Computers, Materials & Continua
DOI: 10.32604/cmc.2022.024211
Article

Tech Science Press

Convergence Track Based Adaptive Differential Evolution Algorithm (CTbADE)

Qamar Abbas[1], Khalid Mahmood Malik[2], Abdul Khader Jilani Saudagar[3,*], Muhammad Badruddin Khan[3], Mozaherul Hoque Abul Hasanat[3], Abdullah AlTameem[3] and Mohammed AlKhathami[3]

[1]Department of Computer Science and Software Engineering, International Islamic University, Islamabad, 44000, Pakistan
[2]Department of Computer Science and Engineering, Oakland University, Rochester, MI, USA
[3]Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia
*Corresponding Author: Abdul Khader Jilani Saudagar. Email: aksaudagar@imamu.edu.sa
Received: 09 October 2021; Accepted: 13 December 2021

Abstract: One of the challenging problems with evolutionary computing algorithms is to maintain the balance between exploration and exploitation capability in order to search global optima. A novel convergence track based adaptive differential evolution (CTbADE) algorithm is presented in this research paper. The crossover rate and mutation probability parameters in a differential evolution algorithm have a significant role in searching global optima. A more diverse population improves the global searching capability and helps to escape from the local optima problem. Tracking the convergence path over time helps enhance the searching speed of a differential evolution algorithm for varying problems. An adaptive powerful parameter-controlled sequences utilized learning period-based memory and following convergence track over time are introduced in this paper. The proposed algorithm will be helpful in maintaining the equilibrium between an algorithm's exploration and exploitation capability. A comprehensive test suite of standard benchmark problems with different natures, i.e., unimodal/multimodal and separable/non-separable, was used to test the convergence power of the proposed CTbADE algorithm. Experimental results show the significant performance of the CTbADE algorithm in terms of average fitness, solution quality, and convergence speed when compared with standard differential evolution algorithms and a few other commonly used state-of-the-art algorithms, such as jDE, CoDE, and EPSDE algorithms. This algorithm will prove to be a significant addition to the literature in order to solve real time problems and to optimize computational models with a high number of parameters to adjust during the problem-solving process.

Keywords: Differential evolution; function optimization; convergence track; parameter sequence; adaptive control parameters

1 Introduction

Differential evolution (DE) is a stochastic algorithm introduced by Storn and Price [1]. Differential evolution is a very powerful heuristic global search evolutionary algorithm for the solution of real parameter optimization. The main benefit of a DE algorithm as compared to other evolutionary algorithms is that it has fewer parameters, it is fast, simple and has a greater chance of finding the optimal value for optimization problems [2,3]. To evolve a current population, a DE algorithm uses crossover, selection, and mutation operators. A number of real world optimization problems in various fields have been successfully solved by DE algorithms, such as image processing [4], microwave engineering [5], signal processing [6], chemical engineering [7], artificial neural networks [8], bioinformatics [9], power systems [10], pattern recognition [11], robotics [12], convolutional neural network training [13], scheduling [14], communication [15].

In a DE algorithm, all population members have the same chance of selection to be a parent but one or more amplified weighted different vector is added in the third vector to generate a new vector [16]. DE algorithms have crossover, mutation, and selection operators in order to evolve the current population to locate optima within the given search space. The setting of control parameters and mutation and crossover strategies has a large impact on the performance of a DE algorithm [17]. The Crossover rate (CR) and mutation probability (F) are two important control parameters of any DE algorithm [3]. The CR parameter contributes to the crossover operator and F contributes to the mutation operator. A greater value of control parameter F focuses on the exploration and a smaller value of F focuses on the exploitation capability [18,19]. The CR control parameter contributes to population diversity; greater diversity increases the algorithm's exploration power and a smaller value reduces population diversity and focuses on the exploitation capability [18,19].

The deadly Covid-19 pandemic has badly affected the whole world in a very short span of time. One of the major issues with Covid-19 is the time taken for the diagnostic process [20]. Computerized Tomography (CT) chest images and X-rays are widely used to diagnose and measure the severity of Covid-19 infection [21]. The automated diagnosis and measurement of Covid-19 infection is one of the most challenging tasks at the moment [22]. The main challenge in this automated diagnostic process is to detect Covid-19 infection more quickly and accurately [23]. A number of researchers have used convolutional neural networks (CNN) which are considered to be very effective for the classification of Covid-19 images into positive and negative classes [24]. But the issue with a CNN is the slow convergence speed to train and test the image data in order to detect Covid-19 infection [25]. The convergence speed of CNNs can be improved by optimizing the hyper parameters [26]. There is an urgent need to develop a powerful optimization algorithm with a quick convergence speed. The main contribution of this research work is to present a novel convergence track based adaptive differential evolution (CTbADE) algorithm. The balance between exploration and exploitation in CTbADE is helpful to avoid any possibility of stagnation, premature convergence, or slow convergence speed. This paper is organized in multiple sections: related work is presented in Section 2 and the proposed methodology is described in Section 3. Test functions and parameter study are covered in Section 4; results and discussions are reported in Section 5; conclusion and references are given at the end of this paper.

2 Related Work

This sections contains details of the original DE algorithm and a literature review.

2.1 Original DE Algorithm

A DE algorithm is a population based stochastic algorithm used to evolve a population of individuals over time by applying various operators, such as the selection operator, crossover operator and mutation operator. The members of a population are initialized by means of small random values from a search space of n-dimensions. This paper uses D-Dimensional minimizations optimization problems without loss of generality:

$$MINIMUM\, f\,(m_1,\, m_2,\, \ldots\ldots\ldots,m_{DIM})\ such\ that\ m_{i,j} \in [m_k^{\min}, m_k^{\max}]\ where\ k = 1, 2, 3, \ldots .DIM \tag{1}$$

where the minimum of minimization, (m1, m2, …..,M_{DIM}) represents DIM dimension members of populations and $[m_k^{\min}, m_k^{\max}]$ is the range of the search space for a given problem.

The details of DE algorithm operators are as follows

2.1.1 Mutation Operator

This operator is used to generate a donor vector, sometimes called a mutant vector, which is generated by scaling the difference vector of different members of population. A number of mutation strategies are available in the literature, the most commonly used mutation strategy has the following equation:

$$V_k^t = M_{r_1}^t + F(M_{r_2}^t - M_{r_3}^t) \tag{2}$$

where V_k^t represents the mutation vector at t^{th} iteration, $M_{r_1}^t$, $M_{r_2}^t$, $M_{r_3}^t$ are three different valid population members selected from the range $[M_1, M_2, \ldots\ldots, M_{POP\_SIZE}]$ and F controls the amplifications of difference vector $M_{r_2}^t - M_{r_3}^t$.

2.1.2 Crossover Operator

In this operation, a trial vector is generated by utilizing the donor vector V_k^t and the target vector M_k. The important crossover operators are binomial and exponential crossover; other types of crossover operator are rarely used. The following equation of crossover operators generates donor vector U_k

$$U_{ki,Gn+1} = \begin{cases} V_{ki,Gn+1} & if\ (rand[0,1] \le CR\ or\ k = rnbr[1, DIM]) \\ M_{k,Gn} & if\ (rand[0,1] > CR\ and\ k \ne rnbr[1, DIM]) \end{cases} \tag{3}$$

The term rand [0, 1] is used to generate random numbers of a uniform nature from the range [0–1], rnbr will generate the index of a random vector in the range [1, *DIM*], $M_{k,Gn}$ represents the k^{th} target vector for Gn^{th} generation, $V_{k,Gn+1}$ represents the k^{th} donor vector for $G_{n+1}{}^{th}$ generation

2.1.3 Selection Operator

After mutation and crossover operations, a trial vector is evaluated by using the fitness function according to the optimization problem. It is basically a greedy method based on the concept of survival of the fittest that will select the best vector of the target vector and trial vector, based on their fitness value. The equation of the selection operator is

$$M_{k,G+1} = \begin{cases} U_{k,Gn+1} & if\ (fitness(U_{k,Gn+1}) < fitness(M_{k,Gn})) \\ M_{k,Gn} & otherwise \end{cases} \tag{4}$$

where *fitness* is a function to evaluate the performance of an individual, $M_{k,G+1}$ is an individual of next generation against $M_{k,G}{}^{\text{th}}$ target vector, $U_{k,Gn+1}$ is the trial vector and fitness () is basically an objective function to evaluate population members for given problem.

