization. Thus FFSGA is static in the sense that the general shape and size of the functional form does not change during the GA search. The pre-defined functional form can be based on any prior knowledge of the response function or the user can formulate several general formulations without any such knowledge. This is different from the GP approach in which the parse tree structures (that represent the functional form) are dynamic and change form as they evolve to obtain the most fit functional form or expression for the data set being analyzed. This static GA approach results in the optimal selection of a functional form or expression that can be simple, easy to use, and compact after training on a given data set. The method uses the basic structure of GA optimization involving selection, crossover, and mutation processes to obtain an optimal functional form by selecting functions of decision variables (model inputs) and mathematical operators that maximize or minimize the fitness of the function. Additionally, the method uses least squares optimization to obtain the optimal coefficients in the functional form.

To demonstrate the utility and performance of the proposed method, the data set extracted from Moody Diagram used by Davidson *et al.* (1999) was used for comparison purposes. The objective function in this case is the explicit functional form sought for the friction factor and the fitness function is based on the mean square error (MSE) of the performance of the objective function in predicting the friction factor as a function of the two decision variables (model inputs). These two decision variables include the relative roughness of the pipe (E/D) and Reynolds number (Re). Note that E is defined as the average height of surface irregularities of the pipe and D is the pipe diameter. These are the same variables that are plotted on Moody Diagram in determining the friction factor for turbulent pipe flow (Potter & Wiggert 1991). Five general functional forms were formulated to represent the expression for friction factor in the turbulent flow zone. Each of these forms is comprised of some logical combination of the two decision variables (or some functions of the two decision variables), some numeric coefficients, and mathematical operators, and are given in Figure 3. It should be noted these functional forms were formulated in a general way so as not to mimic or replicate any particular functional form such as the Swamee and Jain equation (Swamee & Jain 1976), which is an established explicit approximation for the friction factor (Potter & Wiggert 1991). In other words, these general functional forms were formulated without any knowledge of an already existing explicit form. However, any prior knowl-

Functional Form #1 = {C_1 operator_1 [function_1 (E/D) operator_2 function_2 (Re)]} operator_3 {C_2 operator_4 [function_3 (E/D) operator_5 function_4 (Re)]}

Functional Form #2 = {C_1 * [function_1 (E/D) operator_1 function_2 (Re)]} operator_2 {C_2 * [function_3 (E/D) operator_4 function_4 (Re)]}

Functional Form #3 = {C_1 * function_1 (E/D) * function_2 (Re)} operator_1 {C_2 * function_3 (E/D) * function_4 (Re)}

Functional Form #4 = {C_1 * function_1 (E/D)} operator_1 {C_2 * function_2 (Re)}

Functional Form #5 = {[C_1 * function_1 (E/D)] operator_1 [C_2 * function_2 (Re)]} operator_2 {C_3 * function_3 (E/D) * function_4 (Re)}

Figure 3 | Pre-defined functional forms.

edge of the functional form can be used to develop such formulations, thereby facilitating the search process. The user can formulate and evaluate any number of such functional forms for the response function being modeled, and the five given in Figure 3 are formulated here for the sake of demonstration. Table 1 gives a list of a sample of 15 different internal functions for each of the decision variables (model inputs) that are available for selection by the FFSGA model. The number of such internal functions actually used can be expanded further by introducing more functions or combination of functions. Table 2 gives the corresponding mathematical operators that are available for selection by the FFSGA model.

As seen in Figure 3, all of the five functional formulations are defined in terms of some function of the decision variables (selected from a set of 15 or more internal functions identified in Table 1). For example, "function_1 (E/D)" in functional form #1 can be selected to be any of the 15 internal functions defined in Table 1, and so on. Similarly, "operator_1" in functional form #1 can take on any of the five operator values given in Table 2. The coefficients C_1, C_2, and C_3 are double precision real numbers that can take a value from -999 to $+999$. However, the user can specify any limits deemed appropriate. It should be noted that the internal functions given in Table 1 can be further expanded by including other functions and combinations of functions as deemed necessary by the user. Clearly, a larger set of such functions will facilitate the GA search process, thereby increasing the chances of finding a more optimal functional form that fits a given data set.

