The Journal of Supercomputing (2024) 80:8719–8741
https://doi.org/10.1007/s11227-023-05775-2

Optimized radial basis function network for the fatigue driving modeling

José de Jesús Rubio[1] · Marco Antonio Islas[1] · Donaldo Garcia[2] · Jaime Pacheco[1] · Alejandro Zacarias[1] · Carlos Aguilar-Ibañez[2]

Accepted: 27 October 2023 / Published online: 28 November 2023

Abstract
The optimized radial basis function network is a kind of neural network that utilizes a step size inside of the gradient strategy for the modeling, where a small step size will spend much time to reach a minimum, while a big step size will jump over the minimum; hence, it needs an acceptable step size. The genetic optimizer is one option to seek an acceptable step size. In this study, the genetic optimizer is suggested to seek an acceptable step size in the gradient strategy for an optimized radial basis function network. The difference between the other genetic optimizers and our genetic optimizer is that the other genetic optimizers utilize high number of stages, while our genetic optimizer utilizes small number of stages. The idea of utilizing small number of stages in our genetic optimizer is based on the simplex optimizer and bat optimizer which also utilize small number of stages. To validate the performance of the optimized radial basis function network, the fatigue driving modeling in a vehicle is evaluated.

Keywords Genetic optimizer · Bat optimizer · Simplex optimizer · Optimized radial basis function network · Fatigue driving modeling

1 Introduction

The optimized radial basis function network is a kind of neural network that utilizes a step size inside of the gradient strategy [1, 2], and a Gaussian function as the activation function [3, 4] for the modeling, where a small step size will spend much time to reach a minimum, while a big step size will jump over the minimum; hence, it needs an acceptable step size. Seeking of an acceptable step size in the gradient strategy it is not easy, the genetic optimizer is one option to seek an acceptable step size.

Extended author information available on the last page of the article

The genetic optimizer has been evaluated in several optimization problems. In [5], the route calculation is performed by a genetic optimizer. In [6, 7], the hybrid genetic optimizer which combine the genetic mechanism with the gradient strategy is suggested. In [8], the genetic optimizer-based integer-valued optimization is considered for two machine optimization models. In [9], the genetic optimizer is utilized to optimize the terms of variational mode decomposition. In [10], the deep optimization trained by genetic optimizer is suggested. In [11], a cluster-based genetic optimizer that outputs a result of the bin pack problem is developed. In [12], a problem-specific non-dominated sorting genetic optimizer is suggested. In [13], the impact of utilizing constraint priorities on genetic optimizer is studied. In [14], a the genetic optimizer is addressed for the practical medical task of selecting drug combinations. In [15], the automatic design of dispatching rules using the genetic optimizer is addressed. In [16], the genetic optimizer is applied to achieve a balance between privacy protection and resource consumption. In [17], a genetic optimizer that uses a statistical-based chromosome replacement strategy is proposed. In [18], a high-performance solution is presented for parcel exchange based on a genetic optimizer. In [19], a hybrid feature selection method that combines a genetic optimizer with a proposed filter is presented. Nevertheless, the genetic optimizer is not evaluated to seek an acceptable step size in the gradient strategy for an optimized radial basis function network.

In this study, the genetic optimizer is suggested to seek an acceptable step size in the gradient strategy for an optimized radial basis function network. The justification of the genetic optimizer to seek an acceptable step size is described as follows: a small step size will reach small steps and will spend much time to reach a minimum in the gradient strategy, while a big step size will reach big steps and will jump over the minimum in the gradient strategy; hence, it is important to suggest a genetic optimizer to seek an acceptable step size. The advantage of our genetic optimizer with the other genetic optimizers is described as follows: the other genetic optimizers utilize high number of stages denoted as the initialization, fitness function, constraint handling, crossover, mutation, decode chromosomes, calculate fitness value, chromosome selection, replace old chromosome, and stopping criteria, while our genetic optimizer utilizes small number of stages denoted as the initialization, constraint handling, decode chromosomes, chromosome selection, and stopping criteria. The idea of utilizing small number of stages in our genetic optimizer is based on the simplex optimizer [20, 21] and bat optimizer [22, 23] which also utilize small number of stages.

On the other hand, fatigue driving is one of the leading causes of traffic accidents; consequently, the fatigue driving modeling plays a crucial role in road safety. To validate the performance of the optimized radial basis function network, the fatigue driving modeling in a vehicle is evaluated. A data table is utilized in a Arduino Mega to save the final data; after the data collection, the data table is utilized in a personal computer to train the optimized radial basis function network; and after of the training, the trained optimized radial basis function network is utilized in a personal computer for the testing.

The rest of the work is ordered in this sentence. Section 2 describes the literature review. Section 3 describes the research model. Section 4 describes the experimental design and performance evaluation. Section 5 describes the conclusion.

2 Literature review

In this section, the research which provides a broader context for our work is presented.

The genetic optimizers utilize several stages, but there are some works such as [24–26] that does not explain the stages utilized by the genetic optimizers.

There are other works that explain the stages utilized by the genetic optimizers. In [27], the chromosome representation, fitness function, and genetic operators are described as the stages utilized by genetic optimizers and gradient-free methods. In [28], the population initialization, selection operation, crossover operation, and mutation operation are described as the stages utilized by a stochastic gradient descent with genetic optimizer. In [29], the genetic encoding, objective function and fitness function, genetic operation, selection, crossover, and mutation are described as the stages utilized by a genetic optimizer-based fuzzy optimization. In [30], the adaptive crossover and mutation operation are described as the stages utilized by an adaptive genetic optimizer. In [31], the solution representation, generating the initial population and calculation of the fitness function, developed local search optimizer, selection, crossover, and mutation operators are described as the stages utilized by a hybrid genetic optimizer. In [32], the genetic operations, initialization, parent selection, crossover, mutation, and termination are described as the stages utilized by a genetic optimizer and a pore network model. In [33], the chromosome encoding, crossover operation, and mutation operation are described as the stages utilized by a clustering-based extended genetic optimizer. In [34], the initialization operator, design of chromosome coding, design of filter chain, crossover operator, mutation operator, correction operator, design of selection operator, design of termination condition are described as the stages utilized by an improved genetic optimizer. In [35], the chromosome representation, population initialization, cost function, operators, selection, crossover, and mutation are described as the stages utilized by a wireless sensor network using a genetic optimizer. In [36], the population initialization, select the fitness function, select operation, crossover operation, and mutation operation are described as the stages utilized by a neural network and a genetic optimizer. In [37], the initialize population, generate the next generation of individuals, cross operation, mutation operation, decode chromosomes, calculate the fitness value, chromosome selection operation, replace the old chromosome with simulated annealing operator, determine the number of iterations, and output results are described as the stages utilized by a simulated annealing genetic optimizer. In [38], the selection of initial population, crossover, and mutation are described as the stages utilized by a wireless sensor network using genetic optimizer. In [39], the genetic operator design, selection, cross, and variation are described as the stages utilized by a proposed genetic optimizer. In [40], the representation, initialization, fitness function, constraint handling, selection and reproduction, crossover,

mutation, and stopping criteria are described as the stages utilized by a genetic optimizer based probabilistic model.

Some of the main stages utilized by a genetic optimizer are described as follows [37, 40].

Initialization. The size of population is selected. The chromosomes are created in the initial stage, and each unit of a chromosome is initialized with a random number.

Fitness function. The value of fitness corresponding to a chromosome is its efficacy. A good fitness value maximizes the success probability and minimizes the associated cost.

