Chromosomes Identification Based Differential Evolution (CIDE): A New Bio-inspired Variant for Network Intrusion Detection

Md. Nayer
Birla Institute of Technology - Extension Centre Patna
Subhash Chandra Pandey (ubh63@yahoo.co.in)
Birla Institute of Technology - Extension Centre Patna

Research Article

Keywords: Natural Computing, Differential evolution, Chromosomes Identification, Data Classification, Network Intrusion detection

Posted Date: January 19th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-372822/v2

Version of Record: A version of this preprint was published at Cluster Computing on March 18th, 2022.
See the published version at https://doi.org/10.1007/s10586-022-03574-7.

Chromosomes Identification based Differential Evolution (CIDE): A New Bio-inspired Variant for Network Intrusion Detection

[1.]Md. NAYER (First Author)

Research Scholar
Department of Computer Science & Engineering
Birla Institute of Technology, Mesra, Ranchi
(Patna Campus)
Patna, Bihar,
India
[1.]E-mail: mohdnaiyer@gmail.com

[2.]SUBHASH CHANDRA PANDEY (Corresponding Author)

Department of Computer Science & Engineering
Birla Institute of Technology, Mesra, Ranchi
(Patna Campus)
Patna, Bihar,
India
[2.]E-mail: s.pandey@bitmesra.ac.in

***Abstract*— Chromosomes identification is of the paramount importance. The centromeric location of chromosomes is used to classify different species. Plethora of research has already been done in the domain of Bio-inspired computation. However, a Bio-inspired variant of Differential Evolution (DE) is still not designed. In this work attempt has been made for this pursuit and a new variant named as "Chromosomes Identification based Differential Evolution (CIDE)" has been proposed. This technique is used to extract the knowledge from the databases and it constructs effective and reliable decision rules for the purpose of classification. Further, the testing accuracies of proposed CIDE technique has been compared with nearest neighbour classifier (1-NN) optimized by various existing variants of differential evolution and 1-R classifier using six-real datasets, mainly from medical field. The experimental analysis reveals the fact that proposed technique is superior to 1-NN and 1-R classifier. Over and above, Friedman test has also been conducted to validate the superiority of proposed CIDE technique. Moreover, the proposed CIDE algorithm is also tested on KDDCup'99 dataset and different parameters such as classification accuracy, false alarm rate and precision are computed.**

***Keywords* — Natural Computing; Differential evolution; Chromosomes Identification; Data Classification; Network Intrusion detection.**

1. Background

Nature is endowed with diversified phenomena and many of them are related with computing. Natural computing is highly interdisciplinary domain and it mimics nature inspired methodology and techniques. Precisely, it is the amalgamation of natural phenomena and computation. This interdisciplinary domain is widely used in fundamental research and in information sciences [1]. Moreover, different nature inspired techniques are used for performing the specific tasks [2-4] including the information fusion in offspring generation [5]. Further, evolutionary computation (EC) is a peculiar ingredient of natural computing and it is motivated by the Darwinian theory of evolution [6]. It is pertinent to mention that evolutionary algorithm requires parameters specification and these specifications are used in the creation of new generation [7]. In addition, evolutionary algorithms are greatly influenced by the solution of suitable parameter value. Broadly, EC can be classified in three streams. These are [8]:

- Evolutionary strategies
- Evolutionary programming
- Genetic algorithm

In [9, 10], another aspect of EC is proposed which is known as the Differential Evolution (DE). Indeed, the mutation process of DE is different from the genetic algorithm. The mutation operation of DE is an outcome of arithmetic combinations of individuals whereas the mutation operation of genetic algorithm creates minor changes in the positioning of genes. Perhaps, the good convergence and its comprehensiveness renders increased popularity for DE [11]. Recent advances in DE are given in [12]. Like any

evolutionary algorithm, DE also requires parameters specification. Different control parameters used in DE are:

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

These parameters substantially affect the efficacy of DE. However, selection of control parameter (CP) is time consuming and it requires high computational cost. The effect of good parameter selection on the convergence of DE is given in [13-15]. In [16], **t**he adjustment of CP is performed using the problem dependent heuristic rules**.** A hybrid strategy of DE and modified particle swarm optimization for numerical solution of a parallel manipulator is given in [17].