Table 1 | List of functions for decision variables (*Re* and *E/D*)

Function #	Function f(*Re*) or function f(*E/D*)
1	1
2	Re or E/D
3	$1/Re$ or $1/(E/D)$
4	$\mathrm{Exp}(Re)$ or $\exp(E/D)$
5	$\mathrm{Log_e}(Re)$ or $\log_e(E/D)$
6	$\mathrm{Log_{10}}(Re)$ or $\log_{10}(E/D)$
7	$\mathrm{Exp}(1/Re)$ or $\exp(1/(E/D))$
8	$\mathrm{Log_e}(1/Re)$ or $\mathrm{Log_e}(1/(E/D))$
9	$\mathrm{Log_{10}}(1/Re)$ or $\mathrm{Log_{10}}(1/(E/D))$
10	$Re\exp(Re)$ or $(E/D)\exp(E/D)$
11	$Re\log_e(Re)$ or $(E/D)\log_e(E/D)$
12	$Re\log_{10}(Re)$ or $(E/D)\log_{10}(E/D)$
13	$1/Re\exp(Re)$ or $(1/(E/D))\exp(E/D)$
14	$1/Re\log_e(Re)$ or $(1/(E/D))\log_e(E/D)$
15	$1/Re\log_{10}(Re)$ or $(1/(E/D))\log_{10}(E/D)$

Table 2 | List of mathematical operators

Operator #	Operator
1	+
2	−
3	×
4	/
5	^

The basic genetic operations of GA are used to select the best internal functions (from Table 1) and best operators (from Table 2) for inclusion in each of the five pre-defined functional forms given in Figure 3. GA work on a population of possible solutions attempting to find the optimal solution (in this case the most fit computer program or functional form) that maximizes the value of the fitness function. In each generation, some population of solutions improves the fitness function and others get worse. The improved ones are important in producing the next generation populations to continue the search process. Each individual solution set in the GA process consists of a chromosome of decision vectors that makes up the structural components (sub-functions and operators) of the general functional form sought as given in Figure 3. Thus the length of the solution set (chromosome) for each of the five functional forms given in Figure 3 will vary according to the number of terms included in that form. These chromosomes are represented as strings of values in binary form (0 or 1). For example, in the case of functional form #1 given in Figure 3, there are 9 decision vectors in each solution set (chromosome). These are given as follows:

Solution set

operator_1, operator_2, operator_3, operator_4, operator_5, function_1, function_2, function_3, and function_4

In the FFSGA computer code (written in the C language), strings of binary numbers of fixed length represent the values of the decision vectors contained in the solution set given above. The length of each string representation depends on the numeric bounds of the individual parameter being represented. For example, since operator_1 through operator_5 can have a value between 1 and 5 as given in Table 2, they can be represented by a 3-digit binary string. Note that the maximum decimal value of a 3-digit binary string is 7, and appropriate mapping is performed in the decoding of binary numbers. Similarly, if function_1 through function_4 can have a value between 1 and 15 as given in Table 1, each of these internal functions can be represented by a 4-digit binary string. Note that the maximum decimal value of a 4-digit binary string is 15. Consequently, for functional form #1, the total length of each solution set (chromosome) will be 31 for this particular illustration and is represented as shown in Table 3.

For each solution set to be evaluated, the program performs decoding operations to map and assign the corresponding decimal value for each decision variable. For instance, this decoded value can be an integer function value from 1 to 15 for all the functions as defined in Table 1, and an integer operator value from 1 to 5 for all the operators as defined in Table 2. Note that if the library of functions in Table 1 is expanded from 15 to a higher number, the length of the solution set (chromosome) will change accordingly.

The FFSGA model evaluates each of the five functional forms (Figure 3) individually. In other words, a separate search is conducted for each of the five formulations. In each case, the model starts with a random selection of solution sets

Table 3 | Allocation of binary strings to decision vectors in a solution set (chromosome) for Functional Form #1

