TWMS J. App. Eng. Math. V.9, N.4, 2019, pp. 810-821

A MODIFICATION OF ARTIFICIAL BEE COLONY ALGORITHM FOR SOLVING INITIAL VALUE PROBLEMS

K. GÜNEL[1], İ. GÖR[1], §

Abstract. In this paper, some improvements have been made on Artificial Bee Colony (ABC) algorithm to get numerical solutions of both linear and nonlinear differential equations as initial value problems. The solutions are obtained by a feed-forward neural network trained by the modified ABC.

Keywords: Initial Value Problems, Ordinary Differential Equations, Global Optimization, Artificial Bee Colony.

AMS Subject Classification: 68T20, 68T05, 65L05

1. Introduction

Artificial Neural Networks (ANNs) are effective tools for the solution of world problems. There are many different implementation area for ANNs including solving Differential Equations (DEs) numerically with ANNs. Differential Equations (DEs) are very crucial branch of mathematics. The systems are modeled using DEs for solving real problems in many disciplines. On the other hand, obtaining the solution of some special types of differential equations, such as the motion equation of oscillating swing, may not be possible in the continuous space, even knowing the existence of the solution. In order to encounter this problem, numerical solutions of DEs are searched in the interval having specific nodes in it. However the another problem arises in this case. The discrete solution occurs by using numerical approaches. The solution is acquired only some nodes instead of at all points in the interval. One of the method is interpolation in order to get the solution of other points. But, in such a case, not only the error which is appeared using numerical methods but also the error is emerged using interpolation. In last two decades, another approach is presented about this topic as the numerical solution of DEs with Artificial Neural Networks (ANNs). In this manner, some different approaches are studied for the numerical solution of DEs.

Artificial Neural Networks (ANNs) are utilized for solving numerous mathematical problems in the literature. Lee and Kang (1990) get the solution of Ordinary Differential Equations (ODEs) by using ANNs [1]. In their work, the discretization of ODEs are obtained

[1] Adnan Menderes University, Faculty of Arts and Sciences, Department of Mathematics, 09010, Aydın.
e-mail: kgunel@adu.edu.tr; ORCID: https://orcid.org/0000-0002-5260-1858.
e-mail: iclal@adu.edu.tr; ORCID: https://orcid.org/0000-0002-1999-8283.

§ Manuscript received: August 25, 2017; accepted: September 8, 2017.

with finite difference methods and Hopfield Neural Network is used to minimize cost function, which is constructed from differential equations. Malek and Beidokhti (2006) propose an hybrid method with ANNs which is trained by optimization methods for solving first and high order ODEs [2].

The studies are not limited to solve Ordinary Differential Equations (ODEs), in addition some types of Partial Differential Equations (PDEs) are solved with ANNs including initial and boundary value conditions. The studies including different approaches are outlined in the following.

Lagaris et al. (1998) create a method for solving initial and boundary value problems with ANNs including a trial function having two parts [3]. The first part is constructed for satisfying initial and boundary conditions and the other part is for the parameters which are adjusted in Feedforward Neural Network. The authors solve systems of ODEs and PDE besides ODEs in their study. Aarts and Van Der Veer (2001) solve PDEs with Feedforward Neural Network with a white box character [4]. The authors specifically construct the network is trained by an evolutionary algorithm. McFall and Mahan (2009) propose neural network model for the boundary conditions [5]. In the training stage, the network produces error. Then, the weights of ANN are updated to decrease the error. Beidokhti and Malek (2009) solve the initial and boundary value problems using ANNs of which parameters are determined by hybrid method based on Kolmogorov and Cybenko theorms [6]. Tsoulos et al. (2009) solve ODEs, Systems of ODES (SODEs) and PDEs via Feedforward Neural Networks constructed with grammatical evolution and improve the approach with local optimization [7]. In their study, trial solutions are created by hybrid method in neural network with a scheme including grammatical evolution.

Some studies are about the numerical solution of some types of DEs. Anastassi (2014) construct an ANNs method that can produce the best coefficients of two stage Runge-Kutta methods [8]. Raja et al. (2015) create an approach based on neural networks containing optimization with computational intelligence method sequential quadratic programming for the solution of nonlinear Riccati differential equations [9]. Kumar and Yadav (2015) reach approximated solution of one dimensional Bratu's problem using Multi Layer Perceptron (MLP) neural network algorithm [10].

In addition, global optimization techniques are used to get the numerical solution of DEs. Raja et al. (2016) study about numerical solution of nonlinear singular Flierl-Petviashivili equations utilizing ANNs and optimization algortihms as Genetic Algorithms (GAs), Sequential Quadratic Programming (SQP) and their combinations [11].