Constraint handling. Real-world problems are mostly constrained, i.e., solutions may lie outside the feasible region. To avoid constraint violation, chromosomes are encoded in harmony with the constraints.

Crossover. It is the exchange of gene fragments at the same position on two chromosomes. For the optimization of assembly sequence in precast concrete buildings, the cross operation needs to meet the following conditions: (a) inherit the excellent gene of mother chromosome to the greatest extent; (b) all gene values in offspring chromosomes cannot be repeated with each other; (c) offspring chromosomes must meet strict constraints.

Mutation. Because all gene values in the chromosome cannot be repeated with each other, the mutation operation can be realized by randomly exchanging two genes in the chromosome. This method is also called exchange mutation. Mutation operation adopts two-point reciprocity to mutate.

Decode chromosomes. The assembly sequence is obtained by decoding the components.

Calculate the fitness value. The fitness value is calculated according to the assembly difficulty coefficient and assembly time of components in the assembly process.

Chromosome selection. It is to select better individuals from the previous generation and pass them on to the next generation. Selection technology has a great impact on the efficiency of genetic programming. Roulette wheel selection is adopted. Roulette wheel selection is used to select outstanding individuals from a population.

Replace the old chromosome. The combination of simulated annealing and genetic optimizer is to judge whether to replace the old chromosome with simulated annealing operator.

Stopping criteria. A maximum number of generations is taken as the halting condition.

Figure 1 shows the difference between the other genetic optimizers and our genetic optimizer, where the other genetic optimizers utilize high number of stages denoted as the initialization, fitness function, constraint handling, crossover, mutation, decode chromosomes, calculate fitness value, chromosome selection, replace old chromosome, and stopping criteria, while our genetic optimizer utilizes small number of stages denoted as the initialization, constraint handling, decode chromosomes, chromosome selection, and stopping criteria. The idea of utilizing small number of stages in our genetic optimizer is based on the simplex

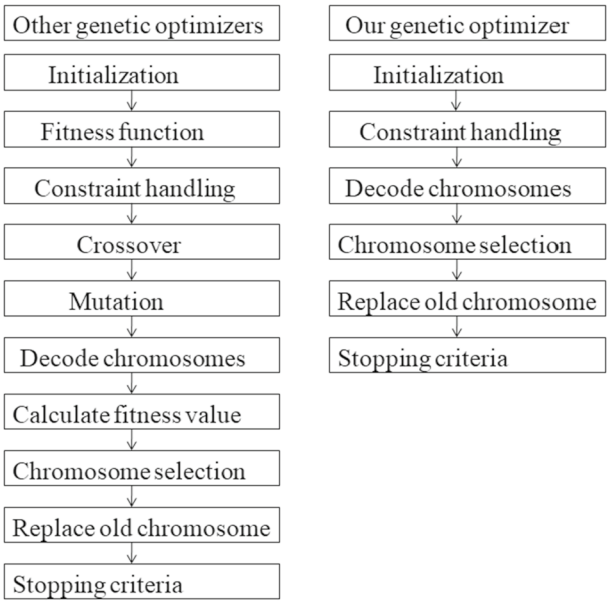

Fig. 1 Comparison between the other genetic optimizers and our genetic optimizer

optimizer [20, 21] and bat optimizer [22, 23] which also utilize small number of stages.

3 Research model

This section introduces the optimized radial basis function network. Later, this section presents the design and the pseudocode of the genetic optimizer to seek an acceptable step size in the gradient strategy for an optimized radial basis function network.

3.1 The optimized radial basis function network

This subsection introduces the optimized radial basis function network.

Figure 2 shows the application of the genetic optimizer to seek an acceptable step size in the gradient strategy for an optimized radial basis function network, where the collected data is used for the training of the optimized radial basis function network, later, the genetic optimizer is utilized to seek a new step size for the optimized radial basis function network; if the root mean square of the new step size is smaller to the root mean square of the previous step size, the new step size is chosen for the optimized radial basis function network, otherwise, the previous step size is chosen for the optimized radial basis function network. The details are in this and in the next section.

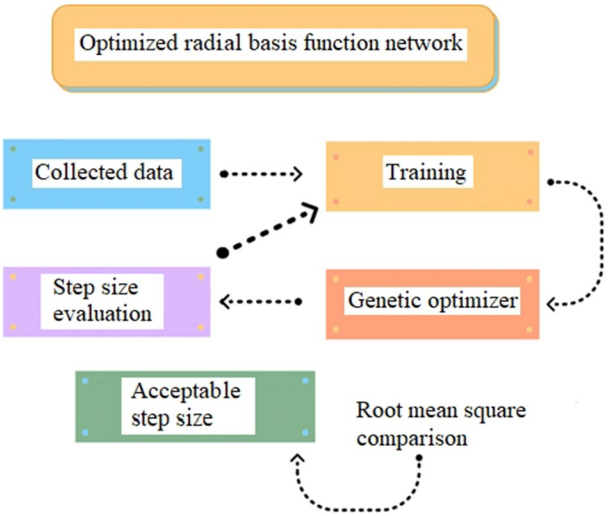

Fig. 2 Application of the genetic optimizer to seek an acceptable step size in the gradient strategy

The optimized radial basis function network with a hidden layer is:

$$\begin{aligned} w_k &= \sum_{j=1}^{M} a_{j,k}\alpha_j(u_{j,k}), \\ \alpha_j(u_{j,k}) &= e^{-u_{j,k}^2}, \\ u_{j,k} &= \sum_{i=1}^{N} b_{ij,k}\left[x_{i,k} - c_{i,k}\right], \\ \gamma_j(u_{j,k}) &= 2u_{j,k}\alpha_j(u_{j,k}), \end{aligned} \tag{1}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, M$, $x_{i,k} \in \mathfrak{R}$ is the optimized radial basis function network input, $w_k \in \mathfrak{R}$ is the optimized radial basis function network output, $a_{j,k} \in \mathfrak{R}$, $b_{ij,k} \in \mathfrak{R}$, $c_{i,k} \in \mathfrak{R}$ are the terms of the output layer, hidden layer, and centers, $\alpha_j(u_{j,k}) \in \mathfrak{R}$ and $\gamma_j(u_{j,k})$ are nonlinear functions, $u_{j,k} \in \mathfrak{R}$ is the addition function, M is the hidden layer neurons numeral. Figure 3 shows the architecture of the optimized radial basis function network with the input layer, hidden layer, and output layer.

In this section, the optimized radial basis function network is suggested. In this part, the tuning law is obtained.

