Chromosomes identification based differential evolution (CIDE): a new bio-inspired variant for network intrusion detection

Md. Nayer1 · Subhash Chandra Pandey2

Received: 17 August 2021 / Revised: 3 January 2022 / Accepted: 28 February 2022 / Published online: 18 March 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Differential Evolution (DE) is perhaps one of the most efficient and versatile evolutionary techniques and its different variants have been proposed in the recent past. However, all these variants often suffer from the shortcomings of slow and/or premature convergence. Further, nearest neighbour (NN) is a frequently used classifier in the domain of data mining but its primitive version is noise sensitive and shows low accuracy. Different data reduction methods such as prototype selection, prototype generation, and DE are widely used to mitigate these drawbacks of NN. However, none of the methods substantially avoids all these shortcomings. This paper proposes a new variant of DE that is based on the biological phenomenon of the centromeric location of chromosomes. The proposed ‘Chromosomes Identification based Differential Evolution (CIDE)’ extracts the knowledge from databases and constructs effective and reliable decision rules for classification. In this paper, the performance of the proposed CIDE technique is computed and compared with 1-NN classifier optimized by fourteen state-of-art variants of DE, and 1-R classifier using six-real datasets as well as KDDCup’99 dataset to illustrate its superiority over the 1-NN and 1-R classifier. Finally, the performance of CIDE is also compared with Naïve Bayes, support vector machine, Neural Network, and Random Forest for the KDDCup’99 dataset using different performance metrics.

Keywords Natural computing · Differential evolution · Chromosomes identification · Data classification · Network intrusion detection

Abbreviations
DE Differential evolution
CIDE Chromosomes identification based differential evolution
CP Control parameter
NN Nearest neighbour
CR Cross-over rate
PS Prototype selection
PG Prototype generation
CNN Condensed nearest neighbour
ENN Edited nearest neighbour
IBL Instance-based learning
DROP Decremental reduction optimization
ICF Iterative case filtering
LVQ Learning quantization vector
ICPL Integrated concept prototype learner
HYB Hybrid
MixtGauss Mixtures of Gaussians
SSMA Steady-state memetic algorithm
ENPC Evolutionary nearest prototype classifiers
PSCSA Prototype selection clonal selection algorithm
FADE Fuzzy adaptive differential evolution
EPSDE Ensemble parameter generation strategy based DE
EDA Estimation of distribution algorithm
UCI University of California Irvin
SADE Self adaptive DE
DEGL DE global and local
SFLSDE Scale factor local search in DE
NIDS Network intrusion detection system

1 Research Scholar, Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi, (Patna Campus), Patna, Bihar, India
2 Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi, (Patna Campus), Patna, Bihar, India
1 Introduction

Nature is endowed with diversified phenomena and many of them are related to computing. Natural computing is a highly interdisciplinary domain and it mimics nature-inspired methodologies and techniques. Precisely, it is the amalgamation of natural phenomena and computation and is widely used in fundamental research and information sciences. In the recent past, different nature-inspired techniques are used for performing specific tasks including the information fusion in offspring generation [1]. In [2], a novel population-based optimization method known as Aquila Optimizer (AO) is proposed. This method is motivated by the Aquila’s behaviours observed in nature during the process of prey catching.

Further, evolutionary computation (EC) is a peculiar ingredient of natural computing and it is motivated by the Darwinian theory of evolution [3]. Broadly, EC can be classified into three streams. These are [4]:

- Evolutionary strategies
- Evolutionary programming
- Genetic algorithm (GA)

EC is widely used for solving the problems of the domain of data mining, and data reduction processes are very useful tasks within the purview of data mining. In [5], a new and efficient clustering technique is proposed for the clustering of the text document. This technique entails a novel attribute selection technique based on particle swarm optimization (PSO) and it also renders efficient dimensionality reduction and thus requires reduced computational cost. Two important data reduction techniques are given in [6, 7]. These are:

- Prototype selection (PS)
- Prototype generation (PG)

In the PS process, a subset of instances is selected from the original training dataset. The literature survey reveals the fact that the PS process can be grouped into three categories. Namely: condensation [8], edition [9], and hybrid methods [10]. The condensation method is an attempt to remove those instances which do not possess sufficient classification capabilities. Whereas, the task of the edition method is to remove the noisy instances from the datasets and hybrid methods combine the features of both approaches. The PG method for data reduction is also known as the prototype abstraction method [10]. PG method can perform varieties of tasks such as it can select the data, modifying the data, and even more, it can perform interpolation and movements of instances.

EC can also be used for solving the PS and PG problems [11, 12]. In [13, 14], a novel aspect of EC is proposed that is known as the Differential Evolution (DE). The functioning of DE differs slightly from GA. Indeed, the mutation process of DE is different from GA. The mutation operation of DE is an outcome of arithmetic combinations of individuals whereas the mutation operation of GA creates minor changes in the positioning of genes. Perhaps, the good convergence and its comprehensiveness render increased popularity for DE [15]. Recent advances of DE are given in [16].

Furthermore, it is pertinent to mention that like other evolutionary algorithms, DE also requires parameters specification. Different control parameters (CP) used in DE are given below:

- Population size
- Scalar factor \( S \)
- Cross over rate (CR).

These parameters substantially affect the efficacy of DE. However, the selection of CP is time-consuming and requires a high computational cost. The effect of good parameter selection on the convergence of DE is given in [17–19]. In [20], the adjustment of CP is performed using the problem-dependent heuristic rules but in this approach, the population size is not adaptive and thus the performance of this approach varies with the population size. This is the major drawback of this approach. A hybrid strategy of DE and modified particle swarm optimization for the numerical solution of a parallel manipulator is given in [21]. The convergence rate of this approach is high and also provides efficient global optimization. However, its performance is problem-dependent and is not versatile.

In this paper, the authors propose a new variant of DE that is based on the phenomenon of chromosomes identification. The chromosomes identification method is mainly based on geometric and morphological features of chromosomes. Indeed, the locus of ‘centromere’ plays a vital role in chromosomes identification. Further, the identification of the chromosome facilitates the task of species classification. Figure 1 shows the biological pattern of centromeric locations in different chromosomes. Figure 1 is drawn based on concepts given in [22]. The contributions of this paper are enumerated below:

1. A new variant of DE known as ‘Chromosomes Identification based Differential Evolution (CIDE)’ is designed and experimented with.
2. The performance of the proposed CIDE technique is compared with the 1-NN classifier optimized by fourteen state-of-art variants of DE and 1-R using six-real datasets. These datasets are Iris, Wine, New-Thyroid, Breast, Wisconsin, and Splice Junction [23].
3. Standard deviation and Friedman test are also performed to illustrate and validate the superiority of the proposed CIDE method over the 1-NN and 1-R.

4. The superiority of the proposed CIDE technique over the 1-NN optimized by fourteen state-of-art variants of DE, and 1-R classifier is also illustrated for the KDDCup’99 dataset using the metrics: classification accuracies, false alarm rate, and precision.

5. The performance of the proposed CIDE technique is also compared with Naive Bayes, support vector machine (SVM), Neural Network, and Random Forest in terms of the performance metrics: accuracy, precision, recall, and F1-score.

The remaining part of the paper has been structured as follows: DE and its recent advances are depicted in Sect. 2. The proposed CIDE is illustrated in Sect. 3. Further, Sect. 4 renders the experimental results and discussions. Experimental Results and Analysis for the KDDCup’99 dataset are illustrated in Sect. 5. Finally, the concluding remarks are given in Sect. 6.

2 DE methods

This section will render introductory information on DE. Section 2.1 presents the basic information of DE and the advancements that took place in the recent past are given in Sect. 2.2.

2.1 DE

DE evolves a given initial population. Let $NP$ be a stochastically chosen population size with uniform distribution and $D, G$ are dimensional parameter vector and generation respectively. At $G = 0$, the commencing population

$$\{x_{i,0} = (x_{i,0}^1, x_{i,0}^2, \ldots, x_{i,0}^D) \text{ where } i = 1, 2 \ldots NP\}$$

is randomly sampled from the feasible solution space. At each generation $G$, the encoded candidate solutions can be symbolically written as:

$$X_{i,G} = (x_{i,G}^1, x_{i,G}^2, \ldots, x_{i,G}^D) \text{ where } i = 1, 2 \ldots NP$$

Analytically, the minimum and maximum parameter bounds of search space can be represented as:

$$X_{\text{min}} = \{x_{\text{min}}^1, x_{\text{min}}^2, \ldots, x_{\text{min}}^D\}$$

$$X_{\text{max}} = \{x_{\text{max}}^1, x_{\text{max}}^2, \ldots, x_{\text{max}}^D\}$$

The initial value of the $j$th parameter in $i$th individual at the generation $G = 0$ is generated by

$$x_{j,i}^0 = x_{j,\min}^i + \text{rand} (0, 1). (x_{j,\max}^i - x_{j,\min}^i), j = 1, 2, \ldots, D$$

(1)

It is important to mention that rand (0, 1) represents a random variable of uniform distribution within the range [0, 1].

DE entails three operations as enumerated below:

1. Mutation operation
   The six frequently used operators for mutation are given below.
   - DE/Rand/1/Bin
   - DE/Best/1/Bin
   - DE/RandToBest/1/Bin
   - DE/Best/2/Bin
   - DE/Rand/2/Bin
   - DE/RandToBest/2/Bin

2. Cross-over operation
3. Selection operation

The first two operations are implemented to create the new vectors. These vectors are known as trial vectors. Subsequently, the selection operation decides the survival of the vectors for the next generation. Details of these operations are elaborated in the Appendix.

