Asia Pac. J. Math. 2025 **12**:69

HYBRIDIZATION OF IMPROVED REAL-CODED GENETIC, TEACHING-LEARNING-BASED OPTIMIZATION AND NEURAL NETWORK (gaTLBO_NNA) ALGORITHM FOR GLOBAL OPTIMIZATION OF FUNCTIONS

HAOUA TINDE[1], WENDDABO OLIVIER SAWADOGO[1,2,*], BEN-STHAL SAKOMA YELINGUE[1,3], PENGDEWENDÉ OUSSÉNI FABRICE OUEDRAOGO[1,4]

[1]LANIBIO Laboratory, University Joseph KI-ZERBO, Ouagadougou, Burkina Faso

[2]LARED Laboratory, University LEDEA Bernard OUEDRAOGO, Burkina Faso

[3]LIS Laboratory, University of Banguy, Central African Republic

[4]LaST Laboratory, University of Thomas SANKARA, Burkina Faso

[*]Corresponding author: wenddabo81@gmail.com

Received Jun. 23, 2025

Abstract. In this paper, we proposed a new hybrid algorithm for the minimization of real functions. This algorithm, called gaTLBO_NNA, is a combination of three algorithms: Improved real-coded genetic algorithm (IRGA), the Teaching–learning-based optimization (TLBO) algorithm and the Neural Network Algorithm (NNA). First we evaluated our algorithm on some benchmark test functions and compared it to some metaheuristics. The results showed that the proposed hybrid algorithm delivers better performance for most of the tested functions.We then applied our algorithm to estimate the parameters of a COVID-19 model using data from Morocco.

2020 Mathematics Subject Classification. 35A01, 92B20, 34C11, 47H10.

Key words and phrases. optimization; metaheuristic; mathematical model in epidemiology; inverse problem.

1. Introduction

Global optimization of functions presents a common challenge in the fields of engineering. Mathematically, this problem can be formulated as follows:

$$\min_{x\in\mathcal{D}} f(x) \tag{1}$$

where $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$, f a numeric function of $\mathbb{R}^n$, $\mathcal{D} = \prod_{i=1}^{n}[a_i, b_i]$, a_i and b_i are reals. Metaheuristic optimization techniques are particularly well-suited to optimization problems where the search space is large, where parameters interact in complex ways, and where available information about the function to be optimized is limited [1,12]. They do not require specific assumptions regarding

DOI: 10.28924/APJM/12-69

the regularity of the objective function. Furthermore, metaheuristic algorithms do not rely on the successive derivatives of the functions to be optimized, which eliminates the need for a continuity assumption. As a result, the function to be optimized can be derived from a simulation. These algorithms often prove to be much more robust in their ability to identify the global optimum, with reduced sensitivity to initial conditions.

Numerous metaheuristic algorithms have been presented in the literature to solve optimization problems. Among them are the genetic algorithm proposed by Holland [12]. The Grey Wolf Optimizer(GWO) algorithm which mimics the leadership hierarchy and hunting mechanism of grey wolves in nature proposed by S. Mirjalili et al [1]. The HmGAGWO algorithm proposed by Sawadogo et al. [8] combine GA and GWO algorithm. Neural Network Algorithm (NNA) proposed by A. Sadollah et al. [2] is inspired by biological nervous system and Artificial Neural Networks (ANNs). Particle Swarm Optimization (PSO) algorithm proposed by Kennedy and Eberhart [3] is inspired from social organization of birds. The Improved Real-coded Genetic Algorithm (IRGA) proposed by [7] is an improvement to the genetic algorithm that allows solving optimization problems where variables are continuous rather than discrete. Teaching–learning-based optimization (TLBO algorithm) is an optimization algorithm based on teaching and learning proposed by by R.V. Rao et al. [10]. In this work, we propose an hybridization of the IRGA, TLBO, and NNA algorithms. The idea is to combine the exploration capacity of the IRGA algorithm, the refinement ability of the NNA algorithm, and the capability of the TLBO algorithm to reach the global optimum.

The rest of this paper is organized as follows: Section 2, we provide a comprehensive review of the Neural Networks (NNA) algorithm, Improved Real-coded Genetic Algorithm (IRGA) and the TLBO algorithm. Section 3 is dedicated to presenting the proposed hybridization method. In section 4, present the results on test functions and the resolution of the of the parameter estimation problem. We conclude with Section 5.

2. Presentation of Algorithm Used

2.1. **Improved real-coded genetic algorithm (IRGA).**

The Improved Real-coded Genetic Algorithm (IRGA) is an improvement to the genetic algorithm that allows solving optimization problems where variables are continuous rather than discrete [7]. The IRGA is based on the Real-coded Genetic Algorithm (RGA) by integrating specific modifications in order to improve its performance or its effectiveness in solving problems.

The IRGA offers three operators, namely tournament selection, directional crossover (DX) and directional mutation.

The discussion focuses on the operating mechanism of the proposed directional mutation (DM) and the the directional crossover (DX) operator developed for the RGA. Both operators are influenced by

the directional information of the given optimization problem. This information directs the search process towards the most promising areas of the variable space where the chances of obtaining the best solutions are high [7].

2.1.1. *The DX directional crossover.* The directional crossover operator (DX) is similar to the (DM) operator. DX mainly has four (04) parameters namely the crossover probability (pc), the crossover probability variable by variable (pcv), the directional probability (pd) and the multiplication factor (α). Suppose p_1^j and p_2^j (j varies from 1 to d), two parents who participate in the crossing and they are not equal. p_{mean}^j and p_{best}^j are the mean of the two parents and the jth variable of the optimal solution, respectively. If p_{best}^j is greater than or equal to p_{mean}^j, the two child solutions denoted c_1 and c_2 are created as follows:

$$val = 1 - (0.5)^{e^{\left[\frac{\left|p_1^j - p_2^j\right|}{\left(y_u^j - y_l^j\right)}\right]}} \tag{2}$$

$$\beta = \frac{r_3}{\alpha^2} \tag{3}$$

$$c_1 = val \times \left(p_1^j + p_2^j\right) + \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_1^j - p_2^j\right| \; si \; r_4 \leq p_d \tag{4}$$

$$c_2 = (1 - val) \times \left(p_1^j + p_2^j\right) - \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_1^j - p_2^j\right| \; si \; r_4 \leq p_d \tag{5}$$

$$c_1 = val \times \left(p_1^j + p_2^j\right) - \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_1^j - p_2^j\right| \; si \; r_4 > p_d \tag{6}$$

$$c_2 = (1 - val) \times \left(p_1^j + p_2^j\right) + \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_1^j - p_2^j\right| \; si \; r_4 > p_d \tag{7}$$

r_3 and r_4: two different random numbers created in the range of (0,1)

val and β are the two intermediate parameters

y_u^j and y_l^j: the upper and lower limits of the jth variable

α: multiplication factor

If p_{best}^j is less than p_{mean}^j, the children are produced as follows:

$$c_1 = val \times \left(p_1^j + p_2^j\right) - \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_1^j - p_2^j\right| \; si \; r_4 \leq p_d \tag{8}$$

$$c_2 = (1 - val) \times \left(p_1^j + p_2^j\right) + \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_1^j - p_2^j\right| \; si \; r_4 \leq p_d \tag{9}$$

$$c_1 = val \times \left(p_1^j + p_2^j\right) + \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_1^j - p_2^j\right| \; si \; r_4 > p_d \tag{10}$$

$$c_2 = (1 - val) \times \left(p_1^j + p_2^j\right) - \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_1^j - p_2^j\right| \; si \; r_4 > p_d \tag{11}$$

If the breeding parents have equal values and if $p_{best}^j \neq p_{mean}^j$, then the solutions for the offspring are obtained as follows:

$$val = 1 - (0.5)^{e^{\left[\frac{\left|p_{best}^j - p_{mean}^j\right|}{\left(y_u^j - y_l^j\right)}\right]}} \tag{12}$$

$$\beta = \frac{r_3}{\alpha^2} \tag{13}$$

$$c_1 = val \times \left(p_{best}^j + p_{mean}^j\right) + \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_{best}^j - p_{mean}^j\right| \; si \; r_4 \leq p_d \tag{14}$$

$$c_2 = (1 - val) \times \left(p_{best}^j + p_{mean}^j\right) - \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_{best}^j - p_{mean}^j\right| \; si \; r_4 \leq p_d \tag{15}$$

$$c_1 = val \times \left(p_{best}^j + p_{mean}^j\right) - \alpha^{r_3} \times e^{(1-\beta)} \times (1 - val) \times \left|p_{best}^j - p_{mean}^j\right| \; si \; r_4 > p_d \tag{16}$$

$$c_2 = (1 - val) \times \left(p_{best}^j + p_{mean}^j\right) + \alpha^{(1-r_3)} \times e^{(-\beta)} \times val \times \left|p_{best}^j - p_{mean}^j\right| \; si \; r_4 > p_d \tag{17}$$

c_1 is recognized as either the first child or the second.