Define an error sum as:

$$\varepsilon_k = \frac{1}{2}\left(w_k - t_k\right)^2, \tag{2}$$

where $\left(w_k - t_k\right)$ is the error, w_k is the optimized radial basis function network output, t_k is the target. The tuning law is achieved utilizing the following equations:

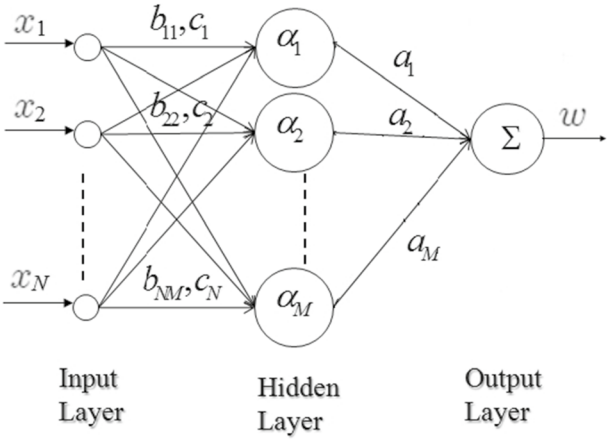

Fig. 3 The optimized radial basis function network

$$\begin{aligned} \triangle a_{j,k} &= a_{j,k+1} - a_{j,k} = -\eta \frac{\partial \varepsilon_k}{\partial a_{j,k}}, \\ \triangle b_{ij,k} &= b_{ij,k+1} - b_{ij,k} = -\eta \frac{\partial \varepsilon_k}{\partial b_{ij,k}}, \\ \triangle c_{i,k} &= c_{i,k+1} - c_{i,k} = -\eta \frac{\partial \varepsilon_k}{\partial c_{i,k}}, \end{aligned} \tag{3}$$

where η is the step size to be defined in the next section. Utilizing the chain rule to achieve $\frac{\partial \varepsilon_k}{\partial a_{j,k}}$ gives:

$$\begin{aligned} c\frac{\partial \varepsilon_k}{\partial a_{j,k}} &= \frac{\partial \varepsilon_k}{\partial \left(w_k - t_k\right)} \frac{\partial \left(w_k - t_k\right)}{\partial w_k} \frac{\partial w_k}{\partial a_{j,k}} \\ &= \alpha_j(u_{j,k})\left(w_k - t_k\right), \end{aligned} \tag{4}$$

Utilizing the chain rule to achieve $\frac{\partial \varepsilon_k}{\partial b_{ij,k}}$ gives:

$$\begin{aligned} c\frac{\partial \varepsilon_k}{\partial b_{ij,k}} &= \frac{\partial \varepsilon_k}{\partial \left(w_k - t_k\right)} \frac{\partial \left(w_k - t_k\right)}{\partial w_k} \frac{\partial w_k}{\partial b_{ij,k}} \\ &= \gamma_j(u_{j,k}) a_{j,k}\left[c_{i,k} - x_{i,k}\right]\left(w_k - t_k\right), \end{aligned} \tag{5}$$

Utilizing the chain rule to achieve $\frac{\partial \varepsilon_k}{\partial c_{i,k}}$ gives:

$$\begin{aligned} \frac{\partial \varepsilon_k}{\partial c_{i,k}} &= \frac{\partial \varepsilon_k}{\partial \left(w_k - t_k\right)} \frac{\partial \left(w_k - t_k\right)}{\partial w_k} \frac{\partial w_k}{\partial c_{i,k}} \\ &= b_{ij,k}\gamma_j(u_{j,k}) a_{j,k}\left(w_k - t_k\right), \end{aligned} \tag{6}$$

By substituting (4), (5) and (6) into (3), gradient strategy of the optimized radial basis function network is as follows:

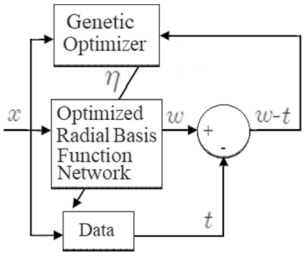

Fig. 4 Block diagram of the genetic optimizer to seek an acceptable step size in the gradient strategy

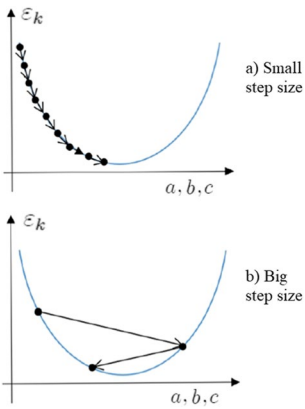

Fig. 5 Justification of the genetic optimizer to seek an acceptable step size in the gradient strategy

$$\begin{aligned} a_{j,k+1} &= a_{j,k} - \eta\alpha_j(u_{j,k})\left(w_k - t_k\right), \\ b_{ij,k+1} &= b_{ij,k} - \eta\gamma_j(u_{j,k})a_{j,k}\left[c_{i,k} - x_{i,k}\right]\left(w_k - t_k\right), \\ c_{i,k+1} &= c_{i,k} - \eta b_{ij,k}\gamma_j(u_{j,k})a_{j,k}\left(w_k - t_k\right), \end{aligned} \tag{7}$$

where $\left(w_k - t_k\right)$ is the error of (2).

3.2 Genetic optimizer to seek an acceptable step size in the gradient strategy for an optimized radial basis function network

This subsection presents the design and the pseudocode of the genetic optimizer to seek an acceptable step size in the gradient strategy for an optimized radial basis function network.

Figure 4 shows the block diagram of the genetic optimizer to seek an acceptable step size in the gradient strategy for an optimized radial basis function network. From the gradient strategy (7), the genetic optimizer will tune the step size η.

Figure 5 shows the justification of the genetic optimizer to seek an acceptable step size η in the gradient strategy for an optimized radial basis function network, where a small step size η will reach small steps and will spend much time to reach a minimum in the gradient strategy, while a big step size η will reach big steps and will jump over the minimum in the gradient strategy.

The information about k is:

$$\langle z_k, v_k, \beta_k, y_k \rangle, \tag{8}$$

where z_k is the chromosome in the time k, v_k is the velocity chromosome in the time k, β_k is the auxiliary velocity chromosome in the time k, y_k is the chromosome precision in the time k.

The step size η_k is assigned to the chromosome z_k of the genetic optimizer:

$$z_k = \eta_k, \tag{9}$$

The chromosome precision y_k is tuned as follows:

$$y_k = \frac{z_{\max} - z_{\min}}{2^C - 1}, \tag{10}$$

where $z_{\min}$ and $z_{\max}$ are the minimum and maximum values that can take the next chromosome z_{k+1}, C is the chromosome number in a population. The auxiliary velocity chromosome β_{ok} is tuned as follows:

$$\beta_{ok} = \begin{cases} 1 & \left(\text{rand}_2 > 0.5\right) \\ 0 & otherwise \end{cases}, \tag{11}$$

where rand_2 is a random numeral in the interval [0, 1]. The velocity chromosome v_k is tuned as follows:

$$v_k = \sum_{o=1}^{C} 2^o \beta_{ok}, \tag{12}$$

where β_{ok} is the auxiliary velocity chromosome, C is the chromosome number in a population.

To reflect the next chromosome, the chromosome is changed with some randomness. The auxiliary chromosome $\bar{z}_{k+1}$ is tuned as follows:

$$\bar{z}_{k+1} = v_k y_k + z_{\min}, \tag{13}$$

where v_k is the velocity chromosome, β_k is the auxiliary velocity chromosome, y_k is the chromosome precision, and z_k is the chromosome.