Further, EC is widely used for solving the problems pertaining to the domain of data mining and data reduction processes are very useful task within the purview of data mining. The two main data reduction techniques are given in [18, 19]. These are:

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

In PS process a subset of instance is selected from the original training dataset and literature survey reveals the fact that the PS process can be grouped in to three categories. Namely: condensation [20], edition [21], and hybrid methods [22]. Condensation method is an attempt to remove those instances or examples which do not possess sufficient classification capabilities whereas the task of edition method is to remove the noisy examples from the datasets. Finally, hybrid methods combine the features of both approaches. There are different PS techniques in literature e.g., CNN [23], ENN [24], IBL [20], and DROP family methods [18], and iterative case filtering (ICF) algorithm [25]. Recently, a new PS technique is proposed in [21].

The PG method for data reduction is also known as prototype abstraction methods [22]. PG method can perform a variety of task. It can select the data, modify the data and even more it can perform interpolation and movements of instances. An important PG method is given in [26]. Moreover, integrated concept prototype learner (ICPL) [22], hybrid (HYB) [27], and mixtures of Gaussians (MixtGauss) [28] are also the PG method. The deep insight about the PS and PG methods are elaborated in [29].

In [30-32], it has been mentioned that EC can successfully be used for solving the prototype selection and generation problem. One popular model for PS problem entails the memetic algorithm. It is known as Steady State Memetic Algorithm (SSMA) [33]. There are many evolutionary algorithms for PG problem. Two main evolutionary algorithms based approach to solve the PG problems are Evolutionary Nearest Prototype Classifiers (ENPC) [33] and Prototype Selection Clonal Selection Algorithm (PSCSA) [34].

From literature survey, it has been revealed that DE technique has also been used to solve the PS and PG problems. The first attempt at using DE for PG can be found in [35].

Further, automated chromosomes classification is of the paramount importance and it is an indispensable problem in the domain of pattern recognition. Moreover, various machine learning and artificial intelligence based approaches have been used for this purpose. In [36], different algorithms used for chromosomes classification are discussed. Furthermore, extensive literature survey revealed the fact that hybrid approaches have also been used for the chromosomes classification e.g., hybridization of GA and SVM is given in [37]. Moreover, genetic algorithm based approaches like GABIL [38], GIL [39], and the algorithm given in [40] employ a single chromosome which contains multiple rules to represent the whole solution.

The contributions of this paper are enumerated below:

1. An innovative variant of DE is proposed. This variant is motivated by the evolutionary technique of chromosome identification.
2. The proposed variant of DE requires no parameter tuning and is easy to implement.
3. The proposed variant of DE is suitable for the PS and PG processes. It first performs the PG operation and creates artificial data and subsequently PS operation is performed.

Remaining part of the paper has been structured as follow. DE and previous works related to DE is depicted in Section 2. The phenomenon of chromosomes identification is given in section 3 and the proposed Chromosomes Identification based Differential Evolution (CIDE) is illustrated in section 4. Further, section 5 renders the results and discussions. Finally, the concluding remarks are given in section 6.

2. Methods

This section will render introductory information of DE. Section 2.1 presents the basic information of DE followed by the advancement took place in recent past given in Section 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

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

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

$$X_{i,G} = \left(x_{i,G}^1, x_{i,G}^2, \dots\dots, x_{i,G}^D\right)\ where\ i = 1,2 \dots NP$$

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

$$X_{min} = \{x_{min}^1, x_{min}^2, \dots\dots, x_{min}^D\}$$
$$X_{max} = \{x_{max}^1, x_{max}^2, \dots\dots, x_{max}^D\}$$

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

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

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

DE entails three operations. These are elaborated in forthcoming sections 2.1.1, 2.1.2, and 2.1.3. First two operations are implemented to create the new vectors. These vectors are known as the trial-vectors. Subsequently, selection operation decides survival of the vectors for next generation.

2.1.1 Mutation operation

This is the first step after initialization. This operation generates mutant vector. At each generation *G*, DE creates a mutant vector $V_{i,G} = \{v_{i,G}^1, v_{i,G}^2, \dots\dots, v_{i,G}^D\}$ for each individual $X_{i,G}$. $X_{i,G}$ is called the target vector. The six frequently used operators for the purpose of mutation are given below.