`Pseudo code of the directional crossover DX of IRGA`

`Input: Two parent (` p_1 `and` p_2`) with d dimensions,` p_c, p_{cv}, p_d, p_{best}, α

Output: offspring (Ch_1 and Ch_2)

---

if rand $\leq p_c$

  forj=1 to d

    if $r_1 \leq p_{cv}$ (% r_1 is random number created in the range (0,1))

      if $\left|p_1^j - p_2^j\right| > 0$

      Determine p_{mean}^j

      Determine val and β by applying 2 and 3

        if $(p_{best}^j \geq p_{mean}^j)$

          if $r_4 \leq p_d$

        calculate c_1 and c_2 by applying 4 and 5

else
calculate c_1 and c_2 by applying 6 7
endif
else
if $r_4 \leq p_d$
calculate c_1 and c_2 using 8 and 9
else
calculate c_1 and c_2 by applying 10 and 11
endif
endif
Apply boundary constraint
Apply child recognition conditions
else
$Ch_1^j = p_1^j$
$Ch_2^j = p_2^j$
endif
endif
else
$Ch_1^j = p_1^j$
$Ch_2^j = p_2^j$
endif
end of for loop
else
$Ch_1^j = p_1^j$
$Ch_2^j = p_2^j$
endif

2.1.2. *Directional mutation* (DM). Let N be the size of the population and d be the total number of variables in an optimization problem. Consider a parent y_i^j and its mutated solutions y_m where i and j vary from 1 to N and from r to d respectively. y_i^j participate in the mutation operation and p_{best}^j the best solution. y_m is created using equation 3:

$$\beta_1 = e^{2r-\frac{2}{r}} \tag{18}$$

$$\beta_2 = e^{r-\frac{2}{r}} \tag{19}$$

$$y_m = \begin{cases} y_i^j + \beta_1 \times \left(y_u^j - y_i^j\right) \ si\ r_2 \leq p_d \\ y_i^j - \beta_2 \times \left(y_i^j - y_l^j\right) \ sinon \end{cases} \tag{20}$$

where:

β_1 and β_2: intermediate parameters

r_1 and r_2: two random numbers created in the range of (0,1) and $r \neq 0$

y_u^j and y_l^j: the upper and lower limits of the jth variable

p_d: the directional probability and between 0.5 and 1

p_m: the probability of mutation

If p_{best}^j is considered less than y_i^j then y_m is generated by equation 4:

$$y_m = \begin{cases} y_i^j - \beta_1 \times \left(y_i^j - y_l^j\right) \ if\ r_2 \leq p_d \\ y_i^j + \beta_2 \times \left(y_u^j - y_i^j\right) \ else \end{cases} \tag{21}$$

The mutated solutions will always be created within the limits of the variables.

`Pseudo code of the proposed DM operator`

`Input: Parent solution (`y_i^j`), DM parameters (` p_m `and` p_d `)`

---

if $r_1 \leq p_m$

  calculate β_1 using 18

  calculate β_2 using 19

  if p_{best}^j

    evaluate y_m using 20

  else

    evaluate y_m using 21

  end if

else

  $y_m = y_i^j$

end if

2.2. **Teaching–learning-based optimization (TLBO) algorithm.** TLBO is an optimization algorithm based on teaching and learning and propose by R.V. Rao , V.J. Savsani, D.P. Vakharia [10]. It has two phases: the teaching phase and the learning phase.

2.2.1. *The teaching phase.* the teaching phase which corresponds to learning alongside the teacher. The teacher can only improve the average performance of the class according to the capacity of the class. Let M_i be the average situation of the population at iteration k and $X_{teacher}$ be the teacher who tries to make M_i converge towards its own level. In this case, a solution is updated based on the difference between the average situation and the teacher's situation as follows [9], [10]:

$$X_{diff} = r_i \times (M_{new} - T_F \times M_i) \tag{22}$$

$$T_F = round\,[1 + rand(0,1)\,\{2-1\}] \tag{23}$$

$$X^i_{new} = X^i_{old} + X_{diff} \tag{24}$$

where:

T_F= learning factor, its value can be 1 or 2

r_i: denotes a vector of random numbers in the interval $[0, 1]$

2.2.2. *The learning phase.* the learning phase represents the acquisition of knowledge through interaction between learners.

Learners improve their knowledge by interacting with each other.

For any learner X_i, the learner X_j is randomly selected $(i \neq j)$,

and if $f(X_i) < f(X_j)$ then:

$$X_{new,i} = X_{old,i} + r_i \times (X_i - X_j) \tag{25}$$

And conversely, if $f(X_i) > f(X_j)$ then:

$$X_{new,i} = X_{old,i} + r_i \times (X_j - X_i) \tag{26}$$

`Pseudo code for TLBO algorithm`

Initialize the population size N and number of generations N_g.

For (k=1 to N_g) **do**

% teacher phase

Find the mean of each design variable X_{mean}

$[X_{teacher} \rightarrow X$ with $f(X)_{max}]$

for (i=1 to n) **do**

Calculate $T_{F,i} = round\,[1 + rand(0,1)\,\{2-1\}]$

$X_{new,i} = X_i + rand(0,1)\,[X_{teacher} - T_{F,i} \times X_{mean}]$

Calculate $f(X_{new,i}) for X_{new,i}$

$f(X_{new,i}) < f(X_i)$ then

$X_i = X_{new,i}$

end if

% End of teacher phase

% Student phase

Select a learner randomly X_j such that $j \neq i$

if $f(X_i) < f(X_j)$ then

$X_{new,i} = X_{old,i} + randi(X_i - X_j)$

else

$X_{new,i} = X_{old,i} + randi(X_j - X_i)$

end if

if $f(X_{new,i}) < f(X_i)$ then

$X_i = X_{new,i}$

end if

% end of student phase

end for

end for

2.3. **Neural Network Algorithm (NNA).** The NNA (Neural Network Algorithm) was developped by Ali Sadollah, Hassan Sayyaadi, Anupam Yadav [4]. Neural Network Algorithm (NNA) is a metaheuristic optimization algorithm inspired by biological nervous systems and artificial neural networks (ANN). It is designed to solve complex optimization problems. In NNA, the best solution obtained at each iteration is assumed to be the target data, and the goal is to minimize the error between the target data and the other predicted model solutions. NNA is an evolutionary algorithm that is population-based and involves initializing the population, updating weight matrices, defining bias operators, as well as transferring and applying the operators.

2.3.1. *Initial Population.* The NNA algorithm begins with the random generation of an initial population within the space of feasible solutions. The generated solutions are called pattern solution.

Let D be the dimension and N the number of generated solutions, then the solution pattern X^l is given by

$X^l = [x_1^l, x_2^l, \cdots, x_D^l]$ where $l = 1, \cdots, N$ and

$$x_j^l = LB_j + rand(UB_j - LB_j) \tag{27}$$

where LB and UB are are respectively the lower and upper bounds of the variables.

The solution matrix X is given by

$$X = \begin{pmatrix} x_1^1 & \cdots & x_i^1 & \cdots & x_D^1 \\ x_1^2 & \cdots & x_i^2 & \cdots & x_D^2 \\ \vdots & \vdots & \vdots & \vdots & \\ x_1^N & \cdots & x_i^N & \cdots & x_D^N \end{pmatrix} \tag{28}$$

2.3.2. *Weight matrix.* In the NNA algorithm, the population is updated using a neural network model-like approach. In the search space, the initial population $X^r = [x_1^r, x_2^r, \ldots, x_N^r]$ is updated through the weight matrix

$W^r = [w_1^r, w_2^r, \ldots, w_N^r]$ for any generation r.Thus, $x_i^r = [x_{i,1}^r, x_{i,2}^r, \ldots, x_{i,D}^r]$ and $w_i^r = [w_{i,1}^r, w_{i,2}^r, \ldots, w_{i,D}^r]$, where $i = 1, 2, \ldots, N_p$.Where, x_i^r represents the i^{th} individual vector and w_i^r represents the i^{th} weight vector, both with D dimensions.

