Identification of Nonlinear Discrete Systems Based Enhanced Genetic Programming

Rami A. Maher, Mohammad J. Mohammad

Abstract— This paper introduces the application of the genetic programming to solve the identification problems of nonlinear discrete dynamic systems. The standard GP is enhanced first to be of a Multi basis function structure and then by a general parameter optimization technique to include one of four proposed techniques. The efficiency of finding the numeric constant node is significantly improved as compared to the traditional methods. The simulation procedure includes first a comparison between the enhanced GP by one of the parameter optimization techniques and the standard GP. Then after for six different models, a comparison between the four utilized techniques is performed. The comparison is made in terms of the number of runs require to find the perfect model, and the average number of generations for successful runs. Finally, a complicated nonlinear discrete is selected to show the powerful of the proposed algorithm.

Keywords- Identification, Nonlinear Discrete systems, Genetic programming, Multi-base Function Genetic Programming MBFGP

I. INTRODUCTION

Identification of a nonlinear discrete dynamic system, which is described by a difference nonlinear equation, represents the mathematical model in many digital control systems or nonlinear additive autoregressive models. Traditionally, there are various model–based as well as non-based-model methods that identify the mathematical model of these systems [1, 2]. Genetic programming has been used extensively for identifying continuous and rarely for discrete nonlinear systems [3, 4].

Genetic programming GP suffers the weakness and difficulty in discovering useful numeric constants for the terminal nodes of its program trees [5]. The difficulty with numeric constants stems from their representation as tree nodes. The reproduction crossover and mutation operations affect only the structure of the tree and not the composition of the nodes. That is, the individual numeric constants are not affected by reproduction operations and hence cannot benefit from them.

Rami A. Maher is with the Isra University, Amman, Jordan, rami.maher@iu.edu.jo while Mohammad J. Mohammad is with the University Of Technology, Baghdad, Iraq, cse-dept@oftechnology.edu.iq

There are two traditional ways of generating a new numeric constant, the arithmetic combination and the arithmetic genesis. Although these two ways are able to create a constant value, they require a tedious computation and in most cases, they are not enough efficient. Some of the early enhancements [6] consist of including a small number of numeric constants and/or ephemeral random constant ERC in the original terminal set. The ERC is helpful because it provides many different numeric constants in the initial generation. In spite of this, most problems require a solution that uses numeric constant other than that in the initial generation, and in turn; it requires evolving the tedious arithmetic combination.

The demand for a more efficient GP is an important research area; for instance, numerous modifications of the basic GP parading are already known. Several researchers have considered GP augmented by hill climbing, simulated annealing and other stochastic techniques [7]. In this paper, four parameter techniques are considered to develop an enhanced GP, which is utilized to solve the identification problem of dynamic discrete nonlinear systems. Therefore, a general procedure for calling the desired technique is augmented with the standard GP to perform an enhanced genetic programming algorithm. In [8, 9] a multi-base function genetic programming MBFGP algorithm shows excellent results in finding optimal controllers. Thus it is thought to experiment with for identification problems.

II. MULTI BASIS FUNCTION GENETIC PROGRAMMING MBFGP

This section is devoted to give a very brief discussion will of the used enhanced genetic programming algorithm, which will be used throughout this paper. There are five basic parts constituting the tree structures of the MBFGP. These are:

i. Representation: The structures of program tree are composed of a random number of linear and/or nonlinear basis functions (terms), which are forced to be linear in parameters. The general tree structure of MPFGP is illustrated in Figure 8.1. As shown, the tree is divided into two main parts. The head part is constructed from independent (fixed) nodes set Σ, f, or R, and the tail part is constructed from dependent (unfixed) nodes set. Figure 8.1 represents an example of such a tree structure of the MBFGP, where three variables $x_1, x_2,$ and x_3 are related through the multiplication, division and power operations and a sine function.