2.2 Literature Survey

The performance of DE algorithms is affected by the selection of mutation variant, crossover variant and the control parameters of the CR, mutation probability, and population size. A large number of studies have been carried out by researchers on various aspects of DE algorithms, such as improvements in control parameter (such as deterministic, adaptive, self-adaptive, fuzzy-based, etc.), enhancement of mutation variants, crossover strategies, population-based variations, and hybridization with other algorithms. Reference [27] introduced a parameter adaption-based population diversity mechanism to DE algorithms. This research explored whether population diversity is helpful in ensuring equilibrium between the exploitation and the exploration behavior of DE algorithms. DE algorithms' convergence is proved by considering a few standard benchmark functions. The concept of fuzzy adaptive control parameters CR and F based DE by using fuzzy logic [28]. In this work, fuzzy adaptive DE was tested by considering a test suite of 10 benchmark functions. A self-adaptive control parameter based DE is presented in [29]. The F and CR control parameters are changed by a random controlled mathematical relationship, where the range of F is from F = 0.1 to F = 0.9. Reference [30] discusses the idea of adaptive external archive memory in DE algorithms, which helps improve the performance of DE algorithms for standard benchmark functions. The arithmetic mean is used to make the CR control parameter adaptive and the Lehmer mean to make the F control parameter adaptive.

A memory-based self-adaptive control parameter with a strategy adaption concept is introduced in [31]. These researchers use the concept of a learning period (LP) to memorize the control parameters F and CR and then probabilistically adapt the value of control parameters during the evolutionary process. The idea of small neighborhood of each population, using an improved family of variant "DE/target-to-best/1/bin," is introduced by [32]. This version improves the power of DE algorithms by balancing exploration and exploitation, and validating their improved algorithm for two real world problems of spread spectrum radar poly-phase code design and frequency modulated sound waves. Reference [33] used the concept of a pool of control parameters and mutation strategies pool in their research work. This method uses F = {0.5, 0.9}, CR = {0.1, 0.5, 0.9} and two different mutation strategies over a test suite of standard benchmark functions. The concept of creating a composite trial vector using a mapping pool of a combination of control parameters and mutation strategies is introduced by [34]. This technique used a pool of "rand/1/bin," "rand/2/bin," and "current-to-rand/1" mutation variants mapping group and [F = 1.0, Cr = 0.9], [F = 1.0, Cr = 0.1], and [F = 0.8, Cr = 0.2] as a control parameter mapping pool. The ensemble-based DE is implemented using a test suite of standard benchmark problems and the results are measured against several state-of-the-art algorithms, showing the significant performance of this algorithm.

Reference [35] introduced novel mutation and crossover strategies in their adaptive version of a DE algorithm by utilizing the p top-ranked individuals from the current population. The authors introduced the "DE/current-to-best/1" mutation strategy and a Cauchy distribution-based random controlled mutation probability and a Gaussian distribution-based random controlled CR control parameter in their work. The concept of a generalized opposition-based self-adaptive parallel DE (GOjDE) algorithm based on graphics processing units to improve the solution quality is introduced in [36]. The authors used an optimization problem of high dimensions and compared it with six other DE

algorithms, showing that the GOjDE performed better or competitively. Random controlled adaptive values of control parameters are used in GOjDE.

A mechanism for adapting strategy parameters and populations in DE based on multivariate probabilistic self-adaptive control parameters was presented in [37]. This strategy uses mean vector m and covariance matrix C where m represents the mean values of CR and F, and C is interdependent between the two parameters. The population adaption from a huge set of individuals helps to balance between exploration and exploitation. The concept of a self-adaptive DE algorithm by incorporating adaptive mutation strategies and zoning-based evolution of control parameters (ZEPDE) [38]. ZEPDE's performance is compared with five other well-known algorithms: jDE, JADE, SaDE, EPSDE, and CoDE by applying them on a test suite of benchmark functions, which shows the significance performance of ZEPDE algorithms. Reference [39] introduced self-adaptive control parameters and self-adaptive strategies based on a symmetric Latin hypercube design to initialize the population in a DE algorithm. This population initialization is helpful to increase diversity. To create a trial vector against the given vector, a mutation strategy from a successful experience-based pool of mutation strategies and control parameters F and CR are updated by normal distribution and Cauchy distribution. A hyper heuristic based self-adaptive DE is introduced by [40] for mixed integer non-linear programming problems in their research work. This algorithm used e-constrained for handling constraint and self-adaptive parameters, a number of crossover and mutation strategies, and normal distribution-based control parameters F and CR. Knowledge based control parameters as well as strategy adaption mechanisms in DE supervise and guide evolution by employing opposition-based learning is introduced by [41]. The concept of a self-adaptive based dual strategy to balance the exploration and exploitation by employing "DE/best/2" and "DE/rand/2" based on self-adaptive scaling parameter is presented in [42]. This algorithm also introduced a dynamic exploration ability control parameter to dynamically adjust the global exploration ability of DE algorithms. Self-adapting parameters and multi mutation variants based enhanced DE algorithms are proposed in [43]. This research measured the convergence speed and accuracy, using three different mutation strategies and control parameters to adapt the dynamic rate of exploitation and exploration. To optimize the structure and parameters of neural networks, a collective intelligent based DE algorithm is presented in [44]. In this work mutation parameters based on collective intelligence were used to enhance the exploitation and the exploration capability of DE algorithms by utilizing promising donor vectors by taking m top ranked donor vectors. They used the weight of m top ranked vectors which is calculated by the linear combination of two random vectors and m top ranked vectors.

3 Convergence Track Based Adaptive Differential Evolution Algorithm (CTbADE)

Mutation strategies and control parameters have a major impact on DE algorithms' performance, which can be enhanced by following a convergence track during the evolutionary process. A smaller value of mutation probability F focuses on exploitation, and a bigger value of F focuses on the exploration capability of a DE algorithm [18]. CR is helpful in maintaining diversity in the population [18]; in early iterations a bigger value of CR is used [e.g., $CR = 0.9$], then it is gradually decreased by using a controlled sequence to a smaller value [$CR = 0.7$]. The reason for decreasing the CR is that in successive iterations diversity should be not much higher, otherwise it will slow down the performance of the algorithm. The concept of memory is used in the controlled sequence and two different control sequences of CR are used by default: one is linear decreasing and the other is linear increasing. The concept of memory is used to calculate the change in the average fitness value, if the fitness value is not changed to learning period (LP), successive iterations then sequence will move to the opposite direction, that is from decreasing to increasing or from increasing to decreasing, according to the

following equations. It is important to mention here that if, for a specified LP, the fitness value of the objective function does not change, then there is a possible diversity issue or exploration issue or local optima issue which means that the values of the parameters need to be changed.

$$\mathrm{CR}_{\_\mathrm{cur}} = \mathrm{CR\_cur} - \left(\frac{\mathrm{CR\_cur} - \mathrm{CR\_min}}{\mathrm{Tot\_iter}}\right) \times \mathrm{j} \tag{5}$$

$$\mathrm{CR}_{\_\mathrm{cur}} = \mathrm{CR\_cur} + \left(\frac{\mathrm{CR\_max} - \mathrm{CR\_cur}}{\mathrm{Tot\_iter}}\right) \times \mathrm{j} \tag{6}$$

where *CR_cur* is current value of crossover rate, *CR_min* is the minimum value and *CR_max* is the maximum value of crossover rate, *Tot_iter* is total number of iterations and j represents the current iteration.

The value of F is increased so that, from a very diverse population, we should start by focusing on the exploration so that we should move in the direction of convergence and its values are used in the increasing sequence. This algorithm will be able to escape from local optima in a controlled manner and in a controlled range which is described in the following range. By default, the sequence of control parameter F is increased using Eq. (4) and then it can be seen whether the fitness value of the problem is changing during the specified LP and the sequence is changed from increasing to decreasing and vice versa accordingly.

$$\mathrm{F}_{\_\mathrm{cur}} = \mathrm{F\_cur} - \left(\frac{\mathrm{F\_cur} - \mathrm{F\_min}}{\mathrm{Tot\_iter}}\right) \times \mathrm{j} \tag{7}$$

$$\mathrm{F}_{\_\mathrm{cur}} = \mathrm{F\_cur} + \left(\frac{\mathrm{F\_max} - \mathrm{F\_cur}}{\mathrm{Tot\_iter}}\right) \times \mathrm{j} \tag{8}$$

where *F_cur* is the current value of mutation probability, *F_min* is the minimum value and *F_max* is the maximum value of mutation probability, *Tot_iter* is total number of iterations and j represents the current iteration.