Utilizing the chromosome z_k of (9), the objective function $g(z_k)$ is tuned as follows:

$$\begin{aligned} a_{\eta j,k+1} &= a_{\eta j,k} - \eta \frac{\partial \varepsilon_k}{\partial a_{j,k}}, \\ w_{\eta,k} &= \sum_{j=1}^{M} a_{j,k} \alpha_j(u_{j,k}), \\ g(z_k) &= \left(w_{\eta,k} - t_{k+1}\right)^2, \end{aligned} \tag{14}$$

Utilizing the auxiliary chromosome $\bar{z}_{k+1}$ of (13), the auxiliary objective function $g(\bar{z}_{k+1})$ is tuned as following:

$$\begin{aligned} \overline{a}_{\eta j,k+1} &= \overline{a}_{\eta j,k} - \eta \frac{\partial \varepsilon_k}{\partial a_{j,k}}, \\ \overline{w}_{\eta,k+1} &= \sum_{j=1}^{M} \overline{a}_{j,k+1} \alpha_j(u_{j,k}), \\ g(\overline{z}_{k+1}) &= \left(\overline{w}_{\eta,k+1} - t_{k+1}\right)^2, \end{aligned} \tag{15}$$

After the auxiliary chromosome $\overline{z}_{k+1}$ is obtained with (13), the next chromosome z_{k+1} is tuned as follows:

$$z_{k+1} = \begin{cases} \overline{z}_{k+1} & \left(g(\overline{z}_{k+1}) < g(z_k)\right) \& \left(\overline{z}_{k+1} \geq z_{\min}\right) \& \left(\overline{z}_{k+1} \leq z_{\max}\right) \\ z_k & otherwise \end{cases}, \tag{16}$$

where $z_{\min}$ and $z_{\max}$ are the minimum and maximum values that can take z_{k+1}. (16) implicates that the chromosome is tuned when two requirements are satisfied: a) it finds the better position of the objective function, i.e., $g(\overline{z}_{k+1}) < g(z_k)$, b) the values of z_{k+1} are bounded by the minimum $z_{\min}$ and maximum values $z_{\max}$.

The next chromosome z_{k+1} of the genetic optimizer is assigned to the next step size η_{k+1}:

$$\eta_{k+1} = z_{k+1}, \tag{17}$$

The optimization by the genetic optimizer uses the constants shown in Table 1. Since in this genetic optimizer, the chromosome is changed with some randomness, other similar constants will obtain similar results.

The pseudo code of the genetic optimizer to seek an acceptable step size in the gradient strategy is as follows:

Inputs: z_1, v_1, y_1.

1. Generate the initial chromosome z_1 and initial velocity chromosome v_1;
2. Define the initial chromosome precision y_1;
3. While ($k <$ maximum iteration numeral) do
4. Tune the chromosome z_k with Eq. (9);
5. Tune the chromosome precision y_k, the auxiliary velocity chromosome β_k, and the velocity chromosome v_k with Eqs. (10), (11), and (12), respectively;
6. Tune the auxiliary chromosome $\overline{z}_{k+1}$ with Eq. (13);
7. Determine the objective function $g(z_k)$ with Eq. (14);
8. Determine the auxiliary objective function $g(\overline{z}_{k+1})$ with Eq. (15);
9. If $\left(\left(g(\overline{z}_{k+1}) < g(z_k)\right) \& \left(\overline{z}_{k+1} \geq z_{\min}\right) \& \left(\overline{z}_{k+1} \leq z_{\max}\right)\right)$ then
10. Accept the new result z_{k+1} with Eq. (16);

Table 1 Genetic optimizer constants

Constant	Symbol	Value
Chromosome number	C	15
Minimum step size	$z_{\min}$	0.1
Maximum step size	$z_{\max}$	1

11. Otherwise, take the past value z_k with Eq. (16).
12. End If
13. Tune the next step size η_{k+1} with Eq. (17);
14. End While

Output: η_{k+1}.

The genetic optimizer of Eqs. (8)–(15) considered to seek an acceptable step size η in the gradient strategy and to reach the objective function g_0 is described by:

$$\begin{gathered} \min g_0 = \min \left(\left(w_{\eta,k} - t_k \right)^2 \right) \\ \text{subject to} \\ \eta_{\min} \leq \eta_k \leq \eta_{\max} \\ z_k = \eta_k \\ z_{\min} = \eta_{\min} \\ z_{\max} = \eta_{\max}, \end{gathered} \tag{18}$$

4 Experimental design and performance evaluation

In this section, we compare the simplex optimizer of (19) [20, 21], the bat optimizer of (20) [22, 23], and the genetic optimizer of Eqs. (1)–(7), (8)–(18), to seek an acceptable step size in the gradient strategy for an optimized radial basis function network. The aim of the strategies is that utilizing the inputs $u_{i,k}$, the optimized radial basis function network output w_k needs to reach the target t_k faster.

4.1 Experimental materials

A data table is utilized in a Arduino Mega to save the final data; after the data collection, the data table is utilized in a personal computer to train the optimized radial basis function network; and after of the training, the trained optimized radial basis function network is utilized in a personal computer for the testing.

For the development of a sleep monitoring system in the vehicle, it is necessary to establish some sensors that will be measuring the state of the driver, taking into account some sleep features, it is convenient to place the following sensors inside the vehicle: the steering wheel force sensor, the steering wheel heart rate sensor, and the steering wheel blood oxygen saturation sensor. The signals of heart rate, pressure, and blood oxygen concentration chosen because these signals are easier and cheaper to collect. The structure of the circuit with the steering wheel heart rate sensor can be seen in Fig. 6; the structure of the circuits with the other two sensors is similar.

To carry out the test of the steering wheel force sensor, the updating is carried out on the steering wheel as shown in Fig. 7, it consists of a pressure sensor FSR connected to the analog port of the Arduino Mega 2560, and through Octave software the reading is stored to obtain the final data. This procedure does not involve any

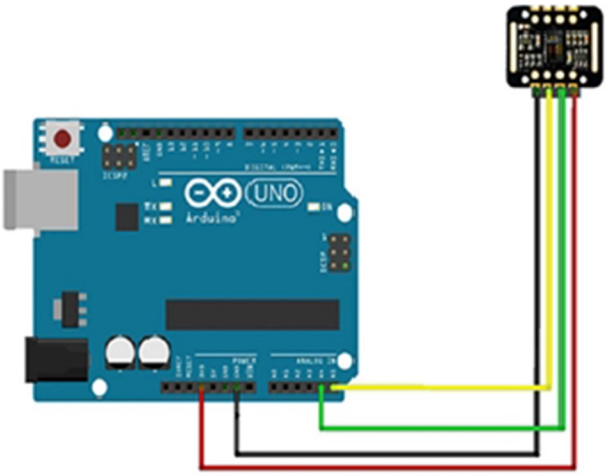

Fig. 6 Structure of the circuit with the steering wheel heart rate sensor

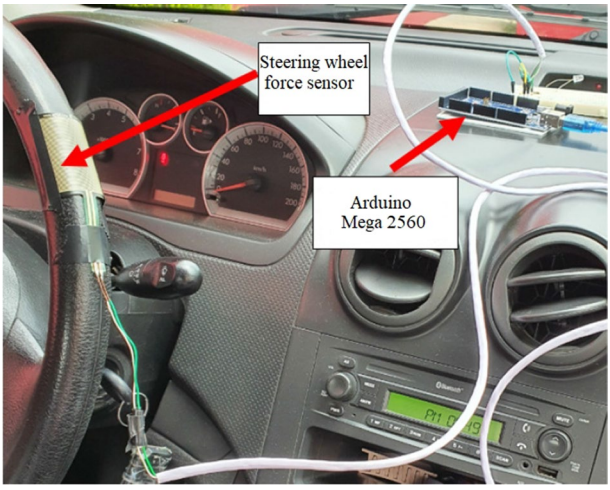

Fig. 7 The steering wheel force sensor

data processing such as data cleaning, resampling, etc. The sampling frequency of the final data is 3200 samples per second.

To carry out the test of the steering wheel heart rate sensor, with the help of the Octave software, the heart rate reading is obtained for the driver, this is done experimentally, for this it was necessary to adjust the sensor MAX30102 on the driver finger as shown in Fig. 8, in this way a good fixation of the sensor is obtained and thus not obtain erroneous readings.