String	Operator_1	Operator_2	Operator_3	Operator_4	Operator_5
No. of binary digits	3	3	3	3	3
	Function_1	**Function_2**	**Function_3**	**Function_4**	
No. of binary digits	4	4	4	4	

resulting in an initial population, each corresponding to a set of functions (from Table 1) and operators (from Table 2) of the functional form being evaluated. Each solution set thus represents an explicit equation for the friction factor. These solution sets are evaluated for the given data set (values of Re and E/D in this case) and the computed values of friction factor are compared against the target or actual function values to determine the mean square error (MSE). The MSE is a measure of how good the given solution set is in representing the data set evaluated and translates into the corresponding fitness function. The FFSGA model continues to evolve new sets of solution vectors as the search marches from one generation to the other. It is possible that some of the individuals (offspring) may be worse than their parents as the average fitness of solutions generally increases. The improved solution sets tend to survive from generation to generation, and those that are inferior (poor fitness values) will tend to die out in the process. This is accomplished through the process of selection of new generation populations, crossover, and mutation. At the termination of specified generations and a corresponding population size used, the functional form that has the highest fitness value is the optimal structure of the explicit expression sought in the search process. Finally, the coefficients of the functional form are obtained by applying least squares optimization to the optimal form obtained from the GA process.

The FFSGA approach works in a manner similar to GP, except that the general functional form is pre-defined and it does not change size and form. The FFSGA approach can be illustrated by the flow chart as shown in Figure 4.

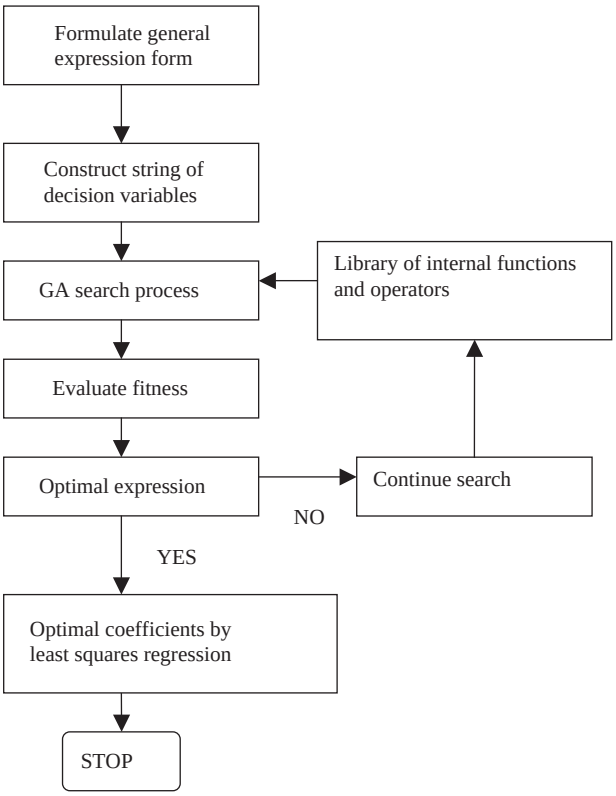

Figure 4 | Flow chart of FFSGA model.

STRATEGY FOR DEFINING INITIAL FUNCTIONAL FORM AND INTERNAL FUNCTIONS OF DECISION VARIABLES (MODEL INPUTS)

FFSGA is a general methodology that can be applied to obtain functional approximation of a response function under study. The open-ended nature of the technique that allows the user to evaluate any desired form and associated functions (linear and/or non-linear) is the strength of this approach that can bring more flexibility and diversity to the search process. By the same token, this can be regarded as a limitation of the approach which requires an input from the user. The following guidelines will aid the user to specify a starting pre-defined formulation and associated library of internal functions (of model inputs) for the GA search process:

1. Given the number of decision variables (model inputs), the starting functional form (as given in Figure 3) can include any number of coefficients, internal functions of model inputs, and mathematical operators as desired by the user. The user can control the complexity of the form in this stage and it is recommended to start from a simple form (fewer terms) and move to more complex forms by adding more terms.
2. Given a starting functional form, the GA would need a library of internal functions of the decision variables (model inputs) to search for the most optimal functions. Table 1 gives an example of one such set of functions for the two model inputs of the example application. A breadth first and depth next approach of elementary function search is suggested here. For any given functional

approximation sought, the user can start with a limited set of generally perceivable functions for the model inputs (such as logarithmic, exponential, sine, square root, etc.) as well as combinations of such functions. The FFSGA model can be run with this initial library of functions to evaluate the goodness of fit obtained for the response function. Based on the results of the model application, if the resulting expression is within the model performance criteria, the user can (1) stop to proceed further, or (2) further improve model performance by focusing on the optimal function types selected by the GA search and introducing more variations of that particular function type in the library of functions.

