International Journal of Academic Scientific Research
ISSN: 2272-6446 Volume 8, Issue 4 (November - December2020), PP 01- 14
www.ijasrjournal.org

Solving Boundary Value Problems for Partial Differential Equations Using Modified Genetic Programming

Bachir Nour Kharrat[1*], Mohamed Khatib[2] and Shaza Alturky[3]

[1,3]*Department of Mathematics, Faculty of Science, Aleppo University, Syria*
[2]*Department of Artificial Intelligence and Natural Languages, Faculty of Informatics Engineering, Aleppo University, Syria*
[*]**Corresponding author**: Prof. B.N. Kharrat, *E-mail address: bachir.kharat@gmail.com*

Abstract: *This paper presents a modified genetic programming to find exact or approximate solution of various nonlinear boundary value problems represented by partial differential equations with boundary conditions. Genetic programming is a metaheuristic algorithm that belongs to the evolutionary algorithms, which simulates some of the ideas of natural evolution inspired by the biological evolution and forms generations of trial solutions expressed in an analytical closed form. In this work, we propose a modification of genetic programming by adding a new step at the end of each generation in which the worst individual (has the largest fitness value) is replaced by the best individual (has the smallest fitness value), with the aim of strengthening the control of the best individual through generations and guiding the exploration of the search space towards the optimum region. To demonstrate the accuracy and effectiveness of the proposed algorithm, we present many numerical examples, and the results we obtained showed the strength of the proposed technique. On the one hand, we compared our results with the simple genetic programming to demonstrate the speed of convergence through generations. On the other hand, we compared our results with classic methods to ensure the accuracy and efficiency of the proposed algorithm.*

Keywords *- Boundary value problems, Modified genetic programming, Non-linear partial differential equations, Simple genetic programming.*

I. INTRODUCTION

The nonlinear partial differential equations arise in various engineering, physical, chemical and hydrodynamic applications. Therefore, the researchers focused their attention on solving this type of problems using various numerical and analytical techniques but there are some difficulties such as calculating integrals, choose a suitable initial approximation as in the homotopy perturbation method and make some transforms as in the natural transform. So, we suggested using one of the metaheuristic algorithms as an optimization tool and effective technique to solve boundary value problems represented by higher order non-linear partial differential equations. Many techniques have been proposed in the literature to solve BVPs, Tsoulos et al. [1] used a genetic programming based on grammatical evolution to solve various differential equations, Jebari et al. [2] solved Poisson equation (PE) with a method based on genetic algorithms and grammatical evolution, Entesar et al. [3] suggested a hybrid technique combined the homotopy analysis method (HAM) with genetic algorithm (GA) for solving fractional partial differential equations. Navarro et al. [4] solved ordinary differential equations using genetic algorithms and the Taylor series matrix method, Panagant et al. [5] proposed an alternative meshless approach to solve partial differential equations based on a new efficient differential evolution, Kadri et al. [6] applied the genetic algorithms in nonlinear heat conduction problems, Hussain et al. [7] proposed a modified genetic algorithm for solving ordinary and partial differential equations, Biazar et al. [8] discussed homotopy perturbation method to obtain exact or approximate solutions for some linear and nonlinear partial differential equations, Spevaka et al. [9] solved a two-dimensional nonlinear heat conduction equation with nonzero boundary conditions by the boundary element method, Pourgholi et al. [10] solved an inverse heat conduction problem using genetic algorithm, Ming Li et al. [11] investigated applications of the method of fundamental solutions (MFS) for the numerical solution of two-dimensional boundary value problems in complex geometries, governed by the Laplace equation and subject to Dirichlet boundary conditions which are not harmonic, Afshar et al. [12] solved the two-dimensional second-order diffusion equation with nonlocal boundary condition, Rawashdeh et al. [13] presented new approximate solutions to fractional nonlinear systems of partial differential equations using the fractional natural decomposition method (FNDM), Cheniguel et al. [14] showed numerical simulations to the two- dimensional diffusion equation using decomposition method, Akter et al. [15] used the homotopy perturbation method for solving highly nonlinear reaction-diffusion-convection problem, Eladdad et al. [16] worked on coupling the homotopy perturbation method with new integral transform for solving systems of partial differential equations, Seaton et al. [17] presented analytic solutions for differential equations under graph-

based genetic programming. Kharrat et al. have been interested in the metaheuristic algorithms, such as suggested hybridization of genetic programming with homotopy perturbation method (HPM) for solving nonlinear heat transfer equations [18], and also expanding the application of genetic algorithm for solving singular boundary-initial value problems arising in physiology applications [19]. Jesuraj and Rajkumar [20] worked on differential equations and problems in engineering and science using a new modified Sumuda transform.

Our main objective in this work is to propose a new modification of the genetic programming that contributes to increasing the speed of convergence towards the optimal region in the search space, and also to obtain exact or approximate solutions of some nonlinear partial differential equations that are difficult to solve using classical methods. The metaheuristic algorithm (GP) is employed to search for the optimum solution which makes the fitness function is minimized.

The organization of the remainder of this paper is as follows: Section 2 introduces a literature review on the simple genetic programming. Section 3 provides a description of the modified genetic programming, Section 4 includes the methodology, Section 5 outlines the numerical example and the experimental results. Finally, concluding remarks are showed in Section 6.

II. SIMPLE GENETIC PROGRAMMING

Genetic Programming (GP) is introduced by Dr. John Koza in 1992, GP is a systematic method for getting computers to automatically solve a problem starting from a high-level statement of what needs to be done. GP iteratively transforms a population of computer programs into a new generation of programs by applying the genetic operations. The genetic operations include crossover, mutation, and reproduction. Genetic programming is an extension of the genetic algorithm GA (Holland 1975) in which the structures in the population are not fixed-length character strings that encode candidate solutions to a problem, but programs that, when executed, are the candidate solutions to the problem [21].

Programs are expressed in genetic programming as syntax trees rather than as lines of code. For example, the simple expression (√(x+3/y)) is represented as shown in Figure. 1. The tree includes nodes (which we will also call point) and links. The nodes indicate the instructions to execute. The links indicate the arguments for each instruction. In the following the internal nodes in a tree will be called functions, while the tree's leaves will be called terminals.