The weigth matrix is given by

$$W(t) = [W_1, W_2, \cdots, W_N] = \begin{pmatrix} w_{11} & \cdots & w_{i1} & \cdots & w_{N1} \\ w_{12} & \cdots & w_{i2} & \cdots & w_{N2} \\ \vdots & \vdots & \vdots & \vdots & \\ w_{1N} & \cdots & w_{iN} & \cdots & w_{NN} \end{pmatrix} \tag{29}$$

It is desirable to impose constraints on the weights associated with new model solutions so that significant biases are prevented in the generation and transmission of these solutions. In this way, NNA was equipped to regulate its behavior through subtle deviations. After initializing the weights, the one corresponding to the desired solution (X_{target}), i.e., the target weight (W_{target}), is chosen from the weight matrix W. Therefore, the summation of the weight matrix must adhere to the following conditions:

$$\sum_{j=1}^{N} w_{i,j}(t) = 1, \quad i = 1, 2, \ldots, N \tag{30}$$

where

$$w_{i,j} \in \mathcal{U}[0, 1], \quad i, j = 1, 2, \ldots, N \tag{31}$$

2.3.3. *Pattern solution and weights update.* The formula for generating a new population at the $(t+1)$th iteration can be expressed by:

$$X_l(t+1) = X_l(t) + \sum_{i=1}^{N} w^{jl}(t) \times X_l(t), \tag{32}$$

After generating the new solutions, the weight matrix must be updated. The following equation represents the process of updating the weights of the matrix based on the newly generated models and the target weight.

$$W_l(t+1) = W_l(t) + 2 \times \text{rand}(0,1) \times (W_{\text{target}}(t) - W_l(t)) \tag{33}$$

where $W_{\text{target}}(t)$ is the vector of optimal target weights obtained in each iteration. The weight matrix must always satisfy constraints during the optimization process.

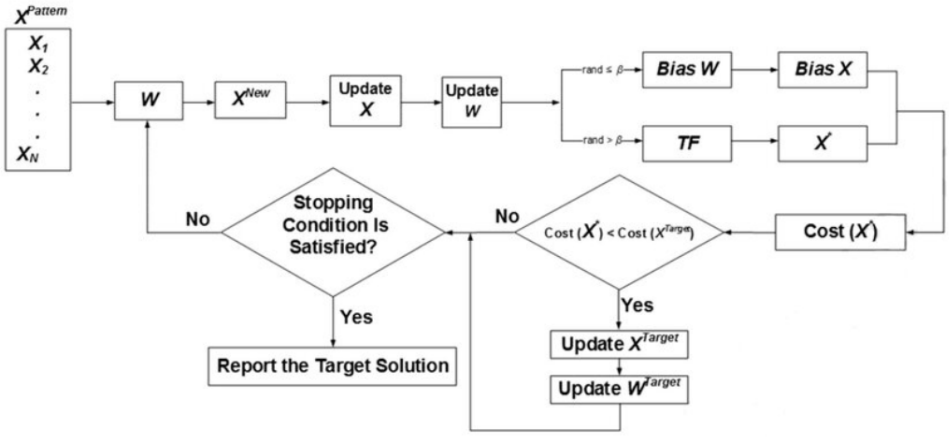

FIGURE 1. Processes of the NNA [4], [5], [6]

2.3.4. *Bias Operator.* In NNA, the bias operator modifies a certain percentage of model solutions in the new population, as well as the updated weight matrix. Therefore, the bias operator is another way to explore the search space. The parameter β determines the percentage of model solutions and is initially set to 100%. Its value has been reduced as described in [4], [5], [6].

$$\beta(t+1) = \beta(t) \times 0.99 \quad \text{for} \quad t = 1, 2, 3, \ldots, \text{Max\_Iteration} \tag{34}$$

The bias population is defined by:

$$x^r_{i,P(s)} = l_{P(s)} + (u_{P(s)} - l_{P(s)}) \times \alpha_1, \quad s = 1, 2, \ldots, N \tag{35}$$

where $L = (l_1, l_2, \ldots, l_D)$ and $U = (u_1, u_2, \ldots, u_D)$ are the lower and upper bounds of the variables. P denotes a set of N_P integers randomly selected from the range 0 to D, and N_P is D multiplied by β_r.

α_1 is a random number between 0 and 1 that obeys a uniform distribution.

he scientific representation for defining the bias weight matrix can be formulated as follows:

$$w^r_{i,Q(t)} = \alpha_2, \quad t = 1, 2, \ldots, P_w \tag{36}$$

where α_2 is a random number between 0 and 1, following a uniform distribution [6].

2.3.5. *Transfer Function Operator.* The transfer function (TF) operator moves the new model solutions from their current positions in the search space to new positions, facilitating the update and generation of higher-quality solutions towards the target solution. Its equation is:

$$X^*_l(t+1) = TF(X_l(t+1)) = X_l(t+1) + 2 \times rand \times (X_{Target}(t) - X_l(t+1)) \tag{37}$$

The pseudocode of NNA is given [5], [6]:

`The pseudocode of the NNA algorithm`

Initialize the population X and the weight matrix W

Calculate the fitness value of each solution and then set X_{target} and W_{target}.

repeat

Generate the new solution $X(t+1)$ by Equation 32 and new weight matrix $W(t+1)$ by Equation 33

for $i = 1$ to N **do**

if$\beta(t) \geq$ rand

Perform the bias operator for updating the new pattern solution $X^i(t+1)$ by Equation 35 and the weight matrix $W^i(t+1)$ by Equation 36.

else

Perform the transfer function operator for updating the solution $X^i(t+1)$ via Equation 37.

endif

endfor

Generate the new modification factor $\beta(t+1)$ by Equation 34.

Calculate the fitness value of each solution and find the target solution $X_{Target}(t+1)$ and the target weight $W_{Target}(t+1)$.

Until (stop condition=false)

---

3. Proposed Hybridization Method

3.1. **Concept of the Proposed Hybridization Method.** The proposed hybridization method combines NNA, IRGA, and TLBO algorithms and proceeds in three (03) stages:

(1) **Exploration Phase:** The IRGA initializes the search process to enhance exploration capability. It uses directional mutation (DM) and directional crossover (DX) operators to explore the search space. At the end of the steps of the IRGA algorithm, we obtain a population of size 2N.

(2) **Refinement Phase:** After the IRGA phase, the NNA uses a neural network to further optimize the refined solutions by focusing on local exploration and performance enhancement. It improves the N best solutions from IRGA by adjusting the neural network weights to obtain more precise results.

(3) **Optimization Phase:** TLBO is used to adjust the remaining N worst solutions. TLBO applies teaching and learning concepts to optimize the remaining individuals, using the best individuals as teachers and the others as students. It adjusts the solutions based on the performance of the teachers and students to avoid local optima and accelerate overall convergence.

3.2. **Steps of Hybridization. Step 1: Initialization of Parameters**

Define the parameters for the IRGA, NNA, and TLBO algorithms:Population size N, Number of variables d, Mutation probability p_m, Crossover probability p_c, Multiplication factor α, Learning factor TF

Step 2: Generation of the Population

Create an initial population P of N possible solutions X_i (with $i = 1, \ldots, N$) covering the search space.

Step 3: Application of IRGA for Each Iteration

For each solution X_i:

- Calculate the directional mutation parameters: eq.18 and eq. 19
- Apply directional mutation: eq. 20

Step 4: For Each Pair of Solutions X_1 and X_2

- Calculate the intermediate value: eq. 2
- Calculate the crossover parameters: eq. 3

Step 5: Teaching and Learning Phase for the N Worst Solutions

Identify the N worst solutions. Apply TLBO to improve these solutions:

Teaching Phase

(1) Calculate the teacher difference: eq. 22

(2) Calculate the learning factor: eq. 23

(3) Update the solution: eq. 24

Learning Phase

(1) For each learner X_i: Randomly select another learner X_j $(i \neq j)$.

(2) Apply the learning rule: eq. 25 and eq. 26

Step 6: NNA Phase for Each Iteration (Top N Solutions)

Identify the top N solutions. Use NNA to refine these solutions by adjusting the neural network weights for local optimization.

Figure 2 illustrates flowchart of the proposed hybridization algorithm gaTLBO_NNA.