1."DE/rand/1":

$$V_{i,G} = X_{r_1^i} + F(X_{r_{2,G}^i} - X_{r_{3,G}^i}) \quad (2)$$

2."DE/best/1":

$$V_{i,G} = X_{best,G} + F(X_{r_{1,G}^i} - X_{r_{2,G}^i}) \quad (3)$$

3."DE/rand-to- best/1":

$$V_{i,G} = X_{i,G} + F\left(X_{best,G} - X_{i,G}\right) + F(X_{r_{1,G}^i} - X_{r_{2,G}^i}) \quad (4)$$

4."DE/best/2":

$$V_{i,G} = X_{best,G} + F\left(X_{r_{1,G}^i} - X_{r_{2,G}^i}\right) + F(X_{r_{3,G}^i} - X_{r_{4,G}^i}) \quad (5)$$

5. *"DE/rand/2":*

$$V_{i,G} = X_{r_{1,G}^i} + F\left(X_{r_{2,G}^i} - X_{r_{3,G}^i}\right) + F(X_{r_{4,G}^i} - X_{r_{5,G}^i}) \quad (6)$$

6. *"DE/rand- to-best/2":*

$$V_{i,G} = X_{i,G} + F\left(X_{best,G} - X_{i,G}\right) + F\left(X_{r_{1,G}^i} - X_{r_{2,G}^i}\right) + F\left(X_{r_{3,G}^i} - X_{r_{4,G}^i}\right) \quad (7)$$

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. $X_{best,G}$ is the vector of highest fitness at generation *G*.

2.1.2 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.

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

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

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, If\ (rand\ j\ [0,1] \le CR) or\ (j = j_{rand}) \\ x_{i,G}^j, \quad otherwise \end{cases} \quad (8)$$

Given that

i=1,2,...,*NP*
j=1,2,....,*D*
$j_{rand} \in [1,D]$
Cross over control parameter $CR \in [0,1]$
$rand_j(0,1)$: Uniformly distributed random number between 0 and 1 created for each j.

2.1.3 Selection operation

This operation is required to choose the better individual from $X_{i,G}$ and $U_{i,G}$. These better individuals are transferred to next generation. The selection operation is given in equation (9).

$$X_{i,G+1} = \begin{cases} U_{i,G}\ if\ f(U_{i,G}) \le f(X_{i,G}) \\ X_{i,G} \qquad otherwise \end{cases} \quad (9)$$

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

2.2 Recent advances

Different variants of DE have been proposed in recent past. In [41], a new variant of DE called FADE is introduced which uses the concept of fuzzy logic for dynamic adjustment of *F* and *CR*. Breast et al. [16], introduced a new variant of DE

which uses *F* and *CR* at individual level. This variant is called self adaptive DE (JDE). In [42], normal and Cauchy distribution are implemented to create the *F* and *CR*. In addition, SADE [13] and EPSDE [43] are two different variants of DE and these variants incorporate a pool of distinct trial vector generation strategies and a pool of values for *F* and *CR*. Further, in [44], a combination approach of DE is introduced and it has experimentally been observed that this approach augments the efficacy and versatility. Moreover, in [45], 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. In [46], a hybridized approach of fuzzy logic controller and genetic operator is implemented to set the CP. Stochastic genetic algorithm is also proposed which employs stochastic coding strategy [47]. In addition, an improved DE with modified orthogonal learning strategy is proposed in [48].

Different strategies for setting the associated parameters are also suggested in recent past. In [49, 50], how to avoid premature convergence or stagnation is discussed. In [51], an innovative mutation operator is introduced to speed up the convergence. In [52], trial vector generation strategy and its various parameters have been suggested. It is suggested in [53] that for the optimum performance of DE, $3D < NP < 8D$ should exist and *F* is not less than the threshold value where the threshold value is problem dependent. These conditions are required to prevent the premature convergence. Further, in [54], it is suggested that the range of *F* should be from 0.4 to 0.95. However, 0.9 is considered as a good choice to initiate.

3. Chromosome identification

Automated chromosomes identification is an essential task in cytogenetic. The chromosomes identification method is mainly based on geometric and morphological features. Indeed, the locus of centromere plays vital role in chromosomes identification.