3. If the model performance with the initial library of functions is not within a prescribed target (such as a target MSE value), the user can continue to add more functions or combinations of functions to the library of functions to provide more flexibility and diversity to the search process. The process continues until a functional expression is obtained that satisfies the user's performance criteria. It should be noted that any prior knowledge of the response function will obviously facilitate the search process as it allows the user to select functions (and its variations) that can better describe the process.

EXAMPLE APPLICATION – FRICTION FACTOR FOR TURBULENT FLOW IN PIPES

The calculation of energy (or head) loss in pipe flow is the one of the most frequently calculated quantities in the area of fluid mechanics. For a given pipe with diameter D, the head loss can be calculated using the Darcy–Weisbach equation as follows:

$$H_L = \frac{fLV^2}{2gD} \tag{1}$$

where H_L is the head loss in the pipe, f is the friction factor, V is the velocity, D is the pipe diameter, and g is the gravitational constant. The friction factor f depends on the relative roughness of the pipe (E/D) and pipe velocity (through Reynolds number $Re \equiv VD/\nu$), and is computed either by implicit or explicit equations available or a chart-based solution such as the Moody Diagram is used. Experimental data that relate the friction factor f to the Reynolds number Re have been developed for fully developed pipe flow over a wide range of wall roughness (Potter & Wiggert 1991). The results of these data are available in the form of what is commonly referred to as the Moody Diagram as given in most fluid mechanics books. In practice, most of the pipe flow calculations lie in the turbulent zone ($4000 < Re < 10^8$). Empirical equations exist that represent the turbulent zone of the Moody Diagram. This paper will not go into a detailed discussion of the numerous equations available for use in computing the friction factor f. Instead, two of the most frequently used equations are discussed in this paper and these include the Colebrook–White equation (Colebrook 1939) and the Swamee and Jain equation (Potter & Wiggert 1991). For turbulent pipe flow, the Moody Diagram is a graphical representation of the Colebrook–White equation given as follows:

$$\frac{1}{\sqrt{f}} = -0.86 \ln\left(\frac{E}{3.7D} + \frac{2.51}{Re\sqrt{f}}\right). \tag{2}$$

The most significant drawback of this formula is its implicit nature, i.e. the friction factor f appears on both sides of the equation, thus requiring the use of iterative methods to solve for f. Regardless of this drawback, the Colebrook–White equation is considered the most accurate formula to compute friction factor f for pipe flow computations in the turbulent zone. The Swamee and Jain equation is an explicit equation for obtaining the friction factor f, and for turbulent flow regime is given as follows:

$$f = \frac{0.25}{\left(\log_{10}\left(\frac{E}{3.7D} + \frac{5.74}{Re^{0.9}}\right)\right)^2} \tag{3}$$

$$10^{-6} < E/D < 0.01 \text{ and } 5000 < \mathrm{Re} < 3 \times 10^8.$$

The Swamee and Jain equation is accurate to within approximately 2% of the Moody Diagram (Potter & Wiggert 1991).

The purpose of this example application is to find an explicit functional approximation for the friction factor f for turbulent pipe flow as a function of Re (Reynolds number) and E/D (relative roughness of the pipe material) for a specific region in the transitional zone of the Moody Diagram. This region lies between Reynolds numbers

ranging from 100 000 to 1000 000 and relative roughness values from 0.001 to 0.01. The data set consists of a two-dimensional grid of 100 data points, created from 10 Reynolds values selected in equal increments of 100 000 on the interval of 100 000 to 1000 000, and 10 relative roughness values selected in equal increments of 0.001 on the interval of 0.001 to 0.01. This is the same data set that was used for finding an explicit polynomial function for friction factor f using GP by Davidson *et al.* (1999), and is used to demonstrate the performance of the FFSGA approach in comparison to the GP approach coupled with polynomial regression. The performance of the explicit functional form using FFSGA will also be compared to the Swamee and Jain equation.