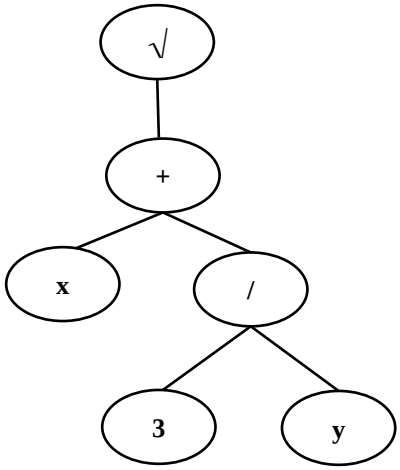

Fig. 1. The tree representation of a GP model(√(x+3/y)) [21].

Note how the variables and constants in the program (x, y, and 3), called terminals in GP, are leaves of the tree, while the arithmetic operations (√, +, and /) are internal nodes (typically called functions). The sets of allowed functions and terminals together form the primitive set of a GP system.

To apply a GP these are often termed the five major preparatory steps. The key choices are [22]:

1. **The set of terminals**:
   - Constants these can be pre-specified, randomly generated as part of the tree creation process, or created by mutation.
   - Variables (e.g., x, y).
   - Functions with no arguments such as the function rand().

2. **The set of functions**,

- Arithmetic function: +, -, *, / .
- Mathematical function: sin, cos, Exp, log, sqrt, ^,…
- Boolean function: And, Or, Not…
- Loop function: For-Repeat
- Conditional function: If-Then-Else

3. **The fitness function** for measuring the fitness values of individuals in the population.

4. **Certain parameters** for controlling the run, where the most important control parameter is the population size. Other control parameters include the probabilities of performing the genetic operations, the maximum depth tree and other details of the run.

5. **The termination criterion** for 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**.**

Algorithmically, Simple GP comprises the following executional steps:

Algorithm (1): Simple Genetic Programming

1. Randomly create an initial population of programs from the available primitives.
2. **Repeat**
3. **Execute** each program and ascertain its fitness.
4. **Select** one or two program(s) from the population with a probability based on fitness to participate in genetic operations.
5. **Create** new individual program(s) by applying genetic operations with specified probabilities.
6. **Until** stopping condition is met (e.g., reaching a maximum number of generations).

Figure. 2 shows the flowchart of the genetic programming.

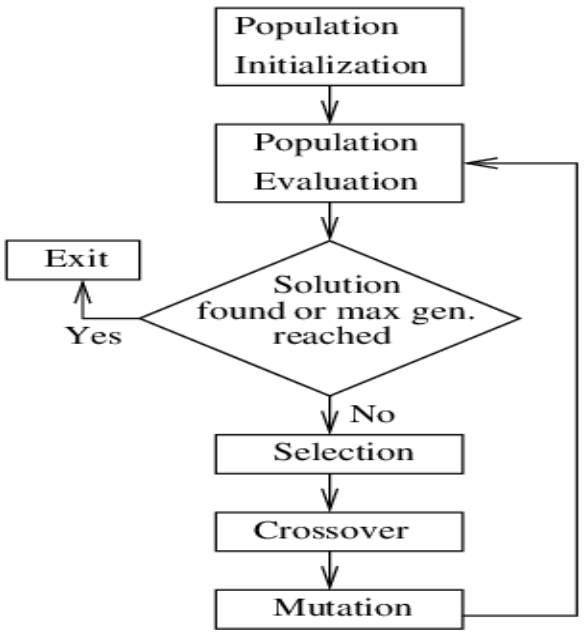

Fig. 2. Genetic programming flowchart [22].

III. MODIFIED GENETIC PROGRAMMING DESCRIPTION

In this part, we will describe the proposed modified GP, where the main idea of the modified GP technique is to ensure the control (survival) of the best individual through generations in order to increase the speed of convergence towards the best solution, so we suggested adding a new step at the end of each generation in which the worst individual (has the largest fitness value) is replaced with the best individual (has the smallest fitness value), to guide exploration of the search space towards the optimal region. It is noted that the modified genetic programming algorithm was applied for the small population size that contains 10 individuals, and this requires an increase in the number of generations to reach the optimal solution, while if the population size is relatively large, this requires fewer generations. The following executional steps explain the proposed modified GP:

Algorithm (2): Modified Genetic Programming

1. Randomly create an initial population of programs from the available primitives.
2. **Repeat**
3. **Execute** each program and ascertain its fitness.
4. **Select** one or two program(s) from the population with a probability based on fitness to participate in genetic operations.
5. **Create** new individual program(s) by applying genetic operations with specified probabilities.
6. **Replace** the worst individual with the best (New Step)
7. **Until** stopping condition is met (e.g., reaching a maximum number of generations).

After implementing the proposed algorithm on a number of applications and repeating it for a large number of generations, we found that the modified GP has the following advantages:

- Retain the best solution or so far the best solution through generations.
- Increase the speed of convergence to reach the best solution.
- Generating possible solutions neighborhood the best solution, which supports exploitation the local search as well.
- The balance between exploitation and exploration.

IV. METHODOLOGY

In this part, we will explain how genetic programming works for solving partial differential equations (PDEs) with boundary conditions. The PDEs can be expressed in the following form [23]:

$$f\left(x, y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}\right) = 0 \qquad (1)$$

With $x \in [a, b] and\ y \in [c, d]$ The boundary conditions are given by:

$$u(\mathrm{a}, \mathrm{y}) = f_0(y)\ ,\ \ u(\mathrm{b}, \mathrm{y}) = f_1(y),\ \ u(x, \mathrm{c}) = g_0(x), u(x, \mathrm{d}) = g_1(x)$$

Where $u = u(x, y)$, Now, to solve PDE with GP, the five major preparatory steps should be specified such that:

1. **Terminal set:** { x, y, random constants}
2. **Function set:** {+, -, *, /, Sin, Cos, ^, sqrt, log, Exp, Tan}
3. **Fitness function (or Error Function) [23]:** The steps for the calculation of the fitness for any individual are:
   a) Choose N uniformly distributed points in the box$[a, b]$ x $[c, d]$.
   b) Choose $N_x$equidistant points in $[a, b]$.
   c) Choose $N_y$equidistant points in $[c, d]$.
   d) Calculate the error of PDE:

$$\varepsilon_1 = \frac{1}{N}\sum_{i=1}^{N} f\left(x, y, M, \frac{\partial M}{\partial x}, \frac{\partial M}{\partial y}, \frac{\partial^2 M}{\partial x^2}, \frac{\partial^2 M}{\partial y^2}\right)^2 \tag{2}$$

c) Calculate the error of boundary conditions:

$$\varepsilon_2 = \frac{1}{4}\Big(\sum_{i=1}^{N_y}\big(M(a, y_{bi}) - f_0(y_{bi})\big)^2 + \sum_{i=1}^{N_y}\big(M(b, y_{bi}) - f_1(y_{bi})\big)^2 + \sum_{i=1}^{N_x}\big(M(x_{bi}, c) - g_0(x_{bi})\big)^2 + \sum_{i=1}^{N_x}\big(M(x_{bi}, d) - g_1(x_{bi})\big)^2\Big) \tag{3}$$

d) Calculate the fitness function:

$$Minimization \leftarrow Fitness\ Function \quad \varepsilon = \varepsilon_1 + \varepsilon_2$$