The history of human chromosome identification is twenty years old. In mid-sixties, automated analysis for chromosomes identification was initiated in several laboratories [55-57]. However, it was observed that scoring an identification accuracy above than 70-80% is hard. A comprehensive description of nomenclature and method for calculation of centromeric location is proposed in [58, 59]. This method of calculating the centromeric locations is briefly described as:

Let the complete span of chromosome be c and the span of 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 [59]:

$$c = l + s \tag{10}$$

$$d = l - s \tag{11}$$

$$r = \frac{l}{s} \tag{12}$$

$$i = \frac{100 \times s}{c} \tag{13}$$

In our proposed algorithm, equation (11) is used for mutation operator and equations (12) and (13) will be used for crossover operation. Further, in selection operation, the $[d, r, i]$ is selected for rule generation. Fig.1 shows the pattern of centromeric locations in different chromosomes and Fig.2 shows the centromeric position of the chromosomes with respect to d and r.

4. Chromosome identification based DE (CIDE)

The concept of chromosomes identification technique as proposed by Denver in [59] motivated the authors to construct the CIDE technique. In this technique, each instance or example 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 attempt has been made to classify different classes on the basis of the centromeric location of chromosomes.

4.1 The structure of CIDE

The algorithmic details of CIDE are shown in Fig. 3. The symbols used in sub-section 4.2 are already explained in section 3.

4.2 Implementation details of CIDE

Table 1 is constructed for the purpose of explanation. This table incorporates four attributes $A, B, C, D$and three classes $X, Y,$ and Z. There are five examples in each class.

Step 1; Mutation:

The maximum and minimum attribute values are calculated for each instance. As per our assumption, the max and min attribute values represent the longer and shorter arm of the chromosomes. Our mutation operator is based on the equation (11). We can represent our mutation operator as: $DE/best - to - worst/1\ or\ DE/(l - s)/1$, where, l and s represents longer and shorter arm of chromosome. The values of $l\ and\ s$ are selected from the range$[1, D]$, where D is the dimension of the data set. The mutation operator provides the mutated vector e.g., the mutated vector of first example given in Table 1can be calculated as:

$$d_1 = l - s$$
$$= 4.6 - 0.3$$

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

$$= 4.3$$

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

Step 2: Cross-over:

In this step we will find the values of r and i for different instances of the dataset by implementing the equation (12) and (13). This step produces the trial vector r and i. For example, the trial vector r for first instance of Table 1 can be calculated as:

$$r_1 = \frac{l}{s}$$

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

$$= 15.33$$

The trial vector r for different instances of sample table 1 is shown in Table 2. We can calculate the trial vector i with the help of equation e.g. (10) and (13). The chromosome's length of the first example given in table 1 can be calculated as:

$$c = l + s$$
$$= 4.6+0.3$$
$$= 4.9$$

Further

$$i_1 = \frac{100s}{c}$$
$$= \frac{100 \times 0.3}{4.9}$$
$$= 6.12$$

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

Step3: Selection:

This operator generates the min and max values of d, r and i . The lower and upper bounds of d, r and i is very useful for the purpose of generating the induction rules. Different classification rules obtained for Table 1are given below. The min and max values of d, r and i obtained after first generation are shown in Fig.4.

(a) Based on mutated vector d*:*
1. $(d < 3.4) \rightarrow$ *Y.*
2. $(d > 5.0) \rightarrow$ *X.*
(b) Based on trial vector r*:*
1. $(r > 5.0) \rightarrow$ *X.*
2. $(3.86 < r \leq 5.0) \rightarrow$ *Y.*
3. $(r < 2.52) \rightarrow$ *Z.*
(c) Based on trial vector i*:*
1. $(i < 16.66) \rightarrow$ *X.*
2. $(16.66 \leq i < 20.54) \rightarrow$ *Y.*
3. $(i > 28.40) \rightarrow$ *Z.*

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

$$d_2 = l - s$$
$$= 3.4 - 1.4$$
$$= 2.0$$

Similarly r and i can be calculated as:

$$r_2 = \frac{l}{s}$$

$$= \frac{3.4}{1.4}$$

$$= 2.42$$

And

$$i_2 = \frac{100 \times s}{c}$$

$$i_2 = \frac{100 \times 1.4}{(3.4+1.4)}$$

$$= 29.16$$

In the same way, $d, r,$ and i can be computed for other instances for second generation of CIDE algorithm.