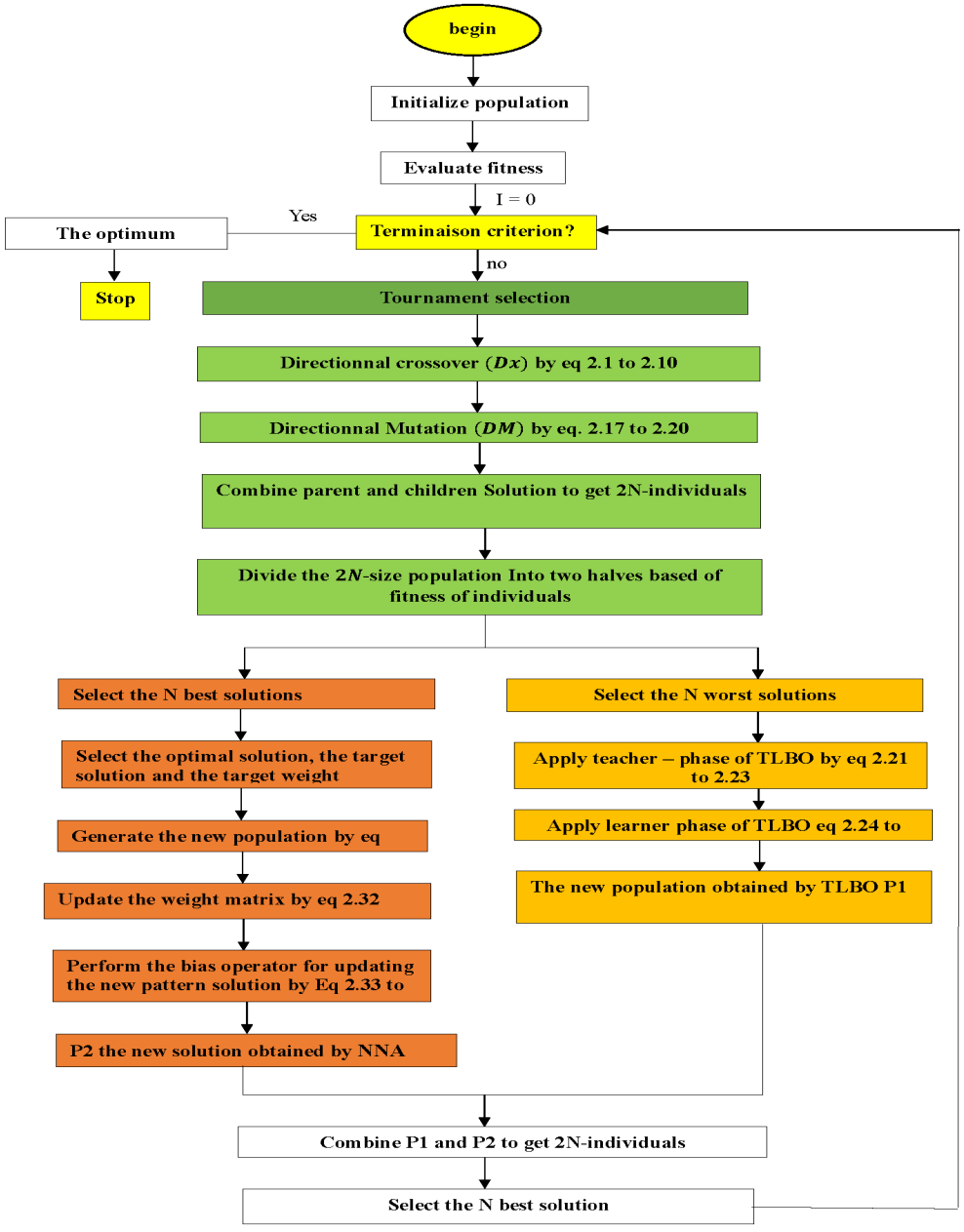

FIGURE 2. flowcharts of gaTLBO_NNA

4. Results and Discussion

4.1. **Results for benchmark functions.** In this part, we first use our algorithm on test functions. These test functions were used by S. Mirjalili and al [1]. In each case the results were compared with others algorithms.

4.1.1. *Test functions.* These functions are of three types. The table 1 presents the unimodal functions. The multimodal functions are given in the table 2 and the table 3 presents the fixed-dimension multimodal. The min column gives the minimum, dim is the dimension, and runge is the search interval. The comparison was made with IRGA, TLBO ,NNA, GWO and PSO.

Functions	Dim	Interval	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{i=j-1}^{i} x_j)^2$	30	$[-100, 100]$	0
$f_4(x) = \max_i \lvert x_i \rvert, 1 \leq i \leq n$	30	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	30	$[-100, 100]$	0
$f_6(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	30	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random(0, 1)$	30	$[-1.28, 1.28]$	0

Table 1. Unimodal test functions.

Functions	Dim	Interval	f_{min}
$f_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$[-500, 500]$	-418.9829×5
$f_9(x) = \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i) + 10)$	30	$[-5.12, 5.12]$	0
$f_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]$	0
$f_{12}(x) = \frac{\pi}{n}(10sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1}) + (y_n - 1)^2]) +$ $\sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = \frac{x_i+1}{,}4$	30	$[-50, 50]$	0
$f_{13}(x) = 0.1(sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 +$ $sin^2(2\pi x_n)]) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0
$f_{14}(x) = -\sum_{i=1}^{n} sin(x_i).\left(sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}, m = 10$	30	$[0, \pi]$	-4.687
$f_{15}(x) = \left[e^{-\sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{2m}} - 2e^{-\sum_{i=1}^{n} x_i^2}\right] . \prod_{i=1}^{n} cos^2\, x_i\,, m = 5$	30	$[-20, 20]$	-1
$f_{16} = \left\{\left[\sum_{i=1}^{n} sin^2(x_i)\right] - exp\left(-\sum_{i=1}^{n} x_i^2\right)\right\} .exp\left[-\sum_{i=1}^{n} sin^2\sqrt{\lvert x_i \rvert}\right]$	30	$[-10, 10]$	-1

Table 2. Multimodal test functions

$$u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m \text{ if } x_i>a \\ 0 \text{ if } -a<x_i<a \\ k(-x_i-a)^m \text{ if } x_i<-a \end{cases}$$

Functions	Dim	Interval	f_{min}
$f_{17}(x)=(\frac{1}{500}\sum_{j=1}^{25}\frac{1}{j+\sum_{j=1}^{2}}(x_i-a_{ij})^6)^{-1}$	2	$[-65,65]$	1
$f_{18}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	4	$[-5,5]$	0.0003
$f_{19}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]$	-1.0316
$f_{20}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2-\frac{10}{8\pi}cosx_1+10$	2	$[-5,5]$	0.398
$f_{21}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)][30+(2x_1-$ $3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2,2]$	3
$f_{22}(x)=-\sum_{i=1}^{4}c_i\,exp\left(-\sum_{j=1}^{3}a_{ij}\,(x_j-p_{ij})^2\right)$	3	$[1,3]$	-3.86
$f_{23}(x)=-\sum_{i=1}^{4}c_i\,exp\left(-\sum_{j=1}^{6}a_{ij}\,(x_j-p_{ij})^2\right)$	6	$[0,1]$	-3.32
$f_{24}(x)=-\sum_{i=1}^{5}\left[(X-a_i)\,(X-a_i)^T+c_i\right]^{-1}$	4	$[0,10]$	-10.1532
$f_{25}(x)=-\sum_{i=1}^{7}\left[(X-a_i)\,(X-a_i)^T+c_i\right]^{-1}$	4	$[0,10]$	-10.4028
$f_{26}(x)=-\sum_{i=1}^{10}\left[(X-a_i)\,(X-a_i)^T+c_i\right]^{-1}$	4	$[0,10]$	-10.5363

TABLE 3. Fixed-dimension multimodal test functions.

4.1.2. *Analysis of results.* For each category of test functions, we conducted 50 simulations. The statistical results are presented in tables 4 to 14. Analyzing these tables reveals that, for the majority of the tested functions, the hybrid algorithm outperforms the other algorithms. This improvement is particularly evident for unimodal functions. This finding is also supported by the convergence curves (see figures 3 to 6). In most cases, gaTLBONNA converges faster than the other algorithms.