Where N_x equidistant points on the boundary at x = a and at x = b, N_y equidistant points on the boundary at y = c and at y = d, M=M(x, y) is the trial solution expressed in the GP, ε_1 is the mean square error of differential equation, ε_2 is the mean square error of boundary conditions, ε is the fitness function represents the mean square error, subject to the availability of the parameters, such that $\varepsilon \to 0$, in case of both $\{\varepsilon_1, \varepsilon_2\} \to 0$, then the approximate results closer to the exact solution.

4. **Control Parameters**: Population size, Crossover probability, Mutation probability, Max depth tree.
5. **Stopping condition**: maximum number of generations.

V. NUMERICAL EXAMPLES

In this section, we will explain the feasibility of applying the proposed GP and its effectiveness through some nonlinear partial differential equations.

5.1. Example (1)

Consider the following nonlinear third order partial differential equation with boundary conditions [24]:

$$\begin{cases} \dfrac{\partial^3 u}{\partial x^3} = y\left(\dfrac{\partial^2 u}{\partial x^2}\right)\left(\dfrac{\partial u}{\partial y}\right) + 6 - 12\,xy^2 \\ \mathrm{u}(0, \mathrm{y}) = y^2\ , \quad \dfrac{\partial\, \mathrm{u}(0, \mathrm{y})}{\partial x} = 0 \\ \mathrm{u}(\mathrm{x}, 0) = x^3\ , \quad \dfrac{\partial^2\, \mathrm{u}(0, \mathrm{y})}{\partial x^2} = 0 \end{cases} \tag{4}$$

To implement the GP, we designed a windows form application in visual C # as in Figure. 3:

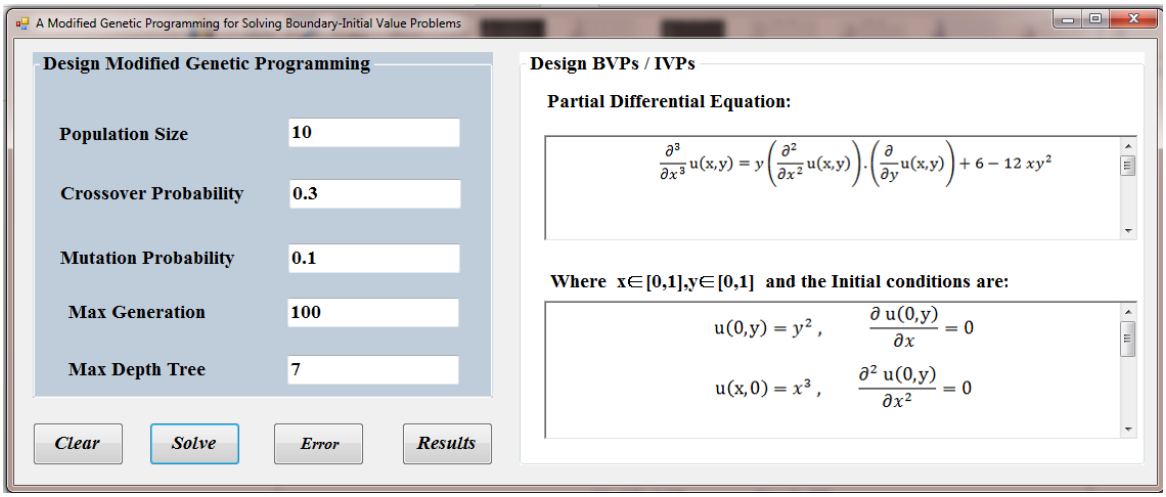

Fig. 3. Windows application C# for Example (1)

The output of the previous C# application provided us with optimal values for the parameters of modified GP, and the approximate solution as follows:

$$u(x,y) = x^3 + y^2 - 11 \times 10^{-30}x^3y$$

Table 1 presents a comparison of the absolute errors between our presented modified GP and the hybrid method combined natural transform (NT) with homotopy perturbation method (HPM) given in [24].

Table 1. Comparison of the absolute error for Example (1)

(x, y)	Hybrid (NT-HPM) [24]	Modified Genetic Programming
(0,0)	0	0
(0.002,0.002)	7.6877 E -16	1.320000 E-31
(0.004,0.004)	4.9250 E -14	2.64000 E-31
(0.006,0.006)	5.6155 E -13	3.9600 E-31
(0.008,0.008)	3.1583 E -12	5.2800 E-31
(0.02,0.02)	7.7568 E -10	1.3200 E-30
(0.04,0.04)	5.0133 E -08	2.640 E-30
(0.06,0.06)	5.7658 E -07	3.96 E-30
(0.08,0.08)	3.2703 E -06	5.28 E-30
(0.1,0.1)	1.2591 E -05	6.6 E-30
(0.2,0.2)	8.3939 E -04	1.32 E-29
(0.3,0.3)	9.8271 E -03	1.8 E-29
(0.4,0.4)	5.5548 E -02	2.2 E-29
(0.5,0.5)	-	2.0 E-29
(0.6,0.6)	-	3.0 E-29
(0.7,0.7)	-	2.0 E-29
(0.8,0.8)	-	4.0 E-29
(0.9,0.9)	-	6.0 E-29
(1,1)	-	1.0 E-28

In Table 2, we list experimental results for observing the progress of (best and worst) solution through generations, as we can notice, the proposed GP is more convergent towards the best solution than the simple GP.

