Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 5, pp. 837-843

Modeling Gilliland Correlation using Genetic Programming

M. Olteanu, N. Paraschiv, O. Cangea

Marius Olteanu, Nicolae Paraschiv, Otilia Cangea
"Petroleum-Gas" University of Ploiesti
Romania, Ploiesti, 100680, Bucuresti blvd., no.39
E-mail: {molteanu,nparaschiv,ocangea}@upg-ploiesti.ro

Abstract: The distillation process is one of the most important processes in industry, especially petroleum refining. Designing a distillation column assesses numerous challenges to the engineer, being a complex process that is approached in various studies. An important component, directly affecting the efficient operation of the column, is the reflux ratio that is correlated with the number of the theoretical stages, a correlation developed and studied by Gililland. The correlation is used in the case of simplified control models of distillation columns and it is a graphical method. However, in many situations, there is the need for an analytical form that adequately approximates the experimental data. There are in the literature different analytical forms which are used taking into account the desired precision. The present article attempts to address this problem by using the technique of Genetic Programming, a branch of Evolutionary Algorithms that belongs to Artificial Intelligence, a recently developed technique that has recorded successful applications especially in process modeling. Using an evolutionary paradigm and by evolving a population of solutions or subprograms composed of carefully chosen functions and operators, the Genetic Programming technique is capable of finding the program or relation that fits best the available data.
Keywords: Gilliland correlation, artificial intelligence, genetic programming.

1 Introduction

The early pioneers of computer science, like Alan Turing, John von Neumann, Norbert Wiener studied natural systems as guiding metaphors for their desire to understand nature and create intelligent computer programs capable to learn and adapt to their environment. In the 1950s and 1960s, several scientists from Germany and United States independently studied evolutionary systems with the aim to use evolution as an optimization tool for engineering problems. Several techniques have been created in this period by different research groups: Evolution Strategies, Evolutionary Programming and Genetic Algorithms (see [5]). The basic idea behind all this techniques was to start with a random population of candidate solutions to the specific problem and by applying a set of genetic operators, inspired from the field of genetics, to modify this candidate solutions in such a way to achieve a better fitness or adequacy of the solution for the engineering problem in an iterative process.

In this article we apply a technique of Evolutionary Algorithms, that of Genetic Programming, with the aim at finding an analytic expression for a well studied and used correlation, the Gililland correlation, applied to the design of control models for distillation columns. The algorithms for implementing Genetic Programming are characterized by many heuristic tuning parameters, this paper underlines the most important ones as a result of the simulations.

2 Genetic programming based symbolic regression

Genetic programming was introduced by John Koza (see [4]) and it can be seen as an extension of the Genetic Algorithms by the increase of the complexity of the structures used to represent the potential solution to the problem. In his 1992 book ([1]), John Koza suggested that these potential solutions should be represented as trees of functions and operators, dynamic structures of varying size and shape. The classes of problems that can be best approached using this technique are symbolic regression, in which an analytic expression for a function has to be discovered such that a set of experimental data is fitted and machine learning, domain that uses a set of possible computer programs that produce the desired behavior in the case of some particular input data.

In genetic programming, the set of possible structures is determined by the set of N_f functions from $F = \{f_1, f_2, ..., f_{N_f}\}$, each function can take a specified number of arguments denoted $a(f_i)$ called its *arity* and the set of N_t terminals from $T = \{t_1, t_2, ..., t_{N_t}\}$. Some examples of functions are: arithmetic operations: plus (+), minus (-) , multiply (*), divide (/); mathematical functions: logarithm (log), trigonometric functions - sin, cos, etc; logical functions: AND, OR, NOT. The terminals are variable atoms that represent input variables, signals form sensors / detectors or constant atoms, for example the number 11.25 or the boolean constant *true*. An example of a simple function represented by such a tree is given in figure 1, the corresponding analytic expression being: $f(x, y) = x + \sqrt{y} - 2$.