The calculation of the oxygen saturation in a driver is carried out with the same sensor MAX30102 with which the heart rate is obtained as shown in Fig. 8, in this

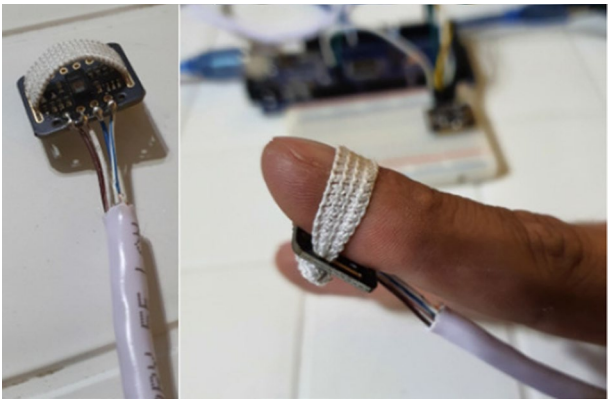

Fig. 8 The steering wheel heart rate sensor

case the same tests are carried out under the same conditions so that the results are similar.

4.2 Experimental environment

Fatigue driving is one of the leading causes of traffic accidents as can be seen in Fig. 9; consequently, the fatigue driving modeling plays a crucial role in road safety. To validate the performance of the optimized radial basis function network, the fatigue driving modeling in a vehicle is evaluated.

As the first comparison [20, 21], the objective function g_0 considered of the simplex optimizer to achieve this aim is:

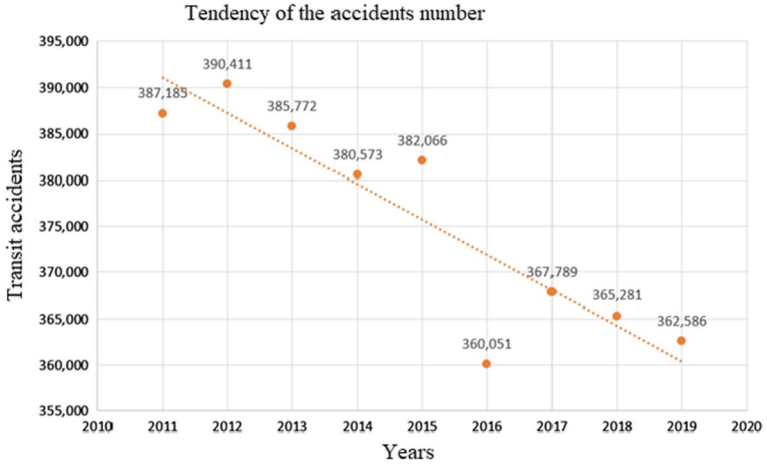

Fig. 9 Fatigue driving is one of the leading causes of traffic accidents

$$\begin{gathered} \min g_0 = \min \left(w_{\eta,k} - t_k \right) \\ \text{subject to} \\ \eta_{\min} \leq \eta_k \leq \eta_{\max} \\ z_k = \eta_k \\ z_{\min} = \eta_{\min} \\ z_{\max} = \eta_{\max}, \end{gathered} \tag{19}$$

As the second comparison [22, 23], the objective function g_0 considered of the bat optimizer to achieve this aim is:

$$\begin{gathered} \min g_0 = \min \left(\left(w_{\eta,k} - t_k \right)^2 \right) \\ \text{subject to} \\ \eta_{\min} \leq \eta_k \leq \eta_{\max} \\ z_k = \eta_k \\ z_{\min} = \eta_{\min} \\ z_{\max} = \eta_{\max}, \end{gathered} \tag{20}$$

4.3 Parameters setting

For the awake driver and somnolent driver, we utilize 2 inputs to train the optimized radial basis function network:

- $u_{1,k}$ = the steering wheel force sensor.
- $u_{2,k}$ = the steering wheel blood oxygen saturation sensor.

and we utilize 1 target to train the optimized radial basis function network:

- t_k = the steering wheel heart rate sensor.

We utilize 2 inputs $u_{1,k}$, $u_{2,k}$, 1 target t_k, and 1 optimized radial basis function network output w_k. The aim is that utilizing the inputs $u_{i,k}$, the optimized radial basis function network output w_k needs to reach the target t_k faster. It is significant to note that the optimization utilizes time-varying terms and time-varying step size for the modeling of 8000 input and target data of an awake driver, the training utilizes time-varying terms and constant step size for the modeling of 8000 input and target data of an awake driver, and the testing utilizes constant terms and constant step size for the modeling of 8000 input and target data of an awake driver and a somnolent driver.

In this part of the study, the suggested optimizer is applied for the awake driver and the somnolent driver, where the root mean square error MSE is utilized as:

$$MSE = \left(\frac{1}{T} \sum_{k=1}^{T} \left(w_k - t_k \right)^2 \right)^{\frac{1}{2}}, \tag{21}$$

where $w_k - t_k$ is the error, w_k is the optimized radial basis function network output, t_k is the target, T is the final iteration.

Equation (19), [20, 21], describes the simplex optimizer with 2 inputs, 1 output, and 3 hidden layer neurons, $z_{s,1} = \eta_{s,1} = 0.5$ is the initial value of the step size before the optimization, $z_{s,T} = \eta_{s,T} = 0.4105$ is the final value of the step size after the optimization, $a_{j,1} = \text{rand}$, $b_{ij,1} = \text{rand}$, $c_{i,k} = \text{rand}$, rand is a random numeral in the interval [0, 1].

Equations (20) [22, 23] describes the bat optimizer with 2 inputs, 1 output, and 3 hidden layer neurons, $z_{b,1} = \eta_{b,1} = 0.5$ is the initial value of the step size before the optimization, $z_{b,T} = \eta_{b,T} = 0.3571$ is the final value of the step size after the optimization, $a_{j,1} = \text{rand}$, $b_{ij,1} = \text{rand}$, $c_{i,k} = \text{rand}$, rand is a random numeral in the interval [0, 1].

Equations (1)–(7), (8)–(18) describe the genetic optimizer with 2 inputs, 1 output, and 3 hidden layer neurons, $z_{g,1} = \eta_{g,1} = 0.5$, is the initial value of the step size before the optimization, $z_{g,T} = \eta_{g,T} = 0.7674$ is the final value of the step size after the optimization, $a_{j,1} = \text{rand}$, $b_{ij,1} = \text{rand}$, $c_{i,k} = \text{rand}$, rand is a random numeral in the interval [0, 1].

4.4 Performance evaluation

Example 1. Figure 10 shows the step size during the optimization for 8000 iterations and for the first 40 iterations. Figures 11 and 12 show the modeling and MSE during the training. Figures 13 and 14 show the modeling and MSE during the testing. Table 2 shows the MSE of (21) during the training and testing.