2.2 Recent advances

Different variants of DE have been proposed in the recent past. In [24], a hybridized approach of the fuzzy logic controller and genetic operator is implemented to set the CP. Further, in [25], a combination approach of DE is introduced and it has experimentally been observed that this approach augments the efficacy and versatility but in this technique, some values are empirically obtained. Further, a stochastic genetic algorithm is also proposed which
employs a stochastic coding strategy [26]. But, it is pertinent to mention that the relationship between the interaction of control parameter setting and performance of GA is complex to establish and is not comprehensible. This makes the stochastic coding strategy less viable. In [27], a new variant of DE, called FADE is introduced which uses the concept of fuzzy logic for dynamic adjustment of $F$ and $CR$. But, the use of a fuzzy logic controller for parameter searching renders its functionality complex.

Moreover, in [28], a hybridization of DE and estimation of distribution algorithm (EDA) is proposed and it has been observed from experimental analysis that the hybridization of DE and EDA algorithms outperforms the DE and the EDA. However, its performance is not good in all scenarios. Breast et al. [20] introduced a new variant of DE which uses $F$ and $CR$ at the individual level. This variant is called self-adaptive DE (JDE). The main drawback of this approach is that the population size is not adaptive and this approach is also not tested for different population sizes. In addition, SADE [17] is a different variant of DE that incorporates a pool of distinct trial vector generation strategies and a pool of values for $F$ and $CR$. In SADE, trial vector generation strategies and associated control parameter values are gradually self-adapted. This self-adaptation takes place through learning from previous experiences and thus renders more computational cost. In [29], normal and Cauchy distributions are implemented to create the $F$ and $CR$. However, its performance for multi-objective evolutionary optimization seems debatable.

In [30], a neighbourhood-based DE mutation is proposed that is equipped with a self-adaptive weight factor but this model does not provide sufficient insight regarding the empirical or theoretical approach to select the neighbourhood size. Further, in [31] a scale factor local search DE (SFLSDE) is proposed. This method uses a mematic algorithm with self-adaptation. However, for the large problem the scale factor local search needs modification.

Like SADE [17], EPSDE [32] is a different variant of DE that entails a pool of values for $F$ and $CR$. However, the self-adaptation in EPSDE is not as robust as in classical DE. In addition, an improved DE with a modified orthogonal learning strategy is proposed in [33]. However, its time complexity is more than other algorithms when solving simple problems.

Different strategies for setting the associated parameters are also suggested in the recent past. In [34, 35], how to avoid premature convergence or stagnation is discussed. In [36], an innovative mutation operator is introduced to speed up the convergence. In [37], a trial vector generation strategy and its various parameters have been suggested. Further in [38], it is suggested that for the optimum performance of DE and to prevent premature convergence, $3D < NP < 8D$ should exist. Moreover, $F$ should not be less than the threshold value, where the threshold value is problem-dependent. Further, in [39], it is suggested that the range of $F$ should be from 0.4 to 0.95 and 0.9 is considered a good choice to initiate. However, all these suggestions are mainly problem-dependant.

### 3 The proposed CIDE algorithm

The concept of chromosomes identification technique given in [22] motivated the authors to construct the CIDE technique. In this technique, each instance of the dataset is considered as the chromosomes and different attributes of the instance are assumed to be different segments of the chromosomes. Further, classes are considered as different species and an attempt has been made to classify different classes based on the centromeric location of chromosomes. The method of calculating the centromeric locations is briefly described as:

- Let the complete length of the chromosome be ‘$c$’ and the length of the longer and shorter arm is ‘$l$’, and ‘$s$’ respectively. With the help of these parameters, we can calculate the centromeric locations in terms of difference ‘$d$’, ratio ‘$r$’, and centromeric index ‘$i$’.

Mathematically, ‘$d$’, ‘$r$’, and ‘$i$’ can be represented as:

$$c = l + s \tag{2}$$

$$d = l - s \tag{3}$$

$$r = \frac{l}{s} \tag{4}$$

$$i = \frac{100 \times s}{c} \tag{5}$$

Further, like conventional DE, our proposed CIDE algorithm also entails three steps. These are mutation, cross-over, and selection. In this proposed technique Eq. (3) is used for mutation operator and Eq. (4) and Eq. (5) will be used for crossover operation. In our proposed technique ‘$d$’ is the mutated vector and ‘$r$’ and ‘$i$’ are the trial vector. Further, in the selection operation, the max–min values of ‘$d$’, ‘$r$’, and ‘$i$’ are selected for generating the classification rules.

#### 3.1 The structure and implementation details of CIDE

The algorithmic details of CIDE are shown in Fig. 2. Further, Table 1 is constructed for the explanation. This table incorporates four attributes $A$, $B$, $C$, $D$, and three classes $X$, $Y$, and $Z$. There are five instances in each class. The first and second generation of the proposed CIDE technique is illustrated in sub-Sect. 3.2.1 and 3.2.2 respectively for the sample Table 1.
Step 1 Set the generation number $G = 1$ and initialize a population of $NP$ individuals. This $NP$ individuals include longer arm of the chromosomes ($l_i$) and shorter arm of the chromosomes ($s_i$) $i = 1, 2, ..., NP$.

Step 2 WHILE stopping criterion is not satisfied.

DO

Step 2.1 Mutation

/*Generate a mutated vector $(d_{i,G})$ for each target vector*/
FOR $i = 1$ to $NP$
    Generate mutated vector $d_{i,G}$ corresponding to the target vector via equation(11)
END FOR

Step 2.2 Crossover

/*Generate a trial vector $r_{i,G}$ and $i_{i,G}$ for each target vector*/
FOR $i = 1$ to $NP$
    Generate trial vector $r_{i,G}$ via equation(12)
    Generate trial vector $i_{i,G}$ via equation(13)
END FOR

Step 2.3 Selection

FOR

1. Evaluate the lower and upper bound of mutated vector $d_{i,G}$ for each class.
2. Evaluate the lower and upper bound of trial vector $r_{i,G}$ for each class.
3. Evaluate the lower and upper bound of trial vector $i_{i,G}$ for each class.
END FOR

Construct the appropriate classification rules with the help of these lower and upper bounds.

Step 2.4 Increment the generation count $G = G + 1$ and repeat the step 2.

Step 3 END WHILE

Fig. 2 Algorithmic description of CIDE

Table 1 Sample table

| Class | Attributes |
|-------|------------|
|       | A  | B  | C  | D  |
| X     | 4.6| 3.4| 1.4| 0.3|
| X     | 5.0| 4.9| 1.5| 0.2|
| X     | 5.8| 4.0| 1.2| 0.2|
| X     | 5.1| 3.7| 1.5| 0.4|
| X     | 4.3| 3.0| 1.1| 0.1|
| Y     | 6.5| 2.8| 4.6| 1.5|
| Y     | 4.9| 2.4| 3.3| 1.9|
| Y     | 5.1| 2.7| 3.9| 1.8|
| Y     | 5.0| 2.0| 3.5| 1.0|
| Y     | 6.3| 3.3| 6.0| 2.5|
| Z     | 5.6| 2.8| 4.9| 2.0|
| Z     | 6.2| 2.8| 4.8| 1.8|
| Z     | 5.9| 3.0| 5.1| 2.5|
| Z     | 5.8| 2.7| 1.5| 1.9|
| Z     | 6.9| 3.2| 5.7| 2.3|

Table 2 Different values of $d, r, i$ for sample Table 1

| Class | First generation of CIDE | Second generation of CIDE |
|-------|--------------------------|---------------------------|
|       | $d_1$ | $r_1$ | $i_1$ | $d_2$ | $r_2$ | $i_2$ |
| X     | 4.3  | 5.13 | 6.12 | 2.00 | 2.42 | 29.16 |
| X     | 4.8  | 25.00 | 3.84 | 1.40 | 1.93 | 34.09 |
| Y     | 5.6  | 29.00 | 3.33 | 2.80 | 3.33 | 23.07 |
| Y     | 4.7  | 12.75 | 7.27 | 2.20 | 2.46 | 28.84 |
| Y     | 4.2  | 43.00 | 2.27 | 1.90 | 2.72 | 26.82 |
| Y     | 5.0  | 4.33 | 18.75 | 1.80 | 1.64 | 37.83 |
| Z     | 3.6  | 2.57 | 27.94 | 0.90 | 1.37 | 42.10 |
| Z     | 3.3  | 2.83 | 26.08 | 1.20 | 1.44 | 40.90 |
| Z     | 4.0  | 5.00 | 16.66 | 1.50 | 1.75 | 36.36 |
| Z     | 3.8  | 2.52 | 28.40 | 2.70 | 1.81 | 35.48 |
| Z     | 3.6  | 2.80 | 23.25 | 2.10 | 1.75 | 36.36 |
| Z     | 4.4  | 3.44 | 22.50 | 2.00 | 1.71 | 36.84 |
| Z     | 3.4  | 2.36 | 29.76 | 2.10 | 1.70 | 37.03 |
| Z     | 4.3  | 3.86 | 20.54 | 0.80 | 1.42 | 41.30 |
| Z     | 4.6  | 3.00 | 25.00 | 2.50 | 1.78 | 35.95 |

Bold values represent the maximum and minimum values of $d$, $r$, and $i$ for first and second generation in different classes.

3.1.1 First generation of CIDE

Step 1: Mutation This is the first step of CIDE after initialization. This operation generates a mutant vector. In this operation, the maximum and minimum attribute values are considered for each instance. As per our supposition, the max and min attribute values represent the longer and shorter arm of the chromosomes. As our mutation operator is based on Eq. (3), we can represent it as $DE/best \to worst/|orDE/(l - s)|/1$, where, ‘$l$’ and ‘$s$’ represent the longer and shorter arm of the chromosome as already stated. The values of ‘$l$’ and ‘$s$’ are selected from the range $[1, D]$, where $D$ is the dimension of the instances of the dataset. The mutated vector for the first instance given in Table 1 is calculated as:

$$d_1 = l_1 - s_1$$

The number written in subscript shows the number of generation

$$= 4.6 - 0.3$$

(i.e., Difference of maximum and minimum attributes values)

$$= 4.3$$

Likewise, another mutated vector can be calculated for other instances. All mutated vectors are shown in Table 2.