The mutation strategy *DE/best/1* is considered a better performing variant among other variants of DE algorithm, which is directed towards the global best value; since we are using the concept of memory to follow the convergence track, this technique will prove to be beneficial in maintaining the improved convergence of DE algorithms. The pseudocode of proposed CTbADE is given in Fig. 1.

0. Initialize the control parameter for the range of time/iteration based controlled sequence:

$[\mathrm{F\_min} = \mathrm{MIN}_1 = 0.5, \mathrm{F\_max} = \mathrm{MAX}_1 = 0.7], [\mathrm{CR\_min} = \mathrm{MIN}_2 = 0.7, \mathrm{CR\_max} = \mathrm{MAX}_2 = 0.9]$

1. In the first step, generate $\mathrm{POP_{Gn}} = \{\mathrm{M_{1,Gn}}, \ldots\ldots \mathrm{M_{Pop\_Size,Gn}}\}$ as the initial population for the first generation $\mathrm{Gn} = 0$, initialize each population member $\mathrm{M_{k,Gn}} = \{\mathrm{M}^1_{k,Gn} \ldots\ldots\ldots \mathrm{M}^{DIM}_{k,G}\}$ using a random number where $i = 1 \ldots\ldots\ldots\ldots$ POP_SIZE [Population Size].
2. Initialize mutation probability F and CR for all individuals of population $\mathrm{M_{k,Gn}} = \{M^1_{k,Gn} \ldots\ldots\ldots \mathrm{M}^{DIM}_{k,Gn}\}$ where $i = 1 \ldots\ldots\ldots\ldots$ POP_SIZE using

$$\mathrm{F} = \mathrm{F\_min} \quad ; \quad \mathrm{CR} = \mathrm{CR\_max} \tag{9}$$

3. FOR $k = 1$ to POP_SIZE

Calculate fitness $fit(M_{k,Gn})$ for each population member $\mathrm{M_{k,Gn}}$ using initial values of mutation probability and crossover rate parameters according to Step 2.

END FOR

4. Initialize the starting value of mutation probability F_cur with an initial value of the mutation probability sequence and the current value of crossover probability CR_cur with an initial value of CR rate using following Eq. (2)

$$\text{F\_cur} = \text{F\_min} \quad ; \quad \text{CR\_cur} = \text{CR\_max} \tag{10}$$

5. WHILE j <= Total_Iter

Step 5.1 Mutation strategy Parent Vectors selection

Select three individual parent individuals that will be used in the following mutation strategy from the current population

$$v_g^i = x_g^{best} + F(x_g^{r1} - x_g^{r2}) \tag{11}$$

Step 5.2 Mutation Operation

FOR $k = 1$ to POP_SIZE

To get a donor vector $V_{k,Gn} = \{v_{k,Gn}^1 \ldots\ldots v_{k,Gn}^{DIM}\}$ against kth given vector $M_{k,Gn}$ use Eq. (3) and jth Mutation Probability $F_{\_cur}$ using equation

(5) or (6) based on the LP convergence track decision

END FOR

Step 5.3 Crossover Operation

FOR $k = 1$ up to POP_SIZE

To get trial vector $U_{k,Gn} = \{u_{k,Gn}^1 \ldots\ldots u_{k,Gn}^{DIM}\}$ against kth given vector $M_{k,Gn}$ use crossover control parameter $CR_{\_cur}$ and Eq. (7) or (8), based on the LP decision.

END FOR

Step 5.4 Selection Step

FOR $k = 1$ up to POP_SIZE

Use fitness function *fit* in order to evaluate vector $U_{k,Gn}$ for vector $M_{k,Gn}$

IF $fit(U_{k,Gn} \leq fit(M_{k,Gn})$, THEN $M_{k,Gn+1} = U_{k,Gn}, fit(M_{k,Gn}) = fit(U_{k,Gn})$

IF $fit(U_{k,Gn} \leq fit(M_{best,Gn})$, THEN $M_{best,Gn+1} = U_{k,Gn}, fit(M_{best,Gn}) = fit(U_{k,Gn})$

END IF

ELSE

$X_{k,Gn+1=X,Gn}$

END IF

END FOR

6. Update the LP based tracking for adaption of control parameters.

Step 6.1 Update Learning Period Convergence Memory

Step 6.2 Calculate Average Convergence

Step 6.3 IF (Average convergence > threshold/zero)

Continue with the same sequence increasing/decreasing control parameters F and CR using Eqs. (4)–(7)

ELSE

Reverse the sequence of CR and F control parameters (increasing to decreasing and decreasing to increasing) using Eqs. (4)–(7)

END IF

7. Increase the generation number $Gn = Gn + 1$
8. END WHILE

Figure 1: Convergence Track Based Adaptive Differential Evolution (CTbADE) algorithm pseudocode

4 Test Functions and Parameter Study

Comprehensive standard problems to test the convergence power of the CTbADE algorithm are taken from [45]; varying natures, such as unimodal, multimodal, separable, and non-separable, are considered in this research work. The details of the nature of the benchmark functions, names of the benchmark functions, optimal values, and search space are given in Tabs. A and B in the appendix. The population size is considered to be fixed for all used dimensions D with values of 10, 20 and 30, see reference [45] for all standard benchmark functions. The average of 30 trials is used to calculate the average fitness performance in the paper for all functions. The range of values of mutation probability [19] used is [0.5, 0.7] and CR [19] is [0.9, 0.5], controlled by the proposed sequences discussed in the proposed algorithm section.

5 Results and Discussion

This section contains the average fitness performance of the original DE and a few more commonly used state-of-the-art algorithms, such as CoDE, EPSDE, jDE, and the proposed CTbADE. These experimental results are generated by implementing the algorithm in C++ using Core-i3 machine 4 GB RAM, 4 GHZ CPU, Windows 10 and by using the performance parameters given in Section 4 of this paper. The benchmark functions of varying nature are used to test the convergence power of the proposed CTbADE algorithm. The experimental results of average fitness values are reported in Tabs. 1–3 for 10 Dimensions, 20 Dimensions, 30 Dimensions and the population size is taken as 50 for all benchmark functions. The training iterations for 10D, 20D, and 30D are used as 5000, 10000, and 15000 respectively. In order to assure the consistency of the proposed algorithm, results are generated over 30 independent runs.

Table 1: Comparison of proposed CTbADE with some key DE algorithm [L = Lo, H = Hg, M = Md, Y = Ys, N = No]

Factor	Control parameter																		
	ADE [27]	FADE [28]	jDE [29]	JADE [30]	SaDE [31]	NSDE [32]	EPSDE [33]	CoDE [34]	MDE-Pbx [35]	GOjDE [36]	iJADE [37]	ZEPDE [38]	SLADE [39]	SADEHH [40]	SAKPDE [41]	SaDSDE [42]	MSaDE [43]	SLCIDE [44]	Proposed CTbADE
Diversity	Lo	Md	Lo	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg	Hg
Local optima handling ability	Lo	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Hg	Hg	Md	Md	Hg
Convergence speed focus	Lo	Md	Md	Md	Md	Md	Hg	Hg	Md	Md	Md	Md	Md	Hg	Hg	Hg	Hg	Hg	Hg
Impact of dimensionality	Hg	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Md	Lo	Lo	Md	Lo	Lo
Deterministic	Ys	Ys	Ys	No	No	Ys	Ys	Ys	Ys	No	No	No	No	No	No	No	No	No	No
Self-adaptive	No	No	No	No	Md	No	No	No	No	No	Md	No	Md	Md	Hg	Hg	No	No	Hg
Parameter memorization	No	No	No	No	Ys	No	No	No	No	No	No	No	No	No	Ys	Ys	No	No	No
Convergence tracking over time	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Ys
Time tracking controlled sequencing	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Ys
Time tracking adaptive	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Ys

Table 2: 10D results of average fitness values and standard deviation of benchmark functions – mean (Stdv)