Table 2. Comparison of experimental results of proposed modified GP and simple GP for Example(1)

Using Modified Genetic Programming				
Generation No.	**The Worst Individual**		**The Best Individual**	
	Trial solution (InOrder)	**Fitness value**	**Trial solution (InOrder)**	**Fitness value**
1	-4*x^2+(y^4-1)	261.6357380825	x^3+y^2-(x/120)-(y/720)	0.000259900424382716
10	x^3-y^2	116.489428	x^3+y^2-0.001*y	1.00823799999994 E-05
40	(x^2-1)(y^2-1)	56.0628820678128	x^3+y^2-(sqrt(y)/2019870)	3.00621968905105 E-12
60	x^2*exp(y)	34.5301947002135	x^3+y^2-2 E-12*y	1.24784364047187 E-24
80	x-y	26.72780825	x^3+y^2-3.22 E-15*x^3*y	2.13035281906461 E-28
100	((x*y)/2)+y*y	21.66425825	x^3+y^2-11 E-30*x^3*y	4.03723064592174 E-32
Using Simple Genetic Programming				
Generation No.	**The Worst Individual**		**The Best Individual**	
	Trial solution (InOrder)	**Fitness value**	**Trial solution (InOrder)**	**Fitness value**
1	2+x*y^2-5*x^2*y	250.1194717	x^3+y^2-Sin(0.05*x)	0.0102237329418139
100	4*x*x-y*y	105.40291325	x^3+y^2-1.0/(105.79693- y)	0.000449128944511318
200	4*x-22.13*y	52.6946609125	x^3+y^2-x/230	6.54536862003781E-05
300	Sqrt(x+1) +y	26.3655325893269	x^3+y^2-14.31074 E-15*y	2.04766766514597 E-27
400	x+y	23.595682	x^3+y^2-11 E-30*x^3*y	4.03723064592174 E-32

Figure. 4 presents a comparison between the modified GP and the simple GP to show the speed of convergence over the generation, where the modified GP converged from the best solution at the generation 100, while the simple GP converged from the best solution at the generation of 400.

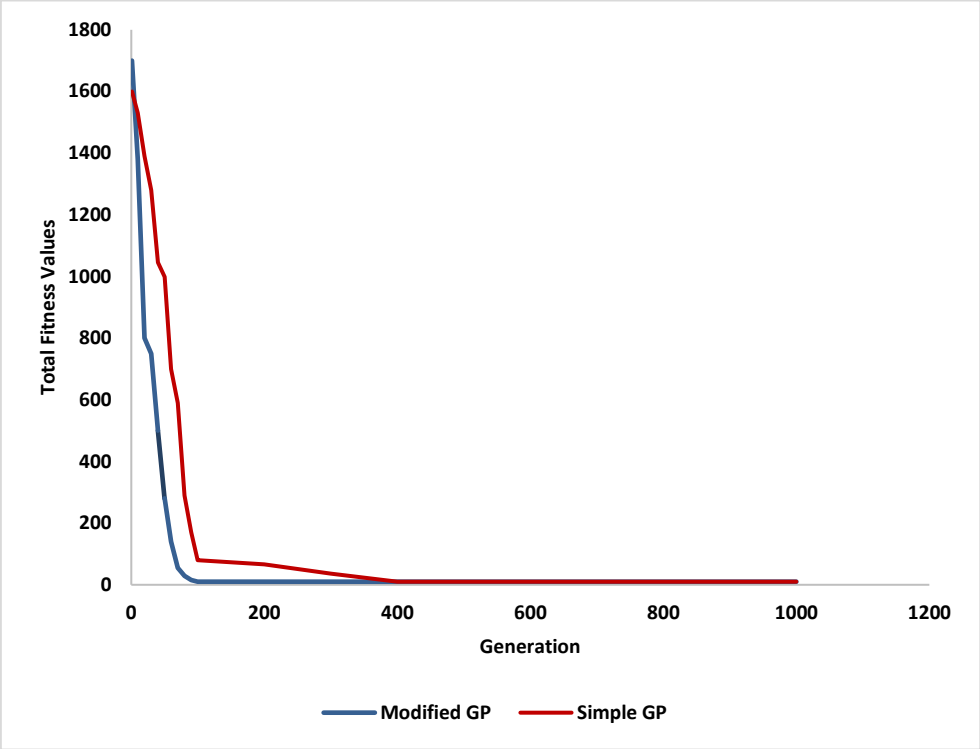

Fig. 4. Comparison between Simple GP and the proposed Modified GP for example 1

5.2. Example (2)

Consider the following nonlinear partial differential equation with boundary conditions [24]:

$$\begin{cases} \dfrac{\partial^4 u}{\partial y^4} = \left(\dfrac{\partial^3 u}{\partial x^2 \partial y}\right)^3 - 69\, y^9 e^{3x} + 384\, y^6 e^{2x} - 768\, y^3 e^x + 24 e^x + 512 \\ \mathrm{u}(\mathrm{x},0) = -1\ , \quad \dfrac{\partial\, \mathrm{u}(\mathrm{x},0)}{\partial y} = -4x^2 \\ \dfrac{\partial^2\, \mathrm{u}(\mathrm{x},0)}{\partial y^2} = 0\ , \quad \dfrac{\partial^3\, \mathrm{u}(\mathrm{x},0)}{\partial y^3} = 0 \end{cases} \tag{5}$$

Figure. 5 shows the windows application C# for design the modified GP:

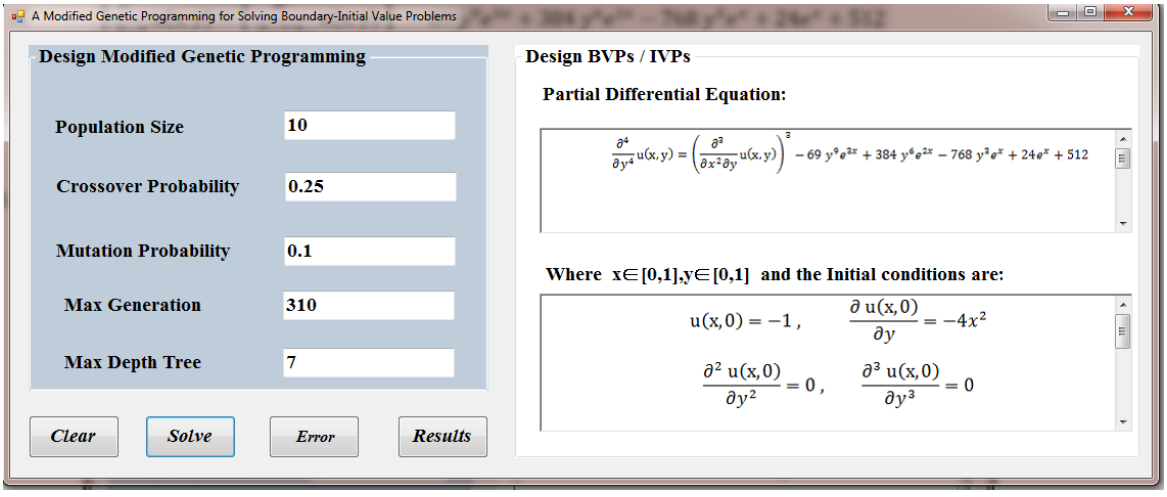

Fig. 5. Windows application C# for Example (2)

The output of the previous C# application provided us by the optimal values for the parameters of the GP, the approximate solution obtained by the modified GP is shown as:

$$u(x,y) = -4x^2y + e^x y^4 - 1$$

Table 3 shows a comparison of the absolute errors between our presented modified GP and the hybrid method combined natural transform (NT) with homotopy perturbation method (HPM) given in [24].