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. Thus, it is genuine to construct only those classification rules from second generation which are associated with class *Y* and *Z*. The min and max of $d, r,$ and i obtained after second generation is shown in Fig.5. Further, we can easily construct the following classification rules associated with class *Y* and *Z*.

(a) Based on mutated vector d*:*
1. $(d < 0.90) \rightarrow$ *Z.*
(b) Based on trial vector r*:*
1. $(r < 1.42) \rightarrow$ *Y.*
2. $(1.78 < r < 1.93) \rightarrow$ *Y.*
(c) Based on trial vector i*:*
1. $(35.48 \leq i \leq 35.95) \rightarrow$ *Y.*
2. $(i > 41.30) \rightarrow$ *Y.*

Now, all the rules obtained after the first and second generation can classify class *X* and *Y* completely. Many instances of class *Z* have also been classified but some instances of class *Z* are still unclassified. Therefore, a usual

classification rule can be generated for class *Z*. This rule can be written as:

"If any instance does not follow the above mentioned rules implicate class Z".

4.3 Classification rules of Iris dataset

Iris dataset is taken from [60] for the purpose of explanation of 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. $(d < 3.82 \text{ or } d > 5.6) \rightarrow$ *Virginica.*

(b) Based on trial vector r*:*
1. $(r > 6.0) \rightarrow$ *Setosa.*
2. $(4.5 < r < 6.0) \rightarrow$ *Versicolor.*
3. $(r < 3.27) \rightarrow$ *Virginica.*

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

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

(a) Based on mutated vector d*:*
1. $(d < 1.8) \rightarrow$ *Versicolor.*
2. $(d > 2.4) \rightarrow$ *Virginica.*

(b) Based on trial vector r*:*
1. $(r < 1.58) \rightarrow$ *Versicolor.*
2. $(r > 2.04) \rightarrow$ *Virginica.*

(c) Based on trial vector i*:*
1. $(i < 18.18) \rightarrow$ *Versicolor.*
2. $(i > 23.27) \rightarrow$ *Virginica.*

The classification rules obtained after the first generation will classify some instances of each class. Similarly, the classification rules obtained in 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 generation will render substantially good result. Fig.8 and Fig.9 shows the subsequent scenario of first and second generation of CIDE algorithm for iris dataset.

4.4 Convergence analysis of CIDE

Many researchers have been investigated the limit behaviour of DE's population. The main issue of their research was to find that under which assumptions it can be guaranteed that DE or its variants can reach an optimal solution [61]. 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:

$$LB = \{f(x_1, x_2, x_3) : x_1 \in A_{d^G}, x_2 \in A_{r^G}, x_3 \in A_{i^G}\}$$

Where $A_d$is the set of mutated vector, A_r and $A_i$be the set of trial vector. $G = 1,2, \dots, NP$ is the number of generation. Similarly, we have to optimize the given problem.

$$UB = \{f(x'_1, x'_2, x'_3) : x'_1 \in A_{d^G}, x'_2 \in A_{r^G}, x'_3 \in A_{i^G}\}$$

Here $f(.,.,.)$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^G}, A_{r^G}, and A_{i^G}. Further, different instances or examples can be obtained by calculating A_{d^G}, A_{r^G}, and A_{i^G} for$G = 1,2, \dots, NP$. 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 $UB\ of\ A_{d^G}, A_{r^G}$, and A_{i^G} 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^G}, A_{r^G}, and A_{i^G}.

Let $\mu(.)$ be a function which 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^G},trial vector A_{r^G}, and $A_{i^G}$for every class of the dataset. Therefore, keeping the accuracy of practical problem in view, we can assume an expanded set S_δ^* such that$(UB - LB) > \delta$, where δ is positive value. We can work out the appropriate value of δ so that accuracy of classification rules can be increased and render the condition $\mu(S^*) > 0$.

The asymptotic convergence of algorithm can be defined in number of ways for the purpose of analysis. Attempt has been made to define the convergence in terms of probability below. We will use this definition for the analysis of CIDE algorithm.

Definition [62]

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

$$\lim_{t \to \infty} p\{X(t) \cap S_\delta^* \neq \emptyset\} = 1 \qquad (14)$$

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

Theorem [62]

Consider using DE to solve the optimization problem $max\{f(x), x \in S\}$. In the $t_k th$ target population $X(t_k)$, $\exists$ individual x (at least one). Further, with the help of reproduction operator the individual x will map to the trial individual y, such that

$$p\{y \in S_\delta^*\} \geq \zeta(t_k) > 0 \quad (15)$$

And the series $\sum_{k=1}^{\infty} \zeta(t_k)$ diverges; then DE converges to the optimal solution set S_δ^*.