Example 2. Figure 15 shows the step size during the optimization for 8000 iterations and for the first 40 iterations. Figures 16 and 17 show the modeling and MSE

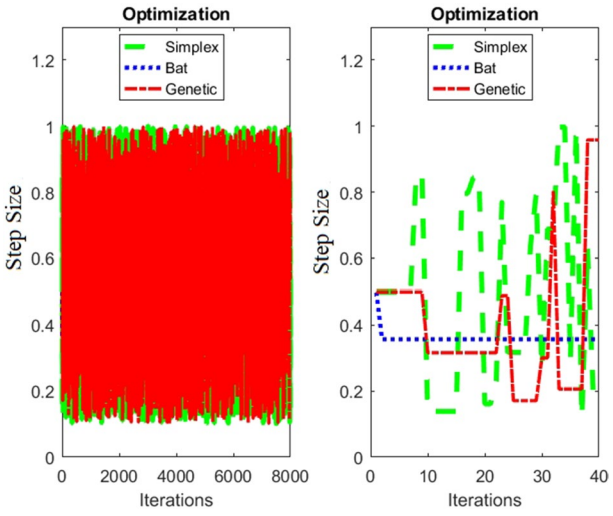

Fig. 10 Step size during the optimization for example 1

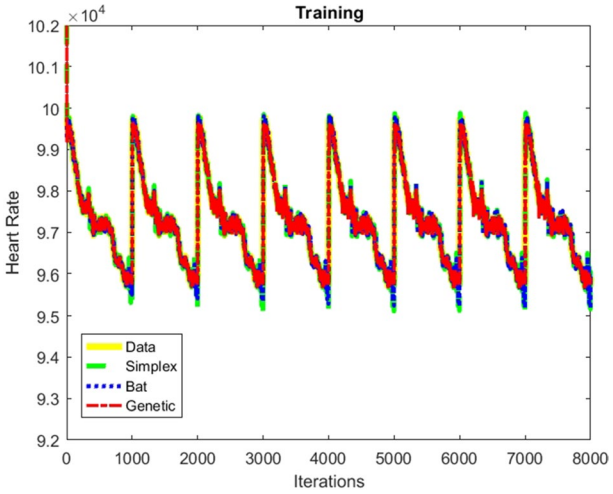

Fig. 11 Modeling during the training for example 1

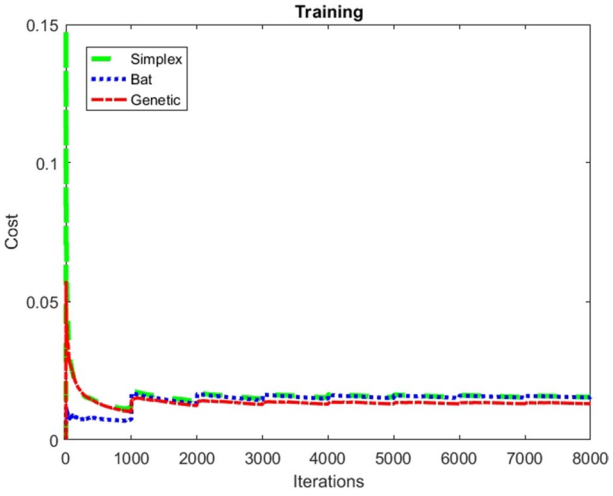

Fig. 12 MSE during the training for example 1

during the training. Figures 18 and 19 show the modeling and MSE during the testing. Table 3 shows the MSE of (21) during the training and testing.

4.5 Discussion

Example 1. In Figs. 11, 13, since the signal of the genetic optimizer reaches better the signal of an awake driver than the signal of the simplex optimizer and bat optimizer, it is observed that the genetic optimizer reaches better step size than the

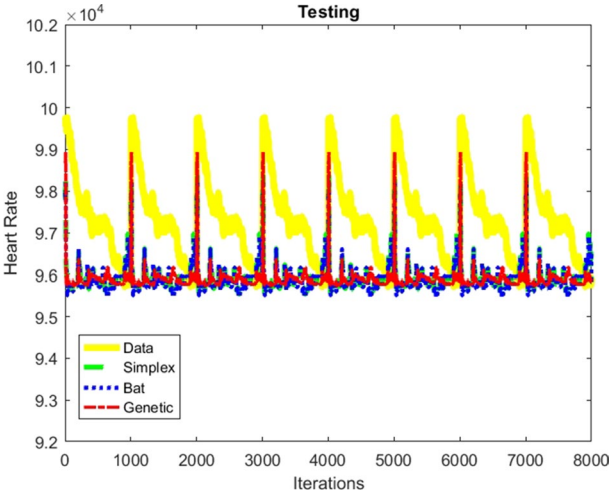

Fig. 13 Modeling during the testing for example 1

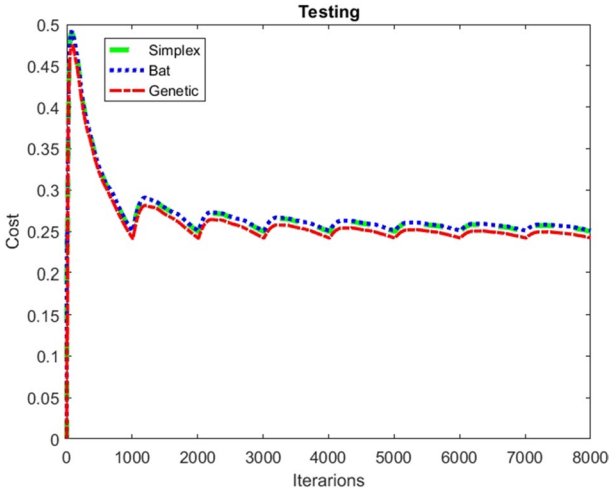

Fig. 14 MSE during the testing for example 1

Table 2 MSE for example 1

	Training	Testing
Simplex optimizer	0.0155	0.2500
Bat optimizer	0.0153	0.2510
Genetic optimizer	0.0130	0.2421

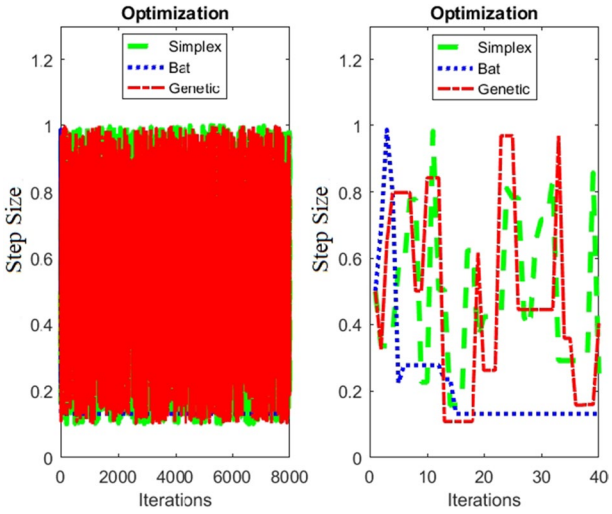

Fig. 15 Step size during the optimization for example 2

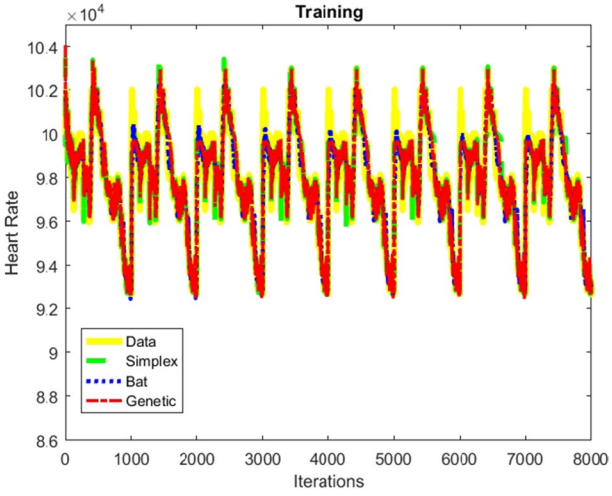

Fig. 16 Modeling during the training for example 2

simplex optimizer and bat optimizer. In Figs. 12, 14, and Table 2, since the MSE is the smallest for the genetic optimizer, it is noticed that the genetic optimizer reaches better step size than the simplex optimizer and bat optimizer. Thus, the genetic optimizer is the optimum one in example 1.