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_1	6.02E − 217 (0)	7.63E − 230 (0)	1.64E − 227 (0)	2.17E − 223 (0)	**0 (0)**
f_2	5.18E − 239 (0)	5.49E − 257 (0)	1.37E − 245 (0)	9.12E − 241 (0)	**0 (0)**
f_3	2.82E − 159 (2.30E − 316)	1.00E − 46 (2.56E − 46)	3.58E − 80 (6.14E − 80)	1.22E − 81 (6.14E − 81)	**4.91E − 241 (0)**
f_4	2.85E + 00 (1.82E + 00)	**0 (0)**	**0 (0)**	1.15E − 01 (8.70E − 02)	3.99E − 01 (1.20E + 00)
f_5	9.15E − 01 (8.70E − 01)	**0 (0)**	**0 (0)**	0 (0)	1.85E + 01 (8.78E + 00)
f_6	2.18E − 02 (1.85E − 02)	**0 (0)**	**0 (0)**	9.04E − 04 (2.81E − 03)	2.91E − 01 (3.12E − 01)
f_7	1.45E − 51 (7.46E − 51)	0 (0)	5.41E − 291 (0)	**0 (0)**	**0 (0)**
f_8	8.85E − 02 (8.32E − 02)	**4.29E − 13 (3.96E − 13)**	1.38E − 12 (1.94E − 12)	4.18E − 03 (2.25E − 02)	9.09E − 01 (3.26E − 01)
f_9	5.25E − 07 (6.70E − 07)	**2.10E − 08 (1.93E − 08)**	1.06E − 07 (1.15E − 07)	2.37E − 08 (3.83E − 08)	2.91E − 02 (9.58E − 02)
f_{10}	9.87E − 160 (5.31E − 318)	2.92E − 64 (4.44E − 64)	3.95E − 96 (7.57E − 96)	1.73E − 98 (5.00E − 98)	**1.59E − 248 (0)**
f_{11}	1.63E − 107 (3.40E − 107)	6.81E − 129 (5.85E − 129)	2.74E − 115 (3.57E − 115)	1.26E − 124 (4.19E − 124)	**2.52E − 181 (0)**
f_{12}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.12E + 01 (1.35E + 01)
f_{13}	6.73E − 88 (3.62E − 87)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{14}	1.05E − 04 (1.25E − 04)	8.98E − 06 (1.04E − 05)	2.00E − 05 (1.99E − 05)	1.19E − 05 (1.38E − 05)	**3.69E − 06 (1.99E − 05)**
f_{15}	**3.27E − 31 (1.31E − 46)**	**3.27E − 31 (1.31E − 46)**	**3.27E − 31 (1.31E − 46)**	**3.27E − 31 (1.31E − 46)**	1.55E − 01 (3.38E − 01)
f_{16}	**8.53E + 01 (7.11E − 14)**	**8.53E + 01 (7.11E − 14)**	**8.53E + 01 (7.11E − 14)**	**8.53E + 01 (7.11E − 14)**	**8.53E + 01 (7.63E − 14)**
f_{17}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	2.41E − 01 (1.50E − 01)
f_{18}	1.08E − 210 (0)	1.41E − 224 (0)	7.01E − 222 (0)	5.92E − 215 (0)	**0 (0)**
f_{19}	5.99E − 217 (0)	3.55E − 230 (0)	7.14E − 228 (0)	2.75E − 223 (0)	**0 (0)**
f_{20}	5.02E − 214 (0)	1.95E − 227 (0)	2.56E − 225 (0)	7.64E − 222 (0)	**0 (0)**
f_{21}	6.04E − 213 (0)	2.75E − 227 (0)	5.39E − 225 (0)	2.83E − 219 (0)	**0 (0)**
f_{22}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{23}	3.50E − 06 (3.65E − 06)	1.89E − 07 (1.66E − 07)	7.74E − 07 (6.72E − 07)	4.38E − 08 (1.50E − 07)	**1.55E − 20 (2.86E − 20)**
f_{24}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.63E − 15 (2.14E − 15)
f_{25}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	2.21E − 15 (2.46E − 15)

(Continued)

Table 2: Continued

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_{26}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{27}	1.94E − 15 (1.47E − 16)	1.34E − 15 (3.89E − 16)	1.99E − 15 (1.16E − 16)	1.90E − 15 (2.13E − 16)	**9.68E − 16 (3.00E − 16)**
f_{28}	**7.83E − 70 (2.73E − 69)**	5.47E − 01 (8.65E − 02)	2.79E − 41 (5.19E − 41)	1.35E − 63 (6.98E − 63)	5.78E − 01 (4.05E − 01)
f_{29}	2.42E − 08 (6.82E − 08)	5.96E − 09 (1.10E − 08)	2.32E − 10 (4.09E − 10)	4.67E − 29 (4.90E − 29)	**3.18E − 29 (3.72E − 29)**
f_{30}	2.13E − 04 (1.80E − 05)	**2.08E − 04 (2.71E − 20)**	**2.08E − 04 (2.21E − 20)**	**2.08E − 04 (1.98E − 20)**	3.49E − 04 (6.03E − 05)

Table 3: 20D results of average fitness values and standard deviation of benchmark functions– mean (Stdv)

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_1	1.72E − 226 (0)	1.99E − 191 (0)	5.06E − 255 (0)	1.70E − 241 (0)	**0 (0)**
f_2	3.35E − 234 (0)	8.43E − 204 (0)	5.29E − 263 (0)	1.07E − 252 (0)	**0 (0)**
f_3	9.33E − 66 (4.99E − 65)	1.24E − 05 (6.91E − 06)	1.72E − 37 (2.73E − 37)	8.01E − 39 (3.05E − 38)	**2.39E − 124 (1.07E − 123)**
f_4	6.21E + 00 (1.90E + 00)	4.34E − 30 (1.46E − 29)	**0 (0)**	1.93E + 00 (1.29E + 00)	1.33E + 00 (1.88E + 00)
f_5	7.06E + 00 (2.48E + 00)	**0 (0)**	**0 (0)**	3.32E − 01 (5.93E − 01)	5.39E + 01 (1.41E + 01)
f_6	1.31E − 03 (2.97E − 03)	**0 (0)**	**0 (0)**	8.22E − 04 (2.49E − 03)	3.39E − 01 (6.38E − 01)
f_7	3.40E − 17 (1.83E − 16)	**0 (0)**	**0 (0)**	1.49E − 162 (6.42e − 323)	**0 (0)**
f_8	2.15E − 01 (5.32E − 02)	**2.19E − 13 (3.43E − 13)**	1.22E − 12 (1.09E − 12)	3.96E − 02 (4.57E − 02)	1.99E + 00 (3.33E − 01)
f_9	2.11E − 07 (2.19E − 07)	2.64E − 08 (2.02E − 08)	8.12E − 08 (8.05E − 08)	**1.61E − 08 (2.57E − 08)**	2.61E + 00 (1.33E + 00)
f_{10}	3.40E − 83 (1.63E − 82)	4.80E − 18 (3.96E − 18)	1.61E − 47 (2.37E − 47)	3.44E − 57 (8.83E − 57)	**6.71E − 145 (1.92E − 144)**
f_{11}	1.40E − 112 (3.86E − 112)	1.89E − 110 (1.22E − 110)	7.92E − 126 (1.02E − 125)	1.10E − 138 (2.13E − 138)	**7.75E − 198 (0)**
f_{12}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.31E + 03 (8.83E + 02)
f_{13}	5.81E − 14 (3.02E − 13)	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{14}	4.36E − 05 (4.61E − 05)	**3.13E − 06 (3.29E − 06)**	1.76E − 05 (1.71E − 05)	1.23E − 05 (1.56E − 05)	4.18E − 06 (6.47E − 06)
f_{15}	1.65E − 31 (4.23E − 33)	**1.63E − 31 (6.57E − 47)**	**1.63E − 31 (6.57E − 47)**	**1.63E − 31 (6.57E − 47)**	1.19E − 01 (1.59E − 01)
f_{16}	4.58E − 05 (5.02E − 05)	7.37E − 06 (5.73E − 06)	3.76E − 05 (3.55E − 05)	5.28E − 06 (1.34E − 05)	**0 (0)**
f_{17}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.02E + 00 (2.92E − 01)

(Continued)