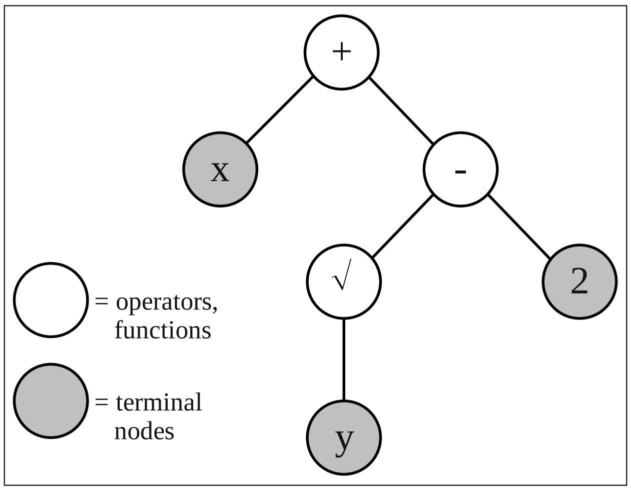

Figure 1: Example of a tree representing a possible solution

As a particular aspect of functions and terminals representation is that of closure which assumes that each of the function from the set must be capable to accept as its input arguments any value and data type that could possibly be returned by any function in the set and any value accepted by any terminal. Another property is that of sufficiency which states that the set of functions and terminals must be capable of defining a solution for the actual problem, the designer of the algorithm is the one that decides what are the most probable functions and constants that could express best a solution. The adequacy or fitness of a particular member of the population has to be measured for a set of fitness cases, in the case of symbolic regression these are the experimental data. The algorithm for implementing Genetic Programming has a structure that resembles that of a Genetic Algorithm:

1. The designers establishes the set of functions / operators and the set of terminals for the specific problem

2. t=0, t - being the generation counter
3. A random initial population of solutions P(0) is generated
4. Fitness evaluation for all the population
5. A new population is created by applying the genetic operators of cloning, mutation and crossover.
6. If the stop condition is met, than the algorithm stops, else t=t+1 and go to step 4

3 Genetic programming applied to the Gilliland correlation

The Gilliland correlation or Gilliland plot (figure 2) correlates the reflux ratio and the number of theoretical stages for a distillation column (see [3]):

$$x = \frac{R - R_{min}}{R+1}, y = \frac{N - N_{min}}{N+1} \tag{3.1}$$

given that the minimum reflux ratio R_{min} is calculated from Underwood's equation and the minimum number of stages N_{min} by Fenske's method.

The resulting curve stretches form (0,1) coordinate at minimum reflux to (1,0) at total reflux. In the literature (see [3]) there are a series of numerical equations derived for this correlation, but because there is some scatter in the fit of data to the Gilliland plot, the expressions that best fits the plot are not always the best reflux-stages correlations.

One of the most used expressions is that of Molokanov:

$$y = 1 - e^{\frac{1-54.4x}{11+117.2x} \cdot \frac{x-1}{\sqrt{x}}} \tag{3.2}$$

used for higher precision, alternative relations being that of Eduljee:

$$y = 0.75(1 - x^{0.5668}) \tag{3.3}$$

and Rusche (see [2]):

$$y = 0.1256 \cdot x - 0.8744 \cdot x^{0.291} \tag{3.4}$$

Also, other correlations have been obtained as a result of research in the domain of optimal control (see [6], [7] - polynomial analytical expression).

For implementing the Genetic Programming algorithm it was used a free MatLab toolbox created by S. Silva, a toolbox well documented and highly modular having many configurable parameters (see [8]).

A usual running of the algorithm is started by entering the following command at the Mathlab prompt: >> [vars, b]=gplab(g,n);
where:

g=maximum number of generations as stop condition
n=the number of individuals in the population
vars=a structure containing all the algorithm variables
b=best fitted individual

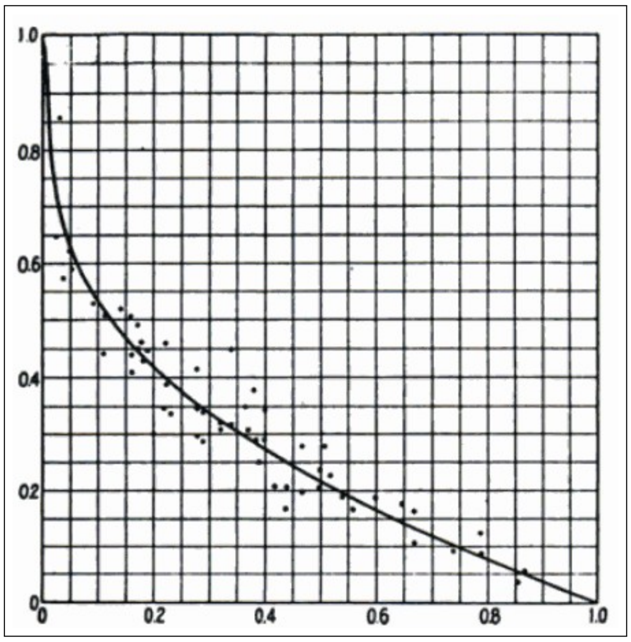

Figure 2: Plot of the original Gilliland correlation

The main modules that compose the toolbox have the following functions: *Variables initializing*, *Initial population* and *Generation creating.* Among the many important features of the toolbox, the parameters that were especially important in the symbolic regression applied for the Gililland correlation:

- Population initialization has three possible methods: *fullinit*, *growinit* and *rampedinit* which was used in the algorithm, and that produces an initial population having very diverse trees (a combination of *fullinit* and *growinit* methods, see [4], [8]);
- With the purpose of avoiding function bloating, the toolbox uses a parameter called *dynamiclevel* that specifies if the trees depth or the trees number of nodes has a fixed limit or not. Another experimental property, called *veryheavy* specifies if this dynamic limit can be decreased under the initial value during the running in the case that the best individual has a smaller depth or number of nodes. By using this option, a much simpler expression for the function has been obtained, and also the running time of the algorithm and the memory resources implied were substantially reduced;
- The methods available for selecting the most adequate individuals are the classical *roulette* and *tournament* methods, in addition there are implemented other methods like *lexictour* or *doubletour* that chooses taking into account the shortest (having the smaller depth or number of nodes) individual. The best results were achieved using the *lexictour* method.

The *crossover* operator (figure 3) randomly choose nodes from both parents and swaps the respective branches creating one or two offspring, in the case of the *mutation* a random node is chosen from the parent individual and substituted by a new random tree, taking into account the imposed restrictions on the depth and number of nodes (see [1]).

A set of six functions have been chosen and two random constants with values between 0 and 1. The functions were: plus, minus, times, custom divide (having protection to divide-by-zero error) also custom square root and custom natural logarithm.