ii. Initial population of the random tree: It must be created by the syntactic rules of construction, and the value for each numeric constant terminal node is chosen randomly in a specified range. The end of the tree structure is bounded by the parameter of the maximum creation depth, which is specified by the user.

iii. Genetic operations: Two types of crossover operation are implemented within MBFGP structures. These are the internal operation which occurs in the tail part, and the external operation, which occurs in the head part. Four types of mutation operation are included. These are the swap, shrink, inverse shrink, and the branch operations, which are applied either to the head or tail parts of the parental tree structure or both.

iv. Additional genetic operations: Besides the basic genetic operations, additional operations are used to vary or adapt the tree structures. Three operations are proposed, the endowment, delete, and merging operations, which make changes on the basis functions. In order to apply any of these operations, random selection of two basis function of the individual is performed.

v. Enhancement operations: To enhance the performance of the MBFGP, five operations are proposed. These are the nested function, structure sorting, enrichment, numerical constant mutation, and contribution operations. These operations are utilized to fulfill different tasks. For example, the last operation is used to resolve the problem of poor diversity of the basis functions.

The above-mentioned operations possess their special programs, which are integrated within the general GP algorithm; the reader can refer throughout the reference [9] for more and detailed information. The use of a certain operation may be either compulsory or problem dependent. Moreover, within the proposed MBFGP algorithm, the numerical method of the 4th order Runga-Kutta is applied to solve the dynamic system of the considered problem.

III. PARAMETER OPTIMIZATION TECHNIQUES

For the purpose of this paper, four known parameter optimization techniques are proposed. They are the numeric constant mutation NCM, the gradient descent GD, the simulated annealing SA, and the genetic algorithm GA. These techniques have different stochastic concepts for finding the numeric constant node values. A brief explanation is given below.

1. *Numeric Constant Mutation*

Numeric constant mutation: Initially, a certain range and resolution are specified for the NC terminal node $\mathcal{R}$ values (say -10 to +10 and 10^{-3}). In each individual, the number of mutated numeric constant node is chosen randomly in a certain range (say, 1 to 3) of cost surface dimension. Then with equal probability, the mutation process is performed in either of two methods. The first method, the new NC is randomly taken from a certain uniform distributed selection range, which is the old value plus or minus a certain percentage of the total allowed range. In the second method, a random selecting of the digit location is mutated in similar as with the first method [8].

2. *Gradient Descent (GD)*

In an analogue way of updating the weights of a numeral network, terminal constants can be adjusted using gradient descent. However, various terminal constants are typically random within a GP tree and are rarely adjusted by gradient methods [7] because of the unavailability of derivatives and computational expense. In principle, the gradient descent search is applied to take a step on the cost surface from the current parameter θ. The gradient of the cost is found as a vector of partial derivative. At each generation all NC terminal nodes are updated several times using the following rule

$$\theta_k^{New} = \theta_k^{New} - \alpha \frac{\partial}{\partial \theta_k}\left[\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i,\theta)^2\right] \qquad (1)$$

where α is a learning rate, x is the vector of NC input values, y is the output vector of NC values, and f is a scalar unknown in advance function. However, finding the NC values is done by gradient descent during the same time the functional structures are evolved. Thus if $n_j(.)$ denotes node functions, then

$$\frac{\partial f(n_1(n_2(n_3(\dots)\dots)..).)}{\partial \theta_k} = \frac{\partial f}{\partial n_1}\frac{\partial n_1}{\partial n_2}\frac{\partial n_2}{\partial n_1}\dots\frac{\partial n_r(\theta_k)}{\partial \theta_k} \qquad (2)$$

Therefore, differentiation of the tree is simply reduced to the product of the node derivatives on the path, which starts at the given NC and ends at the root.