Table 3: Continued

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_{18}	5.07E − 221 (0)	2.07E − 185 (0)	3.14E − 249 (0)	6.65E − 235 (0)	**0 (0)**
f_{19}	8.78E − 226 (0)	2.17E − 191 (0)	2.88E − 255 (0)	2.21E − 242 (0)	**0 (0)**
f_{20}	1.74E − 223 (0)	1.02E − 188 (0)	6.25E − 253 (0)	7.45E − 241 (0)	**0 (0)**
f_{21}	8.78E − 224 (0)	1.69E − 188 (0)	6.99E − 252 (0)	3.54E − 238 (0)	**0 (0)**
f_{22}	3.07E − 31 (4.72E − 31)	0 (0)	0 (0)	6.82E − 31 (7.82E − 31)	**0 (0)**
f_{23}	1.63E − 06 (2.18E − 06)	2.51E − 07 (2.73E − 07)	8.82E − 07 (7.42E − 07)	2.03E − 07 (2.69E − 07)	**8.85E − 21 (2.05E − 20)**
f_{24}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.01E − 14 (7.36E − 15)
f_{25}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	9.28E − 15 (3.34E − 15)
f_{26}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{27}	3.78E − 15 (2.92E − 16)	2.35E − 15 (5.75E − 16)	4.12E − 15 (9.41E − 17)	3.74E − 15 (2.68E − 16)	**1.29E − 15 (4.50E − 16)**
f_{28}	**6.53E − 65 (1.37E − 64)**	4.01E + 00 (3.84E − 01)	1.93E − 30 (3.75E − 30)	8.10E − 57 (2.08E − 56)	2.97E + 00 (9.45E − 01)
f_{29}	1.31E − 08 (2.85E − 08)	3.84E − 09 (6.54E − 09)	7.36E − 11 (1.21E − 10)	4.97E − 29 (4.89E − 29)	**2.25E − 29 (2.98E − 29)**
f_{30}	2.52E − 08 (4.26E − 09)	**1.90E − 08 (3.87E − 24)**	**1.90E − 08 (8.21E − 11)**	**1.90E − 08 (6.70E − 24)**	3.93E − 08 (6.55E − 09)

Overall, from the experimental results it is clear that the CTbADE algorithm has a dominating performance over that of the standard DE, CoDE, EPSDE, and jDE algorithms. The CTbADE algorithm outperforms other well-known and state-of-the-art algorithms in most cases. The CTbADE algorithm's convergence is better than the other algorithms for unimodal, multimodal, separable and non-separable functions. It shows that we can apply the CTbADE algorithm to solve problems of varying natures. In cases of 10D, 20D, and 30D, CTbADE outperforms other algorithms for unimodal problems, such as the axis parallel hyperellipsoid, Schwefel's problem, Schwefel's problem 2.22, ellipse function and tablet function, as well as the Neumaier-2 problem in the cases of 20D and 30D. The performance of CTbADE is better in multimodal problems, such as the sphere model in cases of 10D, 20D, and 30D; sum of difference power in the case of 10D; the Zakharov function in cases of 10D, 20D and 30D; cigar function in cases of 10D, 20D, and 30D; function-15 in cases of 10D, 20D, and 30D; the problem of the deflected corrugated spring in cases of 10D, 20D, and 30D; the problem of multi-model global optimization in the case of 20D; and the Quintic global optimization problem also in the case of 20D. The performance of CTbADE is better in separable problems, such as the sphere model in cases of 10D, 20D, and 30D; the axis parallel hyperellipsoid in cases of 10D, 20D, and 30D; the Neumaier-2 problem in 20D and 30D; cigar in cases of 10D, 20D, and 30D; function-15 in cases of 10D, 20D, and 30D; the ellipse function in case of 10D, 20D, and 30D; the tablet function in case of 10D, 20D, and 30D; the deflected corrugated spring in cases of 10D, 20D, and 30D; the multimodal global optimization problem and the Quintic global optimization problem, both in the case of 20D. The performance of CTbADE is better for non-separable problems, such as Schwefel's problem 1.2 in cases of 10D, 20D, and 30D; sum of different power in the case of 10D; Zakharov function in 10D, 20D, and 30D; Schwefel's problem 2.22 in cases of 10D, 20D, and 30D. Although the fitness value of the given average value fitness tables are same, the convergence speed of CTbADE is better

for the unimodal problem De Jong's no-noise function-4 in cases of 10D and 20D. The performance of CTbADE is better for unimodal problems in 10D and 20D. The convergence speed of CTbADE is better for multimodal problems, such as Alpine function in the case of 10D; Schewel function in case of 10D; the multimodal global optimization problem in the case of 30D; the Quintic global optimization problem in case of 10D and 30D; the stretched_V_global_optimization in cases of 20D and 30D. The performance of CTbADE is better for separable problems, such as the sum of different power in the case of 20D; De Jong's no-noise function-4 in cases of 10D and 20D; Alpine function in the case of 10D; Schewel function in the case of 10D; the multimodal global optimization problem in the case of 30D and the Quintic global optimization problem in cases of 10D and 30D. The performance of CTbADE is better for non-separable problems, such as the sum of different power problem in the case of 20D and the stretched_V_global_optimization problem in the case of 20D and 30D.

The performance of the CTbADE algorithm is competitive compared to other algorithms for unimodal functions: De Jong's no-noise function-4 in the case of 30D and the Neumaier-2 problem in the case of 10D; for multimodal functions sum of different problem in case of 30D; Alpine function in case of 20D and 30D; multimodal global optimization problem in case of 10D and 20D; Quintic global optimization problem in the case of 30D; stretched V global optimization problem in the case of 20D; Xin-SheYang in the case of 30D; separable functions for De Jong's no noise function-4 in the case of 30D; Neumaier-2 Problem in the case of 10D; Alpine function in cases of 20D and 30D; multimodal global optimization problem in cases of 10D and 20D; Quintic global optimization problem in the case of 30D; stretched_V_global_optimization problem in the case of 20D; for the non-separable problem sum of different power in case of 30D; Xin-SheYang in the case of 30D.

Logarithmic convergence graphs of some sampled problems have been shown in Fig. 2 of this paper. The horizontal axis shows the iterations, and the average fitness performance is shown vertically in each logarithmic convergence graph. From the logarithmic convergence graphs, it can be noted that the average fitness performance of the CTbADE algorithm is better in most cases than the standard DE algorithm, jDE algorithm, EPSDE algorithm, and CoDE algorithm.

Fig. 2a shows convergence graph of Schwefel's problem-1.2, showing the number of training iterations against the average fitness performance value. It is evident from this figure that the performance of the CTbADE algorithm is superior to the other algorithms, as it is the only algorithm that gains the global optima in the specified number of generations. The speed of CTbADE in convergence graph 2.b is better from early iterations till the last iteration. It can be observed from Fig. 2c of Schwefel's problem 2.22, that all the algorithms are gradually converging towards global but the performance of CTbADE is much quicker than to the other algorithms. It is noted from Tab. 4 of the average fitness values that the EPSDE, CoDE, and CTbADE algorithms are reaching the global optimal value but the convergence speed of CTbADE is higher as confirmed in Fig. 2b. From Fig. 2d it is clear that, in early iterations, the performance of the CTbADE algorithm remains similar to the others but in succeeding iterations it decreases gradually and performs better than the other algorithms. From Fig. 2e and Tab. 4, it can be observed that all algorithms are achieving global optimal value but the convergence of CTbADE algorithm is better. Fig. 2e shows the better performance of CTbADE algorithm from the initial iteration till the last specified iteration. Thus, it is evident from the convergence graphs that the CTbADE algorithm has the leading performance in most of the cases against the standard DE algorithm, EPSDE algorithm, CoDE algorithm, and jDE algorithm. The CTbADE algorithm will thus prove to be a valuable contribution to the existing DE literature.