Table 3. Comparison of the absolute error for Example (2)

(x, y)	Hybrid (NT-HPM) [24]	Modified Genetic Programming
(0,0)	0	0
(0.002,0.002)	7.8801 E -14	2.57540617 E-24
(0.004,0.004)	5.0533 E -12	1.32654339 E-21
(0.006,0.006)	5.7676 E -11	5.13036847 E-20
(0.008,0.008)	3.2471 E -10	6.87389596 E-19
(0.02,0.02)	8.0231 E -08	2.71830155 E-15
(0.04,0.04)	5.2379 E -06	1.47783267 E-12
(0.06,0.06)	6.0850 E -05	6.03259631 E-11
(0.08,0.08)	3.4860 E -04	8.53120863 E-10
(0.1,0.1)	1.3553 E -03	6.74929403 E-09

In Table 4, the experimental results are listed for observing the progress of (best and worst) solution through generations, as we can notice, the proposed GP is more convergent towards the best solution than the simple GP.

Table 4. Comparison of experimental results of proposed modified GP and simple GP for Example (2)

Using Modified Genetic Programming				
Generation No.	**The Worst Individual**		**The Best Individual**	
	Trial solution (InOrder)	Fitness value	**Trial solution** (InOrder)	Fitness value
1	(x+y^3)*exp(-x)	627.621909387287	-4*x^2*y+y^3	23.4115317402234
100	x^2+y^2	522.311724347447	-4*x^2*y-exp(x)*y^2	15.3335115224022
200	-4*x^2*y-exp(x)*exp(y)	187.423622660079	-4*x^2*y+y^4-1	0.74632816552213
310	x*y+x+(y*y/2)	182.776284564988	-4*x^2*y+exp(x)*y^4-1	2.639156672471E-18
Using Simple Genetic Programming				
Generation No.	**The Worst Individual**		**The Best Individual**	
	Trial solution (InOrder)	Fitness value	**Trial solution** (InOrder)	Fitness value
1	x+y	514.839898091555	-4*x^2*y+exp(x)*y^2	35.3938428117492
200	Sin (x*y)	512.311634040512	-4*x^2*y+exp(x)*y^4+y^4 -exp(3*x)*y^10	22.7993456198442
400	-x*y^2+exp(x)*y	510.860234537959	-4*x^2*y+Exp(x)*y^4 -x*y^4-1	0.685620831583512
600	-4*x^2*y +exp(x)*y^5	23.2734239604527	-4*x^2*y+exp(x)*y^4-1	2.639156672471E-18

Figure. 6 presents a comparison between the modified GP and the simple GP to show the speed of convergence over the generation, where the modified GP converged from the best solution at the generation 310, while the simple GP converged from the best solution at the generation of 600.

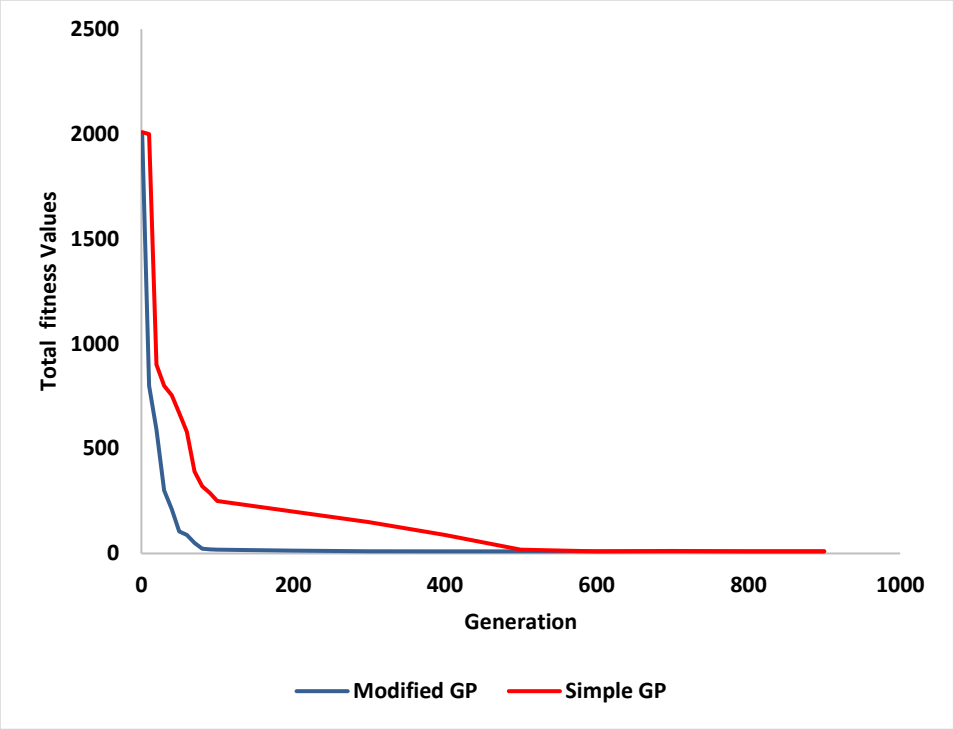

Fig. 6. Comparison between simple GP and the proposed modified GP for Example (2)

5.3. Example (3):

Consider the following elliptic equation with variable coefficients and homogeneous Dirichlet boundary conditions on $\Omega = [0,1]X\,[0,1]$ [25]:

$$\begin{cases} \dfrac{\partial}{\partial x}\left((2-x^2-y^2)\dfrac{\partial u}{\partial x}\right)+\dfrac{\partial}{\partial y}\left((e^{x-y})\dfrac{\partial u}{\partial y}\right)= \\ -16\,x(1-x)(3-2y)e^{x-y}+32y(1-y)(3x^2+y^2-x-2)\ , \quad (x,y)\epsilon\Omega \\ \mathrm{u(x,y)}=0, \qquad (x,y)\epsilon\partial\Omega \end{cases} \tag{6}$$