Step 2: cross-over After mutation, CIDE performs cross-over operation like conventional DE. In this step, we will find the values of ‘$r$’ and ‘$i$’ for different instances of the dataset by implementing the Eq. (4) and Eq. (5). This step produces the trial vectors ‘$r$’ and ‘$i$’. For example, the trial vector ‘$r$’ for the first instance of Table 1 can be calculated as:

$$r_1 = \frac{l_1}{s_1}$$

$$= \frac{4.6}{0.3}$$

$$= 15.33$$
The trial vectors ‘r’, and the graph between ‘d’ and ‘r’ for different instances of sample Table 1 are shown in Table 2 and Fig. 3 respectively. We can calculate the trial vector ‘i’ with the help of Eqs. (2) and (5). The chromosome’s length of the first instance given in Table 1 can be calculated as:
\[ c_1 = l_1 + s_1 = 4.6 + 0.3 = 4.9 \]
Further
\[ i_1 = \frac{100s_1}{c_1} = \frac{100 \times 0.3}{4.9} = 6.12 \]

Similarly, the trial vectors of other instances can be worked out. These trial vectors are shown in Table 2.

**Step3: selection** This operation is required to choose the better individual. In the proposed CIDE algorithm, these better individuals are the min and max values of ‘d’, ‘r’, and ‘i’ for different classes of the dataset. The min and max values of ‘d’, ‘r’, and ‘i’ for different classes obtained after the first generation are shown in bold in Table 2 and its diagrammatic representation is given in Fig. 4. We can easily generate different classification rules by observing the non-overlapping portion of the lines representing different classes in Fig. 4.

(a) Based on mutated vector \( d_1 \):
1. \( (d_1 < 3.4) \) → Y.
2. \( (d_1 > 5.0) \) → X.

(b) Based on trial vector \( r_1 \):
1. \( (r_1 > 5.0) \) → X.
2. \( (3.86 < r_1 \leq 5.0) \) → Y.
3. \( (r_1 < 2.52) \) → Z.

(c) Based on trial vector \( i_1 \):
1. \( (i_1 < 16.66) \) → X.
2. \( (16.66 \leq i_1 < 20.54) \) → Y.
3. \( (i_1 > 28.40) \) → Z.

These rules are sufficient to classify many instances of the sample Table 1. However, instance number 10, 11, 12, 14 and 15 are still unclassified. Therefore, the second generation of the CIDE algorithm is required.

### 3.1.2 Second generation of CIDE

**Step 1: mutation** In the second generation of the CIDE algorithm, the second longer value of the attribute is taken as ‘l’ and the second smaller attribute is taken as ‘s’ e.g., the value of ‘l’ and ‘s’ will be 3.4 and 1.4 respectively for the first instance of sample Table 1 in the second generation. Therefore,
\[ d_2 = l_2 - s_2 = 3.4 - 1.4 = 2.0 \]

**Step 2: cross-over** Similarly, ‘r’ and ‘i’ can be calculated as:
\[ r_2 = \frac{l_2}{s_2} = \frac{3.4}{1.4} = 2.42 \]
And
\[ i_2 = \frac{100 \times s_2}{c_2} = \frac{100 \times 1.4}{(3.4 + 1.4)} = 29.16 \]

In the same way, \( d_2, r_2, \) and \( i_2 \) can be computed for other instances for the second generation of the CIDE algorithm.

**Step 3: selection** It is explicit that the classification rules obtained after the first generation are sufficient to classify class X completely and many instances of class Y and Z as well. This is explicit from Fig. 4 because the red line representing class X is not overlapped with the line of class...
Y and Z in Fig. 4b and c. Thus, it is genuine to construct only those classification rules from the second generation which are associated with class Y and Z. The min and max of \(d_2, r_2,\) and \(i_2\) obtained after the second generation is shown bold and italic in Table 2 for different classes and is diagrammatically represented in Fig. 5. Further, we can easily construct the following classification rules associated with classes Y and Z with the help of a non-overlapped portion of lines shown in Fig. 5.

(a) Based on mutated vector \(d_2:\)
1. \((d_2 < 0.90) \rightarrow Z.\)
2. \((d_2 \geq 2.50) \rightarrow Y.\)

(b) Based on trial vector \(r_2:\)
1. \((r_2 < 1.42) \rightarrow Y.\)
2. \((r_2 \geq 1.78) \rightarrow Y.\)

(c) Based on trial vector \(i_2:\)
1. \((35.48 \leq i_2 < 35.95) \rightarrow Y.\)
2. \((i_2 \geq 41.30) \rightarrow Y.\)

Now, all the rules obtained after the first and second generation can classify classes X and Y completely. Many instances of class Z have also been classified but some instances of class Z are still unclassified. Therefore, a general classification rule can also be generated for class Z. This rule can be written as:

“If any instance does not follow the above-mentioned rules implicate class Z”.

3.2 Classification rules of Iris dataset

Iris dataset is taken from [23] for the explanation of the CIDE technique. Details of this dataset are given in
Table 3. Following classification rules can be generated for this dataset after the first generation with the help of Fig. 6.

(a) Based on mutated vector $d_1$:
1. $(d_1 < 3.82 \text{ord}_1 > 5.6) \rightarrow \text{Virginica}.$

(b) Based on trial vector $r_1$:
1. $(r_1 > 6.0) \rightarrow \text{Setosa}.$
2. $(4.5 < r_1 < 6.0) \rightarrow \text{Versicolor}.$
3. $(r_1 < 3.27) \rightarrow \text{Virginica}.$

(c) Based on trial vector $i_1$:
1. $(i_1 < 14.28) \rightarrow \text{Setosa}.$
2. $(14.28 \leq i_1 < 18.18) \rightarrow \text{Versicolor}.$
3. $(i_1 > 23.27) \rightarrow \text{Virginica}.$

Similarly, the rules given below can be obtained after the second generation with the help of Fig. 7.

(a) Based on mutated vector $d_2$:
1. $(d_2 < 1.8) \rightarrow \text{Versicolor}.$

(b) Based on trial vector $r_2$:
1. $(r_2 < 1.58) \rightarrow \text{Versicolor}.$
2. $(r_2 > 2.04) \rightarrow \text{Virginica}.$

(c) Based on trial vector $i_2$:
1. $(i_2 < 18.18) \rightarrow \text{Versicolor}.$
2. $(i_2 > 23.27) \rightarrow \text{Virginica}.$

The classification rules obtained after the first generation will classify some instances of each class. Similarly, the classification rules obtained in the second generation will again classify some new instances of different classes. In this way, we can construct the classification rules generation after generation and some new instances will be classified in each generation. Therefore, it is obvious that for a given dataset the cumulative classification accuracy of all generations will render substantially good results. Figures 8 and 9 show the scatter plot for the first and second generation of the CIDE algorithm for the iris dataset.

![Figure 6](image-url)

**Fig. 6** Rule extraction from iris data set after the first generation of CIDE algorithm (a) rule extraction on the basis of $d$ (b) rule extraction on the basis of $r$ (c) rule extraction on the basis of $i$
Fig. 7 Rule extraction from iris dataset after the second generation of CIDE algorithm. (a) rule extraction on the basis of d (b) rule extraction on the basis of r (c) rule extraction on the basis of i.

Fig. 8 Scatter plot for the distribution of mutated vector d, trial vector r, and trial vector i after the first generation of CIDE for iris dataset.

Fig. 9 Scatter plot for the distribution of mutated vector d, trial vector r, and trial vector i after the second generation of CIDE for iris dataset.
3.3 Convergence analysis of CIDE

Many researchers have been investigated the limited behaviour of DE’s population. The main issue of their research was to find under which assumptions it can be guaranteed that DE or its variants can reach an optimal solution [40]. The convergence in probability can be used for the convergence analysis of different DE variants.

Our proposed DE variant i.e., CIDE can be considered as an optimization problem in which we have to optimize the min i.e., lower bound (LB) and max i.e., upper bound (UB) of mutated vector ‘d’, trial vector ‘r’, and ‘i’. Mathematically, this can be described as:

\[ LB = \{ f(x_1, x_2, x_3) : x_1 \in A_d, x_2 \in A_r, x_3 \in A_i \} \]

where \( A_d \) is the set of mutated vectors, \( A_r \) and \( A_i \) be the set of the trial vector. \( G = 1, 2, \ldots, NP \), is the number of generations. Similarly, we have to optimize the given problem.

\[ UB = \{ f(x'_1, x'_2, x'_3) : x'_1 \in A_d, x'_2 \in A_r, x'_3 \in A_i \} \]

Here \( f(\ldots) \) is the objective function.

Let \( S \) be a measurable space. We can consider \( S \) similar to a dataset whose different attributes or features are \( A_d, A_r, \) and \( A_i \). Further, different instances can be obtained by calculating \( A_d, A_r, \) and \( A_i \) for \( G = 1, 2, \ldots, NP \). Therefore, we can say that \( \{ (x_1, x_2, x_3), (x'_1, x'_2, x'_3) \} \in S \) and \( S \) is bounded. The optimal solution set which consists of \( LB \) and \( UB \) of \( A_d, A_r, \) and \( A_i \) is denoted as \( S^* \).

\[ S^* = \{ (x_1, x'_1), (x_2, x'_2), (x_3, x'_3) \} \]

\[ = x^* \]

Here, \( x^* \) is the collection of \( LB \) and \( UB \) of \( A_d, A_r, \) and \( A_i \).