In this work, we structure a Feedforward Neural Network (FNN)in order to get the solution of Initial Value Problems (IVPs). The output of the network depends on the trial function, which satisfies the initial conditions. Furthermore, to minimize the cost function consisting with the derivative of the trial function, we modified the ABC algorithm as a metaheuristic optimization algorithm, and use it to train the neural net. Our original contribution is to made the improvements on ABC for solving IVPs. In experiments we show that the modifications enhance the exploration and exploitation capability of the ABC. The obtained results show the ability of ABC solution of IVPs.

In the sequel, first we briefly explain the ABC algorithm. Then, we propose some improvements on classical ABC algorithm, and we describe that how the network is constructed and trained by the modified version of ABC for the numerical solution of IVPs. In addition, we present experiments including the numerical solution of ODEs. In the fourth section, the some experimental studies are given. Final section presents the limitations and findings of the study.

2. Artificial Bee Colony (ABC) Algorithm

Artificial Bee Colony (ABC) Algorithm is inspired by the behaviour of honey bees [12]. The details of the behaviour of honey bees in nature are examined in the study of Karaboga, D. [13]. In ABC, a food source means a solution of the problem and the amount of the nectar means the qualify of the solution. The algorithm has three group of bees as employed, onlooker, and scout bees. Employed bees find food sources and give information to onlooker bees about the food. After this transformation of information, onlookers select the best food source. When the qualify of the food source is not convenient, employed bee abandon the source. Then the employed bee becomes a scout bee and search a new food source.

Initially, the food positions are generated by randomly as given in Eq. 1

$$x_{i,j} = \underline{x_j} + \eta(\overline{x_j} - \underline{x_j}) \tag{1}$$

where $i \in \{1, 2, ..., S_n\}$ and $j \in \{1, 2, ..., D\}$, $\eta \in (0, 1)$ is a randomly generated real number, $\underline{x_j}$ is the lower bound and $\overline{x_j}$ is the upper bound in the j-th dimension.

In this algorithm, onlooker bees select the convenient food source with estimating probability as calculated in Eq. 2

$$P_i = \frac{fit(\vec{x}_i)}{\sum_{i=1}^{S_n} fit(\vec{x}_i)} \tag{2}$$

where $fit(\vec{x}_i)$ is the amount of the i-th food source.

After determining the food source $\vec{x}_i$, the position of this source is changed with Eq. 3. Then the nectar amount of the candidate source or solution can be detected.

$$x_{i,j}(t+1) = x_{i,j}(t) + \psi(x_{i,j}(t) - x_{k,j}(t)) \tag{3}$$

where $x_{i,j}$ is determined neighboured to $x_{k,j}$, ψ is random in $[-1, 0]$ for $i, k \in \{1, 2, ..., S_n\}$, $k \neq i$. Also the indexes, k and j, are chosen randomly. As seen in Eq. 3, only one component of the position vector of the bee $\vec{x}_i$ is updated for obtaining the new position, in ABC. According to this position update, the nectar amount of solution is calculated. If the nectar amount of position better than the older one, the new position is selected and the older position is ignored [14].

In the next section, we explain that how the feed forward neural network is constructed for the solution of IVPs, and we clarify the modifications over ABC algorithm for training of neural net.

3. Modification of ABC Algorithm for Solving IVPs

For the first order differential equation, the initial value problem is given in Eq. 4.

$$\begin{cases} y'(t) = f(t, y(t)), \\ y(t_0) = y_0 \end{cases} \tag{4}$$

The solution of Eq. 4 occurred by creating trial function as $y_T(t_j, \vec{p}) = y_0 + (t_j - t_0)N(t_j, \vec{p})$ which is satisfied initial condition. In the trial function, $N(t_j, \vec{p}) = \sum_{i=1}^{m} \alpha_i \sigma(z_i)$ shows the solution of feedforward neural network for the neuron $z_i = w_i t_j + \beta_i$ where w_i is the weight and β_i is the bias value for the input t_j for $1 \leq j \leq n$. $N(t_j, \vec{p})$ has the unknown parameters vector as $\vec{p} = \vec{p}(\vec{\alpha}, \vec{\beta}, \vec{w})$ which is determined by ABC algorithm. In the vector $\vec{p}$, the unknown parameters defined as $\vec{\alpha}, \vec{\beta}, \vec{w} \in \mathbb{R}^m$ for m is the number of neurons

in the hidden layer. In $N(t_j, \vec{p})$, the sigmoid function is selected as activation function, $\sigma(z) = \dfrac{1}{1+\exp(-z)}$.

In feedforward neural network, the error appears after training of the network for each input at any iteration. The minimization of the error occurs after updating the unknown parameters. The aim of this process, the propagation of the error to whole network.