Figure. 7 Shows the windows form application in visual C # for Example (3):

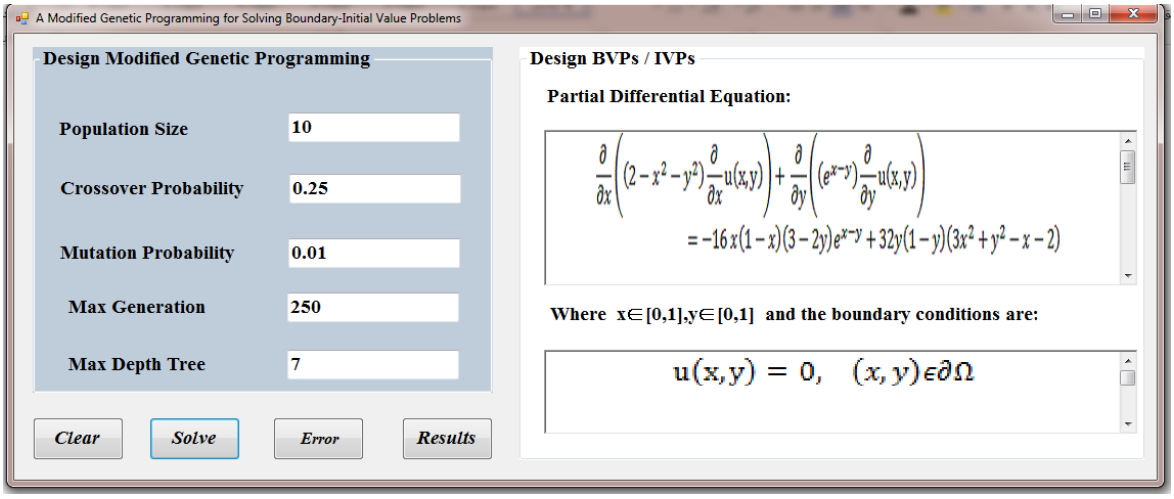

Fig. 7. Windows application C# for Example (3)

The exact solution obtained by the modified GP is:

$$u(x,y) = 16x(1-x)y(1-y)$$

Figure. 8 presents a comparison between the modified GP and the simple GP for observing the progress of solution through generations, as we can notice; the proposed GP is more convergent towards the best solution than the simple GP.

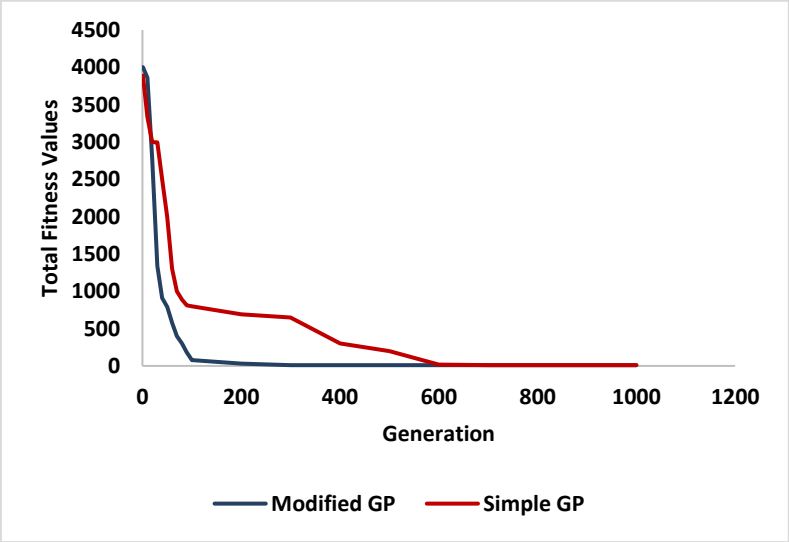

Fig. 8. Comparison between simple GP and the proposed modified GP for Example (3)

We can notice the speed of convergence over the generation, where the modified GP converged from the best solution at the generation 250, while the simple GP converged from the best solution at the generation of 610.

5.4. Example (4):

Consider the following boundary value problem [23]:

$$\begin{cases} \nabla^2 u = e^{-x}(x-2+y^3+6y) \ , & x \in [0,1] \ , \quad y \in [0,1] \\ u(0,y) = y^3 \ , & u(1,y) = (1+y^3)e^{-1} \\ u(x,0) = x\,e^{-x} \ , & u(x,1) = (x+1)\,e^{-x} \end{cases} \tag{7}$$

Figure. 9 Shows the windows form application in visual C # for Example (3):

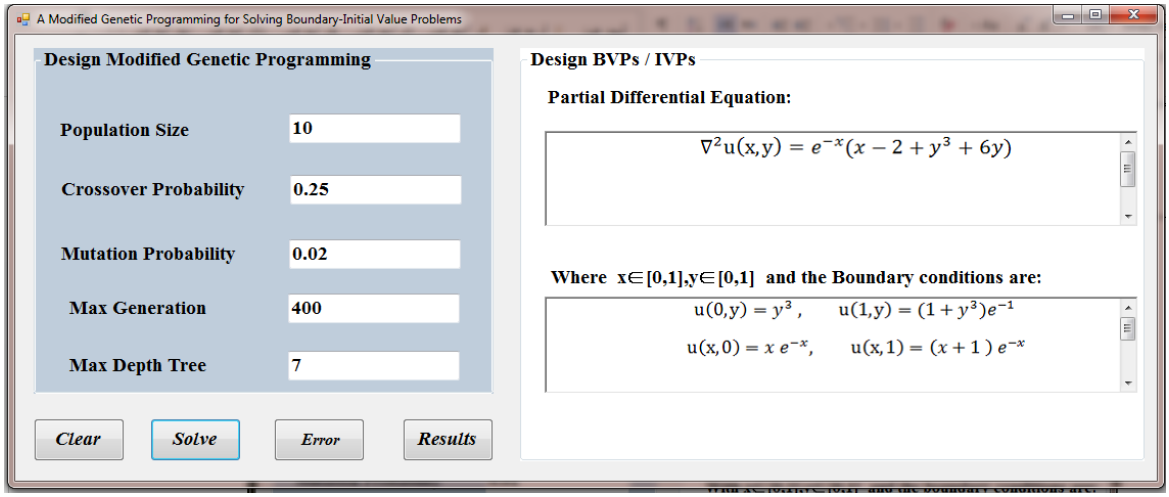

Fig. 9. Windows application C# for Example (4)

The exact solution obtained by the modified GP is:

$$u(x,y) = (x+y^3)e^{-x}$$

Figure. 10 presents a comparison between the modified GP and the simple GP for observing the progress of solution through generations, as we can notice; the proposed GP is more convergent towards the best solution than the simple GP.