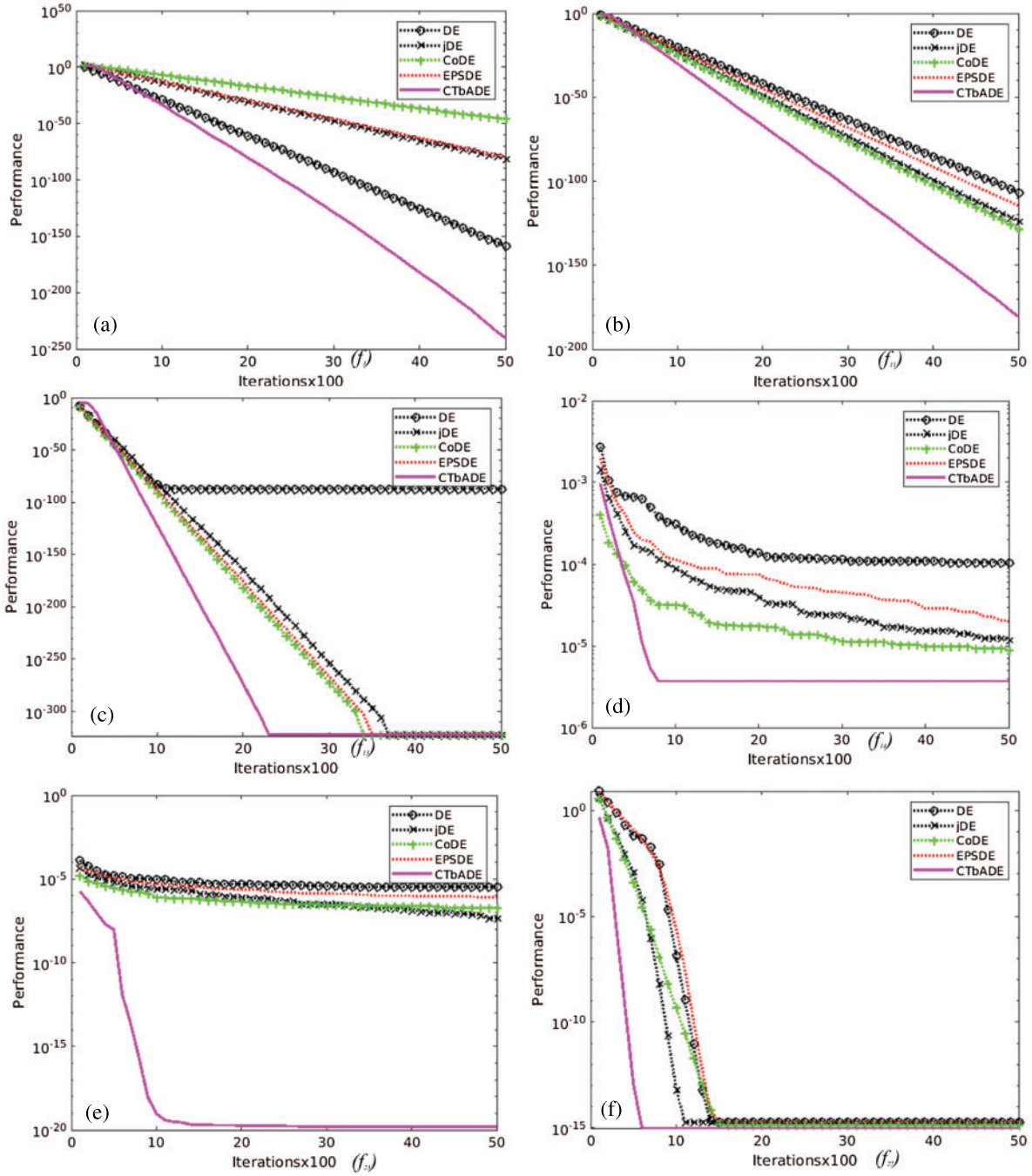

Figure 2: 10 Dimensional convergence graphs of a few benchmark functions, showing performance vertically and iterations horizontally (a) Schwefel's problem 1.2 (b)Schwefel's problem 2.22 (c) De Jong's function 4 (no noise), (d) Alpine function (e) Deflected corrugated spring (f) Quintic global optimization problem

Table 4: 30D results of average fitness values and standard deviation of benchmark functions– mean

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_1	9.98E − 237 20)	7.84E − 164 (0)	1.48E − 282 (0)	7.08E − 259 (0)	**4.94E − 324 (0)**
f_2	2.12E − 242 (0)	3.30E − 171 (0)	1.20E − 288 (0)	3.42E − 265 (0)	**4.94E − 324 (0)**
f_3	4.52E − 39 (2.17E − 38)	3.52E + 01 (2.38E + 01)	7.37E − 22 (2.41E − 21)	3.37E − 22 (8.11E − 22)	**9.18E − 75 (3.21E − 74)**
f_4	1.28E + 01 (6.32E + 00)	2.28E − 19 (2.45E − 19)	**1.22E − 30 (4.57E − 30)**	1.01E + 01 (3.91E + 00)	6.64E − 01 (1.49E + 00)
f_5	1.35E + 01 (5.01E + 00)	**0 (0)**	**0 (0)**	1.29E + 00 (1.12E + 00)	7.66E + 01 (1.62E + 01)
f_6	2.05E − 03 (4.76E − 03)	1.81E − 21 (9.73E − 21)	**0 (0)**	2.47E − 04 (1.33E − 03)	2.43E − 01 (8.99E − 01)
f_7	1.01E − 10 (5.26E − 10)	0 (0)	**0 (0)**	5.75E − 12 (3.10E − 11)	0 (0)
f_8	2.53E − 01 (4.49E − 02)	2.25E − 13 (2.07E − 13)	1.68E − 12 (1.58E − 12)	8.75E − 02 (4.78E − 02)	2.36E + 00 (3.47E − 01)
f_9	7.68E − 08 (5.84E − 08)	1.67E − 08 (1.27E − 08)	7.43E − 08 (6.60E − 08)	**1.44E − 08 (2.25E − 08)**	9.01E + 00 (3.92E + 00)
f_{10}	3.37E − 52 (1.35E − 51)	1.01E − 07 (5.38E − 08)	1.79E − 29 (2.01E − 29)	7.57E − 40 (1.75E − 39)	**9.59E − 88 (2.37E − 87)**
f_{11}	6.86E − 123 (8.30E − 123)	1.20E − 96 (7.18E − 97)	2.84E − 137 (3.49E − 137)	1.64E − 148 (5.54E − 148)	**7.44E − 206 (0)**
f_{12}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	5.50E + 03 (2.09E + 03)
f_{13}	1.62E − 10 (8.74E − 10)	1.06E − 271 (0)	**0 (0)**	**0 (0)**	0 (0)
f_{14}	3.43E − 05 (3.19E − 05)	2.00E − 06 (1.84E − 06)	8.72E − 06 (6.83E − 06)	4.25E − 06 (3.62E − 06)	3.37E − 06 (3.28E − 06)
f_{15}	**1.09E − 31 (1.10E − 34)**	**1.09E − 31 (8.76E − 47)**	**1.09E − 31 (8.76E − 47)**	**1.09E − 31 (9.99E − 34)**	1.66E − 01 (2.22E − 01)
f_{16}	7.63E − 05 (5.44E − 05)	2.75E − 05 (3.79E − 05)	6.55E − 05 (9.10E − 05)	7.78E − 06 (1.38E − 05)	**3.05E − 07 (6.70E − 07)**
f_{17}	9.85E − 03 (3.69E − 02)	**0 (0)**	**0 (0)**	**0 (0)**	1.73E + 00 (4.77E − 01)
f_{18}	4.16E − 231 (0)	3.63E − 158 (2.57e − 315)	2.80E − 277 (0)	9.08E − 255 (0)	**0 (0)**
f_{19}	5.44E − 236 (0)	2.19E − 163 (0)	9.77E − 283 (0)	1.28E − 259 (0)	**0 (0)**
f_{20}	2.60E − 233 (0)	4.45E − 161 (2.33e − 321)	2.08E − 280 (0)	1.68E − 257 (0)	**9.88e − 324 (0)**
f_{21}	8.53E − 235 (0)	5.16E − 161 (3.85e − 321)	3.50E − 280 (0)	8.76E − 255 (0)	**0 (0)**
f_{22}	5.88E − 30 (7.86E − 30)	**1.27E − 32 (6.86E − 32)**	9.04E − 32 (1.71E − 31)	4.48E − 30 (5.10E − 30)	1.40E − 31 (2.65E − 31)

(Continued)

Table 4: Continued

Function	DE	CoDE	EPSDE	jDE	CTbADE
f_{23}	1.64E − 06 (1.61E − 06)	2.87E − 07 (3.07E − 07)	6.26E − 07 (5.38E − 07)	2.23E − 07 (2.86E − 07)	**9.83E − 10 (3.92E − 09)**
f_{24}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.00E + 09 (0)
f_{25}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	1.00E + 09 (0)
f_{26}	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**
f_{27}	5.20E − 15 (2.88E − 16)	2.61E − 15 (6.13E − 16)	6.18E − 15 (8.78E − 17)	5.30E − 15 (3.48E − 16)	**1.81E − 15 (4.16E − 16)**
f_{28}	**1.07E − 69 (2.15E − 69)**	9.55E + 00 (5.88E − 01)	9.50E − 27 (1.43E − 26)	3.70E − 52 (1.85E − 51)	7.01E + 00 (1.67E + 00)
f_{29}	6.87E − 09 (2.03E − 08)	7.69E − 09 (2.09E − 08)	4.01E − 11 (9.90E − 11)	2.80E − 29 (4.48E − 29)	**2.01E − 29 (2.72E − 29)**
f_{30}	2.63E − 12 (9.83E − 13)	**1.29E − 12 (5.84E − 28)**	3.82E − 12 (4.38E − 13)	1.30E − 12 (2.90E − 14)	2.81E − 12 (4.47E − 13)