Generally, in the feedforward neural network $E = \dfrac{1}{2}\sum_{j=1}^{n}(d_j - y_j)^2$ is determined for the error calculation called cost function. In this equation, d_j is the desired output and y_j is the output of the network for the input t_j. In this study, the cost function is calculated as $E = \dfrac{1}{n}\sum_{j=1}^{n}\left(\dfrac{\partial y_T}{\partial t_j} - f(t_j, y_T(t_j))\right)^2$ so as to get the value of Mean Squared Error (MSE). In this equation, the partial derivatives of y with respect to t_j is $\dfrac{\partial y_T}{\partial t_j} = N(t_j, \vec{p}) + t_j \dfrac{\partial N(t_j, \vec{p})}{\partial t_j}$ where $y_T(t_j, \vec{p}) = y_0 + (t - t_0)N(t_j, \vec{p})$ and $\dfrac{\partial N(t_j, \vec{p})}{\partial t_j} = \sum_{i=1}^{m} \alpha_i w_i \sigma(z_i)(1 - \sigma(z_i))$.

As for the second order differential equation as seen in Eq. 5, the calculations are fulfilled the same way, except that the trial function is $y_T(t_j, \vec{p}) = A + B.(t - t_0) + (t - t_0)^2 N(t_j, \vec{p})$.

$$\begin{cases} y''(t) = f(t, y(t), y'(t)), \\ y(t_0) = A \\ y'(t_0) = B \end{cases} \tag{5}$$

In this type of DEs, $E = \dfrac{1}{n}\sum_{j=1}^{n}\left(\dfrac{\partial^2 y_T}{\partial t_j^2} - f\left(t_j, y_T(t_j), \dfrac{\partial y_T}{\partial t_j}\right)\right)^2$ is the cost function for the minimization of MSE. For calculation of MSE,

$$\frac{\partial y_T}{\partial t_j} = B + 2(t_j - t_0)N(t_j, \vec{p}) + (t_j - t_0)^2 \frac{\partial N(t_j, \vec{p})}{\partial t_j}$$

and

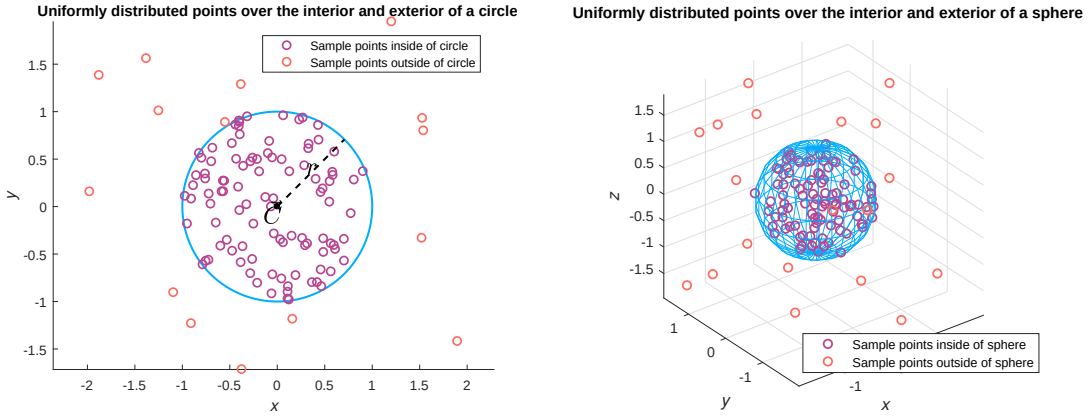
$$\frac{\partial^2 y_T}{\partial t_j^2} = 2N(t_j, \vec{p}) + 4(t_j - t_0)\frac{\partial N(t_j, \vec{p})}{\partial t_j} + (t_j - t_0)^2 \frac{\partial^2 N(t_j, \vec{p})}{\partial t_j^2}$$

are estimated where $\dfrac{\partial^2 N(t_j, \vec{p})}{\partial t_j^2} = \sum_{i=1}^{m} \alpha_i w_i \sigma(z_i)(1 - \sigma(z_i))(1 - 2\sigma(z_i))$.

Unlike the original ABC algorithm, we use adaptive swarm size in this study. Generally, the metaheuristics that use the adaptive swarm size paradigm generate a uniformly distributed population over the whole search space. However, if a solution is abandoned in ABC algorithm, then the generating a new bee population around the abandoned solution is an unnecessary process leading waste of time. To prevent the mentioned case, we propose the generating a new population uniformly distributed on the interior or exterior of a hypersphere as a subdomain of whole search space. The incomplete gamma function allows us to perform the randomization process using Eq. 6.

$$\vec{p_i} = \vec{C} + r.\vec{U_i}\frac{\left(\int_0^{S_i^2} t^{\left(\frac{D}{2}-1\right)} e^{-t} dt\right)^{\frac{1}{D}}}{S_i.\Gamma\left(\frac{D}{2}\right)^{\frac{1}{D}}} \tag{6}$$

where $\vec{U}_i$ is randomly generated vector by continuous uniform distribution over the search space. $\vec{C}$ represents the position of best solution so far as the center of hypersphere, and r is the radius of the hypersphere. In Eq. 6, $S_i = \sqrt{\sum_{j=1}^{D} p_{i,j}^2}$ for $i = 1, 2, \ldots M$ such that M denotes the size of new population, and D specifies the dimension of search space. Eventually, $\vec{p}_i$ is the position of i^{th} individual belonging the new generated population as a neural network parameter. Figure 1 illustrates the randomization process on a hypersphere. In Figure 1a, the generation of uniformly distributed points on the unit circle are demonstrated. Figure 1b repeats the demonstration on a unit sphere.



