Genetic Programming – An Evolutionary Approach for Modeling

Prof B V Babu

Assistant Dean – ESD &
Group Leader (Head) - Chemical Engineering
Birla Institute of Technology and Science
Pilani – 333 031 (Rajasthan) India

Phones: +91-1596-245073 Ext. 205 / 224 (Work)
Fax: +91-1596-244183
E-mail: bvbabu@bits-pilani.ac.in
Homepage: http://discovery.bits-pilani.ac.in/discipline/chemical/BVb/

Introduction

Improvement in the performance of any process usually requires thorough understanding and knowledge about the system, with mathematical models being the most common means of representing this knowledge. While it may be possible to develop a model using a detailed knowledge of the physics of a system, there are a number of drawbacks to this approach. Industrial process systems are often extremely complex and non-linear in nature, thus it may take a considerable amount of time and effort to develop a realistic model. Moreover, in many instances simplifying assumptions have to be made in order to provide a tractable solution. A first-principles model will, therefore, often be costly to develop and may be subject to inaccuracies. However, if an accurate process model were available, then many of the benefits of improved process operability would be achievable. The current trend within the process industries is to use data based modeling techniques to develop accurate, cost-effective input-output process descriptions (McKay et al., 1997). The popular techniques may be divided into two categories. The first are based on the use of various statistical techniques and regression analysis, while the second involves the use of artificial neural networks.

For systems, where the physical understanding is complete, the conservation laws in combination with the corresponding rate laws would give the representative mathematical models (Babu, 2004). In the absence of the physical understanding of the systems, we have to rely upon empiricism based on the experimental investigations followed by data regression to get a representative model in terms of an empirical correlation (dimensional or non-dimensional). These empirical models have the limitations of not being applicable to the ranges beyond the experimental investigations. Then came the artificial neural networks which are basically black or grey box models, which work better for many complex processes. But, unfortunately the physics or knowledge about the system is totally hidden. The data-driven identification of these models involves the tasks of structure selection, input sequence design, noise modeling, parameter estimation, and model validation.

Genetic programming (GP), which is an evolutionary approach and a population based search algorithm, is used to develop nonlinear models of various systems. This technique is quite useful, and successful models can be developed using only plant input-output data. Genetic programming is different from all other approaches to artificial intelligence, machine learning, neural networks, adaptive systems, reinforcement learning, or automated logic in all (or most) of the following seven ways (www.genetic-programming.com/sevendiffs.html):

- Representation: Genetic programming overtly conducts it search for a solution to the given problem in program space.
- Role of point-to-point transformations in the search: Genetic programming does not conduct its search by transforming a single point in the search space into another single point, but instead transforms a set of points into another set of points.
- Role of hill climbing in the search: Genetic programming does not rely exclusively on greedy hill climbing to conduct its search, but instead allocates a certain number of trials, in a principled way, to choices that are known to be inferior.
- Role of determinism in the search: Genetic programming conducts its search probabilistically.
- Role of an explicit knowledge base: None.
- Role of formal logic in the search: None.
- Underpinnings of the technique: Biologically inspired.

The performance of an individual organism in its environment determines the likelihood of it passing on its genetic material to future generations. This basic biological principle is known as Darwinian survival of the fittest, and has inspired a class of algorithms known as Genetic Algorithms (GA). GA attempts to find the best solution to a problem by mimicking the process of evolution in nature (Goldberg, 1989; Onwubolu and Babu, 2004). Thus, a typical algorithm will 'breed' a population of individuals that represent possible solutions to a particular problem. GA is not appropriate for symbolic regression problems where the structure and parameters of a model are to be determined simultaneously. This is because simple GA generally uses fixed length binary strings to code potential solutions to a problem. Clearly this is unsuitable for symbolic regression, where the model structure is allowed to vary during evolution. However, GP is a closely related approach that does lend itself to the implementation of symbolic regression. GP differs from GA by following ways:

- Tree structured variable length chromosomes (rather than chromosomes of fixed length and structure).
- Chromosomes coded in a problem specific fashion (that can usually be executed in their current form) rather than binary strings.
- Genetic operators that preserve the syntax of the tree structured chromosomes during 'reproduction'.