Example 2. In Figs. 16, 18, since the signal of the genetic optimizer reaches better the signal of a somnolent driver than the signal of the simplex optimizer and bat optimizer, it is observed that the genetic optimizer reaches better step size than the simplex optimizer and bat optimizer. In Figs. 17, 19, and Table 3, since the MSE is

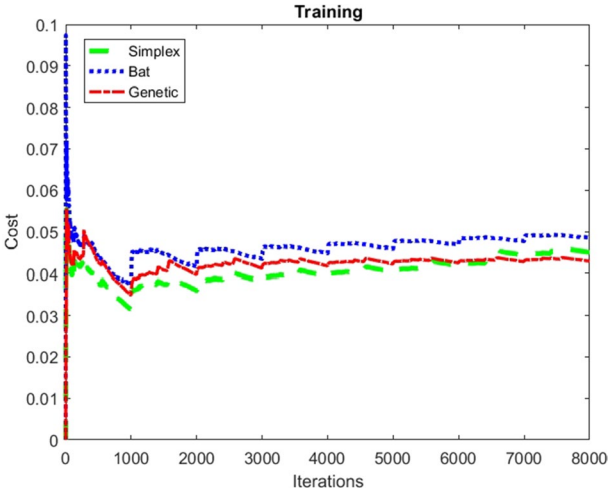

Fig. 17 MSE during the training for example 2

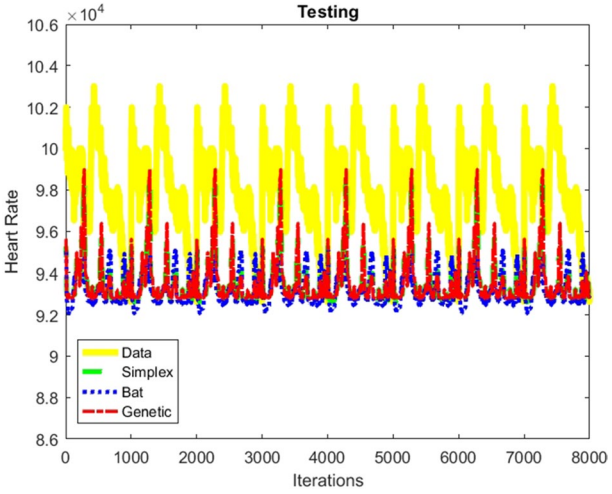

Fig. 18 Modeling during the testing for example 2

the smallest for the genetic optimizer and bat optimizer, it is noticed that the genetic optimizer reaches better step size than the simplex optimizer and bat optimizer. Thus, the genetic optimizer is the optimum one in example 2.

Remark 1 Even if the genetic optimizer, simplex optimizer, and bat optimizer have similar structure, the genetic optimizer outperform the simplex optimizer and bat optimizer because the genetic optimizer utilizes different equations than the utilized by the simplex optimizer and bat optimizer.

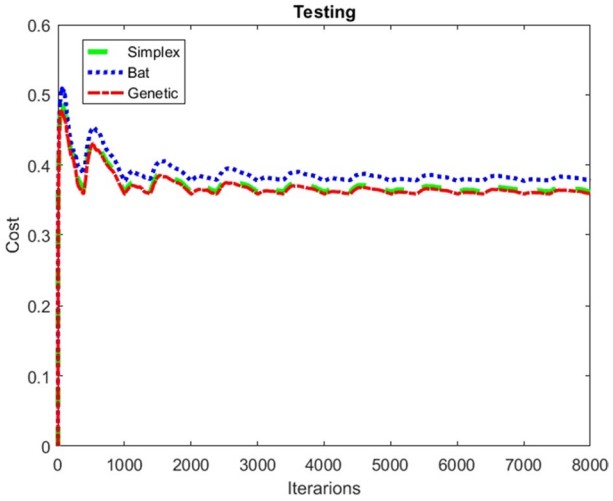

Fig. 19 MSE during the testing for example 2

Table 3 MSE for example 2

	Training	Testing
Simplex optimizer	0.0450	0.3616
Bat optimizer	0.0485	0.3775
Genetic optimizer	0.0429	0.3587

5 Conclusion

On the basis of the modeling, three strategies denoted as simplex optimizer, bat optimizer, and our genetic optimizer were compared to seek an acceptable step size in the gradient strategy for an optimized radial basis function network. To validate the performance of the optimized radial basis function network, the fatigue driving modeling in a vehicle was evaluated by two examples, where the genetic optimizer achieved better step size in the optimized radial basis function network than the simplex optimizer and bat optimizer. The suggested strategy has other applications as are the mechatronic, robotic, energy, electric, electronic, or computing. In the future, other optimizer will be studied to seek an acceptable step size in the gradient strategy for an optimized radial basis function network, or the suggested optimizer will be utilized in other kind of optimized radial basis function network.

Acknowledgements Authors thank the Instituto Politécnico Nacional, the Consejo Nacional de Humanidades Ciencias y Tecnologías, the Secretaría de Investigación y Posgrado, and the Comisión de Operación y Fomento de Actividades Académicas for their support.

Author contributions J.J.R. and M.A.I. were involved in the investigation and formal analysis; D.G. and J.P. contributed to the software and validation; A.Z. and C.A-I assisted in writing—original draft, review and editing. All authors have read and agreed to the published version of the manuscript.

Funding This research was funded by the Secretaría de Investigación y Posgrado (SIP), and the Comisión de Operación y Fomento de Actividades Académicas (COFAA), both from the Instituto Politécnico Nacional, and by the Consejo Nacional de Ciencia y Tecnología (CONACYT), México.

Declarations

Conflict of interest The authors declare that there are no competing interests regarding the publication of this paper.