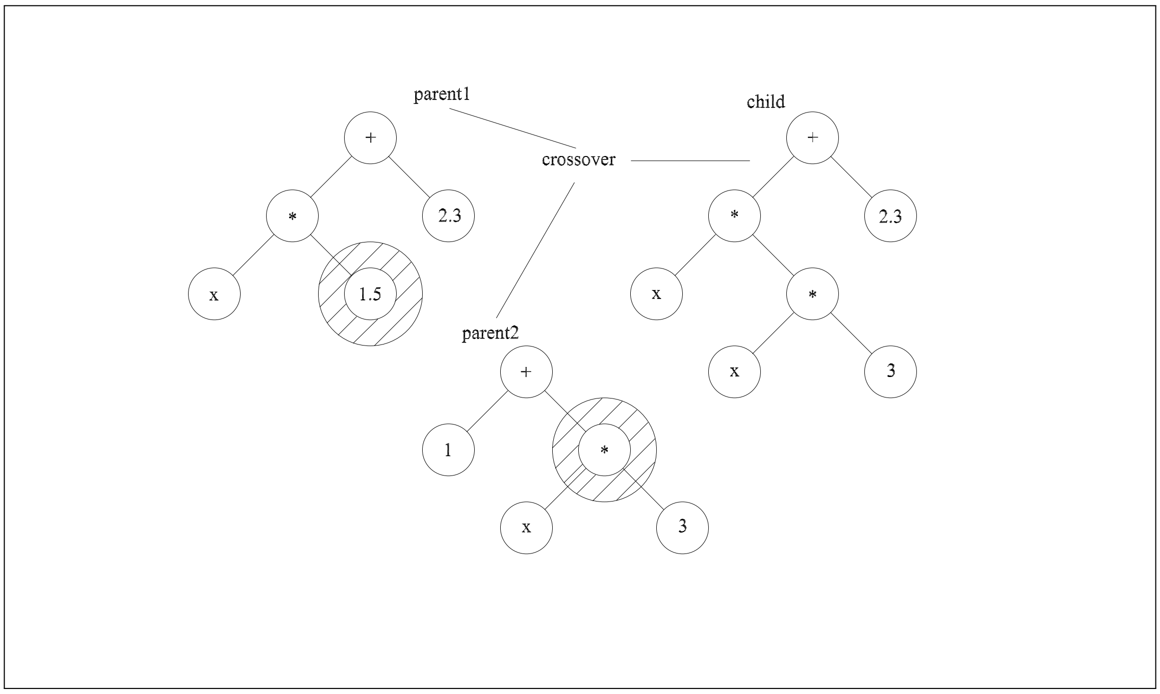

Figure 3: Example of applying the crossover operator

The algorithm ran on a computer with an Intel Core 2 Duo processor, having 2GB of memory and Matlab 7.4 for Windows XP. For a population with 1000 members and a number of 20 generations the running time was in the range of 4 minutes to 4.5 minutes. If the number of generations is increased too much it always results a very big expression for the final function, making it very hard for implementing and studying, a very slow increase in performance (fitness) being obtained. Adding the two final points (0,1) and (1,0) in the data set conducts to a function that does not approximate well the middle data points having a poor general performance because of the relative big scatter of the terminal points from the plot, so for most of the simulations, the two ending points were not included. The fitness function used calculates the sum of the absolute difference between the desired output values and the value computed by the individual on all fitness cases.

$$\mathsf{fitness} = \sum_{i=1}^{N} |y_i - f(x_i)| \tag{3.5}$$

where N is the number of fitness cases, y_i the desired output and $f(x_i)$ is the value returned by the individual.

With a generation number of 40 and a population of 500 individuals for a running time of 169 seconds, the expression obtained is presented in the following tree plot (figure 4):

Another common representation is the string representation, used in Matlab to represent the function:

f=plus(times(minus(mysqrt(mydivide(0.96486,0.56835)),times(plus(0.96486,0.33765),mysqrt (X1))),mylog(mysqrt(mydivide(0.8073,0.22837)))),mysqrt(plus(times(0.33765,times(X1,times (plus(mysqrt(0.8073),0.56835),mysqrt(X1)))),mylog(minus(mysqrt(mydivide(0.9393,0.56835)), times(plus(mysqrt(0.8073),0.56835),mysqrt(X1))))))),

from which we can write the following simplified relation:

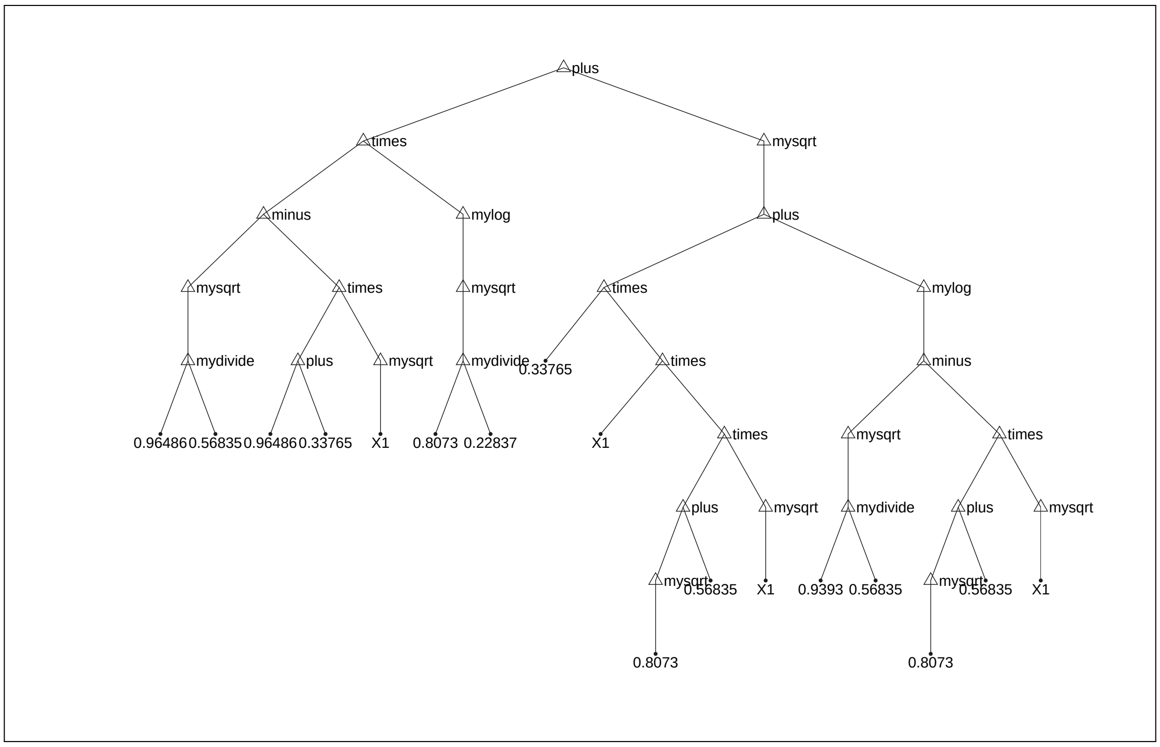

Figure 4: The tree representation of the final solution

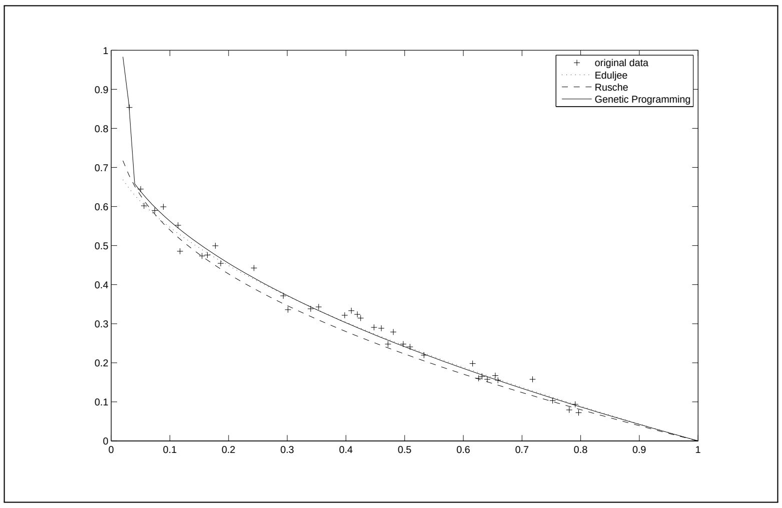

Figure 5: Plot of the final solution and other correlations together with the original data

$$f = 0.631(1.30294 - 1.30251\sqrt{x}) + \sqrt{|0.495 \cdot x\sqrt{x} + \ln(|1.285 - 1.467\sqrt{x}|)|} \quad (3.6)$$

From the plot of the classic correlations (figure 5) and that of the Genetic Programming function we can conclude that a very good approximation is obtained, challenging the methods used in the domain of identification.

4 Conclusions and Future Works

The study presented in this article aims at applying a recently and not too well studied technique of Artificial Intelligence, that of Genetic Programming to the problem of finding the best function that fits some data. The well known Gililland correlation, applied in the domain of process control of the distillation process has been studied. Using the classic representation of the potential solutions as trees, many interesting results have been obtained. After running the algorithm with varying parameters for initial population for the method of selection the best individuals, the method of creating the new offspring, the genetic operators of crossover and mutation and ending with the number of generations and of individuals in the population, it can be stated that the technique of Genetic Programming has a promising future potential, proved by the good estimation of the experimental data. One aspect that proved to be important is the careful chosen of the running parameters which directly influence the quality of the solution.

Bibliography

[1] M. Affenzeller, S. Winkler, S. Wagner, A. Beham, *Genetic Algorithms and Genetic programming - Modern Concepts and Practical Applications,* CRC Press, 2009.

[2] J.R. Couper, W.R. Penney, J.R. Fair, S.M. Wallas, *Chemical Process Equipment - Second Edition,* Elsevier - Gulf Professional Publishing, 2005.

[3] H.Z. Kister, *Distillation Design,* McGraw-Hill, 1992.

[4] J.R. Koza, *Genetic Programming - On the Programming of Computer by Means of Natural Selection,* MIT Press, 1992.

[5] M. Mitchell, *An Introduction To Genetic Algorithms,* MIT Press, 1996.

[6] N. Paraschiv, *Equipment and Programs for Optimal Control of Fractionation Processes*, PhD Thesis, "Petroleum-Gas" University of Ploiesti, Ploiesti, 1987.

[7] N. Paraschiv, *An Analytical Form of the Gilliland Graphical Correlation for the Advanced Control of Fractionation Processes,* Chemistry Magazine no.7-8, 1990.

[8] S. Silva, *GPLAB - A Genetic Programming Toolbox for Matlab User Manual,* ECOS - Evolutionary and Complex Systems Group, University of Coimbra, Portugal, 2007 - accessible from http://gplab.sourceforge.net/index.html.