Preparatory Steps for Genetic Programming

Genetic programming starts from a high-level statement of the requirements of a problem and attempts to produce a computer program that solves the problem. The human user communicates the high-level statement of the problem to the genetic programming system by performing certain well-defined preparatory steps. The five major preparatory steps for the basic version of genetic programming require the human user to specify are:

- The set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) for each branch of the to-be-evolved program,
- The set of primitive functions for each branch of the to-be-evolved program,
- The fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population),

- Certain parameters for controlling the run, and
- The termination criterion and method for designating the result of the run.

The five major preparatory steps for the basic version of genetic programming are terminal set, function set, fitness measure, parameters, and termination criterion and result designation (http://www.genetic-programming.com/gppreparatory.html). These preparatory steps are the human-supplied input and the computer program is the output of the genetic programming system. The first two preparatory steps specify the ingredients that are available to create the computer programs. A run of genetic programming is a competitive search among a diverse population of programs composed of the available functions and terminals.

Function set and Terminal set: The identification of the function set and terminal set for a particular problem (or category of problems) is usually a straightforward process. For some problems, the function set may consist of merely the arithmetic functions of addition, subtraction, multiplication, and division as well as a conditional branching operator. The terminal set may consist of the program's external inputs (independent variables) and numerical constants. This function set and terminal set is useful for a wide variety of problems (and corresponds to the basic operations found in virtually every general-purpose digital computer).

For many other problems, the ingredients include specialized functions and terminals. For example, if the goal is to get genetic programming to automatically program a robot to mop the entire floor of an obstacle-laden room, the human user must tell genetic programming what the robot is capable of doing. For example, the robot may be capable of executing functions such as moving, turning, and swishing the mop.

If the goal is the automatic creation of a controller, the function set may consist of signal-processing functions that operates on time-domain signals, including integrators, differentiators, leads, lags, gains, adders, subtractors, and the like. The terminal set may consist of signals such as the reference signal and plant output. Once the human user has identified the primitive ingredients for a problem of controller synthesis, the same function set and terminal set can be used to automatically synthesize a wide variety of different controllers.

Fitness measure: The third preparatory step concerns the fitness measure for the problem. The fitness measure specifies what needs to be done. The fitness measure is the primary mechanism for communicating the high-level statement of the problem's requirements to the genetic programming system. The first two preparatory steps define the search space whereas the fitness measure implicitly specifies the search's desired goal.

Control parameters: The fourth and fifth preparatory steps are administrative. The fourth preparatory step entails specifying the control parameters for the run. The most important control parameter is the population size. In practice, the user may choose a
population size that will produce a reasonably large number of generations in the amount of computer time we are willing to devote to a problem (as opposed to, say, analytically choosing the population size by somehow analyzing a problem's fitness landscape). Other control parameters include the probabilities of performing the genetic operations, the maximum size for programs, and other details of the run.

Termination: The fifth preparatory step consists of specifying the termination criterion and the method of designating the result of the run. The termination criterion may include a maximum number of generations to be run as well as a problem-specific success predicate. In practice, one may manually monitor and manually terminate the run when the values of fitness for numerous successive best-of-generation individuals appear to have reached a plateau. The single best-so-far individual is then harvested and designated as the result of the run.

Running GP: After the human user has performed the preparatory steps for a problem, the run of genetic programming can be launched. Once the run is launched, a series of well-defined, problem-independent execution steps are executed.

Executional Steps in GP

Genetic programming typically starts with a population of randomly generated computer programs composed of the available programmatic ingredients. Genetic programming iteratively transforms a population of computer programs into a new generation of the population by applying analogs of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness (as measured by the fitness measure provided by the human user in the third preparatory step). The iterative transformation of the population is executed inside the main generational loop of the run of genetic programming.

- GP uses four steps to solve problems:
- Generate an initial population of random compositions of the functions and terminals of the problem (Computer programs).
- Execute each program in the population and assign it a fitness value according to how well it solves the problem.
- Create a new population of computer programs:
    - Copy the best existing programs.
    - Create new computer programs by mutation.
    - Create new computer programs by crossover (sexual reproduction).
- The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of genetic programming.

References

Babu, B. V. (2004). *Process Plant Simulation*, Oxford University Press, India.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*, Reading, MA: Addison-Wesley, 1989.

McKay B., Willis M. and Barton G. "Steady-state Modeling of Chemical Process Systems using Genetic Programming", *Computers and Chemical Engineering*, **21**, pp 981 – 996 (1997).

Onwubolu, G.C. and Babu, B.V. (2004). *New Optimization Techniques in Engineering*. Springer Verlag, Heidelberg, Germany.