References

1. Liu H, Wang Z, Fei W, Li J (2020) H-infinity and l2-l-infinity state estimation for delayed memristive neural networks on finite horizon: the round-robin protocol. Neural Netw 132:121–130
2. Li J, Wang Z, Dong H, Fei W (2020) Delay-distribution-dependent state estimation for neural networks under stochastic communication protocol with uncertain transition probabilities. Neural Netw 130:143–151
3. Zhou Q, Li Y, Zhao D, Li J, Williams H, Xu H, Yan F (2022) Transferable representation modelling for real-time energy management of the plug-in hybrid vehicle based on k-fold fuzzy learning and gaussian process regression. Appl Energy 305:117853
4. Li J, Zhou Q, He Y, Shuai B, Li Z, Williams H, Xu H (2019) Dual-loop online intelligent programming for driver-oriented predict energy management of plug-in hybrid electric vehicles. Appl Energy 253:113617
5. Abbas S, Javaid N, Almogren A, Gulfam SM, Ahmed A, Radwan A (2021) Securing genetic algorithm enabled sdn routing for blockchain based internet of things. IEEE Access 9:139739–139754
6. DAngelo G., Palmieri F., (2021) Gga: a modified genetic algorithm with gradient-based local search for solving constrained optimization problems. Inf Sci 547:136–162
7. Yang J, Hu Y, Zhang K, Wu Y (2021) An improved evolution algorithm using population competition genetic algorithm and self-correction bp neural network based on fitness landscape. Soft Comput 25:1751–1776
8. Hamdia KM, Zhuang X, Rabczuk T (2021) An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Comput Appl 33:1923–1933
9. Huang Y, Gao Y, Gan Y, Ye M (2021) A new financial data forecasting model using genetic algorithm and long short-term memory network. Neurocomputing 425:207–218
10. Pan Y, Yang Y, Li W (2021) A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-uav. IEEE Access 9:7994–8005
11. Zhang B, Wang X, Wang H (2021) Virtual machine placement strategy using cluster-based genetic algorithm. Neurocomputing 428:310–316
12. Zhou Y, Zhang W, Kang J, Zhang X, Wang X (2021) A problem-specific non-dominated sorting genetic algorithm for supervised feature selection. Inf Sci 547:841–859
13. Alouane B, Boulif M (2023) Fuzzy constraint prioritization to solve heavily constrained problems with the genetic algorithm. Eng Appl Artif Intell 119:105768
14. Pavlovskii VV, Derevitskii IV, Kovalchuk SV (2022) Hybrid genetic predictive modeling for finding optimal multipurpose multicomponent therapy. J Comput Sci 63:101772
15. Durasevic M, Jakobovic D (2022) Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. J Comput Sci 61:101649
16. Liu Z, Wang J, Gao Z, Wei J (2023) Privacy preserving-edge computing offloading scheme based on whale optimization algorithm. J Supercomput 79:3005–3023
17. Egrioglu E, Grosan C, Bas E (2023) A new genetic algorithm method based on statistical-based replacement for the training of multiplicative neuron model artificial neural networks. J Supercomput 79:7286–7304
18. Teijeiro D, Amor M, Doallo R, Corbelle E, Porta J, Parapar J (2022) Land consolidation through parcel exchange among landowners using a distributed spark-based genetic algorithm. J Supercomput 78:19522–19544
19. Abasabadi S, Nematzadeh H, Motameni H, Akbari E (2022) Hybrid feature selection based on sli and genetic algorithm for microarray datasets. J Supercomput 78:19725–19753

20. Silva MQM, Valente IM, Almeida PJ, Rodrigo Santos J (2023) Carbon dioxide spectrophotometric determination in whole roasted coffee beans using a total analysis system after super-modified simplex optimization. J Food Compos Anal 115:104978
21. Mok MCH, Yeoh CV, Tan MK, Foo JJ (2023) Space-filling single square and square fractal grids induced turbulence: reynolds stress model parameters-optimization. Results Eng 17:100806
22. Rubio JJ (2023) Bat algorithm based control to decrease the control energy consumption and modified bat algorithm based control to increase the trajectory tracking accuracy in robots. Neural Netw 161:437–448
23. Soriano LA, Rubio JJ, Orozco E, Cordova DA, Ochoa G, Balcazar R, Cruz DR, Meda-Campaña JA, Zacarias A, Gutierrez GJ (2021) Optimization of sliding mode control to save energy in a scara robot. Mathematics 9(24):3160
24. Wang H, Ji C, Shi C, Yang J, Wang S, Ge Y, Chang K, Meng H, Wang X (2023) Multi-objective optimization of a hydrogen-fueled wankel rotary engine based on machine learning and genetic algorithm. Energy 263:125961
25. Khan PW, Yeun CY, Byun YC (2023) Fault detection of wind turbines using scada data and genetic algorithm-based ensemble learning. Eng Fail Anal 148:107209
26. Li B, Shen L, Zhao Y, Yu W, Lin H, Chen C, Li Y, Zeng Q (2023) Quantification of interfacial interaction related with adhesive membrane fouling by genetic algorithm back propagation (gabp) neural network. J Colloid Interface Sci 640:110–120
27. Acampora G, Chiatto A, Vitiello A (2023) Genetic algorithms as classical optimizer for the quantum approximate optimization algorithm. Appl Soft Comput 142:110296
28. Ye Y, Huang Q, Rong Y, Yu X, Liang W, Chen Y, Xiong S (2023) Field detection of small pests through stochastic gradient descent with genetic algorithm. Comput Electr Agric 206:107694
29. Wang Y, Zhang Y, Zhang C, Zhou J, Hu D, Yi F, Fan Z, Zeng T (2023) Genetic algorithm-based fuzzy optimization of energy management strategy for fuel cell vehicles considering driving cycles recognition. Energy 263:126112
30. Cui H, Qiu J, Cao J, Guo M, Chen X, Gorbachev S (2023) Route optimization in township logistics distribution considering customer satisfaction based on adaptive genetic algorithm. Math Comput Simul 204:28–42
31. Tutumlu B, Sarac T (2023) A mip model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting. Comput Oper Res 155:106222
32. Gorp R, V., Heijden M. V. d., Sadeghi M. A., Gostick J., Forner-Cuenca A., (2023) Bottom-up design of porous electrodes by combining a genetic algorithm and a pore network model. Chem Eng J 455:139947
33. Wang Y, Wei Y, Wang X, Wang Z, Wang H (2023) A clustering-based extended genetic algorithm for the multidepot vehicle routing problem with time windows and three-dimensional loading constraints. Appl Soft Comput 133:109922
34. Liu H, Niu Z, Du J, Lin X (2023) Genetic algorithm for delay efficient computation offloading in dispersed computing. Ad Hoc Netw 142:103109
35. Shahryari M-S, Farzinvash L, Feizi-Derakhshi M-R, Taherkordi A (2023) High-throughput and energy-efficient data gathering in heterogeneous multi-channel wireless sensor networks using genetic algorithm. Ad Hoc Netw 139:103041
36. Wang Y, Zhu Z, Sha A, Hao W (2023) Low cycle fatigue life prediction of titanium alloy using genetic algorithm-optimized bp artificial neural network. Int J Fatigue 172:107609
37. Liu C, Zhang F, Zhang H, Shi Z, Zhu H (2023) Optimization of assembly sequence of building components based on simulated annealing genetic algorithm. Alex Eng J 62:257–268
38. Bahadur DJ, Lakshmanan L (2023) A novel method for optimizing energy consumption in wireless sensor network using genetic algorithm. Microprocess Microsyst 96:104749
39. Qin Y, Li Z, Ding J, Zhao F, Meng M (2023) Automatic optimization model of transmission line based on gis and genetic algorithm. Array 17:100266
40. Shameem M, Nadeem M, Zamani AT (2023) Genetic algorithm based probabilistic model for agile project success in global software development. Appl Soft Comput 135:109998

Authors and Affiliations

José de Jesús Rubio[1] · Marco Antonio Islas[1] · Donaldo Garcia[2] · Jaime Pacheco[1] · Alejandro Zacarias[1] · Carlos Aguilar-Ibañez[2]

✉ José de Jesús Rubio
rubio.josedejesus@gmail.com

Marco Antonio Islas
anto_islas@outlook.com

Donaldo Garcia
donigj00@gmail.com

Jaime Pacheco
jpachecoma@gmail.com

Alejandro Zacarias
azacariass@gmail.com

Carlos Aguilar-Ibañez
carlosaguilari@cic.ipn.mx

[1] Sección de Estudios de Posgrado e Investigación, Esime Azcapotzalco, Instituto Politécnico Nacional, Av. de las Granjas no. 682, Col. Santa Catarina, 02250 México D.F., México

[2] Centro de Investigación en Computación, Instituto Politécnico Nacional, Av. Juan de Dios Bátiz, Col. San Pedro Zacatenco, 07738 México City, México