	F1				F2			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	4,9405E-237	7,0567E-220	1,4113E-220	0	1,4335E-135	1,8529E-129	7,1338E-130	9,7643E-130
TLBO	1,37683E-82	3,9426E-81	1,21227E-81	1,59316E-81	3,80048E-41	1,68469E-40	9,04677E-41	6,14221E-41
MNA	4,20078E-08	3,37807E-07	1,85066E-07	1,15212E-07	1,59194E-05	9,73689E-05	4,28638E-05	3,17464E-05
PSO	9,48691E-35	9,40112E-31	2,04967E-31	4,11416E-31	3,15586E-11	2,69177E-06	5,48115E-07	1,19844E-06
GWO	2,91436E-42	9,55976E-41	2,75248E-41	3,86969E-41	1,98639E-24	9,27032E-24	4,56319E-24	2,84829E-24
IGA	1,193E-11	8,06713E-11	4,29479E-11	3,17375E-11	4,66772E-08	7,42594E-08	5,91751E-08	1,13168E-08

TABLE 4. Results for test functions -1

	F3				F4			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	1,089E-104	1,82014E-82	3,64029E-83	8,13993E-83	2,16411E-96	5,23728E-90	1,75624E-90	2,45594E-90
TLBO	2,91092E-16	7,46405E-15	4,09799E-15	2,99277E-15	3,94731E-33	8,04116E-33	5,68164E-33	1,74547E-33
MNA	5,186856	23,19358	11,25134	7,258951	0,084142	0,302758	0,166427	0,093845
PSO	0,032864	0,777496	0,529999	0,301551	0,069329	0,167601	0,127047	0,040792
GWO	2,13097E-13	2,71802E-12	9,21416E-13	1,04598E-12	5,24873E-11	1,65829E-09	4,8042E-10	6,70303E-10
IGA	81,78037	300,6139	178,01788	80,88151	0,290834	0,97095	0,54322	0,255875

Table 5. Results for unimodal test functions-2

	F5				F6			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	24,19995	24,8382	24,47988	0,233853	1,56089E-13	8,5255E-13	3,2531E-13	2,96753E-13
tlbo	15,8572	19,27742	17,51799	1,365953	2,18258E-19	6,66594E-17	2,24312E-17	2,64159E-17
mna	24,19478	77,79509	36,17072	23,30995	1,04949E-08	1,46497E-07	4,14141E-08	5,89757E-08
pso	7,146065	76,2392	30,57586	26,47320	1,2326E-32	9,01335E-30	1,98818E-30	3,93065E-30
Gwo	25,11981	26,19161	25,91423	0,451840	2,17191E-05	0,256310	0,101119	0,13845
iga	9,766462	82,92044	34,28521	28,12968	1,86075E-11	8,77665E-11	4,34317E-11	2,61044E-11

Table 6. Results for unimodal test functions-3

	F7				F8			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	-12569,47	-12451,04	-12545,8	52,97228	0	0	0	0
TLBO	-9521,248	-7667,773	-8652,383	673,1520	0,004355	7,963454	4,579853	4,195738
MNA	-11733,34	-10472,53	-10995,57	517,9359	21,90292	29,65968	26,47626	3,1208691
PSO	-7910,808	-5502,19	-6643,314	854,2481	32,83361	62,68227	49,34989	10,84456
GWO	-7104,147	-5682,756	-6371,445	608,7203	0	0,995783	0,199157	0,44533
IGA	-12323,47	-11959,35	-12201,20	166,2304	1,68015E-05	1,990903	0,398407	0,890232

Table 7. Results for unimodal test functions-4

	F9				F10			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBONNA	4,44089E-16	4,44089E-16	4,44089E-16	0	0	0	0	0
TLBO	3,9968E-15	3,9968E-15	3,9968E-15	0	0	0	0	0
MNA	5,08365E-05	8,3965E-05	6,12065E-05	1,31928E-05	6,29223E-08	0,012320	0,004928	0,006747
PSO	2,88658E-14	1,07025E-13	5,37348E-14	3,17764E-14	0	0,049176	0,016240	0,019168
GWO	2,53131E-14	2,88658E-14	2,81553E-14	1,58882E-15	0	0	0	0
IGA	1,01303E-06	1,79593E-06	1,44419E-06	3,84634E-07	4,05235E-11	1,72202E-10	9,45437E-11	5,38723E-11

Table 8. Results for unimodal test functions-5

	F11				F12			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	2,5945E-12	6,3122E-11	2,25807E-11	2,37103E-11	2,45617E-10	2,36323E-09	8,72838E-10	8,71201E-10
TLBO	4,79933E-20	6,51061E-19	2,07023E-19	2,60007E-19	6,41339E-19	0,010987	0,00879	0,004914
MNA	2,10608E-09	2,86032E-05	5,73135E-06	1,27857E-05	2,06984E-08	1,27004E-07	5,10189E-08	4,38597E-08
PSO	1,61088E-32	0,10367	0,020734	0,046362	1,47304E-32	0,010987	0,002198	0,004914
GWO	0,006499	0,07169	0,023546	0,027466	1,73259E-05	0,402586	0,202536	0,157927
IGA	1,85213E-13	5,61777E-13	3,22515E-13	1,57786E-13	3,78151E-12	5,76853E-11	2,93533E-11	2,50484E-11

TABLE 9. Results for unimodal test functions-6

	F13				F14			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	0,998004	0,998004	0,998004	1,57009E-16	0,000308	0,001223	0,000497	0,000406
TLBO	0,998004	0,998004	0,998004	0	0,000307	0,000624	0,00038	0,000140
MNA	0,998004	0,998004	0,998004	1,24127E-16	0,000342	0,000602	0,00044	0,000109
PSO	0,998004	1,9920	1,395615	0,544451	0,000307	0,00159	0,000565	0,000575
GWO	0,998004	12,67051	3,729324	5,071573	0,000307	0,020363	0,004319	0,008969
IGA	0,998004	0,998004	0,998004	0	0,000509	0,000725	0,000626	9,44299E-05

TABLE 10. Results for unimodal test functions-7

	F15				F16			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	-1,031628	-1,031628	-1,031628	1,92296E-16	0,397887	0,397887	0,397887	0
TLBO	-1,031628	-1,031628	-1,031628	0	0,397887	0,397887	0,397887	0
MNA	-1,031628	-1,031628	-1,031628	1,11022E-16	0,397887	0,397887	0,397887	0
PSO	-1,031628	-1,031628	-1,031628	0	0,397887	0,397887	0,397887	0
GWO	-1,031628	-1,031628	-1,031628	4,65713E-09	0,397887	0,397888	0,397888	4,24961E-07
IGA	-1,031628	-1,031628	-1,031628	0	0,397887	0,397887	0,397887	0

TABLE 11. Results for unimodal test functions-8

	F17				F18			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	3	3	3	2,30756E-15	-3,862782	-3,862782	-3,862782	4,44089E-16
TLBO	3	3	3	6,28037E-16	-3,862782	-3,862782	-3,862782	0
MNA	3	3	3	1,9984E-15	-3,862782	-3,862782	-3,862782	4,44089E-16
PSO	3	3	3	6,28037E-16	-3,862782	-3,862782	-3,862782	0
GWO	3	3	3	2,22367E-06	-3,862782	-3,8628	-3,862780	9,94071E-07
IGA	3	3	3	1,73422E-15	-3,862782	-3,862782	-3,862782	0

TABLE 12. Results for unimodal test functions-9

	F19				F20			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	-3,321995	-3,20310	-3,250659	0,065120	-10,1532	-5,10077	-9,142714	2,259514
TLBO	-3,321995	-3,321995	-3,321995	6,7136E-08	-10,1532	-10,15319968	-10,1532	8,88178E-16
MNA	-3,321995	-3,20310	-3,226881	0,05317	-10,1532	-5,055198	-7,112628	2,775712
PSO	-3,321995	-3,20310	-3,250659	0,065120	-10,1532	-2,630472	-4,63956	3,260157
GWO	-3,321994	-3,197375	-3,272147	0,068252	-10,15306	-10,15193	-10,15265	0,000460
IGA	-3,321995	-3,321995	-3,321995	3,14018E-16	-10,1532	-10,1532	-10,1532	0

TABLE 13. Results for unimodal test functions-10

	F21				F22			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
gaTLBO_NNA	-10,40294	-5,128823	-9,348117	2,358657	-10,53641	-10,53641	-10,53641	2,17558E-15
TLBO	-10,40294	-10,40294	-10,40294	1,53837E-15	-10,53641	-10,53641	-10,53641	0
MNA	-10,40294	-5,128823	-9,348117	2,358657	-10,53641	-5,128481	-8,382671	2,949175
PSO	-10,40294	-2,765897	-6,2933	3,87361	-3,835427	-2,4273	-2,936530	0,528658
GWO	-10,40287	-10,40162	-10,40229	0,00045	-10,53578	-10,53537	-10,53563	0,000154
IGA	-10,40294	-10,40294	-10,40294	8,88178E-16	-10,53641	-10,53641	-10,53641	8,88178E-16