The target or actual friction values used in this exercise are obtained for the 100 data points by using the implicit Colebrook–White formula. A computer program was written in C to implement the FFSGA approach for obtaining the optimal functional approximation for the friction factor f for the turbulent flow regime investigated. For each of the five different general functional forms selected (Figure 3), the FFSGA model finds the optimal expression by selecting the optimal functions of decision variables (model inputs) and operators that minimize the MSE over the data set. Each functional form is evaluated individually by varying the number of populations, generations, and the probability of crossover and mutation in the GA search process. The results of the analysis reveal that the GA search process produces the optimal expressions when the probability of crossover and mutation is fixed at 0.7 and 0.03, respectively. The resulting functional expressions are then used in a least squares optimization to obtain the optimal coefficients (constants) for each expression. The final expressions representing the optimal functional forms for each of the five general forms are given below (Equations (4)–(8)). Table 4 gives the corresponding MSE values and maximum error of interpolation for the data set analyzed by each of the optimal expressions, namely FFSGA-Function 1 through 5. It is evident from the results that the FFSGA approach produces several compact, and easy to use expressions for obtaining the friction factor f for a given set of Re and E/D in the turbulent flow zone. Also given in Table 4 are the MSE values for the same data set for the expressions resulting from the GP approach used by Davidson *et al.* (1999) as well as the Swamee and Jain equation:

Table 4 | MSE and maximum error values for example application (data set from Moody Diagram)

Expression type/method	MSE	Max. absolute error
FFSGA-Function 1	0.000 00020	0.001 229 585
FFSGGA-Function 2	0.000 00050	0.001 848 040
FFSGA-Function 3	0.000 00315	0.003 719 280
FFSGA-Function 4	0.000 00011	0.001 807 310
FFSGA-Function 5	0.000 00022	0.001 281 080
FFSGA-Function 6	0.000 00002	0.000 883 490
2-term GP term by Davidson *et al.* (1999)	0.000 00082	0.002 293 470
4-term GP term by Davidson *et al.* (1999)	0.000 00016	0.001 499 050
5-term GP term by Davidson *et al.* (1999)	0.000 00009	0.001 491 670
10-term GP term by Davidson *et al.* (1999)	0.000 00002	0.000 693 730
14-term GP term by Davidson *et al.* (1999)	0.000 000 002	0.000 193 930
Jane and Swamee equation	0.000 000 02	0.000 253 750

$$f = \frac{-32.48 - \left[\left(\frac{1}{E/D}\right)\exp(E/D) - \log_{10}(Re)\right]}{16.14\left[\left(\frac{1}{E/D}\right)\log_{10}(E/D) + \log_{10}\left(\frac{1}{Re}\right)\right]} \quad (4)$$

$$f = \frac{10.84\left[(\exp(E/D))^{\log_{10}(Re)}\right]}{84.69[\ln(E/D) - 1]} \quad (5)$$

$$f = \frac{[11.71(\exp(E/D))Re\log_{10}(Re)]}{\left[71.71\left(\log_{10}\left(\frac{1}{E/D}\right)\right)Re\ln(Re)\right]} \quad (6)$$

$$f = \left[4.161\log_{10}\left(\frac{1}{E/D}\right)\right]^{[-1.543\exp(\frac{1}{Re})]} \quad (7)$$

$$f = \frac{[3.60(E/D)\exp(E/D)] + \left[-0.027\log_{10}\left(\frac{1}{Re}\right)\right]}{-0.082[\ln(E/D)*\ln(Re)]}. \quad (8)$$

It can be seen in Table 4 that the expressions resulting from the FFSGA approach compete well with all other