3. *Simulated Annealing (SA)*

SA is a stochastic global search optimization approach which does not require derivatives. The SA algorithm is inspired by a physical thermal process called annealing, which is used to obtain a minimum energy crystalline structure of the metal [10]. The SA algorithm as a numeric parameter optimizer has five basic features:

i. The Solution representation: It represents a set of NC values that are optimized in a specified individual tree. Assuming a certain range, say -10 to +10 with a resolution 10^{-3} (it is 2001 code instances) and five decimal digits for each value.
ii. The generation of a new solution: A new solution is generated by perturbing a random number of NC of the GP individual. The decimal code changes are performed by randomly adding value to a randomly selected position among the last three least significant decimal digits. This operation respects the upper and lower limits of the decimal code.
iii. The acceptance function: Let Δc be the change of cost function between the current state and new state. The acceptance of new state is for $\Delta c < 0$ or with a probability of $e^{-\Delta c/t}$, where t is control parameter (temperature).
iv. Cooling schedule: The proposed SA applies a geometric Boltzmann annealing, which is simply be effective, $t_{k+1} = \lambda\, t_k, k = 0,1,2..$ where λ is a constant in the range 0.50-0.99.
v. The stopping criterion: The optimization process will be stopped when the control parameter reaches a value less than a pre-specified value.

4. *Genetic Algorithm (GA)*

The proposed strategy is very similar conceptually to gradient or SA technique. GA has the advantage of being less local than the gradient descent technique and able to produce better constant values even for that lie within the region within the search space, which are well far apart from present ones [11]. The GA includes the encoding of NC tree, the generation of random initial population, fitness evaluation, the selection of individuals to be continued in the next generation, and the genetic crossover and mutation operations.

IV. PROBLEM STATEMENT AND GP SOLUTION

Let us have a series of observed data points $(k, u(k), y(k))$, collected from a one dimension discrete-time dynamic system that can be arranged in an array form as

$$Data := \big(k, u(k), y(k)\big) = \begin{bmatrix} 0 & u(0) & y(0) \\ 1 & u(1) & y(1) \\ 2 & u(2) & y(2) \\ \vdots & \vdots & \vdots \\ m & u(m) & y(m) \end{bmatrix} \qquad (3)$$

where m is a number of available data points or the number of iterations in the experiment of collecting the data.

The task is to identify (modeling) the given data by a certain nth order difference equation that can describe as well as possible the behavior of a discrete-time dynamic system DDS

$$y(k) = f(k, u(k-1), u(k-2) \ldots u(k-n_b), y(k-1), y(k-2), y(k-3) \ldots . y(k-n_a)) \qquad (4)$$

where f is a composite function including of some elementary functions such as trigonometric functions, power function, etc. n_b, and n_a are the input and output orders respectively.

The GP solution is proposed to creates all types of nonlinear models and not only models that are linear in parameters. In order to restrict the evolved tree structures in a GP algorithm only to these trees that represent linear in parameter models, the MBFGP algorithm has to be adapted. The MBFGP algorithm is made to have the ability to decompose the tree into function terms. Each sub-tree, which is rooted by a basis function node f represents one of the functional terms in the model. To avoid nonlinear in parameter models, the parameters must be removed from the unfixed terminal set; that is to contain only the variables $\{k, x_1(k), x_2(k) \ldots . x_n(k)\}$, where $x_i(k)$ denotes the ith regressor variable.

The parameters of the model are represented by the fixed numeric constant terminal node. The unfixed function set can be used for selection from special model classes that are linear in parameters. For a general linear in parameters model, the unfixed function set can be as $F = \{*, /, \hat{}2, \hat{}3, \sqrt{}, \sin, \cos, \exp, \text{etc.}\}$. However, if polynomial models are considered then the set can be customizing to the set $F = \{*, /, \hat{}2, \hat{}3, ..\}$.

The MBFGP algorithm keeps evolving new generations until the mean square error between the calculated and measured output values reaches a prescribed minimum value, i.e. the Rawfitness is given

$$Rawfitness = \frac{1}{m}\sum_{k=1}^{m}\left[y(k) - \sum_{i=1}^{N_b} p_i F_i(X(k))\right]^2 \qquad (5)$$

where N_b is the number of terms in the candidate model.