(a) 2-dimensional case on unit circle (b) 3-dimensional case on unit sphere

FIGURE 1. Uniformly distributed points generated randomly on the interior or exterior of an hypersphere.

In our study, we generate a new bee population around the best solution ever found using Eq. 6 in each iteration. It applies exploitation by making a pressure to obtain better solution rather than the best found so far. Moreover, we generate a new candidate solution on the exterior of the neighbourhood of any abandoned solution, in the scout bee phase. After new population is generated the elitism stage is applied. In elitism, the fitness values of newly generated population is calculated, and the weakest ones are eliminated. The process maximises the probability of exploring the global optimum over the whole of search space. In each iteration, we also reduce the radius of the hypersphere by a damping ratio for making dynamization of exploration. The modified algorithm of ABC for obtaining the solution of the first order differential equations with initial conditions is given following in Algorithm 2 using the common functions given in Algorithm 1. One can easily update the cost function and the other related functions in Algorithm 2 for second order ODEs. In the next section, some different types of ODEs are solved numerically.

4. NUMERICAL EXPERIMENTS

In the numerical experiments, we solve some types of initial value problems with FNN by training with both of classical and our proposed ABC algorithm. In all experiment, all the parameters used in FNN trained by ABC algorithm are selected identical as seen in Table 1 for making robust comparison.

The step size is chosen as h and a partition of domain interval $[a, b]$ is created for training set. In addition, the step size is halved $\frac{h}{2}$ for the test set except the nodes which

Algorithm 1 Common functions used in ABC for IVPs

function Net(t,$\vec{\alpha}$,$\vec{w}$,$\vec{\beta}$) ▷ returns the neural net solution at t
  $m \leftarrow$ the length of $\vec{\alpha}$ ▷ m indicates the number of neurons in the NN.
  return $\sum_{i=1}^{m} \alpha_i \sigma(w_i t + \beta_i)$
end function

function dNet(t,$\vec{\alpha}$,$\vec{w}$,$\vec{\beta}$) ▷ returns the derivation of neural net solution at t
  $m \leftarrow$ the length of $\vec{\alpha}$ ▷ m indicates the number of neurons in the NN.
  return $\sum_{i=1}^{m} \alpha_i w_i \sigma(w_i t + \beta_i)(1 - \sigma(w_i t + \beta_i))$
end function

function Trialy(t,t_0,y_0,$\vec{p}$) ▷ returns the trial solution at t according to the parameters of neural network $\vec{p}$
  return $y_0 + (t - t_0)$Net(t,$\vec{p}$)
end function

function dTrialy(t,t_0,y_0,$\vec{p}$) ▷ returns the value of $\frac{\partial y_T}{\partial t}$
  return Net(t,$\vec{\alpha}$,$\vec{w}$,$\vec{\beta}$) $+(t - t_0)$dNet(t,$\vec{\alpha}$,$\vec{w}$,$\vec{\beta}$)
end function

function Cost($\vec{t}$,t_0,y_0,$\vec{p}$) ▷ returns the fitness value depending on all inputs
  $n \leftarrow$ the length of vector $\vec{t}$ specifies the number of inputs
  $E \leftarrow \frac{1}{n} \sum_{j=1}^{n} \{$ dTrialy(t_j,t_0,y_0,$\vec{p}$) $- f(t_j$,Trialy(t_j,t_0,y_0,$\vec{p}$)) $\}^2$
  return E
end function

TABLE 1. Free parameters used in this study.

The parameters	**The values of the parameters**
Number of neurons in ANN, m	5
Number of population, N	100
Lower bound of search space	-1
Upper bound of search space	1
Dimension of search space, D	$3m$
The step size for the training set, h	0.1
The step size for the test set, $\frac{h}{2}$	0.05
Number of iteration	300

are used for training set. However, only the node given by initial conditions is belonging to the test set. After applying the training process to the network, it is able to give the numerical solution of any points in the interval.

In Example 4.1, we solve classical first order linear differential equation. Example 4.2 demonstrates to ability of the proposed algorithm for solving nonlinear ODEs. Example 4.3 is a real world application introduced by Newton. The problem is a classical heat transfer problem. Finally, an example of second order ODEs is solved in Example 4.4. Obtained numerical solutions by traditional ABC algorithm are compared with proposed approach regarding with absolute errors, and mean squared errors.