Where $\{t_k, k = 1,2, \dots\}$ is the natural number, $p\{y \in S_\delta^*\}$ is probability of $y \in S_\delta^*$, where S_δ^* is the optimal solution set. $\zeta(t_k)$ is a small positive value which can alter as t_k.

Our proposed algorithm generates artificial data after the prototype generation. Indeed, after every generation proposed algorithm generates an artificial data which is the measurable space S as already discussed. We can say that $S = X(t)$ as given in eq. (14). Further, it has already been declared that S_δ^* is a set which contains LB and $UBof\ A_{d^G}, A_{r^G}, A_{i^G}$.

Let S_1 be the measurable space and $(S_\delta^*)_1$ be LB and $UB\ of\ A_{d^G}, A_{r^G}, A_{i^G}$ obtained after the first generation of CIDE algorithm respectively. Then it is obvious that $(S_\delta^*)_1$ is contained in S_1 i.e., $(S_\delta^*)_1 \subseteq S_1$. Similarly, it will also be true that $(S_\delta^*)_2 \subseteq S_2$ and so on. Therefore, the condition given in eq. (14) will always be satisfied. Further, it is obvious that the probability $p\{y \in S_\delta^*\}$ will always be equal to one because $S_\delta^* \subseteq S$ is true for every generation. Therefore, the sufficient condition of convergence given in eq. (15) will also be satisfied. Thus, we can infer that the proposed CIDE algorithm converges to global optimum.

4.5 Asymptotic analysis

The runtime complexity analysis of DE algorithm is hard because of its non-predictable nature [63]. However, following the work of Zielinski et al., [64] we note that the average runtime of a standard DE algorithm is $O(NP, D, G_{max})$ where G_{max} is the highest digit of generations. Further, in [65] 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 $O(NP, D, G_{max})$. Furthermore, the runtime complexity of 1-R is $O(n)$ while there is 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 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. Further, total runtime complexity of NP population will be $O(NP, D)$; therefore, for G_{max} i.e., the highest digit of generation, net runtime complexity of CIDE will be $O(NP, D, G_{max})$.

4.6 Limitations of CIDE

It is explicit from previous discussion that the proposed CIDE technique performs its computation based on maximum and minimum attribute values. However, this technique has its own 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 first generation in closed vicinity of the value of d obtained in 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 lower and upper bound of *d,r,* and *i* for the purpose of constructing 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 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 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 then the number of generations take place will also be less. Eventually, it will result in less number of classification rules obtained. This can also cause mitigation in classification accuracy.

5. Experimental Results and Discussion

The performance of CIDE algorithm has been checked experimentally on six actual datasets given in [60]. Properties of these datasets are given in Table 3. The fivefold cross-validation is used for partitioning the datasets and the mean of five results obtained from fivefold cross-validation is calculated which produces a single result.

Further, the classification accuracy of proposed algorithm is compared with different state-of-art variants of DE. Different variants of DE are used to optimize the nearest neighbour (NN) classifier [65] and 1- NN is considered as the basic of performance. Brief descriptions of algorithms used for the purpose of comparison are given in sub-section 5.1.

5.1 Characteristics of algorithms

Following algorithms are used for the purpose of comparison with CIDE.

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

The mutation operators for these variants of DE have been given in Section 2. In addition to these variants of DE, we have also included four recent variants of DE in our experimental analysis. These four most recent DE algorithms are briefly introduced in forthcoming sub-sections.

5.1.1 Self-adaptive DE

In [13], 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. (2), (6), (7), (9)) that are called candidate pool.

5.1.2. Adaptive DE with optional external archive (JADE)

JADE is proposed in [42]. In this variant, normal and Cauchy distribution 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.

5.1.3. DE with neighbourhood-based mutation operator (DEGL)

DEGL variant is introduced in [66]. In this variant neighbourhoods scheme is used. Further, "Local" and "Global" neighbourhood are incorporated in this variant for the purpose of 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.