In order to simulate with the four parameter optimization technique, a general procedure is augmented. The steps of this procedure are:

- The number of the NC terminal nodes in the program tree is counted and their location is recorded.
- The original values of the NC terminal nods are stored in specified memory locations as a best set and the corresponding fitness values of the individual are calculated.
- Starting from original values, a vector of these values is created so that the parameter optimizer can work on them easily and clear the iteration counter, i.e. set j = 0.
- Using the selected technique to adjust the current vector, and letting $j = j + 1$.
- The adjusted vector is inserted back in program tree and the fitness is calculated for this adjusted vector.
- If the fitness of the new adjusted vector is less than that found in the previous iterations then store this vector in a specified memory as the best vector via all iterations. Otherwise go to the next step.
- If the number of iterations reaches the maximum number, the go to step 8, otherwise go to step 4.
- The best vector is inserted back in the program tree.
- The fitness value of the program tree is calculated.
- End.

V. RESULTS AND DISCUSSION

The identification of the nonlinear discrete dynamic system will be demonstrated in two examples. This first considers the solution based on standard GP and the enhanced MBFGP for linear in parameter model. The second example gives a comparison between the four parameter optimization techniques for five different models.

***Example* 1**

Consider the linear in parameter system

$$y(k) = 0.8u^2(k-1) + 1.2y(k-1) - 0.9y(k-2) - 0.2$$

The task is to compare the identification performance between the standard GP and MBFGP algorithms.

The measurements are generated by simulation using uniform distributed random input in the range -5 to +5, and a total of 50 points of measurements are taken. The standard GP and the MBFGP algorithms are used. Each of these algorithms has 200 population size, ramped-half-and-half creation type with 20% creation probability and 5 maximum depth of creation, the NCM method with -10 to +10 range and 0.001 resolution, Tournament selection of size 4, and 300 generations termination criterion. Also, both algorithms have the same maximum depth of crossover equal to 8, 8% probability of swap and shrink mutations, the feature of adding best solutions to a new generation, and a number of 50 best solutions are used in Elitism.

Differences between the two algorithms include the use of internal and external probability (4% each) of the inverse shrink and branch mutations with MBFGP, while with standard GP a single probability of 8% for the inverse shrink and branch mutations. Furthermore, MBFGP has 8% probability of delete and endowment operations that have included with standard GP. Finally, only the MBFGP algorithm uses contribution operation of 3 generations frequency and an error reduction value of 0.001, besides the margining operation in all generations for all individuals.

The unfixed function set and terminal set of the MBFGP algorithm are

$$F = \{*, /, pow, -unary\}$$

and

$$T = \{\mathcal{R}, u(k-1), u(k-2), y(k-1), y(k-2),$$
$$y(k-3), one\}$$

The equivalent function of the standard GP are

$F = \{-, +, *, /, pow, -unary\}$,

and

$$T = \{\mathcal{R}, u(k-1), u(k-2), y(k-1), y,$$
$$y(k-3), one\}$$

Both GP algorithms are simulated using C++ Language for 10 times for two cases. In the first case, the input data are noise free, while in the second case a random uniform distribution in the range of $\pm 3\%$ of the model output is added to the correct output of the target model. The standard GP catches the perfect model 4 times in the first case and 3 times in the second case, while the MBFGP catches correspondingly 6 and 5 perfect models. Furthermore, the MBFGP has an average of 45 generations to reach the perfect model against 120 generations for the standard GP. In the second case, for the MBFGP, the evolved models of the successful runs have relatively near exact model parameters due to the injected noise. One of these five models is.

$$y(k) = 0.798u^2(k-1) + 1.202\mathrm{y(k-1)} - 0.889\mathrm{y(k-2)}$$
$$-0.208$$

Accordingly, it can be concluded that the MBFGP outperforms the standard GP.