TABLE 14. Results for unimodal test functions11

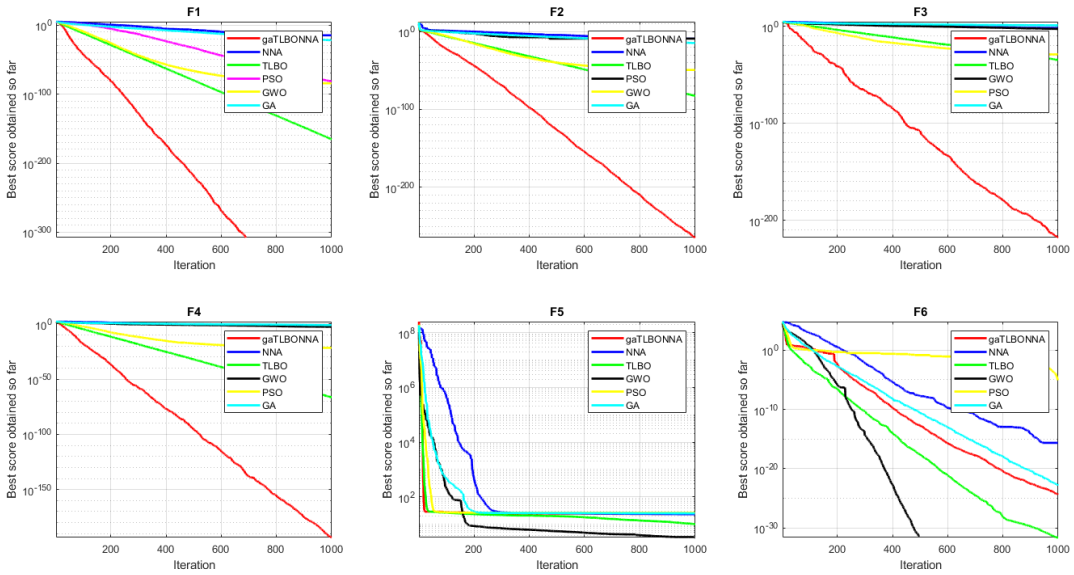

FIGURE 3. curves of convergence.

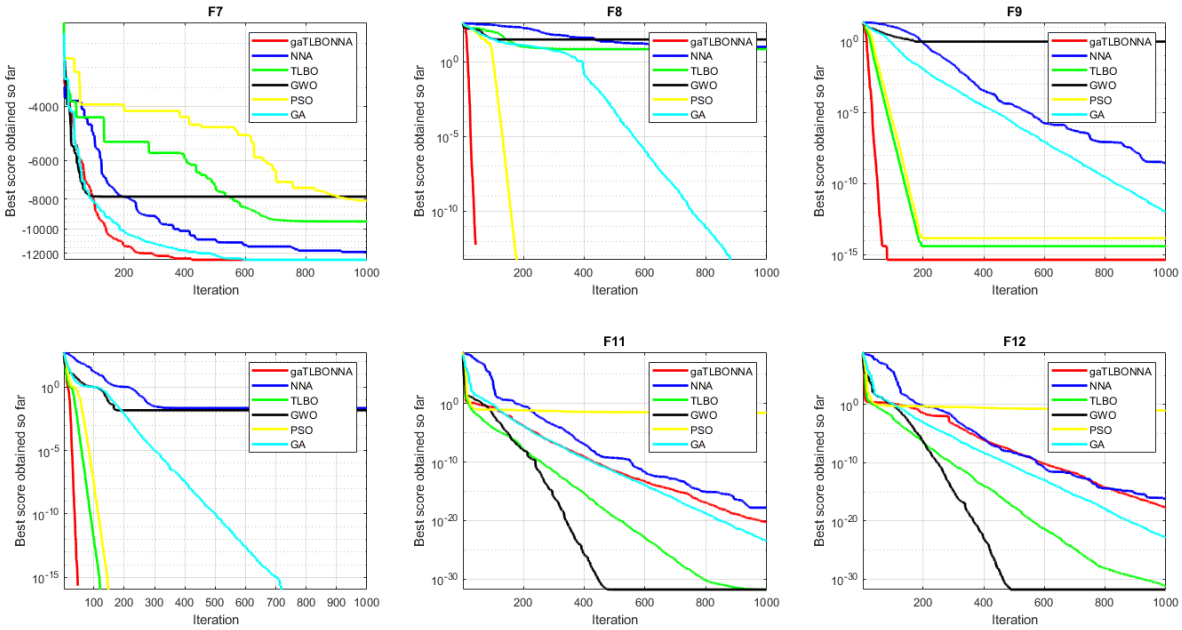

FIGURE 4. curves of convergence

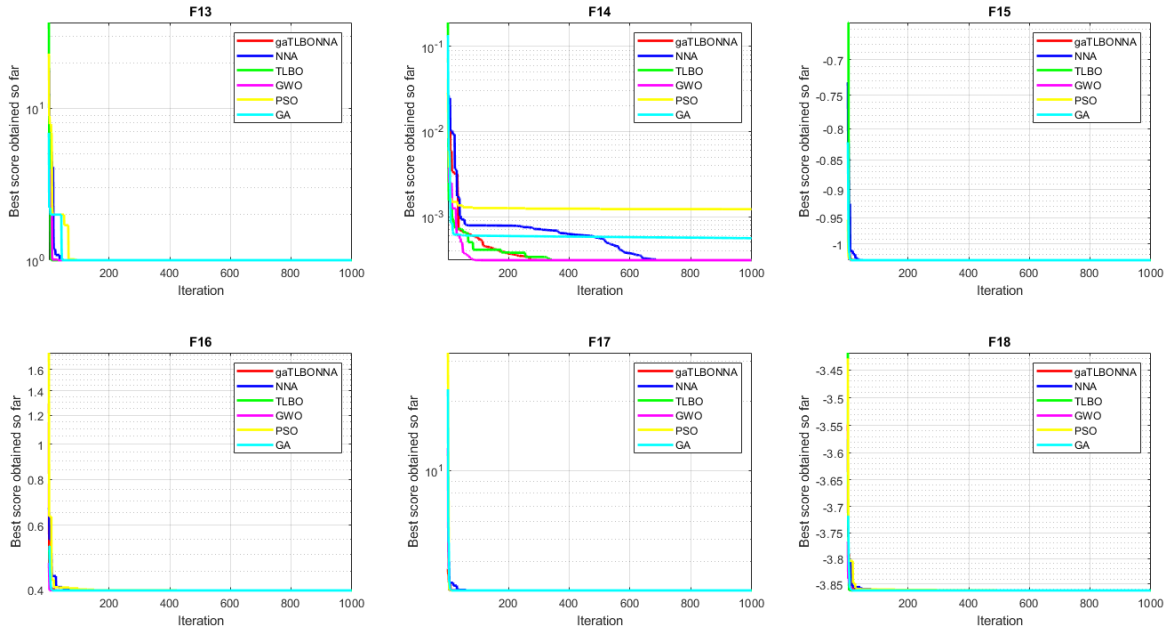

FIGURE 5. curves of convergence

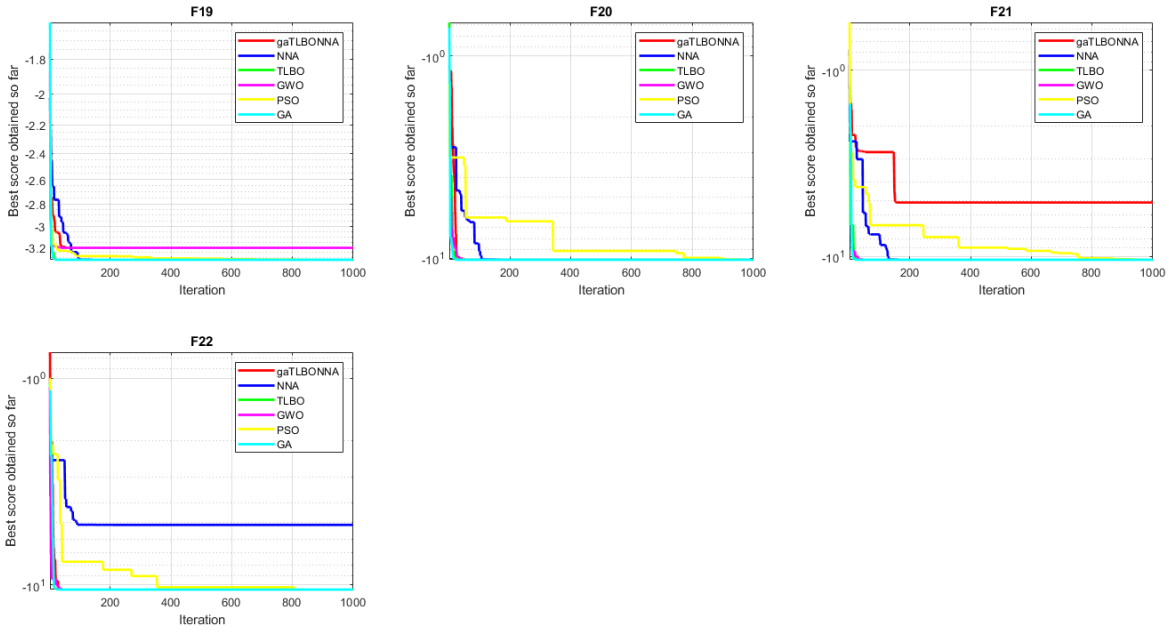

FIGURE 6. curves of convergence

4.2. **Application of the gaTLBO_NNA algorithm for estimating the parameters of a mathematical model in epidemiology.** In this section, we employ the gaTLBO_NNA algorithm to estimate the parameters of a mathematical model related to epidemiology. The used model was proposed by Hamza Alaa and Al. [11]. The authors developed a new Reservoir-Population (RP) transmission network model to simulate the potential spread of the Covid-19 virus within the Moroccan population, taking into account the different lockdown phases implemented by the government.