methods. Of the five functional forms (as given in Figure 3) evaluated, the best results are obtained by using the FFSGA-Function 4 (listed as Functional Form #4 in Figure 3), and even though it does not compete with some of the higher order polynomial expressions derived by Davidson *et al.* (1999) in terms of accuracy, the expression is simple, compact, and easy to use. It should be noted that in the GP approach used by Davidson *et al.* (1999) to derive expressions for friction factor in the turbulent zone, the independent and target values were transformed to fit on a scale ranging from 1 to 10. This was done to reduce ill-conditioning in the computations. Thus the resulting expressions are in terms of a transformed friction factor, which would need to be converted back into the actual friction factor using the transformation function used. The fixed set GA approach uses the actual data in terms of the actual independent variables and actual friction factor resulting in expressions that do not need to be transformed. The 14-point expression derived by Davidson *et al.* (1999) may be of greater accuracy (MSE is superior to the most accurate FFSGA expression by two significant digits), but results in an expression that is not as compact and simple as the ones derived by the FFSGA model. FFSGA-Function 6 has an improved performance as compared to the five starting functional forms given in Figure 3 (MSE values matches that of the 10-term expression by Davidson *et al.* (1999) as well as the Jane and Swamee equation). This improved performance is attributed to bringing more variability to the internal functions library as given in Table 1 by incorporating functions that are dependent on both Re or E/D. This improvement is a result of sensitivity analysis that takes advantage of prior knowledge of the response function and is explained in the section to follow under sensitivity analysis. The number of generations it takes the FFSGA model for different functional formulations will vary with the complexity of the form and the number of internal functions available for selection. For the five functional forms in Figure 3, the number of generations in which the FFSGA returns the most optimal expressions ranges from 500 to 1500.

SENSITIVITY ANALYSIS

As described previously, none of the five general functional forms given in Figure 3 are structured to replicate the Swamee and Jain equation with regard to the functions of decision variables, operators, and coefficients. For instance, none of the internal functions given in Table 1 include functions that are dependent on both Re and E/D. In fact, all of them are either a function of Re or E/D. A sensitivity analysis was performed to evaluate the performance of the proposed technique by formulating a new functional form #6 (in addition to the five given in Figure 3) that replicates the structure of the Swamee and Jain equation. This is done by introducing functions that are dependent both on Re and E/D. This functional form #6 is given by Equation (9) as follows:

$$f = \frac{C_1\left(function\left(C_2\frac{E}{D}\ OPER(+\ -\ \times/)\ C_3Re\right)\right)^{C_4}}{C_5\left(function\left(C_6\frac{E}{D}\ OPER(+\ -\ \times/)\ C_7Re\right)\right)^{C_8}}. \tag{9}$$

The FFSGA model when applied to such a starting functional formulation then finds the optimal functions and operators by a GA search from the available library of functions. The optimal coefficients C_1 through C_8 are obtained by least squares optimization. In one particular solution set resulting from the FFSGA model, the optimal expression obtained returns a 1 for the numerator function (note that one of the elementary functions is 1 as given in Table 1) and a logarithmic function in the denominator. The performance of this optimal expression (Equation (10)) matches that of the Swamee and Jain equation (MSE = 0.000 000 02) and is given below:

$$f = \frac{0.267}{\left(\log_{10}\left(0.222(E/D) + \frac{6.69}{Re}\right)\right)^2}. \tag{10}$$

The ability of the FFSGA model to match the performance of the explicit Swamee and Jain equation given a functional form (Equation (9)) that benefits from some knowledge of the response function highlights two things: (1) FFSGA improves in performance when some knowledge of the response function is known, and (2) FFSGA produces an optimal expression that looks similar to the Swamee and Jain equation. The improved performance validates the promise that this approach has shown in the functional approximation of response functions. This exercise was carried out to show that prior knowledge

about the response function (form and structure) will benefit the FFSGA approach by facilitating the search process and achieving better accuracy. It should, however, be noted that the GA may return other optimal expressions with different internal functions that may provide the same accuracy. This is anticipated as the search process is driven by the fitness of the optimal expressions and there may be other functions or combinations of functions and corresponding coefficients that would result in expressions of comparable accuracy. At this point it is worth noting that, given the limited number of internal functions of the decision variables (initial library of function provided in Table 1) available for selection by the FFSGA model and the fact that all five general forms (Figure 3) were formulated without prior knowledge of the response function, the previously derived expressions (Equations (4)–(8)) are quite encouraging. It can thus be anticipated that by expanding the library of internal functions and/or combination of functions will bring more variability to the FFSGA model, and it can be expected to further improve its performance, ultimately approaching that of the Colebrook–White equation. Such an expansion can be carried out either by conducting a breadth first and depth next search for optimal functions or with the help of any knowledge that the user might have about the response function. It can be concluded that the proposed method holds the prospect of identifying useful functions for describing physical processes resulting in simple, compact, and easy to use expressions.