***Example* 2**

The MBFGP algorithm considered in the previous example, will be used for identifying five different linear in parameter models. For each of these models, the four mentioned parameter optimization techniques will be merged one by one to make a comparison between them. Table 1 lists the main control parameters of all techniques.

TABLE 1 MAIN CONTROL PARAMETERS OF ALL OPTIMIZERS

Optimizer Type	Parameters of Optimizer
Numeric Constant Mutation	- Maximum number of mutation is 3 - Selecting range is 100% of the total range that is used in method 1 - One of the four digits is selected to be mutated by method 2 with uniform selection probability - Probability of applying each method is 0.5 - Number of iterations is 660
Gradient Descent	- First learning rate is 0.1 - Second leaning rate is 0.4 - The probability to choose each learning is 0.5 - Number of iterations is 600
Simulating Annealing	- Integer coding - First initial value $t_0 = 1$ - Second initial value $t_0 = 0.001$ - Probability to choose initial value is 0.5 - Number of state transitions under each control variable value is 10 - Cooling rate is 0.9 - Stopping criterion is $t < 0.001t_0$; this gives 660 iterations in both initial values.
Genetic Algorithm	- Integer coding - Population size is 20 - Maximum number of iterations is 33 - Tournament selection with $k = 0.7$ - Single crossover point - Crossover rate 0.8 - Mutation rate 0.2 -Probability to mutate each parameters is 0.3 - Elitism strategy is used (retain one best solution)

The nonlinear discrete models are:

1. $y(k) = 0.8\,u^2(k-1) + 1.2\,y(k-1) - 0.9\,y(k-2) - 0.2$
2. $y(k) = 0.3\,y(k-1) + 0.6\,y(k-2) - 0.4\,y(k-1) +$
   $+0.3\,u^2(k-1) + u^3(k-1)$
3. $y(k) = \sin\big(y(k-1)\big)$
   $+ u(k-1)\cos\big(y(k-1)u(k-1)\big) + 5\,u(k$
   $-1)$
4. $y(k) = 0.5\,u(k-1) + 0.3\,y(k-1) + y^2(k-1)$
5. $y(k) = u(k-1) + u(k-1)u(k-2)y(k-2) + u(k-1)y(k$
   $-2)$

Except for the third model, the used MBFGP algorithm is the same as that for example 1; however for the third model, the unfixed function set is $F = \{*, /, pow, -unary, sin, cos\}$. In order to make a fair comparison, each technique is applied to the underlying candidate model, i.e. unless a zero mean square error is reached, the stop of evaluation will be after ending the 660 iterations. Again, the simulation is performed for 10 independent runs for each of the four optimizers. Table 2 depicts the simulation results. Clearly, the GD technique is the worse among the others, because it fails to find exact models for many cases and for successful run, the average number of generations is very high. Better performance has been obtained with the simulated annealing technique as compared to gradient descent technique. The NCM technique is the best choice among the others. However, although the GA fails to find exact models in a small number of runs, it has less total average number of generations than for the NCM.

Example 3

To show the powerful of the proposed MBFGP, which is enhanced by the numeric constant mutation for identifying a complex discrete time dynamic system, the following system is considered [12]:

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)u(k-1)[y(k-2)-1)+u(k)}{1+y^2(k-1)+y^2(k-2)}$$

The task is to identify optimally the structure and parameters using both MBFGA algorithms. Both is equipped with numeric constant mutation. Table 3 lists the parameters and their values of the proposed enhanced MBFGA algorithm. The values are assigned to obtain as much as accurate results, which are defined by a minimum fitness value. The fitness is the mean square error between the actual and the evolved outputs of the system. Several trails have carried out before the final assignment of these values