6 Conclusion

This paper presents a convergence tracking over time based parametric adaptive version of differential evolution algorithms. The two important operators of parameters of mutation probability that contribute are mutation operation and crossover rate, which are controlled by using a new sequence over time. The proposed sequences are helpful to track and follow the convergence path and remove the DE algorithm from the local optima problem. The concept of a small memory based on a user defined learning period is used in the sequence of control parameters of the algorithm to improve the convergence behavior of DE algorithms and escaping from the stagnation problem. A comprehensive suite of well-known and varying nature benchmark standard optimization functions was used to test the convergence power of the proposed convergence track based adaptive evolution algorithm (CTbADE). The experimental results are generated using the same set of parameters for a fair comparison. The results of the new CTbADE algorithm are compared with standard DE algorithms and some other well-known and commonly used state-of-the-art algorithms, such as CoDE, jDE, and EPSDE algorithms. The research result shows that the proposed CTbADE algorithm has more powerful convergence than the other algorithms used. The novel optimization algorithm presented in this research work will help the development of a fast automated diagnostics model to detect Covid-19 infection. The future work of this research work is to apply the CTbADE algorithm to optimize the hyper parameters of convolutional neural networks for Covid-19 CT/X-ray images feature extraction and classification to construct a fast automated diagnostics model.

Acknowledgement: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through project number 959.

Funding Statement: This work was supported by the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, which funded this research work through project number 959.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," CA, Berkeley, Tech. Rep TR-95-012, 1995.

[2] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transaction on Evolutionary Computing*, vol. 10, no. 6, pp. 646–657, 2006.

[3] K. Price, R. M. Storn and J. A. Lampinen, "*Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*," 1st ed. New York, USA: Springer-Verlag, 2005.

[4] A. Karsaz, "Evaluation of lung involvement in patients with coronavirus disease from chest ct images using multi-objective self-adaptive differential evolution approach," *Journal of Control*, vol. 14, no. 5, pp. 1–14, 2021.

[5] X. Yi, L. Li, T. Yu, J. Shi, M. Cai *et al.,* "Double-layer wideband reflectarray using polynomial optimization technology," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 31, no. 5, pp. e22594, 2021.

[6] G. Liu, Y. Li and G. He, "Design of digital fir filters using differential evolution algorithm based on reserved genes," in *IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, pp. 1–7, 2010.

[7] X. Zhang, L. Jin, C. Cui and J. Sun, "A self-adaptive multi-objective dynamic differential evolution algorithm and its application in chemical engineering," *Applied Soft Computing*, vol. 106, pp. 107317, 2021.

[8] M. Baioletti, G. Di Bari, A. Milani and V. Poggioni, "Differential evolution for neural networks optimization," *Mathematics*, vol. 8, no. 1, pp. 69, 2020.

[9] E. Marchiori, J. H. Moore and J. C. Rajapakse, "Evaluating evolutionary algorithms and differential evolution for the online optimization of fermentation processes" *Machine Learning and Data Mining in Bioinformatics*, Berlin, Heidelberg: Springer, 2007.

[10] S. Li, W. Gong, C. Hu, X. Yan, L. Wang *et al.,* "Adaptive constraint differential evolution for optimal power flow," *Energy*, vol. 235, pp. 121362, 2021.

[11] G. F. Plichoski, G. F. Chidambaram and R. S. Parpinelli, "A face recognition framework based on a pool of techniques and differential evolution," *Information Sciences*, vol. 543, pp. 219–241, 2021.

[12] A. Smirnov and R. P. Jastrzebski, "Differential evolution approach for tuning an h∞controller in amb systems," in *35th IEEE Annual Conf. of Industrial Electronics*, Lappeenranta, Finland, pp. 1514–1518, 2009.

[13] D. Singh, V. Kumar and M. Kaur, "Classification of covid-19 patients from chest ct images using multi-objective differential evolution–based convolutional neural networks," *European Journal of Clinical Microbiology & Infectious Diseases*, vol. 39, no. 7, pp. 1379–1389, 2020.

[14] O. H. Ahmed, J. Lu, Q. Xu, A. M. Ahmed, A. M. Rahmani *et al.,* "Sing differential evolution and moth–flame optimization for scientific workflow scheduling in fog computing," *Applied Soft Computing*, vol. 112, pp. 107744, 2021.

[15] Y. Yu, H. Wang, S. Liu, L. Guo, P. Yeoh *et al.,* "Distributed multi-agent target tracking: A nash-combined adaptive differential evolution method for uav systems," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 8122–8133, 2021.

[16] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, no. 11, pp. 341–359, 1997.

[17] X. Dong, C. S. Deng, Y. Zhang and Y. C. Tan, "Enhancing local search of differential evolution algorithm for high dimensional optimization problem," *Chinese Control Conference* , Hangzhou, China, pp. 8411–8415, 2015.

[18] M. Duan, H. Yang, S. Wang and Y. Liu, "Self adaptive dual strategy differential evolution algorithm," *Plos one*, vol. 14, no. 10, pp. e0222706, 2019.

[19] Q. Abbas, J. Ahmad and H. Jabeen, "Random controlled pool base differential evolution algorithm," *Intelligent Automation & Soft Computing*, vol. 2017, pp. 1–14, 2017.

[20] A. Rehman, T. Sadad, T. Saba, A. Hussain and T. Tariq, "Real-time diagnosis system of covid-19 using x-ray images and deep learning," *It Professional*, vol. 23, no. 4, pp. 57–62, 2021.

[21] V. Ravi, H. Narasimhan, C. Chakraborty and T. D. Pham, "Deep learning-based meta-classifier approach for covid-19 classification using ct scan and chest x-ray images," *Multimedia Systems*, vol. 2021, pp. 1–15, 2021.
[22] S. Ahuja, B. K. Panigrahi, N. Dey, V. Rajinikanth and T. K. Gandhi, "Deep transfer learning-based automated detection of covid-19 from lung ct scan slices," *Applied Intelligence*, vol. 51, no. 1, pp. 571–585, 2021.
[23] P. Saha, D. Mukherjee, P. K. Singh, A. Ahmadian, M. Ferrara *et al.,* "Graphcovidnet: A graph neural network based model for detecting covid-19 from ct scans and x-rays of chest," *Scientific Reports*, vol. 11, no. 1, pp. 1–16, 2021.
[24] A. Narin, C. Kaya and Z. Pamuk, "Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks," *Pattern Analysis and Applications*, vol. 24, pp. 1–14, 2021.
[25] K. Rao, K. Xie, Z. Hu, X. Guo, C. Wen *et al.,* "Covid-19 detection method based on svrnet and svdnet in lung x-rays," *Journal of Medical Imaging*, vol. 8, no. S1, pp. 017504, 2021.
[26] P. Singh, S. Chaudhury and B. K. Panigrahi, "Hybrid mpso-cnn: Multi-level particle swarm optimized hyperparameters of convolutional neural network," *Swarm and Evolutionary Computation*, vol. 63, pp. 100863, 2021.
[27] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proc. of the Mendel 9th Int. Conf.*, University of Technology, Brno, Czech Republic, vol. 9, pp. 41–46, 2003.
[28] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, pp. 448–462, 2005.
[29] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.
[30] J. Zhang and A. C. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
[31] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 13, pp. 398–417, 2009.
[32] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.
[33] R. Mallipeddi, P. N. Suganthan, Q. K. Pan and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, pp. 1679–1696, 2011.
[34] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.
[35] S. Minhazul-Islam, S. Das, S. Ghosh, S. Roy and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics-PART B: Cybernetics*, vol. 42, no. 2, pp. 482–500, 2012.
[36] H. Wang, S. Rahnamayan and Z. Wu, "Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 62–73, 2013.
[37] V. Gonuguntla, R. Mallipeddi and K. C. Veluvolu, "Differential evolution with population and strategy parameter adaption," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–11, 2015.
[38] Q. Fan and X. Yan, "Self adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 219–232, 2016.
[39] Z. Zhao, J. Yang, Z. Hu and H. Che, "A differential evolution algorithm with self adaptive strategy and control parameters based on symmetric latin hypercube design for unconstrained optimization problems," *European Journal of Operation Research*, vol. 250, pp. 1–16, 2015.