Algorithm 2 Modified ABC Algorithm for IVP

procedure Modified_ABC_IVP ▷ Main code blocks of modified ABC for solving IVP
  Specify the interval $[a, b]$ as a search space
  $h \leftarrow$ the step size ▷ $h > 0$
  $t_0 \leftarrow a, y_0 \leftarrow$ the initial condition for the problem
  $m \leftarrow$ the number of neurons in Neural Net
  Initialize the radius of hypersphere, $r \leftarrow \frac{|b-a|}{2}$
  for $j \leftarrow 0$ **to** n **do**
    $t_j \leftarrow a + hj$ ▷ Create a partition of search space
  end for

  for $i \leftarrow 1$ **to** m **do**
    Initialize artificial bee population as Neural Net parameters $\vec{p_i} = (\vec{\alpha_i}, \vec{\beta_i}, \vec{w_i})$ such that $\vec{\alpha}, \vec{\beta}, \vec{w} \in \mathbb{R}^m$
    Evaluate the fitness values for candidate solutions by calling Cost($\vec{t}$,t_0,y_0,$\vec{p_i}$) function
  end for

  iteration $\leftarrow 1$
  repeat
    for all employeed bee $\vec{p_i}$ **do** ▷ employeed bee phase
      Generate a new solution as $\vec{p_i'}$ in the neighbour of employeed bee $\vec{p_i}$ using the Eq. 3 where k is randomly selected not equal to i
      Evaluate the fitness values of new solutions by calling Cost($\vec{t}$,x_0,y_0,$\vec{p_i'}$) function
      Apply the greedy selection by comparing the fitness values of new solutions $\vec{p_i'}$ and fitness values of employeed bees $\vec{p_i}$ and determine the new population
    end for
    Check the probability values given Eq. 2 and normalize them to select onlooker bees

    for all onlooker bee $\vec{p_i}$ **do** ▷ onlooker bee phase
      Generate new solution as $\vec{p_i'}$ in the neighbour of onlooker bee $\vec{p_i}$ according to the probability value P_i
      Evaluate the fitness values of new solutions by calling Cost($\vec{t}$,x_0,y_0,$\vec{p_i'}$) function
      Apply the greedy selection by comparing the fitness values of new solutions $\vec{p_i'}$ and fitness values of onlooker bees $\vec{p_i}$ and determine the new population
    end for

    for all candidate solution $\vec{p_i}$ **do** ▷ Scout bee phase
      if the candidate is an abandoned solution **then**
        Replace it with a new randomly produced solution satisfying the condition $\sum_{i=1}^{D}(p_i - C_i)^2 > r^2$ for ensuring to stay the outside of the hypersphere as a neighboured of the abandoned solution.
      end if
    end for
    ▷ Elitism phase
    Using Eq. 6, generate a new bee population guaranteeing to stay inside of the hypersphere as a neighboured of the best solution ever found.
    Evaluate the fitness values of new solutions
    Eliminate the weakest solutions according to the fitness values

    Determine the best food source position and keep it into a memory.
    Decrease the hypersphere radius by damping ratio
    iteration $\leftarrow$ iteration $+1$
  until The maximum number of iteration is reached
end procedure

Furthermore, the best cost values encountered in each iteration of neural network training stage are confronted. Figure 2 depicts the best cost values for each example. We also compare the execution time required for getting ODE solution in test stage. Algorithms are executed 10 times with randomly generated bee population, so the elapsed times are calculated for each execution. As a result, the mean and the standard deviation of the execution time is given in Table 7.

Example 4.1.

$$\begin{cases} y'(t) + \dfrac{y(t)}{t+1} = 0, \quad t \in [2,4], \\ y(2) = 3 \end{cases} \tag{7}$$

The first example is first order homogenous linear differential equation as given in Eq. 7 having the exact solution as $y(t) = \dfrac{9}{t+1}$. *After 10 trials, the best cost values are calculated as* 1.568×10^{-5} *and* 4.240×10^{-9} *for ABC and the modified version, respectively. Table 2 summarizes the obtained absolute errors for the problem.*

TABLE 2. The absolute errors obtained from feed forward neural network outputs with test set using the step size $h = 0.05$ for Eq. 7 in Example 4.1.

k	t_k	**ABC**	**Modified ABC**
1	2.00	0.000	0.000
2	2.05	1.568×10^{-3}	7.583×10^{-5}
3	2.15	3.724×10^{-3}	1.228×10^{-4}
4	2.25	4.880×10^{-3}	9.832×10^{-5}
5	2.35	5.339×10^{-3}	5.365×10^{-5}
6	2.45	5.342×10^{-3}	1.602×10^{-5}
7	2.55	5.077×10^{-3}	3.370×10^{-6}
8	2.65	4.685×10^{-3}	3.225×10^{-6}
9	2.75	4.271×10^{-3}	1.234×10^{-5}
10	2.85	3.905×10^{-3}	3.682×10^{-5}
15	3.35	3.663×10^{-3}	9.838×10^{-5}
20	3.85	3.970×10^{-3}	1.681×10^{-5}
22	4.00	3.100×10^{-3}	1.681×10^{-5}