5.1.4 DE with Scale factor local search (SFLSDE)

In [67], SFLSDE is introduced. In this variant two local search is implemented. These are: scale factor golden section (SFGSS) and scale factor hill-climb (SFHC). In SFLSDE, trial vector $U_{i,G}$ is worked out by the five numbers chosen randomly. Mathematical expressions are given below.

$$F = \begin{cases} SFGSS & if\ rand_s < \tau_3 \\ SFHC & if \tau_3 \le rand_5 < \tau_4 \\ \begin{cases} F_l + F_u.rand_1 & if rand_2 < \tau_1 \\ F_l & Otherwise \end{cases} & if rand_5 < \tau_4 \end{cases} \quad (16)$$

$$CR_i = \begin{cases} rand_3 & if rand_4 < \tau_2 \\ CR_i & Otherwise \end{cases} \quad (17)$$

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

5.2 Parameter specification

Specification of various parameters used is shown in Table 4. These specifications are suggested by the developer of different variants. Authors have considered these specified parameters in every problem. It is pertinent to mention that this work is motivated by the research given in [68]. Therefore, the algorithm and parameters considered are similar to [68].

5.3 Analysis of the Results

The average classification accuracies obtained after five fold 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 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 wine dataset we observed different pattern of testing accuracies for different algorithms. For wine dataset SADE (Learning Period=50) renders a maximum testing accuracy of 80.02% while the second best testing accuracy for same dataset is obtained by 1-R. This algorithm provides a testing accuracy of 79.96% which is very near to the validation accuracy obtained by SFLSDE/ReadToBest/1/Bin. Our proposed CIDE algorithm also performs well for new-thyroid dataset and displays a validation accuracy of 83.72% which is very close to the best result. For breast dataset, proposed algorithm again produces best validation accuracy of 84.38%. However, SADE (Learning Period=100) algorithm also produces a considerably good result for breast dataset which is 83.14%. Moreover, for Wisconsin dataset the performance of SADE (Learning Period=50), SADE (Learning Period=100), SFLSDE/Read/1/Bin, SFLSDE/ReadToBest/1/Bin, and CIDE produce almost similar results but the result produced by SADE (Learning Period=100) is comparatively better.

The standard deviations (SD) of these algorithms in five fold cross-validation are displayed in Table 6. The mean SD of CIDE technique for all six datasets is 2.39 and comparatively it is lowest.

5.3.1 Friedman test

In [69], it has been mentioned that statistics is of the paramount importance to validate the innovative method. Therefore, Friedman test is conducted for this purpose. Ranking of the algorithms for different datasets are shown in Table 7.

This test compares $R_j = \left(\frac{1}{N}\right)\sum_i r_i^j$ where, R_j is the average rank of algorithms. If we have k- algorithms and N datasets then the r_i^jrepresents 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 is equal. The mean rank $R_j = 8.99$. The Friedman statistic can be calculated using $\chi_F^2 = \left(\frac{12N}{k(k+1)}\right)\left[\sum_i R_j^2 - \frac{k(k+1)^2}{4}\right]$. Further, χ_F^2 with $(k-1)$ is considered for the purpose of distribution. It is important to mention that $(k-1)$ is the 'degree of freedom (DF)'. It has been observed that χ_F^2 = 77.04. In addition, in [70], a better statistic$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$ is used and in this statistic the F-distribution is considered. Moreover, in this distribution$(k-1)$and $(k-1)(N-1)$ are the number of DF. The F_F= 20.31and 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 suggest the rejection of null-hypothesis. Detailed information regarding implementation of Friedman test is given in [71].

6. Experimental Results and Analysis for KDDCup'99 Dataset

Network security is of the paramount importance in order to keep confidential data and information safe from unauthorized third party access [72, 73]. Network Intrusion Detection System (NIDS) is a succinct domain of research. Perhaps, it is due to the fact that this tackles miscellaneous possibilities and issues of the real-time applications for the purpose of network security. The NIDS detects the intrusion and send the information to the authority [74, 75]. Role of swarm and evolutionary algorithms are discussed thoroughly in [76].