The unfixed function set and terminal set are as

$$F = \{*, /, \hat{}, -unary, sin, cos\}$$

$$T = \{u(k), u(k-1), y(k), y(k-1), y(k-2), one\}$$

TABLE 2 RESULTS OF EXAMPLE 2

Model No.	Type of Parameter Optimizer	No. of successful runs	No. of Generations
1	NCM	10	43
	SA	10	45
	GD	6	244
	GA	10	64
2	NCM	10	120
	SA	9	151
	GD	10	100
	GA	9	97
3	NCM	10	114
	SA	8	120
	GD	6	114
	GA	9	94
4	NCM	10	31
	SA	10	96
	GD	8	224
	GA	9	38
5	NCM	10	59
	SA	9	88
	GD	7	286
	GA	10	49

The numeric constant terminal nodes are in the range (-10, +10) with a resolution of 10^{-5}.

The input $u(k)$ is randomly generated in the range $[-1,1]$, and 200 data samples are generated. The first 100 data points are used for the training and the other 100 data are used for result validation.

The best evolved model with a fitness value equal to 3.345×10^{-4} is the following model (in the model, for simplicity it is written u, u_1, y_1 instead of $u(k), u(k-1), y(k-1)$ and so on)

$$\begin{aligned} y(k+1) = {} & 1.667u\cos(y_1)\cos(y_2) + 0.039195\sin(u_1^2 y_1) \\ & + 0.1416535\sin(-u_1^2 y_1 y_2) + 0.01445 y_1 \\ & - 0.23555\sin(u) \\ & + 0.42263 u_1 y_1^2 y_2 \cos(u_1)\sin(u) \\ & + 0.02217 u^4 cos^4(y_1) cos^4(y_2) \end{aligned}$$

Clearly, the evolved model contains only the operation, terms and functions that are considered in the unfixed and terminal nodes. In addition, it can be noted that the evolved model is completely different than the original system. Figure 1 shows the actual and model outputs for validation data set, and figure 2 shows the identification error for the best evolved model by MBFGP algorithm. As it can be shown, the identification error is of order 10^{-2}, which indicates an acceptable accuracy for such a complex system.

In fact, the simulation shows that the standard GP algorithm provides also acceptable results, but with a little larger identification error and undesirable nested mathematical terms for the same training and validation data set. Furthermore, the improvement of accuracy increases with smaller resolution, and as the number of training data points increases. However, the computation time will be increased significantly.