[40] H. Peraza-Vázquez, A. M. Torres-Huerta and A. Flores-Vela, "Self adaptive differential evolution hyper-heuristic with applications in process design," *Computación y Sistemas*, vol. 20, no. 2, pp. 173–193, 2016.
[41] Q. Fan, W. Wang and X. Yan, "Differential evolution algorithm with strategy adaption and knowledge based control," *Artificial Intelligence Review*, vol. 51, no. 2, pp. 219–253, 2019.
[42] M. Duan, H. Yang, S. Wang and Y. Liu, "Self-adaptive dual-strategy differential evolution algorithm," *Plos one*, vol. 14, no. 10, pp. e0222706, 2019.
[43] M. A. Attia, M. Arafa, E. A. Sallam and M. M. Fahmy, "An enhanced differential evolution algorithm with multi-mutation strategies and selfadapting control parameters," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 4, pp. 26–38, 2019.
[44] R. Jiang, J. Zhang, Y. Tang, C. Wang and J. Feng, "A collective intelligent based differential evolution algorithm for optimizing the structure and parameters of a neural network," *IEEE Access*, vol. 8, pp. 69601–69614, 2020.
[45] Q. Abbas, J. Ahmad and H. Jabeen, "A novel tournament selection based differential evolution variant for continuous optimization problems," *Mathematical Problems in Engineering*, vol. Article ID 205709, pp. 1–21, 2015.

Appendix: Benchmark Functions

Table A: Test suit of benchmark functions ($f_1 - f_{15}$)

Function	Name of function (type)	Equation	Search space	Optima
f_1	Sphere model (Separable, Multimodal)	$f(x) = \sum_{i=0}^{n} x_i^2$	$-5.12 \leq x_i \leq 5.12$	0
f_2	Axis parallel hyperellipsoid (Separable, Unimodal)	$f(x) = \sum_{i=0}^{n} i \cdot x_i^2$	$-5.12 \leq x_i \leq 5.12$	0
f_3	Schwefel's problem 1.2 (Non-Separable, Unimodal)	$f(x) = \sum_{i=0}^{n} \left(\sum_{j=0}^{i} x_j \right)^2$	$-65 \leq x_j \leq 65$	0
f_4	Rosenbrock's valley (Non-Separable, Unimodal)	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	$-30 \leq x_i \leq 30$	0
f_5	Rastrigin's function (Separable, Multimodal)	$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	$-5.12 \leq x_i \leq 5.12$	0
f_6	Griewank's function (Non-Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} (\frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$-600 \leq x_i \leq 600$	0
f_7	Sum of different power (Non-Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$	$-1 \leq x_i \leq 1$	0

(Continued)

Table A: Continued

Function	Name of function (type)	Equation	Search space	Optima
f_8	Ackley's path function (Non-Separable, Multimodal)	$f(x) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=i}^{n} Cos(2\pi x_i}{n}\right) + 20 + e$	$-32 \leq x_i \leq 32$	0
f_9	Levy function (Separable, Multimodal)	$0.1\Big[\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2 \times (1+\sin^2(3\pi x_i+1)) + (x_n-1)(1+\sin^2(2\pi x_n))\Big]$	$-10 \leq x_i \leq 10$	0
f_{10}	Zakharov function (Non-Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=i}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=i}^{n} 0.5ix_i\right)^4$	$-5 \leq x_i \leq 10$	0
f_{11}	Schwefel's problem 2.22 (Non-Separable, Unimodal)	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	$-10 \leq x_i \leq 10$	0
f_{12}	Step function (Separable, Unimodal)	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5\rfloor)^2$	$-100 \leq x_i \leq 100$	0
f_{13}	De Jong's function 4 (no noise) (Separable, Unimodal)	$f(x) = \sum_{i=1}^{n} ix_i^4$	$-1.28 \leq x_i \leq 1.28$	0
f_{14}	Alpine function (Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i\rvert$	$-10 \leq x_i \leq 10$	0
f_{15}	Levy and Montalvo Problem (Separable, Multimodal)	$f(x) = \left(\frac{\pi}{n}\right)\Big(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1 + 10\sin 2(\pi y_{i+1})]\Big) + (y_n-1)^2$ $y_i = 1 + \frac{1}{4}(x_i+1)$ where	$-10 \leq x_i \leq 10$	0

Table B: Test suit of benchmark functions ($f_{16} - f_{30}$)

Function	Name of function (type)	Equation	Search space	Optima
f_{16}	Neumaier 2 Problem (Separable, Unimodal)	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} (x_i x_{i-1})$	$-n^2 \leq x_i \leq n^2$	0
f_{17}	Cosine Mixture (Separable, Multimodal)	$f(x) = -0.1 \sum_{i=1}^{n} \cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2 \quad \|\|x\|\| = \sqrt{\sum_{i=1}^{n} x_i^2}$	$-1 \leq x_i \leq 1$	
-0.1x (n)				
f_{18}	Cigar (Separable, Multimodal)	$f(x) = {x_1}^2 + 100000 \sum_{i=1}^{n} {x_i}^2$	$-10 \leq x_i \leq 10$	0
f_{19}	Function '15' (Separable, Multimodal)	$f(x) = \sum_{i=1}^{n-1} [0.2{x_i}^2 + 0.1{x_i}^2 \sin(2x_i)]$	$-10 \leq x_i \leq 10$	0
f_{20}	Ellipse Function (Separable, Unimodal)	$f(x) = \sum_{i=1}^{n} (10^{6(\frac{i-1}{n-1})}.x_i^2)$	$-100 \leq x_i \leq 100$	0
f_{21}	Tablet Function (Separable, Unimodal)	$f(x) = 10^4 x_1^2 + \sum_{i=2}^{n} x_i^2$	$-100 \leq x_i \leq 100$	0
f_{22}	Schewel (Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} ((x_1 - x_i^2)^2 + (x_i - 1)^2)$	$-32 \leq x_i \leq 32$	0
f_{23}	Deflected Corrugated Spring (Separable, Multimodal)	$f(x) = 0.1 \sum_{i=1}^{n} \left((x_i - \alpha)^2 - \cos\left(K \sqrt{\sum_{i=1}^{n} ((x_i - \alpha)^2)} \right) \right)$	$0 \leq x_i \leq 10 \quad K = 5$ $\alpha = 5 \quad x_i \in [0, 2\alpha]$	0
f_{24}	Mishra 1 global optimization problem (Non-Separable, Multimodal)	$f(x) = (1 + x_n)^{x_n}$ where $x_n = n - \sum_{i=1}^{n-1} x_i$	$0 \leq x_i \leq 1$	2

(Continued)

Table B: Continued

Function	Name of function (type)	Equation	Search space	Optima
f_{25}	Mishra 2 global optimization problem (Non-Separable, Multimodal)	$f(x) = (1 + x_n)^{x_n}$ where $x_n = n - \sum_{i=1}^{n-1} \frac{(x_i + x_{i+1})}{2}$	$0 \leq x_i \leq 1$	2
f_{26}	MultiModal global optimization problem (Separable, Multimodal)	$f(x) = \left(\sum_{i=1}^{n} \lvert x_i \rvert\right)\left(\prod_{i=1}^{n} \lvert x_i \rvert\right)$	$-10 \leq x_i \leq 10$	0
f_{27}	Quintic global optimization problem (Separable, Multimodal) 1	$f(x) = \sum_{i=1}^{n} \lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \rvert$	$-10 \leq x_i \leq 10$	-1
f_{28}	Stochastic global optimization problem (Separable, Multimodal)	$f(x) = \sum_{i=1}^{n} \varepsilon_i \left\lvert x_i - \frac{1}{i} \right\rvert$	$-5 \leq x_i \leq 5$	0
f_{29}	Stretched V global optimization problem (Non-Separable, Multimodal)	$f(x) = \sum_{i=1}^{n-1} t^{1/4}[\sin(50t^{0.1}) + 1]^2$ where $t = x_{i+1}^2 + x_i^2$	$-10 \leq x_i \leq 10$	0
f_{30}	XinSheYang (Non-Separable, Multimodal)	$f(x) = \left(\sum_{i=1}^{n} \lvert x_i \rvert\right) \Big/ e^{\sum_{i=1}^{n} \sin(x_i^2)}$	$-2\pi \leq x_i \leq 2\pi$	0