Four phases (4) have been considered:

- Day 1 (first case) to day 150: phase 1
- Day 151 to day 220: phase 2
- Day 221 to day 300: phase 3
- Day 301 to day 379: phase 4

The model stands out by incorporating phase-dependent parameters, allowing the calculation of the basic reproduction number $\mathcal{R}_0$ for each phase.

The authors used genetic algorithms to optimize the model's parameters, minimizing a cost function to fit the results to real-world data. Simulations demonstrated that total lockdown significantly reduced the virus's spread ($\mathcal{R}_0 < 1$).

4.2.1. *Mathematical model.* The following assumptions have been made:

(H1) The population is divided into five compartments: susceptibles S, exposed E, symptomatic infected I, asymptomatic infected A, and recovered R.

(H2) The reservoir W represents the source of infection in the seafood market where the virus is present. The virus in the reservoir is exported at a rate ϵW, where $\frac{1}{\varepsilon}$ represents the virus's lifespan in the environment, depending on the infected individuals exporting the virus into the market: symptomatic infected μI and asymptomatic $c\mu A$.

(H3) The entry rate of new individuals into the population is given by $\Lambda = n \times N$, where n is the birth rate and N is the total human population size. Natural mortality affects each compartment at a rate m.

(H4) Susceptible individuals S are infected through contact with the reservoir W and the symptomatic infected I, with respective transmission rates β_W and β. Asymptomatic individuals A can also transmit the virus, but with a reduced rate $\kappa\beta$, where $0 \leq \kappa \leq 1$.

(H5) The proportion of asymptomatic individuals is given by δ, meaning that a fraction δ of the exposed E will become asymptomatic, and the other fraction $(1-\delta)$ will become symptomatic.

(H6) The incubation and latency periods of human infection are defined as $\frac{1}{\rho}$, and the infectious period of the I and A compartments is defined as $\frac{1}{\gamma}$.

The compartmental diagram is given in the figure 7.

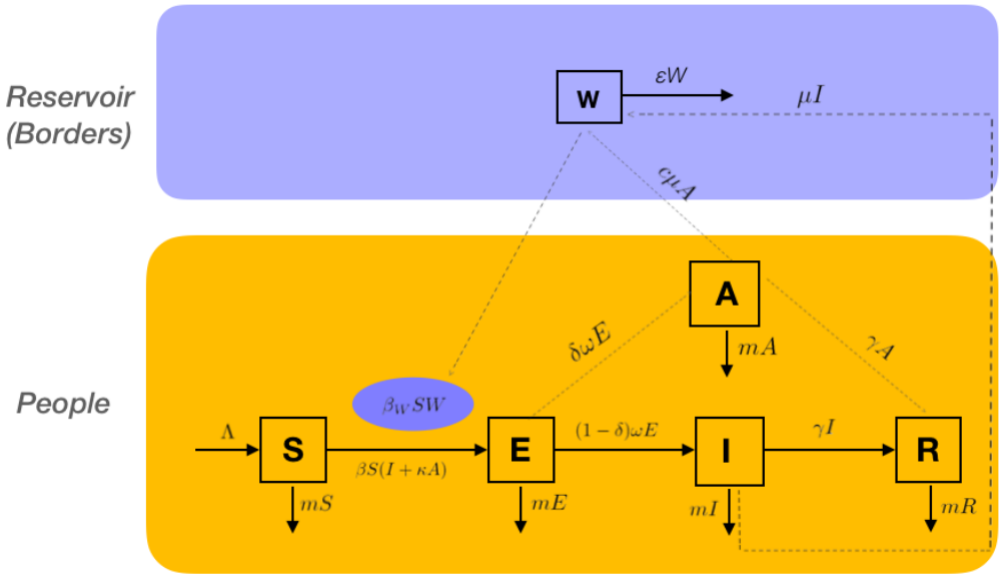

FIGURE 7. Compartmental diagram [11]

4.3. **Mathematical model.** By assuming:

$s = \frac{S}{N}, e = \frac{E}{N}, i = \frac{I}{N}, a = \frac{A}{N}, r = \frac{R}{N}, w = \frac{\epsilon W}{\mu N}, b = \beta N, b_w = \frac{\mu \beta W N}{\epsilon}$

The normalized model is given by [11]:

$$\begin{cases} \frac{ds}{dt} = n - ms - b(t)s(i + \kappa(t)a) - b_w(t)sw \\ \frac{de}{dt} = -(m+\rho)e + b(t)s(i + \kappa(t)a) + b_w(t)sw \\ \frac{di}{dt} = -(\gamma(t) + m)i + (1 - \delta(t))\rho e \\ \frac{da}{dt} = -(\gamma(t) + m)a + \delta(t)\rho e \\ \frac{dr}{dt} = -mr + \gamma(t)i + \gamma(t)a \\ \frac{dw}{dt} = -\epsilon w + \epsilon(i + c(t)a) \end{cases} \tag{38}$$

The model parameters $(b, c, b_w, \delta, \kappa, \gamma)$ are assumed to be variable but constant within each phase.

4.3.1. *The basic reproduction number $\mathcal{R}_0$.* The disease-free equilibrium (DFE) is: $\left(\frac{n}{m}, 0, 0, 0, 0, 0\right)$. In this model, the authors assumed that the parameters are constant within each phase $[T_i^l, T_f^l]$, where $l = 1, \ldots, 4$. After applying the next-generation matrix, the expression for the basic reproduction number $\mathcal{R}_0^l$ is given as follows [11]:

$$\begin{aligned} R_0^l = b^l \frac{n}{m} \frac{(1-\delta^l)\omega^l}{(\omega^l + m)(\gamma^l + m)} + \kappa^l b^l \frac{n}{m} \frac{\delta^l \omega^l}{(\omega^l + m)(\gamma^l + m)} + b_w^l \frac{n}{m} \frac{(1-\delta^l)\omega^l}{(\omega^l + m)(\gamma^l + m)} \\ + b_w^l \frac{n}{m} \frac{c^l \delta^l \omega^l}{(\omega^l + m)(\gamma^l + m)} \end{aligned} \tag{39}$$

4.4. **Parameter estimation problem.** Problem 38 can be formulated as follows:

$$\begin{cases} y'(t) = f(y, k) & t \in [t_i, t_f] \\ y_0 & given \end{cases} \tag{40}$$

where: $k = (b, c, b_w, \delta, \kappa, \gamma)$: the parameter vector

$$y = \begin{pmatrix} s \\ e \\ i \\ a \\ r \\ w \end{pmatrix} \quad f(y,k) = \begin{pmatrix} n - ms - b(t)s(i + \kappa(t)a) - b_w(t)sw \\ -(m+\rho)e + b(t)s(i + \kappa(t)a) + b_w(t)sw \\ -(\gamma(t) + m)i + (1 - \delta(t))\rho e \\ -(\gamma(t) + m)a + \delta(t)\rho e \\ -mr + \gamma(t)i + \gamma(t)a \\ -\epsilon w + \epsilon(i + c(t)a) \end{pmatrix}$$

During each phase l, $l = 1, \ldots, 4$, Subdivide $[T_i^l, T_f^l]$ into $M \in \mathbb{N}^+$ subdomains, ie $[T_i^l, T_f^l] = \bigcup_{j=0}^{M-1} [t_j^l, t_{j+1}^l]$ with $T_i^l = t_0^l < t_1^l < t_2^l < \ldots < t_{M-1}^l < t_M^l = T_f^l$. $t_j^l = jh$, with $h = \frac{T_f^l - T_i^l}{M}$ and

apply the 4th order Runge-Kutta scheme given

$$\begin{cases} y_0 = (s_0, e_0, i_0, a_0, r_0, w_0) \\ k_1 = hf(y_j, k) \\ k_2 = hf(y_j + \frac{k_1}{2}, k) \\ k_3 = hf(y_j + \frac{k_2}{2}, k) \\ k_4 = hf(y_j + k_3, k) \\ y_{j+1} = y_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \tag{41}$$

to solve model 38 for each element $k = (b, c, b_w, \delta, \kappa, \gamma)$ of the population, to obtain the unique solution (s, e, i, a, r, w). This solution is associated with the cost function F_l, called fitness.

$$F_l(k) = \int_{T_i^l}^{T_f^l} \left((i(t) - i_{\text{obs}}(t))^2 + (a(t) - a_{\text{obs}}(t))^2 \right) dt \tag{42}$$

$y_0^1 = (s_0, e_0, i_0, a_0, r_0, w_0)$ the initial condition at phase 1. The initial condition $y_0^l = y_M^{l-1}$, $l = 2, 3, 4$. The objective is to determine the global minimun $k_l^* = (b^*, c^*, b_w^*, \delta^*, \kappa^*, \gamma^*)$ of the functional F_l for each phase.