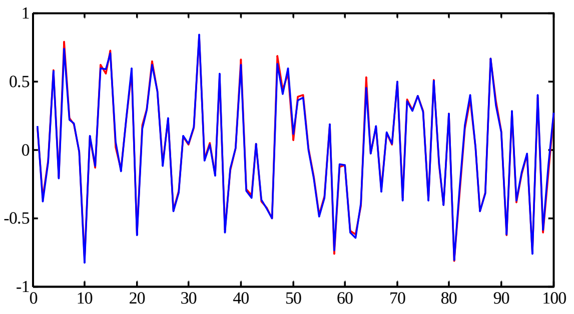

Figure 1 Actual and model outputs for validation data set

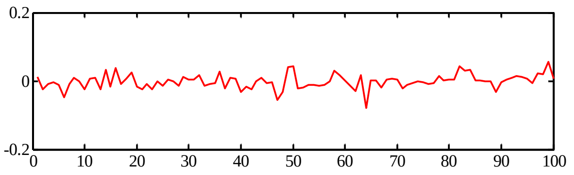

Figure 2 Identification error

TABLE 3 RESULTS OF EXAMPLE 3

MBFGP Control Parameters	Values	
Population Size	200	
Terminal Criterion	100 generation	
Creation Probability	20%	
Criterion Type	Ramped-Half-and-Half	
Crossover Probability	Internal	19%
	External	13%
Max. Depth for Creation	5	
Max. Depth for Crossover	8	
Selection Type	Tournament	
Tournament Size	4	
Swap Mutation Probability	8%	
Shrink Mutation Probability	8%	
Inverse Shrink Mutation Probability	Internal	4%
	External	4%
Branch Mutation Probability	Internal	4%
	External	4%
Probability of Delete Operation	8%	
Probability of Endowment Operation	8%	
Max. Number of Basis Functions	7	
Add Best Solutions to s New Generation	Yes	
Number of Best Solution Used in Elitism	50	
Use Contribution Operation	Yes	
Frequency of Contribution Operation F_{co}	Each 3 generations	
The error reduction Value	0.01	
Use Merging Operation	Yes, in all generations for all individuals	
Number of Iterations Applying by Numeric Constant Mutation	10 Iterations for each individual in the population, and 500 iterations for the elected individuals by the elitism operation	

VI. CONCLUSIONS

As a central conclusion is the possible use of GP for identifying a class of nonlinear discrete dynamic systems. MBFGP algorithm shows better results than the standard GP especially when data is incorporated with measurement noise. In terms of the number of successful runs and required number of generations, a comparison is carried out between four parameter optimization techniques. The numeric constant mutation technique seems to be the best choice. Perfect model finding is achieved when the MBFGP algorithm is enhanced with a numeric constant mutation technique for all linear in parameter models. Furthermore, the MBFGP enhanced by numeric constant mutation gives acceptable identification results even with complex dynamic systems. Although the identification process gives a different mathematical expression as compared to the original model, the accuracy of matching is better than with other methods. The accuracy of the identification process can be controlled for a customize computation time.

REFERENCES

[1] Isermann, Rolf Munchhof, Marco. 2011. Identification of Dynamic Systems An Introduction with Applications, Springer-Verlag Berlin Heidelberg
[2] Fatin Mohamed Ali. 2002. Neural Network Based Identification Utilizing Genetic Algorithm. Comm. M.Sc. Thesis. University of Technology, Baghdad
[3] Amir Gandmi, Amir H. Alavi, Conor Ryan (Ed). 2015. Handbook of Genetic Programming Application. Springer International Publishing , Swaziland
[4] Xiao-lei Yuan, Yan Bai, Ling Dong. 2008. SSCH: Identification of Linear Time-Invariant, Nonlinear and Time Varying Dynamic Systems Using Genetic Programming. Evolutionary Computation, The IEEE World Congress on Computation Intelligence, Honk Kong, 56-61.
[5] J. R. Koza. 1995. Survey of Genetic Algorithms and Genetic Programming, Proceeding on 1995 WESCON, Piscata way, NJ, IEEE 7-9 Nov. 589-594
[6] J. R. Koza 1992. Genetic Programming: On Programming of Computers by Mean of Natural Selection. Cambridge, MA, The MIT press

[7] A. Topchy, W. Punch. 2001. Faster Genetic Programming based on local Gradient Search of Numeric Leaf Values, GECCO-2001, 155-162, Morgan Kaufmann

[8] Rami A. Maher, Mohammed J. Mohammed 2013. An enhanced Genetic Programming Algorithm for Optimal Controller Design, Intelligent Control and Automation, Published Online (http://www.scirp.org/journal/ica)

[9] Rami A. Maher 2013. Optimal Control Engineering with MATLAB, Nova Science, NY

[10] L. Zhang, L. Wang 2003. Optimal Parameters Selection for Simulate Annealing with Limited Computation Effort. IEEE Conference, Neural Networks and Signal Processing, Nanjing, China

[11] Michael Affenzeller, Stephan Winkler, Stefan Wagner, Andreas Beham 2009. Genetic Algorithm and Genetic Programming Modren Concepts and Practical Applications, CRC Press Taylor & Francis, NY

[12] G.P. Liu, V. Kadirkamamathan, S.A. Billings 1998, On-Line Identification of Nonlinear Systems Using Volterra Polynomial Basis Function Neural Networks, Neural Networks No. 11 p.p 1645-1657