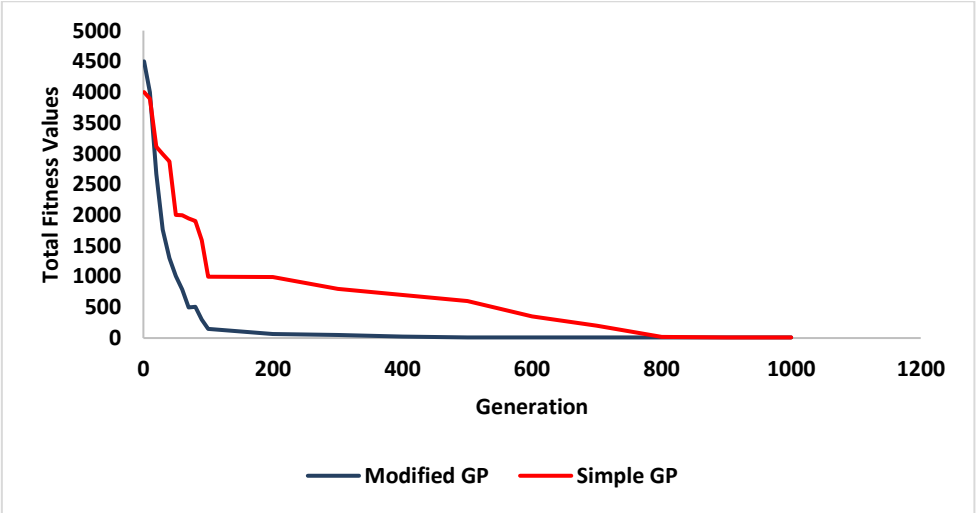

Fig. 10. Comparison between simple GP and the proposed modified GP for example (4)

We can notice the speed of convergence over the generation, where the modified GP converged from the best solution at the generation 400, while the simple GP converged from the best solution at the generation of 830.

5.5. Example (5)

Consider the following boundary value problem [23]:

$$\begin{cases} \nabla^2 u = (x-2)e^{-x} + xe^{-y} \;, & x \in [0,1] \;, \quad y \in [0,1] \\ u(0,y) = 0 \;, \quad u(1,y) = \sin(y) \\ u(x,0) = 0 \;, \quad u(x,1) = \sin(x) \end{cases} \tag{8}$$

Figure. 11. Shows the windows form application in visual C # for Example (5):

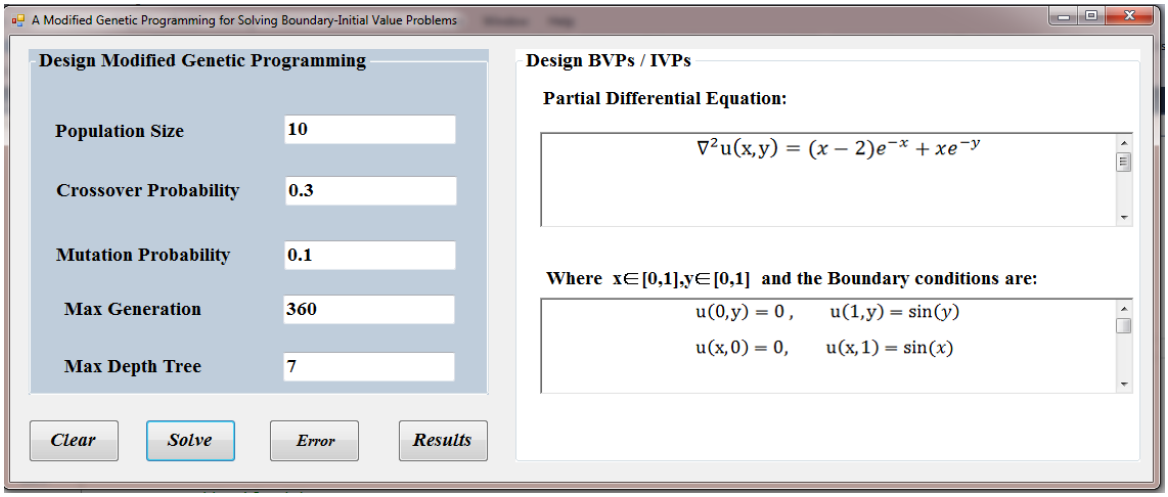

Fig. 11. Windows application C# for Example (5)

The exact solution obtained by the modified GP is:

$$u(x,y) = \sin(xy)$$

Figure. 12, presents a comparison between the modified GP and the simple GP for observing the progress of solution through generations, as we can notice; the proposed GP is more convergent towards the best solution than the simple GP.

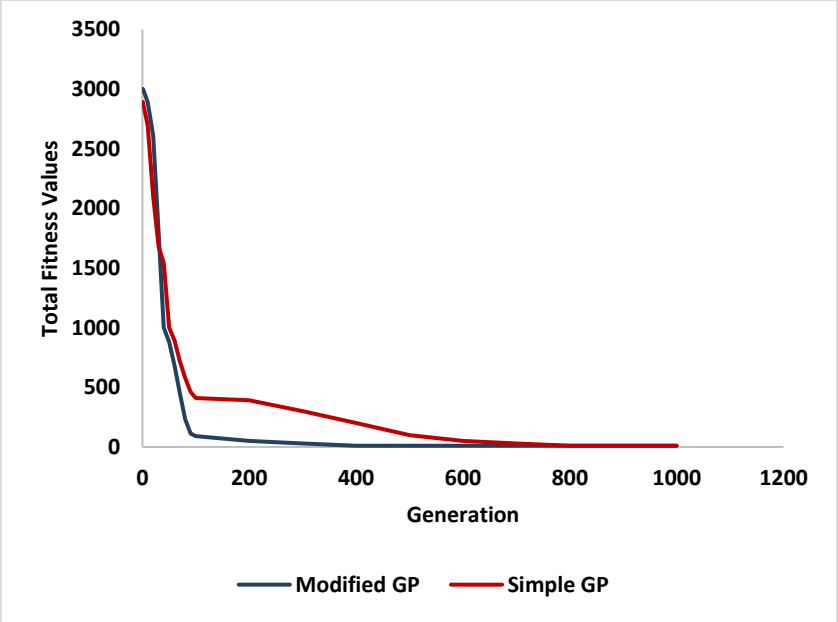

Fig. 12. Comparison between simple GP and the proposed modified GP for Example (5)

We can notice the speed of convergence over the generation, where the modified GP converged from the best solution at generation 360, while the simple GP converged from the best solution at the generation of 800.