Example 4.2.

$$\begin{cases} y'(t) - \dfrac{t}{y(t)} = 0, \quad t \in [3,4], \\ y(3) = 4 \end{cases} \tag{8}$$

The First order homogenous linear differential equation as given in Eq. 8 has the exact solution as $y(x) = \sqrt{25 - t^2}$. *The experiments show that the best cost values are* 2.957×10^{-5} *and* 3.099×10^{-7} *for ABC and modified ABC orderly. The absolute errors are reported with Table 3, briefly.*

Example 4.3.

$$\begin{cases} y'(t) = -0.5(y(t) - 25), \quad x \in [0,15], \\ y(0) = 32 \end{cases} \tag{9}$$

Eq. 9 describes the Newton's Laws of cooling problem. Newton's Law of Cooling is utilized to model the temperature change of an object to an environment of a different temperature. Eq. 9 has the exact solution as $y(t) = 7\exp(-0.5t) + 25$. *For ABC, best of MSEs is*

TABLE 3. The numerical solution of Feedforward Neural Network trained by ABC for test set in Example 2.

k	t_k	**ABC**	**Modified ABC**
1	3.00	0.000	0.000
2	3.05	6.420×10^{-4}	3.286×10^{-4}
3	3.15	2.517×10^{-3}	6.394×10^{-4}
4	3.25	4.633×10^{-3}	6.255×10^{-4}
5	3.35	6.450×10^{-3}	4.474×10^{-4}
6	3.45	7.591×10^{-3}	2.503×10^{-4}
7	3.55	7.877×10^{-3}	1.509×10^{-4}
8	3.65	7.341×10^{-3}	2.201×10^{-4}
9	3.75	6.243×10^{-3}	4.613×10^{-4}
10	3.85	5.086×10^{-3}	7.815×10^{-4}
11	3.95	4.639×10^{-3}	9.533×10^{-4}
12	4.00	5.000×10^{-3}	8.627×10^{-4}

computed as 3.847×10^{-2}. *This value is* 2.173×10^{-5} *for the modified version. Table 4 exposes the numerical errors by the mentioned methods.*

TABLE 4. The numerical solution of Feedforward Neural Network trained by ABC for test set in Example 3.

k	t_k	**ABC**	**Modified ABC**
1	0.00	0.000	0.000
5	0.35	9.686×10^{-2}	7.647×10^{-3}
10	0.85	1.188×10^{-1}	5.945×10^{-4}
15	1.35	1.095×10^{-1}	3.171×10^{-3}
20	1.85	1.279×10^{-1}	3.112×10^{-4}
25	2.35	1.650×10^{-1}	4.388×10^{-3}
30	2.85	1.870×10^{-1}	6.977×10^{-3}
35	3.35	1.760×10^{-1}	6.356×10^{-3}
40	3.85	1.366×10^{-1}	3.406×10^{-3}
45	4.35	3.059×10^{-2}	3.185×10^{-4}
50	4.85	1.147×10^{-2}	3.457×10^{-3}
75	7.35	9.834×10^{-3}	1.155×10^{-3}
100	9.85	2.192×10^{-1}	6.211×10^{-3}
125	12.35	3.464×10^{-1}	5.451×10^{-3}
150	14.85	1.883×10^{-1}	4.640×10^{-3}
152	15.00	1.692×10^{-1}	6.936×10^{-3}

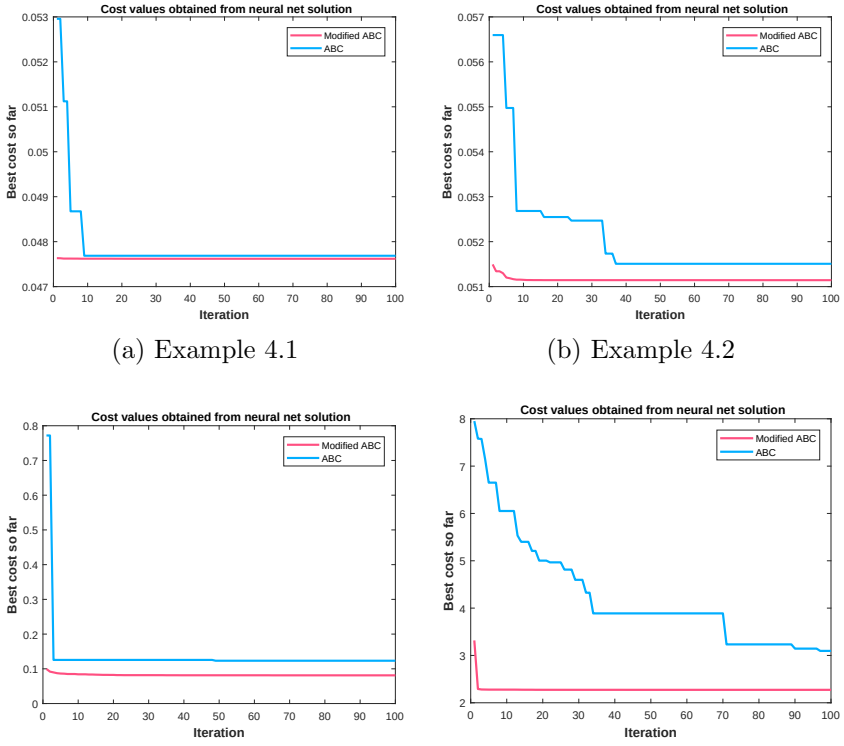
Example 4.4.