GENERAL REMARKS

The use of inductive models that replicate existing deductive models is gaining popularity and this method provides a simple and useful tool for developing explicit relationships for a response function. This method can serve as a useful candidate for application in areas such as rainfall–runoff modeling, watershed response models (for instance, fecal load and nutrient load estimation), and receiving water models (for instance, DO concentration predictions) (Tufail & Ormsbee 2005). Given good and reliable observed data (such as discrete and continuous water quality sampling data), the FFSGA approach can be used to develop functional expressions for different response functions. Such expressions can then be effectively used as inductive models in making quick management decisions without requiring referring to a more complex deductive model. For instance, once an optimal expression is obtained for a response function and validated on an independent data set, it can be easily integrated into an optimization framework to formulate management strategies. Such integration is quite cumbersome and computationally expensive if more complex deductive models are to be used. An added benefit of the approach is that it allows the user to control the complexity of the functional form sought by incorporating fewer or more terms in the starting functional formulation. Other methods such as GP may be able to develop more accurate expressions, but it may be at the expense of increasing complexity in the form and size of such expressions. A limitation of the proposed method is the selection of pre-defined general functional forms for the response function. However, the fact that one can formulate such general forms without requiring any prior knowledge of the functional form sought still makes it an effective technique. Work is in progress in using the proposed FFSGA approach in deriving closed form expressions for determining fecal coliform concentrations for surface water bodies using rainfall, flow, turbidity, and temperature as model inputs (Tufail & Ormsbee 2005).

CONCLUSIONS

A fixed functional set genetic algorithm (FFSGA) methodology is developed to derive closed form expressions for a given data set by using the basic genetic operations of genetic algorithms (GA) namely selection, crossover, and mutation, and a subsequent least squares optimization. The method is simple to implement and has the advantage of producing simple and compact expressions. The method competes well with other evolutionary methods such as genetic programming (GP), and shows great promise of improved performance. The method involves a GA search for the optimal internal functions and mathematical operators for a predefined general functional form formulated to represent a response function. Such a general form can be formulated with or without the knowledge of any existing explicit

expressions available for the response function. The library of elementary functions can be expanded as deemed necessary by the user to bring more diversity and flexibility in the selection process. From a strategic sense, the model can start with an initial library of functions or combination of functions (as given in Table 1) and once a particular function provides favorable results, the user can provide various variations of such functions to further diversify the GA search process. Also, the user can use any prior knowledge of the response function to introduce functions that will facilitate the search process. The resulting optimal structure is then further tuned by obtaining appropriate numeric coefficients using least squares optimization. The example application proves the utility and usefulness of the FFSGA approach and shows that its performance competes well with other methods such as GP. It should be noted here that the purpose of the example application is not to prove that FFSGA necessarily outperforms other methods such as the GP method used by Davidson *et al.* (1999) or other ANN-based models. In fact, the purpose is to demonstrate the ability and performance of this conceptually acceptable new approach based on GA for functional approximation. The MSE values reported for expressions generated by FFSGA and other methods are all comparable (up to five or six significant digits). FFSGA competes well and in fact provides a slightly improved MSE when compared to the 2- and 4-term expressions by Davidson *et al.* (1999). The expressions with 5 and more terms generated by Davidson *et al.* (1999) may provide a better MSE but it is at the cost of greater complexity in terms of the number of terms in the functional form.

ACKNOWLEDGEMENTS

The work presented in this paper was funded by a grant from the Kentucky Division of Water (Grant # 200304300958). The authors sincerely thank the reviewers for their objective and constructive criticism that led to the improvement of the manuscript.

REFERENCES

Alvarez, L. F., Toropov, V. V., Hughes, D. C. & Ashraf, A. F. 2000 Approximation model building using genetic programming methodology: applications. In *Second ISSMO/AIAA Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization*, May 25–June 2, pp. 1–2. Available online: http://www-tm.wbmt.tudelft.nl/~ wbtmavk/2aro_conf/Toropov/Fred4.pdf.

Babovic, V. 1996 *Emergence, Evolution, Intelligence: Hydroinformatics*. A.A. Balkema, Rotterdam, The Netherlands.

Babovic, V. & Abbott, M. B. 1997 The evolution of equations from hydraulic data. Part II: Applications. *Journal of Hydraulic Research* **35** (3), 411–427.