Let \( \mu(\cdot) \) be a function that implicates the measure of space \( S \). Perhaps \( \mu(S') = 0 \), which reflects the fact that set \( S' \) is with measure zero. Precisely, we can say that this will be the situation when \( LB = UB \) for mutated vector \( A_d \), trial vector \( A_r \), and \( A_i \) for every class of the dataset. Therefore, keeping the accuracy of the practical problem in view, we can assume an expanded set \( S_0 \) such that \( (UB - LB) > \delta \), where \( \delta \) is a positive value. We can work out the appropriate value of \( \delta \) so that the accuracy of classification rules can be increased and render the condition \( \mu(S') > 0 \).

The asymptotic convergence of the algorithm can be defined in several ways for analysis. An attempt has been made to define the convergence in terms of probability below.

We will use this definition for the analysis of the CIDE algorithm.

### Definition [41]

Let \( \{X(NP), NP = 0, 1, 2, \ldots\} \) be a series of populations created by CIDE for solving the optimization problem \( \max\{f(x), x \in S\} \). Then the condition for its convergence to global optimum is given by:

\[ \lim_{NP \to \infty} p\{X(NP) \cap S_0^* \neq \emptyset\} = 1 \]  
\[ (6) \]

i.e., the probability of population created, and expanded set \( S_0^* \) such that \( UB - LB > \delta \) will increase as the number of generations increases.

Let us give a sufficient condition for the global convergence of CIDE.

### Theorem [41]

Consider using CIDE to solve the optimization problem \( \max\{f(x), x \in S\} \). Indeed, in the \( NP_k \)th target population \( X(NP_k) \), \( \exists \) individual \( x \) (at least one). Further, with the help of the crossover operator, the individual \( x \) will map to the trial individual \( y \), and thus its probability of occurrence will be greater than zero. Mathematically, we can say:

\[ p\{y \in S_0^*\} \geq \zeta(NP_k) > 0 \]  
\[ (7) \]

i.e., as the series \( \sum_{k=1}^{\infty} \zeta(NP_k) \) diverges; the CIDE converges to the optimal solution set \( S_0^* \).

Where \( \{NP_k, k = 1, 2, \ldots\} \) is the natural number, \( p\{y \in S_0^*\} \) is the probability of \( y \in S_0^* \), where \( S_0^* \) is the optimal solution set. The \( \zeta(NP_k) \), is a small positive value that can alter as \( NP_k \).

Indeed, after every generation proposed algorithm generates artificial data which is the measurable space \( S \) as already discussed. We can say that \( S = X(NP) \) as given in Eq. (6). Further, it has already been declared that \( S_0^* \) is a set that contains \( LB \) and \( UB \) of \( A_d, A_r, A_i \).

Let \( S_1 \) be the measurable space and \( (S_0^*)_1 \) be \( LB \) and \( UB \) of \( A_d, A_r, A_i \) obtained after the first generation of CIDE algorithm respectively. Then it is obvious that \( (S_0^*)_1 \) contained in \( S_1 \), i.e. \( (S_0^*)_1 \subseteq S_1 \). Similarly, it will also be true that \( (S_0^*)_2 \subseteq S_2 \) and so on. Therefore, the condition given in Eq. (6) will always be satisfied. Further, it is obvious that the probability \( p\{y \in S_0^*\} \) will always be equal to one because \( S_0^* \subseteq S \) is true for every generation. Therefore, the sufficient condition of convergence given in Eq. (7) will also be satisfied. Thus, we can infer that the proposed CIDE algorithm converges to the global optimum.

3.4 Asymptotic analysis

The runtime complexity analysis of the DE algorithm is hard because of its non-predictable nature [42]. However, following the work of Zielinski et al. [43], we note that the average runtime of a standard DE algorithm is \( O(NP, D, G_{max}) \), where the highest digit of generation is
$G_{\text{max}}$. Further, in [30] it is given that the runtime complexity of different DE variants like DE/rand to best/1, DEGL, etc. also possesses the average runtime complexity of $O(NP, D, G_{\text{max}})$. Furthermore, the runtime complexity of 1-R is $O(n)$ while there are 1-rules and $n$-features. In addition, the runtime complexity of 1-R is $O(n^2)$ while 1-rule pairs. Perhaps, it can inhibit 1-R as a viable tool.

In the proposed CIDE algorithm, finding the max and min quantity of each attribute considering all instances will have a runtime complexity of $O(D)$; where $D$ is the number of attributes of the dataset. Furthermore, the total runtime complexity of the NP population will be $O(NP, D)$; therefore, for $G_{\text{max}}$ i.e., the highest digit of generation, net runtime complexity of CIDE will also be $O(NP, D, G_{\text{max}})$.

### 3.5 Limitations of CIDE

It is explicit from the previous discussion that the proposed CIDE technique performs its computation based on maximum and minimum attribute values. However, this technique has its limitations as given below.

- In some datasets, the maximum and minimum attribute values of the first generation are very close to the maximum and minimum attribute values of the second generation. This will render the difference ($d$) obtained during mutation i.e., step 1 for all instances of different classes in the first generation in closed vicinity of the value of ‘$d$’ obtained in the second generation. Similar is the case for trial vector ‘r’ and ‘i’ calculated during the crossover operation i.e., step 2. Finally, in selection operation i.e., step 3 we generate the min and max values of ‘$d$’, ‘r’, and ‘i’ to construct the classification rules. Thus, it is intuitive to visualize that if the maximum and minimum attribute values of the first generation are very close to the maximum and minimum attribute values of the second generation then the efficiency of classification rules constructed during the selection operation will not display the substantial classification accuracy. This fact is also valid if the maximum and minimum attribute values of the second generation are very close to the third generation and so on. This can be considered as the main limitation of the proposed algorithm.

- It is explicit that if the number of attributes in the dataset is less, the number of generations that take place will also be less. Eventually, it will result in a fewer number of classification rules obtained. This can also cause mitigation in classification accuracy.

### 4 Experimental results and discussion

#### 4.1 Datasets used

The performance of the CIDE algorithm has been checked experimentally on six actual datasets given in [23]. These datasets are Iris, Wine, New-Thyroid, Breast, Wisconsin, and Splice Junction. The number of instances and attributes of these datasets are given in Table 3. The number of instances in these datasets is of the increasing order from Iris to Splice Junction. Indeed, these datasets are taken in an attempt to visualize the performance of the proposed algorithm. However, substantially increased datasets are not taken in this experimentation.

#### 4.2 Algorithms

The performance of the proposed CIDE technique is compared with two algorithms. These are 1-NN and 1-R.

Following variants of DE are used to optimize the 1-NN algorithms.

- DE/Rand/1/Bin
- DE/Best/1/Bin
- DE/RandToBest/1/Bin
- DE/Best/2/Bin
- DE/Rand/2/Bin
- DE/RandToBest/2/Bin

In addition to these variants of DE, the authors also included four recent variants of DE to optimize the 1-NN. These are given below.

- Self-adaptive DE (SADE)
- Adaptive DE with optional external archive (JADE)
- DE with neighbourhood-based mutation operator (DEGL)
- DE with Scale factor local search (SFLSDE)

The mutation operators for all these variants of DE have been given in Appendix.

#### 4.3 Parameter specification

Specification of various parameters used is shown in Table 4. These specifications are suggested by the developer of different variants. The authors have considered these specified parameters in every problem. It is pertinent to mention that this work is motivated by the research given in [44]. Therefore, the algorithm and parameters considered are similar to [44].
4.4 Analysis of the results

Cross-validation is frequently used to estimate and validate the model. In this experimentation, the test mode for evaluation is fivefold cross-validation. In fivefold cross-validation, the dataset is randomly partitioned into five disjoint sets. Further, different classifiers are run using one set as a test dataset and the remaining four sets as a training dataset. This phenomenon is repeated five times. Finally, the mean of five results obtained from fivefold cross-validation is calculated which produces a single result.

Further, the classification accuracy of the proposed algorithm is compared with different state-of-art variants of DE given in the Appendix. These variants of DE are used to optimize the nearest neighbour (NN) classifier [30], and 1-R is considered as the basis of performance.

The average classification accuracies obtained after fivefold cross-validation of selected algorithms for six datasets are given in Table 5. The bold result in each column shows the best value obtained.

Our validation results show that for the iris dataset the proposed CIDE algorithm displays maximum testing accuracy i.e., 86.66%. The second best testing accuracy for this dataset is 85.94% which is shown by 1-R. Further, for the wine dataset, we observed different patterns of testing accuracies for different algorithms. For the wine dataset, 1-NN optimized by SADE (Learning Period = 50) renders a maximum testing accuracy of 80.02% while the second-best testing accuracy for the same dataset is obtained by 1-R. 1-R algorithm displays a testing accuracy of 79.96% which is very near to the accuracy obtained by 1-NN optimized by SFLSDE/ReadToBest/1/Bin. Our proposed CIDE algorithm also performs well for the new-thyroid dataset and displays a validation accuracy of 83.72% which is very close to the best result given by 1-NN optimized by JADE i.e., 83.98%. For the breast dataset, the proposed CIDE algorithm again produces the best validation accuracy of 84.38%. However, 1-NN optimized by SADE (Learning Period = 100) algorithm also produces a considerably good result for the breast dataset which is 83.14%. Moreover, for the Wisconsin dataset, the performance of 1-NN optimized by SADE (Learning Period = 100) is comparatively best. 1-NN optimized by SADE (Learning Period = 100) produces an accuracy of 81.07% for Wisconsin dataset. The second-best accuracy for Wisconsin dataset is produced by 1-NN optimized by SFLSDE/RandToBest/1/Bin i.e., 80.32%. The proposed CIDE algorithm displays accuracy of 79.89% for Wisconsin dataset that is very close to the accuracy produced by the proposed CIDE algorithm (i.e., 79.89%). In addition, for Splice dataset, the proposed CIDE technique renders the best validation accuracy of 80.32%. Further, 1-R displays the second best validation accuracy for Splice dataset (i.e., 79.22%), and 1-NN optimized by SFLSDE/RandToBest/1/Bin produces the third-best accuracy (i.e., 79.02%) for this dataset.