$$\begin{cases} y''(t) - 5y'(t) + 4y(t) = 0, \quad x \in [0,1], \\ y(0) = 0 \\ y'(0) = -1 \end{cases} \tag{10}$$

In Eq. 10, second order homogenous linear differential equation with Cauchy condition has the exact solution as $y(t) = \dfrac{(\exp(t) - \exp(-4t))}{3}$. *The best of MSEs values are 53.530 and 52.330 for the mentioned methods, respectively. In Table 5, the absolute errors are pointed out for both of the methods.*

TABLE 5. The numerical solution of Feedforward Neural Network trained by ABC for test set in Example 4.

k	t_k	**ABC**	**Modified ABC**
1	0.00	0.000	0.000
2	0.05	2.951×10^{-2}	2.965×10^{-2}
3	0.15	1.893×10^{-1}	1.908×10^{-1}
4	0.25	4.348×10^{-1}	4.443×10^{-1}
5	0.35	8.301×10^{-1}	8.397×10^{-1}
6	0.45	1.445×10^{0}	1.448×10^{0}
7	0.55	2.383×10^{0}	2.376×10^{0}
8	0.65	3.805×10^{0}	3.783×10^{0}
9	0.75	5.956×10^{0}	5.907×10^{0}
10	0.85	9.198×10^{0}	9.104×10^{0}
11	0.95	$1.407\times10^{+1}$	$1.390\times10^{+1}$
12	1.00	$1.735\times10^{+1}$	$1.714\times10^{+1}$



(a) Example 4.1 (b) Example 4.2

(c) Example 4.3 (d) Example 4.4

FIGURE 2. Graphs of best cost values found so far.

5. CONCLUSION

In this work, we solve numerically some types of differential equations with initial conditions with feed forward neural network trained by a modifed variant of ABC algorithm. First of all, we construct a trial function depending on the solution of a neural network. Trial function satisfies the initial conditions, and it used for unsupervised learning of neural network. The error appears after training of the network. We define a cost function regarding with the trail function and the neural network outputs. To minimize the cost

TABLE 6. Mean of cost values encountered in both training and testing stages.

Stage	Example	ABC	Modified ABC
Training	Ex. 4.1	$3.054 \times 10^{-5} \pm 3.599 \times 10^{-5}$	$1.541 \times 10^{-8} \pm 1.054 \times 10^{-8}$
	Ex. 4.2	$2.170 \times 10^{-5} \pm 2.738 \times 10^{-5}$	$2.139 \times 10^{-7} \pm 1.060 \times 10^{-7}$
	Ex. 4.3	$3.798 \times 10^{-2} \pm 2.465 \times 10^{-2}$	$2.975 \times 10^{-5} \pm 1.509 \times 10^{-5}$
	Ex. 4.4	$4.743 \times 10^{+1} \pm 3.635 \times 10^{-1}$	$4.695 \times 10^{+1} \pm 1.996 \times 10^{-1}$
Test	Ex. 4.1	$3.072 \times 10^{-5} \pm 3.072 \times 10^{-5}$	$1.486 \times 10^{-8} \pm 1.012 \times 10^{-8}$
	Ex. 4.2	$2.177 \times 10^{-5} \pm 2.800 \times 10^{-5}$	$2.240 \times 10^{-7} \pm 1.101 \times 10^{-7}$
	Ex. 4.3	$3.805 \times 10^{-2} \pm 2.447 \times 10^{-2}$	$2.959 \times 10^{-5} \pm 1.499 \times 10^{-5}$
	Ex. 4.4	$5.341 \times 10^{+1} \pm 3.999 \times 10^{-1}$	$5.286 \times 10^{+1} \pm 2.248 \times 10^{-1}$

TABLE 7. Mean of elapsed time in seconds for test set.

Example	ABC	Modified ABC
Ex. 4.1	$1.510 \times 10^{-4} \pm 1.162 \times 10^{-4}$	$1.516 \times 10^{-4} \pm 1.344 \times 10^{-4}$
Ex. 4.2	$2.134 \times 10^{-4} \pm 2.883 \times 10^{-4}$	$1.775 \times 10^{-4} \pm 2.105 \times 10^{-4}$
Ex. 4.3	$3.853 \times 10^{-4} \pm 1.562 \times 10^{-4}$	$4.681 \times 10^{-4} \pm 2.267 \times 10^{-4}$
Ex. 4.4	$1.745 \times 10^{-4} \pm 2.120 \times 10^{-4}$	$1.429 \times 10^{-4} \pm 1.289 \times 10^{-4}$

function, we trained the network both traditional Artificial Bee Colony algorithm and a variant of ABC proposed by us. Proposed algorithm uses dynamically constructed hypersphere to generate new bee population. The individuals in new population fall into the hypersphere to increase the exploitation ability of traditional ABC. Similarly, the individuals generated outside of the hypersphere supports the exploration quality of ABC.