Babovic, V. & Keijzer, M. 2000 Genetic programming as a model induction engine. *Journal of Hydroinformatics* **2** (1), 35–60.

Babovic, V., Keijzer, M., Aguilera, D. R. & Harrington, J. 2001 An evolutionary approach to knowledge induction: genetic programming in hydraulic engineering. In *Proceedings of the World Water & Environmental Resources Congress, 21–24 May, Orlando, FL*, p. 64. ASCE. Reston, VA.

Banzhaf, W., Nordin, P., Keller, R. E. & Frannkone, F. D. 1998 *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann. San Francisco.

Box, G. E. P. & Jenkins, G. M. 1976 *Time Series Analysis: Forecasting and Control*, 2nd edn, pp. 17–18. Holden Day, San Francisco.

Colebrook, C. F. 1939 Turbulent flow in pipes, with particular reference to the transition between the smooth and rough pipe laws. *Journal of the Institution of Civil Engineers*, **11**, 133–156.

Davidson, J. W., Savic, D. A. & Walters, G. A. 1999 Method for identification of explicit polynomial formulae for the friction in turbulent pipe flow. *Journal of Hydroinformatics* **1** (2), 115–126.

Drecourt, J. -P. 1999 *Application of Neural Networks and Genetic Programming to Rainfall-Runoff Modeling. D2K Technical Report 0699-1*. DHI, Copenhagen.

Giustolisi, O. & Savic, D. A. 2004 A novel genetic programming strategy: evolutionary polynomial regression. In *Proc. 6th International Conference on Hydroinformatics, 21–24 June, Singapore*, vol. 1 (eds Liong, S. -Y. Phoon, K. -K. & Babovic, V.) pp. 787–794. World Scientific, Singapore.

Giustolisi, O., Savic, D. A., Doglioni, A. & Laucelli, D. 2004 Knowledge discovery by evolutionary polynomial regression. In *Proc. 6th International Conference on Hydroinformatics, 21–24 June, Singapore*, vol. 2 (eds Liong, S. -Y. Phoon, K. -K. & Babovic, V.), pp. 1647–1654. World Scientific, Singapore.

Goldberg, D. E. 1989 *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley. New York.

Koza, J. R. 1992 *Genetic Programming: On The Programming of Computers By Means of Natural Selection*. MIT Press, Cambridge, MA.

Potter, M. C. & Wiggert, D. C. 1991 *Mechanics of Fluids*. Prentice Hall, Englewood Cliffs, NJ.

Rogers, D. & Hopfinger, A. J. 1994 Application of genetic function approximation to quantitative structure-activity relationships

and quantitative structure-property relationships. *J. Chem. Inf. Comput. Sci.* **34**, 854–866.

Savic, D. A., Walters, G. A. & Davidson, J. W. 1999 A genetic programming approach to rainfall-runoff modeling. *Water Resources Management* **13** (3), 219–231.

Shi, L. M., Fan, Y., Myers, T. G., O'Connor, P. M., Paull, K. D., Friend, S. H. & Weinstein, J. N. 1998 Mining the NCI Anticancer Drug Discovery Database: genetic function approximation for the QSAR study of anticancer ellipticine analogues. *J. Chem. Inf. Comput. Sci.* **38**, 189–199.

Swamee, P. K. & Jain, A. K. 1976 Explicit equations for pipe-flow problems. *J. Hydraul. Div. Am. Soc. Civil. Eng.*, **102** (5), 657–664.

Tufail, M. & Ormsbee, L. E. 2005 The utility of inductive models using artificial intelligence (AI) techniques for water quality management in a TMDL framework. In *Proceedings of the Water Environment Federation (WEF) TMDL 2005 Specialty Conference, 26–29 June*, Philadelphia, PA, pp. 934–954. World Environment Federation, Alexandria, VA.

Tufail, M., Ormsbee, L. E. & Teeagavarapu, R. 2004 A comparison of two artificial intelligence (AI) techniques to develop functional relationships for water quality loadings. In *ASCE World Water and Environmental Resources Congress Annual Conference, 28 June–1 July, Salt Lake City, UT*. ASCE, Reston, VA.

Zurada, J. M. 1992 *Introduction to Artificial Neural Systems*. PWS, Boston, MA.