4.4.1. *Results of idenfication.* The results obtained by our method are summarized in the table 15. Table 16 summarizes the results obtained by the genetic algorithm [11].

	phase 1	Phase 2	Phase 3	Phase 4
κ	0,032391063	0,4253588	0,5675845	0,5
c	0,017869446	0,036946999	0,798936826	0,010101335
b	0,010035221	0,30819454	0,058620818	0,788555126
b_w	0,8	0,796277997	0,8	0,779667433
δ	0,4	0,5	0,43806583	0,6
γ	0,176865399	0,375315963	0,177006875	0,5
Values of objective function	0,000123	0,000552	0,00324	0,00038

TABLE 15. Parameters estimation

	phase 1	Phase 2	Phase 3	Phase 4
κ	0.0554	0.5249	0.7318	0.7002
c	0.2381	0.6898	0.6484	0.3930
b	0.6604	0.2807	0.2295	0.5117
b_w	0.7552	0.6368	0.7218	0.7065
δ	0.4312	0.3519	0.2124	0.5066
γ	0.2699	0.3150	0.1880	0.5575
Values of objective function	$0,0385$	$0,0386$	$0,0179$	$0,00067$

TABLE 16. Parameters estimation [11]

In Figure 8, we present the curves of new COVID-19 cases recorded in Morocco, along with the results obtained from the gaTLBONNA algorithm and GA algorithm [11].
This figure illustrates that gaTLBO_NNA algorithm can reproduce the evolution of new COVID-19 cases observed during the study period. However, the gaTLBO_NNA algorithm provides a better reproduction, as indicated by the values of the objective functions presented in the tables 15 and 16.

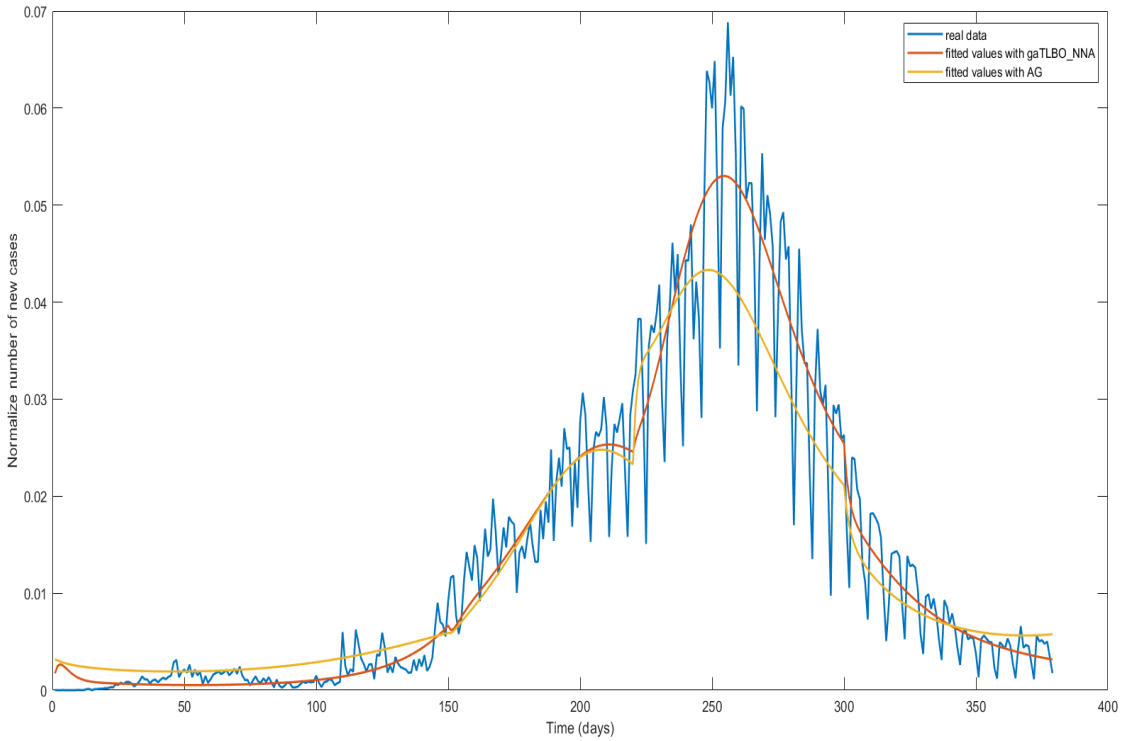

FIGURE 8. Best fitted curve

5. Conclusion

In this study, we developed an algorithm for the minimization of real functions, called gaTLBONNA. This algorithm is a hybrid of the Improved real-coded genetic algorithm (IRGA), Teaching–learning-based optimization (TLBO) algorithm, and Neural Network Algorithm (NNA). Initially, we tested our algorithm on benchmark functions and compared it to several well-established metaheuristics. The results, in terms of convergence and accuracy, indicate that gaTLBONNA outperforms the compared methods for the majority of the tested functions. Subsequently, the algorithm was applied to identify parameters of a mathematical model for the spread of COVID-19, using real data from Morocco. This demonstrated the effectiveness of our algorithm in estimating parameters of mathematical models in epidemiology.

Authors' Contributions. All authors have read and approved the final version of the manuscript. The authors contributed equally to this work.

Conflicts of Interest. The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Adv. Eng. Softw. 69 (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

[2] A. Sadollah, H. Sayyaadi, A. Yadav, A Dynamic Metaheuristic Optimization Model Inspired by Biological Nervous Systems: Neural Network Algorithm, Appl. Soft Comput. 71 (2018), 747–782. https://doi.org/10.1016/j.asoc.2018.07.039.

[3] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, Perth, WA, Australia, pp. 1942–1948. https://doi.org/10.1109/icnn.1995.488968.

[4] A. Sadollah, H. Sayyaadi, A. Yadav, A Dynamic Metaheuristic Optimization Model Inspired by Biological Nervous Systems: Neural Network Algorithm, Appl. Soft Comput. 71 (2018), 747–782. https://doi.org/10.1016/j.asoc.2018.07.039.

[5] X. Xiong, S. Li, F. Wu, An Enhanced Neural Network Algorithm with Quasi-Oppositional-Based and Chaotic Sine-Cosine Learning Strategies, Entropy 25 (2023), 1255. https://doi.org/10.3390/e25091255.

[6] H. Fahim, O. Sawadogo, N. Alaa, M. Guedda, An Efficient Identification of Red Blood Cell Equilibrium Shape Using Neural Networks, Eurasian J. Math. Comput. Appl. 9 (2021), 39–56. https://doi.org/10.32523/2306-6172-2021-9-2-39-56.

[7] A.K. Das, D.K. Pratihar, Solving Engineering Optimization Problems Using an Improved Real-Coded Genetic Algorithm (irga) with Directional Mutation and Crossover, Soft Comput. 25 (2021), 5455–5481. https://doi.org/10.1007/s00500-020-05545-9.

[8] W.O. Sawadogo, P.O.F. Ouédraogo, K. Somé, N. Alaa, B. Somé, Modified Hybrid Grey Wolf Optimizer and Genetic Algorithm (HmGWOGA) for Global Optimization of Positive Functions, Adv. Differ. Equ. Control. Process. 20 (2019), 187–206. https://doi.org/10.17654/de020020187.

[9] K. Divyesh R, A. Kulshrestha, Minimizing the Detection Error in Cooperative Spectrum Sensing using Teaching Learning Based Optimization(TLBO), Int. J. Eng. Res. Technol. 6 (2017), 495–500.

[10] R. Rao, V. Savsani, D. Vakharia, Teaching–learning-based Optimization: a Novel Method for Constrained Mechanical Design Optimization Problems, Comput.-Aided Des. 43 (2011), 303–315. https://doi.org/10.1016/j.cad.2010.12.015.

[11] H. Alaa, E.A. Alaa, F. Aqel, Development and Simulation of a Mathematical Model to Simulate the Phase Transmissibility Ofcovid19 in Morocco, Ann. Univ. Craiova Math. Comput. Sci. Ser. 49 (2022), 75–83. https://doi.org/10.52846/ami.v49i1.1489.

[12] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.