The standard deviations (SD) of these algorithms in fivefold cross-validation are displayed in Table 6. The SD of the proposed CIDE technique is lowest for Iris, New Thyroid, and Wisconsin datasets. It is explicit from Table 6 that the SD of the proposed CIDE algorithm is 1.41, 1.94, and 1.87 respectively. For Wine dataset, 1-NN optimized by DEGL (Linear) shows the lowest SD which is 2.46. Similarly, for Breast dataset, the SD of 1-NN optimized by SFLSDE/RandToBest/1/Bin is 1.90 and it is the lowest. Moreover, for Splice dataset 1-NN optimized by DEGL (Exponential) produces the lowest SD of 3.21.

Over and above, the mean SD of the CIDE technique for all six datasets is 2.39 and comparatively, it is also the lowest. The second-lowest mean SD for six datasets is 2.84 that is displayed by 1-NN optimized by SFLSDE/ReadToBest/1/Bin and the third lowest mean SD is shown by 1-NN optimized by DEGL (Linear) (i.e., 2.90). However, the mean SD of 1-NN optimized by SFLSDE/ReadToBest/1/Bin and DEGL (Linear) is almost equal.
**Table 5** Classification accuracies of different DE models

| Serial Number | Algorithms | iris     | wine   | New-Thyroid | Breast | Wisconsin | Splice |
|---------------|------------|----------|--------|-------------|--------|-----------|--------|
| 1             | DE/Rand/1/Bin | 74.31    | 71.78  | 74.14       | 74.08  | 73.26     | 74.13  |
| 2             | DE/Best/1/Bin  | 74.98    | 73.32  | 75.32       | 76.04  | 75.79     | 72.67  |
| 3             | DE/RandToBest/1/Bin | 73.63    | 74.58  | 75.63       | 75.38  | 73.62     | 73.24  |
| 4             | DE/Best/2/Bin  | 75.76    | 72.46  | 74.41       | 73.15  | 74.27     | 72.22  |
| 5             | DE/Rand/2/Bin  | 74.93    | 76.01  | 71.93       | 74.37  | 76.13     | 75.41  |
| 6             | DE/RandToBest/2/Bin | 75.37    | 74.51  | 77.38       | 80.03  | 78.69     | 74.52  |
| 7             | SADE (Learning Period = 50) | 78.04    | 80.02  | 79.68       | 80.08  | 79.41     | 77.04  |
| 8             | SADE (Learning Period = 100) | 77.13    | 74.13  | 80.21       | 83.14  | 81.07     | 76.02  |

**Bold values display the maximum testing accuracies and best accuracies of standard deviations obtained for different datasets by different algorithms respectively.**

**Table 6** Accuracy of standard deviation for different dataset

| Serial Number | Algorithms | Iris     | Wine   | New-Thyroid | Breast | Wisconsin | Splice | Average |
|---------------|------------|----------|--------|-------------|--------|-----------|--------|---------|
| 1             | DE/Rand/1/Bin | 3.17     | 3.84   | 4.79        | 4.11   | 5.81      | 6.01   | 4.62    |
| 2             | DE/Best/1/Bin  | 4.17     | 3.76   | 4.24        | 5.10   | 3.22      | 4.82   | 4.21    |
| 3             | DE/RandToBest/1/Bin | 2.18     | 3.97   | 4.36        | 5.21   | 4.88      | 3.98   | 4.09    |
| 4             | DE/Best/2/Bin  | 3.25     | 3.10   | 4.56        | 5.03   | 4.78      | 5.11   | 4.30    |
| 5             | DE/Rand/2/Bin  | 6.13     | 4.11   | 3.73        | 4.06   | 2.91      | 4.02   | 4.16    |
| 6             | DE/RandToBest/2/Bin | 4.02     | 3.92   | 4.13        | 2.98   | 3.07      | 3.26   | 3.56    |
| 7             | SADE (Learning Period = 50) | 3.81     | 2.97   | 4.03        | 3.62   | 3.74      | 3.83   | 3.66    |
| 8             | SADE (Learning Period = 100) | 3.26     | 3.81   | 4.03        | 2.98   | 3.24      | 3.64   | 3.49    |
| 9             | JADE        | 4.12     | 3.81   | 5.11        | 2.89   | 3.04      | 4.02   | 3.83    |
| 10            | DEGL (Exponential) | 3.04     | 2.77   | 3.41        | 4.03   | 3.74      | 3.21   | 3.36    |
| 11            | DEGL (Adaptive) | 2.11     | 3.04   | 2.84        | 2.96   | 3.68      | 3.53   | 3.02    |
| 12            | DEGL (Random) | 3.02     | 4.01   | 2.79        | 4.11   | 3.24      | 4.21   | 3.56    |
| 13            | DEGL (Linear) | 2.89     | 2.46   | 2.04        | 3.68   | 2.86      | 3.47   | 2.90    |
| 14            | SFLSDE/Rand/1/Bin | 4.18     | 3.21   | 3.73        | 2.17   | 2.42      | 4.28   | 3.33    |
| 15            | SFLSDE/RandToBest/1/Bin | 3.47     | 2.81   | 3.03        | 1.90   | 2.14      | 3.72   | 2.84    |
| 16            | 1-R         | 2.98     | 3.13   | 2.32        | 3.88   | 4.88      | 5.32   | 3.75    |

**Bold values display the maximum testing accuracies and best accuracies of standard deviations obtained for different datasets by different algorithms respectively.**
The authors also conducted the Friedman test to validate the proposed CIDE algorithm. The outcome of this test is briefly presented in sub-Sect. 4.4.1.

4.4.1 Friedman test

In [45], it has been mentioned that statistics is of paramount importance to validate the innovative method. Therefore, the Friedman test is conducted for this purpose. The ranking of the algorithms for different datasets is shown in Table 7.

This test compares \( R_j = \frac{1}{N} \sum_i r_{ij} \) where \( R_j \) is the average rank of algorithms. If we have \( k \)-algorithms and \( N \) datasets then the \( r_{ij} \) represents the rank of \( j \)th of \( k \) algorithms on the \( i \)th of \( N \) datasets. It is important to mention that as per the null hypothesis algorithms are entirely equivalent. It implicates that ranks \( R_j \) are equal. The mean rank \( \bar{R}_j = 8.99 \). The Friedman statistic can be calculated using \( \chi^2_F = \left( \frac{12N}{k(k+1)} \right) \sum_j R_j^2 - \frac{k(k+1)^2}{4} \). Further, \( \chi^2_F \) with \( (k-1) \) is considered for distribution. It is important to mention that \( (k - 1) \) is the ‘degree of freedom (DF)’. It has been observed that \( \chi^2_F = 77.04 \). In addition, in [46], a better statistic \( F_F = \frac{(N-1) \chi^2_F}{N(k-1)-\chi^2_F} \) is used and in this statistic the \( F \)-distribution is considered. Moreover, in this distribution \( (k-1) \) and \( (k-1)(N-1) \) is the number of DF. The \( F_F = 20.31 \) and \( F_F \) is distributed as per \( F \)-distribution. The number of DF calculated is \( 17 - 1 = 16 \) and \( (17 - 1)(6 - 1) = 80 \). Over and above, the ‘critical value’ of \( F(16, 80) \) at the ‘level of confidence’ \( \alpha = 0.05 \) is 2.08 which suggests the rejection of the null hypothesis. Detailed information regarding the implementation of the Friedman test is given in [47].

5 Experimental results and analysis for KDDCup’99 dataset

Network security is of paramount importance to keep confidential data and information safe from unauthorized third-party access. Network Intrusion Detection System (NIDS) is a succinct domain of research. Perhaps, it is because this tackles miscellaneous possibilities and issues of the real-time applications for network security. The NIDS detects the intrusion and sends the information to the authority.

The role of the swarm and evolutionary algorithms in the domain of NIDS is thoroughly discussed in [48].