VI. CONCLUSION

In this work, we proposed a new modified genetic programming to solve nonlinear boundary value problems for partial differential equations. The accuracy and effectiveness of the proposed algorithm are illustrated by solving some of nonlinear problems. When the solution cannot be expressed in an analytical closed form then our proposed method produces an approximate solution with a level of accuracy. We proposed the idea of modifying the GP algorithm and tested its performance on

the small population sizes and the results were effective as the speed of convergence increased significantly compared to the simple GP algorithm. We suggest ,as a future work, to extend the application of the modified GP for handling the large population sizes. In addition, we report the experimental results to show the progress of the best solution through generations.

REFERENCES

[1] I. G. Tsoulos, I. E. Lagaris, Solving Differential Equations With Genetic Programming, *Genetic Programming and Evolvable Machines*, 2006.
[2] Kh. Jebari, M. Madiafi, A. El Moujahid, Solving Poisson Equation by Genetic Algorithms, *International Journal of Computer Applications*, 2013.
[3] A. Entesar, O. Saber, W. Al-Hayani, Hybridization of Genetic Algorithm with Homotopy Analysis Method for Solving Fractional Partial Differential Equations, *Eurasian Journal of Science & Engineering*, 2019.
[4] D. G. Navarro, S. L. Aguayo, Solving Ordinary Differential Equations Using Genetic Algorithms And The Taylor Series Matrix Method, *Journal of physics communications*, 2018.
[5] N. Panagant, S. Bureerat, Solving Partial Differential Equations Using a New Differential Evolution Algorithm, *Mathematical Problems in Engineering*, 2014.
[6] M. B. Kadri, W. A. Khan, Application of Genetic Algorithms in Nonlinear Heat Conduction Problems, *The Scientific World Journal*, 2014.
[7] E. A. Hussain, Y. M. Abdul – Abbass, Solving Differential Equation by Modified Genetic Algorithms, *Journal of University of Babylon for Pure and Applied Sciences*, 2018.
[8] J. Biazar, H. Ghazvini, Convergence Of The Homotopy Perturbation Method For Partial Differential Equations. *Nonlinear Analysis: Real World Applications*, *Elsevier*, 2009.
[9] L. F. Spevaka and O. A. Nefedova , Solving a Two-Dimensional Nonlinear Heat Conduction Equation with Nonzero Boundary Conditions by the Boundary Element Method, *AIP Conference Proceedings*, 2017.
[10] R. Pourgholi, H. Dana, S. H. Tabasi, Solving An Inverse Heat Conduction Problem Using Genetic Algorithm, *Applied Mathematical Modelling*, *Elsevier*, 2014.
[11] M. Li, C.S. Chen, A. Karageorghis, The MFS For The Solution Of Harmonic Boundary Value Problems With Non-Harmonic Boundary Conditions, *Computers and Mathematics with Applications*, *Elsevier*, 2013.
[12] S. Afshar, B. Soltanalizadeh, Solution Of The Two-Dimensional Second-Order Diffusion Equation With Nonlocal Boundary Condition, *International Journal of Pure and Applied Mathematics*, 2014.
[13] M. S. Rawashdeh, H. Al-Jammal, New Approximate Solutions To Fractional Nonlinear Systems Of Partial Differential Equations Using The FNDM, *Advances in Difference Equations*, *Springer*, 2016.
[14] A. Cheniguel, Numerical Simulation of Two-Dimensional Diffusion Equation with Non Local Boundary Conditions, *International Mathematical Forum*, 2012.
[15] M. T. Akter, M. A. Chowdhury, Homotopy Perturbation Method for Solving Highly Nonlinear Reaction-Diffusion-Convection Problem, *American Journal of Mathematics and Statistics* 2019.
[16] E. E. Eladdad and E. A. Tarif , On the Coupling of the Homotopy Perturbation Method and New Integral Transform for Solving Systems of Partial Differential Equations, *Advances in Mathematical Physics*, 2019.
[17] Tom Seaton1, Gavin Brown2, and Julian F. Miller1 ,Analytic Solutions to Differential Equations under Graph-Based Genetic Programming", *EuroGP Springer-Verlag Berlin Heidelberg* 2010
[18] B. N. Kharrat, M. Khatib, Sh. Alturky, Hybrid Evolutionary Algorithm with Homotopy Perturbation Method for Solving Nonlinear Heat Transfer Equations. *International Journal of Academic Scientific Research*. 2019.
[19] B. N. Kharrat, M. Khatib, Sh. Alturky, Numerical Solution of Singular Boundary Value Problems Using Genetic Algorithm. *International Journal of Academic Scientific Research*. 2019.
[20] C. Jesuraj, A. Rajkumar, A New Modified Sumudu Transform Called Raj Transform To Solve Differential Equations And Problems In Engineering And Science, *International Journal on Emerging Technologies*, 2020.
[21] R. Poli, B. Langdon, F. N. McPhee, J. R. Koza, Genetic Programming An Introductory Tutorial and a Survey of Techniques and Applications. *Technical Report CES-475 ISSN: 1744-8050*, 2007.
[22] Riccardo Poli , William B. Langdon , Nicholas F. McPhee , John R. Koza , A Field Guide to Genetic Programming,. *Computational Intelligence: A Compendium* , *Springer*, 2008.
[23] I. G. Tsoulos, D. Gavrilis, E. Glavas, Solving Differential Equations With Constructed Neural Networks, *Neurocomputing*, *Elsevier*, 2009.
[24] B. N. Kharrat, George A. Toma, A Hybrid Homotopy Perturbation Method with Natural Transform to Solve Partial Differential equations, *International Journal of Scientific Research in Mathematical and Statistical Sciences*, 2020.
[25] M. Esmaeilbeigi, M.M. Hosseinim, A New Approach Based On The Genetic Algorithm For Finding A Good Shape Parameter In Solving Partial Differential Equations By Kansa's Method, *Applied Mathematics and Computation*, *Elsevier*, 2014.