In this work, we give some numerical examples some different types of differential equations such as first order and second order ODEs. The empirical studies precisely clarify that the modified version of ABC outperforms the classical ABC by means of absolute and mean squared errors. Table 6 exposes that cost values obtained in training and the testing stages are quite similar. Only, the desired improvement has not been achieved for the second order differential equation. Furthermore, it can be observed that the improvement has been drastic at the initial steps of the algorithm with Figure 2. However, it has been slight for following steps of the proposed algorithm.

According to the Table 7, the modification over classification does not affect the running time of the algorithm. The solution at each node of the interval $[a, b]$ are reached as short as 10-thousandth of a second approximately. Consequently, the proposed metaheuristic provides an advancement to ABC algorithm. In addition, the mentioned suggestions can be applied to other population based global optimization algorithms as a future work.

ACKNOWLEDGEMENT

We would like to acknowledge the support for this project from the Council of Higher Education in Turkey (YÖK), Coordination of Academic Member Training Program (ÖYP) in Adnan Menderes University, under Grant no. ADÜ-ÖYP-14011.

REFERENCES

[1] Lee, H. and Kang, I. 1990. Neural algorithms for solving differential equations, Journal of Computational Physics, 91, 110.

[2] Malek, A. and Beidokhti, R.S. 2006. Numerical solution for high order differential equations using a hybrid neural network-optimization method, Applied Mathematics and Computation, 183, 260-271.

[3] Lagaris, I. E., Likas, A. and Fotiadis, D. I. 1998. Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9(5), 987-1000.
[4] Aarts, L. P. and Van Der Veer, P. 2008. Neural Network Method for Solving Partial Differential Equations, Neural Process. Lett., 14(3), 261-271.
[5] McFall, K. S. and Mahan, J. R. 2009. Artificial Neural Network Method for Solution of Boundary Value Problems With Exact Satisfaction of Arbitrary Boundary Conditions, IEEE Transactions On Neural Networks, 20(8).
[6] Beidokhti, R. S. and Malek A. 2009. Solving initial-boundary value problems for systems of PDE using NN and optimization techniques, Journal of the Franklin Institute, 346, 898–913.
[7] Tsoulos, G. I., Gavrillis, D. and Glavas, E. 2009. Solving differential equations with constructed neural networks, Neurocomputing, 72, 2385-2391.
[8] Anastassi, A. A. 2014. Constructing Runge–Kutta methods with the use of artificial neural networks, Neural Computing and Applications, 25, 229-236.
[9] Raja, M. A. Z., Manzar, M. A. and Raza, S. 2015. An efficient computational intelligence approach for solving fractional order Riccati equations using ANN and SQP, Applied Mathematical Modelling, 39, 3075-3093.
[10] Kumar, M. and Yadav, N. 2015. Numerical Solution of Bratu's Problem Using Multilayer, Natl. Acad. Sci. Letter, 38(5), 425–428.
[11] Raja, M. A. Z., Khan J. A. and Chaudhary, N. I. 2016. Reliable numerical treatment of nonlinear singular Flierl–Petviashivili eq. for unbounded domain using ANN, GAs, and SQP, Applied Soft Computing, 38, 617-636.
[12] Karaboga, D. 2005. An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.
[13] Karaboga, D. and Akay, B. 2009. A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation, 214(1), 108-132.
[14] Chun-Feng, W., Kui, L. and Pei-Ping, S. 2014. Hybrid artificial bee colony algorithm and particle swarm search for global optimization, Hindawi Publishing Corporation, Mathematical Problems in Engineering, Article ID 832949, 8 pages.

Korhan Günel graduated from Ege University as a mathematician. He received his one of the M.Sc. degrees in computer engineering from Dokuz Eylul University, and received the other one in applied mathematics from Adnan Menderes University. He completed his Ph.D. degree in computer science at the Department of Mathematics in Ege University. His interests revolve around natural language processing, artificial intelligence applied to education, global optimization and machine learning for solving differential equations. Currently he works as Assistant Professor at the Department of Mathematics in Adnan Menderes University.

İclal Gör graduated from Mimar Sinan Fine Arts University as a mathematician. She received her M.Sc. degree in applied mathematics from Adnan Menderes University. She is a Ph.D. candidate in the Department of Mathematics at Adnan Menderes University. She currently works on numerical solutions of differential equations via neural networks and metaheuristics.