The performance of the proposed CIDE algorithm has been checked experimentally for the intrusion detection dataset i.e., KDDCup’99 [49]. There are 4,898,430 labelled and 311,029 unlabeled connection records in this dataset. The numbers of attributes in labelled connection records are 41. Different types of attacks for labelled records of the KDDCup’99 dataset are depicted in Table 8.

| Serial Number | Algorithms                          | Iris | Wine | New-Thyroid | Breast | Wisconsin | Splice | Average |
|---------------|------------------------------------|------|------|-------------|--------|-----------|--------|---------|
| 1             | DE/Rand/1/Bin                      | 16   | 17   | 16          | 16     | 17        | 14     | 16      |
| 2             | DE/Best/1/Bin                      | 14   | 15   | 14          | 13     | 14        | 16     | 14.33   |
| 3             | DE/RandToBest/1/Bin                | 17   | 11   | 13          | 14     | 16        | 15     | 14.33   |
| 4             | DE/Best/2/Bin                      | 11   | 16   | 15          | 17     | 15        | 17     | 15.16   |
| 5             | DE/Rand/2/Bin                      | 15   | 9    | 17          | 15     | 13        | 11     | 13.33   |
| 6             | DE/RandToBest/2/Bin                | 12   | 12   | 9           | 7      | 7         | 12     | 9.83    |
| 7             | SADE (Learning Period = 50)        | 5    | 1    | 6           | 6      | 6         | 5      | 4.83    |
| 8             | SAD (Learning Period = 100)        | 7    | 13   | 5           | 2      | 1         | 7      | 5.83    |
| 9             | JADE                               | 9    | 6    | 1           | 10     | 9         | 6      | 6.83    |
| 10            | DEGL (Exponential)                 | 6    | 4    | 11          | 8      | 8         | 9      | 7.66    |
| 11            | DEGL (Adaptive)                    | 13   | 8    | 7           | 12     | 10        | 10     | 10.00   |
| 12            | DEGL (Random)                      | 8    | 14   | 10          | 9      | 11        | 13     | 10.83   |
| 13            | DEGL (Linear)                      | 10   | 7    | 12          | 11     | 12        | 8      | 10.00   |
| 14            | SFLSDE/Rand/1/Bin                  | 4    | 10   | 8           | 3      | 3         | 4      | 5.33    |
| 15            | SFLSDE/RandToBest/1/Bin            | 3    | 3    | 4           | 5      | 2         | 3      | 3.33    |
| 16            | CIDE                               | 1    | 5    | 2           | 1      | 4         | 1      | 2.33    |
Table 8 Details of attacks of labeled records

| Category of attack | Name of attack          |
|--------------------|-------------------------|
| Normal             | Normal                  |
| DoS                | Neptune, Smurf, Pod, Teardrop, Land, back |
| Probe              | Portsweep, Ipsweep, Nmap, satan |
| R2L                | Guesspassword, Ftpwrite, Imap, Phf, Multihop, Warezmaster, Warezclient |
| U2R                | Bufferoverflow, LoadModule, Perl, Rootkit |

Table 9 Set of features of KDDCup’99 dataset

| Feature No. | Feature name           | Feature No. | Feature name           | Feature No. | Feature name           |
|-------------|------------------------|-------------|------------------------|-------------|------------------------|
| 1           | Duration (c)           | 11          | num_failed_logins (c)  | 21          | is_host_login (s)      |
| 2           | Protocol_type (s)      | 12          | logged_in (s)          | 22          | is_guest_login (s)     |
| 3           | Service (s)            | 13          | num_compromised (c)    | 23          | count (c)              |
| 4           | Flag (s)               | 14          | root_shell (c)         | 24          | srv_count (c)          |
| 5           | src_bytes (c)          | 15          | su_attempted (c)       | 25          | serror_rate (c)        |
| 6           | dst_bytes (c)          | 16          | num_root (c)           | 26          | srv_serror_rate (c)    |
| 7           | Land (s)               | 17          | num_file_creations (c) | 27          | rerror_rate (c)        |
| 8           | Wrong_fragment (c)     | 18          | num_shells (c)         | 28          | srv_diff_srv_rate (c)  |
| 9           | Urgent (c)             | 19          | num_access_files (c)   | 29          | same_srv_rate (c)      |
| 10          | Hot (c)                | 20          | num_outbound_cmds (c)  | 30          | diff_srv_rate (c)      |
| 41          | dst_host_serror_rate (c) |          |                         | 40          | dst_host_rerror_rate (c) |

Table 10 Distribution of connection types in KDDCup’99 10% training

| Class      | Number of records | Occurrence in %age |
|------------|-------------------|--------------------|
| Normal     | 97,277            | 19.69              |
| DoS        | 391,458           | 79.24              |
| Probe      | 4107              | 0.83               |
| R2L        | 1126              | 0.23               |
| U2R        | 52                | 0.01               |
| Total      | 494,020           | 100.00             |

Table 9 entails the complete list of the set of features of this dataset. It is important to mention that in Table 9 ‘c’ and ‘s’ stand for continuous and symbolic respectively. Further, this dataset incorporates one type of normal data and 22 different attack types categorized into four classes. These four classes are given below.

- **Denial of Service (DoS):** In this attack, the attacker attempts to stop legitimate users from using a service.
- **Probe:** In this attack, the attacker strives to elicit information about the target host.
- **User-to-Root (U2R):** In this attack, the attacker has local access to the victim machine and attempts to obtain the privileges of the superuser.
- **Remote-to-Login (R2L):** In this attack, an attacker attempts to get access to the victim’s machine even he does not have an account on the victim machine.

In this experimentation, 10% of the entire KDDCup’99 labelled dataset is taken for experimentation purposes. It includes 4, 94,020 records and 41 features. The distribution of connections types for this dataset is shown in Table 10. The most substantial drawback of the KDDCup’99 dataset is the number of redundant instances. The presence of redundant instances renders the training of the learning algorithm difficult. The overall effect of these redundant instances makes the learning algorithm biased towards the
frequent instances and restricts the learning from infrequent instances. Undoubtedly, this phenomenon is harmful to the network’s intrusion detection. Therefore, to cope with this problem duplicate instances are removed and selected a random sample of 10\% normal data and 10\% Neptune attack in the DoS class. This comprises 8783 normal instances, 7935 DoS instances, 2131 Probe instances, 52 U2R instances, and 999 R2L instances. Four new sets of data are generated with the normal class and four categories of attack as mentioned earlier (DoS, Probe, U2R, and R2L). It is important to mention that in each dataset, instances with the same attack category and 10\% normal instances are included, where each dataset has its distribution of categories of instances [49].

The performance metrics like accuracy, false alarm rate, and precision are registered for 1-NN optimized by different DE variants, 1-R, and the proposed CIDE algorithm. The formulae used to calculate these metrics are given in Eqs. (8), (9), and (10).

\[
\text{Accuracy} = \frac{TP + TN}{TP + TP + FN + TN} \tag{8}
\]

\[
\text{False alarm rate} = \frac{FP}{TN + FP} \tag{9}
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{10}
\]

Table 11 shows the classification accuracies (A), false alarm rate (B), and precision (C) of DoS + 10\% Normal and Probe + 10\% Normal datasets. For DoS + 10\% Normal dataset and Probe + 10\% Normal, the classification accuracy and precision are highest for the proposed CIDE algorithm. For the DoS + 10\% Normal dataset, the classification accuracy and precision are 83.68\% and 84.56\% respectively and for the Probe + 10\% Normal dataset, the classification accuracy, and precision are 85.57\% and 86.87\% respectively. The false alarm rate of the proposed CIDE algorithm is also lowest for DoS + 10\% Normal and Probe + 10\% Normal datasets. This is 0.0387 and 0.0343 respectively for these datasets. The second-best classification accuracy for the DoS + 10\% Normal dataset is displayed by 1-NN optimized with SADE (Learning Period = 100). It is 82.24\%. Further, for the Probe + 10\% Normal dataset, 1-R shows the second-best classification accuracy of 84.84\%. Precision and false alarm rate of 1-NN optimized by CIDE algorithm is also second-best for the DoS + 10\% Normal dataset whereas its false alarm rate is not good for the Probe + 10\% Normal dataset. It is 0.0422.

Table 12 shows the classification accuracies (A), false alarm rate (B), and precision (C) of R2L + 10\% Normal
Table 12 Classification accuracies (A), False alarm rate (B) and Precision (C) of R2L + 10% Normal and U2R + 10% Normal datasets

| Serial Number | Algorithms                          | R2L + 10% Normal | U2R + 10% Normal |
|---------------|-------------------------------------|------------------|------------------|
|               |                                     | A    |    | B    |    | C    |    |
| 1             | DE/Rand/1/Bin                        | 70.68| 0.0501 | 72.15 | 73.14 | 0.0502 | 75.02 |
| 2             | DE/Best/1/Bin                        | 72.32| 0.0572 | 71.97 | 74.32 | 0.0532 | 75.81 |
| 3             | DE/RandToBest/1/Bin                  | 73.48| 0.0504 | 74.86 | 74.53 | 0.0503 | 76.21 |
| 4             | DE/Best/2/Bin                        | 71.36| 0.0506 | 72.32 | 73.31 | 0.0497 | 75.03 |
| 5             | DE/Rand/2/Bin                        | 75.01| 0.0473 | 76.87 | 71.43 | 0.0501 | 73.02 |
| 6             | DE/RandToBest/2/Bin                  | 73.51| 0.0465 | 75.21 | 75.36 | 0.0464 | 74.96 |
| 7             | SADE (Learning Period = 50)           | 79.02| 0.0401 | 80.12 | 78.58 | 0.0476 | 79.73 |
| 8             | SADE (Learning Period = 100)          | 73.03| 0.0489 | 75.03 | 79.21 | 0.0493 | 80.65 |
| 9             | JADE                                 | 75.67| 0.0476 | 76.23 |       |       |       |
| 10            | DEGL (Exponential)                   | 77.83| 0.0487 | 79.01 | 75.19 | 0.0401 | 76.89 |
| 11            | DEGL (Adaptive)                      | 75.12| 0.0443 | 76.75 | 78.10 | 0.0406 | 79.93 |
| 12            | DEGL (Random)                        | 73.07| 0.0501 | 76.02 | 76.04 | 0.0486 | 77.78 |
| 13            | DEGL (Linear)                        | 75.41| 0.0488 | 76.94 | 74.79 | 0.0459 | 76.34 |
| 14            | SFLSDE/Rand/1/Bin                    | 73.71| 0.0403 | 75.23 | 77.73 | 0.0496 | 79.21 |
| 15            | SFLSDE/RandToBest/1/Bin              | 78.17| 0.0411 | 80.12 | 79.41 | 0.0523 | 80.84 |
| 16            | 1-R                                  | 78.86| 0.0402 | 80.23 | 82.32 | 0.0497 | 83.79 |
|               | CIDE                                 | 77.65| 0.0423 | 79.86 | 82.62 | 0.0488 | 84.03 |

Bold values show the best values of classification accuracy and precision for different algorithms.

and U2R + 10% Normal datasets. The 1-NN optimized by SADE (Learning Period = 50) algorithm shows the highest classification accuracy and precision for the R2L + 10% Normal dataset. These are 79.02% and 80.12% respectively. However, the classification accuracy and precision of the proposed CIDE algorithm are also very close to the 1-NN optimized by SADE (Learning Period = 50) algorithm. For the CIDE algorithm, the values of classification accuracy and precision are 77.65% and 79.86% respectively.

In contrast, for the U2R + 10% Normal dataset, the 1-NN optimized by the JADE algorithm renders the highest classification accuracy and precision. The 1-NN optimized by the JADE algorithm shows 82.88% classification accuracy and 84.03% precision for this dataset. However, the classification accuracy and precision of the proposed CIDE algorithm are 82.62% and 84.03% respectively. Classification accuracies, false alarm rate, and precision of these datasets are also illustrated in Fig. 10.

Further, we also compared the performance of the proposed CIDE technique with different existing state-of-art techniques such as Naive Bayes, SVM, Neural Network, and Random Forest in terms of the performance metrics like accuracy, precision, recall, and F1-score. This comparison is shown in Table 13. The values of these performance metrics for the above-mentioned techniques are taken from [50, 51]. The formulae used to calculate the recall and F1-score are given in Eqs. (11) and (12).

\[
Recall = \frac{TP}{TP + FN} \tag{11}
\]

\[
F1 \text{- score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}
\]

The average classification accuracy of CIDE can be obtained from Tables 11 and 12. This will be 82.38% i.e., \((83.68 + 85.57 + 77.65 + 82.62)/4\). Similarly, the average precision of CIDE is 83.28% i.e., \((84.56 + 86.87 + 77.65 + 84.03)/4\).

From Table 13, it is explicit that the accuracy, precision, recall, and F1-score of the proposed CIDE technique is better than the Naive Bayes, SVM, and Neural Network. However, these performance metrics for the proposed CIDE technique are not as good as Random Forest.

6 Conclusions

The proposed CIDE technique renders the processes of prototype selection and prototype generation and is easy to implement. Experimental analysis reveals the fact that the proposed CIDE technique performs better than the 1-NN classifier optimized from different variants of DE, and 1-R for Iris, Breast, and Splice datasets. Its performance is also very near to the 1-NN optimized from JADE for the New Thyroid dataset. The classification accuracy of 1-NN optimized from JADE is 83.98% whereas the proposed
technique renders the accuracy of 83.72%. However, the classification accuracy of CIDE is slightly less than the 1-NN optimized by SADE (Learning Period = 50) for the Wine dataset. Moreover, the classification accuracies, False alarm rate, and Precision of the proposed CIDE algorithm are superior to 1-NN and 1-R for DoS + 10% Normal and U2R + 10% Normal.
Probe + 10% Normal datasets and its performance is also satisfactory for R2L + 10% Normal and U2R + 10% Normal datasets.

Over and above, the performance of the proposed CIDE technique is also compared with Naive Bayes, SVM, Neural Network, and Random Forest for KDDCup’99 dataset and it is observed that its performance is better than the Naive Bayes, SVM, and Neural Network in terms of accuracy, precision, recall, and F1-score. However, the performance of Random Forest is superior for this dataset. The future research scope of this work is pervasive. It can be used within the purview of the medical domain for disease classification and prediction. Further, the implementation of the proposed technique and observing its performance for increased datasets as well as its use in the pursuit of anomaly detection can also be considered as subsequent future research work.

Appendix 1

DE

Mutation operation

This is the first step after initialization. This operation generates a mutant vector. At each generation $G$, DE creates a mutant vector $V_{i,G} = \{v_{i,G}^1, v_{i,G}^2, \ldots, v_{i,G}^D\}$ for each individual $X_{i,G}$. $X_{i,G}$ is called the target vector. The six frequently used operators for mutation are given in Eqs. (1) to (6). Details of these operators are given in [44].

a. $\text{DE/rand/1:}$

$$V_{i,G} = X_{i,G} + F(X_{\text{best},G} - X_{i,G}) + F(X_{r_1^i,G} - X_{r_2^i,G})$$  \hspace{1cm} (15)

d. $\text{DE/best/2:}$

$$V_{i,G} = X_{\text{best},G} + F(X_{r_1^i,G} - X_{r_2^i,G}) + F(X_{r_3^i,G} - X_{r_4^i,G})$$  \hspace{1cm} (16)

e. $\text{DE/rand/2:}$

$$V_{i,G} = X_{r_1^i,G} + F(X_{r_2^i,G} - X_{r_3^i,G}) + F(X_{r_4^i,G} - X_{r_5^i,G})$$  \hspace{1cm} (17)

f. $\text{DE/rand-to-best/2:}$

$$V_{i,G} = X_{i,G} + F(X_{\text{best},G} - X_{i,G}) + F(X_{r_1^i,G} - X_{r_2^i,G})$$  \hspace{1cm} (18)

In the above equations, $r_1^i, r_2^i, r_3^i, r_4^i, r_5^i \in [1, NP]$ and $F$ is scaling factor. $X_{\text{best},G}$ is the vector of highest fitness at generation $G$. $NP$ is the population size.

Cross-over operation

After mutation, DE performs a binomial crossover operation. This step is implemented with the target vector $X_{i,G}$, and its mutant vector $V_{i,G}$, and a trial vector is produced.

$$V_{i,G} = \{u_{i,G}^1, u_{i,G}^2, \ldots, u_{i,G}^D\}$$

The binomial (uniform) crossover is generally used for this purpose. Mathematically, this crossover is given below.

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if } \text{rand}[0,1] \leq CR \text{ or } j = j_{\text{rand}} \\ x_{i,G}^j, & \text{otherwise} \end{cases}$$  \hspace{1cm} (19)

Given that,

- $i=1,2,\ldots, NP$,
- $j=1,2,\ldots, D$,
- $j_{\text{rand}} \in [1,D]$.

Cross over control parameter $CR \in [0,1]$. 

| Algorithms            | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|-----------------------|--------------|---------------|------------|--------------|
| Naive Bayes [50]      | 70           | 70            | 80         | 86 [51]      |
| SVM [50]              | 81           | 82            | 60         | 94 [51]      |
| Neural Network [50]   | 79           | 83            | 62         | 94 [51]      |
| Random Forest [50]    | 88           | 85            | 87         | 86           |
| CIDE (Proposed)       | 82.38        | 83.28         | 82.46      | 94.01        |

Bold values show the best values of accuracy, precision, recall, and F1-score.

| Algorithms            | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|-----------------------|--------------|---------------|------------|--------------|
| Naive Bayes [50]      | 70           | 70            | 80         | 86 [51]      |
| SVM [50]              | 81           | 82            | 60         | 94 [51]      |
| Neural Network [50]   | 79           | 83            | 62         | 94 [51]      |
| Random Forest [50]    | 88           | 85            | 87         | 86           |
| CIDE (Proposed)       | 82.38        | 83.28         | 82.46      | 94.01        |

Bold values show the best values of accuracy, precision, recall, and F1-score.

Table 13 Comparative analysis of proposed CIDE technique with Naive Bayes, SVM, Neural Network, and Random Forest in terms of Accuracy, Precision, Recall, and F1 Measure
$\text{rand}_j(0, 1)$: Uniformly distributed random number between 0 and 1 created for each $j$.

**Selection operation**

This operation is required to choose the better individual from $X_{j,G}$ and $U_{j,G}$. These better individuals are transferred to the next generation. The selection operation is given in Eq. (21).

$$X_{j,G+1} = \begin{cases} U_{j,G} & \text{if } f(U_{j,G}) \leq f(X_{j,G}) \\ X_{j,G} & \text{otherwise} \end{cases}$$  \hspace{1cm} (20)

The above 3 steps are recursively called unless and until a predefined termination strategy is not achieved.

**Appendix 2**

**Self-adaptive DE**

In [17], a new variant of DE known as SADE is proposed which reduces the excessive search time. Four mutation strategies are implemented in this variant. These four mutation strategies are given in [Eqs. (13), (17), (18), (20)] that are called candidate pool.

**Appendix 3**

**Adaptive DE with optional external archive (JADE)**

JADE is proposed in [29]. In this variant, normal and Cauchy distributions are implemented. These distributions are used to create the scaling factor ($F$) and crossover rate ($CR$). This variant of DE uses the recent successful $F$ and $CR$ for getting the information. Further, this information is used for creating the new $F$ and $CR$.

**Appendix 4**

**DE with neighbourhood-based mutation operator (DEGL)**

DEGL variant is introduced in [30]. In this variant neighbourhoods scheme is used. Further, “Local” and “Global” neighbourhoods are incorporated in this variant for mutation. In addition, two kinds of mutation operators are also used. This variant implements a new parameter, known as “scalar weight”. Like other adaptive methods, this method employs a different scheme for adaptation.

**Appendix 5**

**DE with scale factor local search (SFLSDE)**

In [31], SFLSDE is introduced. In this variant, two local searches are implemented. These are scale factor golden section (SFGSS) and scale factor hill-climb (SFHC). In SFLSDE, trial vector $U_{j,G}$ is worked out by the five numbers chosen randomly. Mathematical expressions are given below.

$$F = \begin{cases} \text{SFGSS if } \text{rand}_5 < \tau_3 \\ \text{SFHC if } \tau_3 \leq \text{rand}_5 < \tau_4 \\ F_i & \text{otherwise} \end{cases}$$  \hspace{1cm} (21)

$$CR_i = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_2 < \tau_2 \\ CR_i & \text{otherwise} \end{cases}$$  \hspace{1cm} (22)

In this experiment, we used two mutation strategies for SFLSDE. These are DE/Rand/1/Bin and DE/RandToBest/1/Bin.

**Author contributions** A new bio-inspired variant of differential evolution is developed and applied for intrusion detection and classification of many real datasets.

**Funding** Not Applicable.

**Data availability** Available.

**Code availability** Available.

**Material availability** Available.

**Declarations**

**Conflict of interest** No conflict of interest.

**Ethical approval** No human or animal is involved in experimentation.

**Consent to participate** Not Applicable.

**Consent for publication** The authors are giving full consent for the publication of the manuscript.

**References**

1. Fang, H., Zhou, A., Zhang, H.: Information fusion in offspring generation: a case study in DE and EDA. Swarm Evol. Comput. 42, 99–108 (2018)
2. Abualigah, L., Yousri, D., Elaziz, M.A., Ewees, A.A., Al-qaness, M.A.A., Gandomi, A.H.: Aquila optimizer: a novel metaheuristic optimization algorithm. Comput. Ind. Eng. 157, 107250 (2021)
3. Darwin, C.: The Origin of Species by Means of Natural Selection. Adamant media Corp., Tampa (2001)
4. De Jong, K.: Evolutionary computation: a unified approach. MIT Press, Cambridge (2006)
5. Abualigah, L.: Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. Springer, Cham (2018)
6. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Mach. Learn. 38(3), 257–286 (2000)
7. Fayed, H.A., Hashem, S.R., Atiya, A.F.: Self-generating prototypes for pattern classification. Pattern Recogn. 40(5), 1498–1509 (2007)
8. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (1991)
9. Pandey, S.C., Nandi, G.C.: TSD based framework for mining the induction rules. J. Comput. Sci. 5, 184–195 (2014)
10. Lam, W., Keung, C.K., Liu, D.: Discovering useful concept prototypes for classification based on filtering and abstraction. IEEE Trans. Pattern Anal. Mach. Intell. 14(8), 1075–1090 (2002)
11. Nannin, L., Lumini, A.: Particle swarm optimization for prototype reduction. Neurocomputing 72(4–6), 1092–1097 (2008)
12. Cervantes, A., Galvan, I.M., Isasi, P.: AMPSO: a new particle swarm method for nearest neighborhood classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. 39(5), 1082–1091 (2009)
13. Storn, R., Price, K.: Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces. Tech. Rep.TR-95-012, Berkeley (1995)
14. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11, 341–359 (1997)
15. Liu, J., Lampinen, J.: On setting the control parameter of the differential evolution method. In: Proceedings of the Eighth International Conference on Soft Computing (MENDEL2002), pp. 11–18 (2002)
16. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution—an updated survey. Swarm Evol Comput. 27, 1–30 (2016)
17. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13, 398–417 (2009)
18. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition based differential evolution. IEEE Trans. Evol. Comput. 12(1), 64–79 (2008)
19. Ronkkonen, J., Kukkonen, J.S., Price, K.V.: Real parameter optimization with differential evolution. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC-2005), pp. 506–513. IEEE Press, Piscataway (2005)
20. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameter in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10, 646–657 (2006)
21. Mao, B., Xie, Z., Wang, Y., Handroos, H.: A hybrid strategy of differential evolution and modified particle swarm optimization for the numerical solution of a parallel manipulator, mathematical problems in engineering. Math. Prog. Eng. 2018, 1–9 (2018)
22. Denver study group: A proposed standard system of nomenclature of human mitotic chromosomes. Acta Gery 10, 322–328 (1960)
23. Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html (for information contact ml-repository@ics.uci.edu) (1995)
24. Herrera, F., Lozano, M.: Adaptive genetic operators based on coevolution with fuzzy behaviors. IEEE Trans. Evol. Comput. 15(2), 149–165 (2001)
25. Aic, M.M., Torn, A.: Population set based global optimization algorithms: some modifications and numerical studies. Comput. Oper. Res. 31(10), 1703–1725 (2004)
26. Tu, Z., Lu, Y.: A robust stochastic genetic algorithm (S(G)a) for global numerical optimization. IEEE Trans. Evol. Comput. 8(5), 456–470 (2004)
27. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. 9(6), 448–462 (2005)
28. Sun, J., Zhang, Q., Tsang, E.: DE/EDA: a new evolutionary algorithm for global numerical optimization. Info. SCI. 169, 249–262 (2005)
29. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with the optional external archive. IEEE Trans. Evolut. Comput. 13(5), 945–958 (2009)
30. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a neighbourhood-based mutation operator. IEEE Trans. Evol. Comput. 13(3), 5216–5553 (2009)
31. Neri, F., Tirronen, V.: Scale factor local search in differential evolution. Memet. Comput. 1(2), 153–171 (2009)
32. Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. 11(2), 1679–1696 (2011)
33. Lei, Y.-X., Gou, J., Wang, C., Luo, W., Cai, Y.-Q.: Improved differential evolution with a modified orthogonal learning strategy. IEEE Access 5, 9699–9716 (2017)
34. Price, K., Storn, R., Lampinen, J.: Differential Evolution—A Practical Approach to Global Optimization. Springer, Berlin (2005)
35. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Osmera, P. (ed.) Proceedings of 6th International Mendel Conference on Soft Computing, pp. 76–83 (2002)
36. Fan, H.Y., Lampinen, J.: A trigonometric mutation operator to differential evolution. J. Global Optim. 27(1), 105–129 (2003)
37. Price, K.V.: An introduction to the differential equation. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill, London (1999)
38. Gamperle, R., Muller, S.D., Kounoutsakos, P.: A parameter study for differential evolution. In: Gmurma, A., Mastorakos, N.E. (eds.) Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, pp. 293–298. WSEAS Press, Interlaken (2002)
39. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 506–513. IEEE, Edinburgh (2005)
40. Eiben, A.E., Rudolph, G.: Theory of evolutionary algorithms: a bird’s eye view. Theoret. Comput. Sci. 229(1–2), 3–9 (1999)
41. Hu, Z., Xiong, S., Su, Q., Zhang, X.: Sufficient conditions for global convergence of differential evolution algorithm. J. Appl. Math. 2013, 1–14 (2013)
42. Lampinen, J., Elinka, I.: On stagnation of the differential evolution algorithm. In: Proceedings of 6th International Mendel Conference on Soft Computing, Brno, Czech Republic, Jun, pp. 76–83 (2000)
43. Zielinski, K., Peters, D., Laur, R.: Run time analysis regarding stopping criteria for differential evolution and particle swarm optimization. In: Proceedings of the 1st International Conference on Experiments/Processes/System Modelling/Simulation/Optimization, Athens, Greece (2005)
44. Isaac, T., Salvador, G., Francisco, H.: Differential evolution for optimizing the positioning of prototypes in nearest neighbour classification. Pattern Recogn. 44, 901–916 (2011)
45. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1(1), 3–18 (2011)
46. Iman, R.L., Davenport, J.M.: Approximations of the critical region of the Friedman statistic. Commun. Stat. A9(6), 571–595 (1980)
47. Demsar, J.: Statistical comparison of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
48. Thakkar, A., Lohiya, R.: Role of the swarm and evolutionary algorithms for intrusion detection system: a survey. Swarm Evol. Comput. (2020). https://doi.org/10.1016/j.swevo.2019.100631
49. KDDCup 1999 data. http://kdd.ics.uci.edu/Databases/kddcup99/10percent.gz
50. Sandosh, S., Govindasamy, V., Akila, G.: Enhanced intrusion detection system via agent clustering and classification based on outlier detection. Peer-to-Peer Netw. Appl. 13, 1038–1045 (2020)
51. Divekar, A., Parekh, M., Savla, V., Mishra, R., Shirole, M.: Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In: 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), pp. 1–8. IEEE, Kathmandu (2018). https://doi.org/10.1109/CCCS.2018.8586840

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Md. Nayer is pursuing his Ph.D. from Birla Institute of Technology, Patna Campus, India. He has completed his B.E in computer science & engineering from Maulana Azad College of Engineering & Technology, Patna, India, and M.Tech. in computer science & Engineering from the Jaypee University of Information Technology, Waknaghat, India. He has worked as a Lecturer in the department of computer science & IT in Prince Sattam bin Abdulaziz University, Kingdom of Saudi Arabia. His research area includes Machine Learning, Evolutionary Algorithm, Wireless sensor networks, etc.

Subhash Chandra Pandey completed his Ph.D from Dr. APJ Abdul Kalam Technical University, Lucknow, India, and M.Tech. in computer engineering from Motilal Nehru National Institute of Technology, Allahabad, UP, India. Presently, he is associated with Birla Institute of Technology, Mesra, Ranchi (Patna Campus) as a Professor. He is a member of many professional bodies including “International Association of Engineers – Hong Kong”, “Computer Science Teachers Association – USA” and “International Association of Artificial Intelligence and Law”. Dr. Pandey has also worked with the Ministry of Health, Saudi Arabia, and possesses deep insight into different technological issues related to the healthcare sector. He is regularly publishing research papers in reputed international journals and international conference proceedings. His research interests include the use of soft computing, evolutionary computing, bio-inspired computing, machine intelligence, and data mining in the domain of the medical sector. Moreover, he has published research papers in high-impact journals in the realm of “Philosophical aspects of machine cognition and IoT”. Nowadays, Dr. Pandey is working in the domain of network security and ‘Cloud Computing and IoT’ and how these emerging technologies can pragmatically be used in the healthcare sector.