The performance of CIDE algorithm has also been checked experimentally for the intrusion detection dataset KDDCup'99[77]. There are 4,898,430 labelled and 311,029 unlabeled connection records in the dataset. The numbers of attributes in labelled connection records are 41. Different types of attacks for labelled records of KDDCup'99 dataset are depicted in Table 8. Further, 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.
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, attacker attempts to stop legitimate users from using a service.
- Probe: In this attack, attacker strives to elicit information about the target host.
- User-to-Root (U2R): In this attack, attacker has local access to victim machine and attempts to obtain privileges of super user.
- Remote-to-Login (R2L): In this attack, an attacker attempts to get access of victim's machine even he does not have 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 KDDCup'99 dataset is number of redundant instances. Presence of redundant instances renders the training of 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. This phenomenon is harmful to networks. Therefore, to cope with this problem duplicate instances are removed and selected random sample of 10% normal data and 10% Neptune attack in DoS class. This comprises of 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 data set, instances with the same attack category and 10% normal instances are included, where each dataset has its own distribution of categories of instances [77].

The performance metrics like accuracy, false alarm rate, and precision are registered for different DE variants and CIDE algorithm. The formulae used to calculate these metrics are given below.

Accuracy = (TP + TN) / (TP + FP + TN + FN) (18)
False Alarm Rate = FP / (TN + FP) (19)
Precision = TP / (TP + FP) (20)

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 proposed CIDE algorithm. For Dos+10% normal dataset, the classification accuracy and precision are 83.68% and 84.56% respectively and for Probe+10% Normal dataset, the classification accuracy and precision are 85.57% and 86.87% respectively.

Table 12 shows the classification accuracies (A), false alarm rate (B) and precision (C) of R2L+10% Normal and U2R+10% Normal datasets. The SADE (Learning Period = 50) algorithm shows the highest classification accuracy and precision for R2L+10% Normal dataset. These are 79.02% and 80.12% respectively. However the values of classification accuracy and precision for proposed CIDE algorithm are also very close to SADE (Learning Period = 50) algorithm. For CIDE algorithm the values of classification accuracy and precision are 77.65% and 79.86% respectively.

In contrast, for U2R+10% Normal dataset, the JADE algorithm renders the highest classification accuracy and precision. The JADE algorithm shows 82.88% classification accuracy and 84.03% precision for U2R+10% Normal. However, the classification accuracy and precision of 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.

7. Conclusions

The proposed CIDE technique renders the processes of prototype selection and prototype generation and requires no parameter tuning and thus it is easy to implement. Experimental analysis reveals its superiority with selected classifiers. The performance of proposed CIDE algorithm is also satisfactory for KDDCup'99 dataset. The experimental results render the substantial better classification accuracy and low false alarm rate for IDS. The future research scope of this work is pervasive. It can be used within the purview of medical domain for the purpose of disease classification and prediction. Further, implementation of proposed technique in the pursuit of anomaly detection can also be considered as subsequent future research work.

List of 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

Declarations

Funding: Not Applicable.

Conflicts of interest/Competing interests: No conflict of Interest.

Availability of data and material: Available.

Code availability: Available.

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

Ethics approval: No human or animal is involved in experimentation.

Consent to participate: Not Applicable.

Consent for publication: Author's are giving full consent for the publication of manuscript.

References

[1] Rozenberg G 2008 Computer Science, Informatics and Natural Computing, Personal Reflections in New Computational Paradigms: Changing concepts of what is computable. Springer, pp. 373-379
[2] Deb K, Reddy A R 2003 Reliable classification of two-class cancer data using evolutionary algorithms. Bio-Systems. 72(1-2): 111-129
[3] Sorensen K, Janssens G K 2003 Data mining with genetic algorithms on binary trees. European Journal of Operation Research 151: 253-264
[4] De Falco I, Cioppa A D, and Tarantino E 2002 Mutation-based genetic algorithm: performance evaluation. Applied Soft Computing 1: 285-299
[5] HuiFang, AiminZhou,HuZhang, 2018, Information fusion in offspring generation: A case study in DE and EDA, Swarm and evolutionary computation, elsevier. 42: 99-108
[6] Darwin C 2001 The origin of species by means of natural selection Adamant media corp., original 1859
[7] Marcus M H, Lopes H S, and Delgado M R 2005 Self-adaptive evolutionary parameters: Encoding aspects for combinational optimization problems. In: Lecture Notes in Computer Science, G.R.Raidl and J.Gottlieb, Eds.Laussane, Switzerland: Springer-verlog.3448, Proc.Evol.comput. Combinatorial optimization, pp.155-166
[8] De Jong K 2006 Evolutionary computation: A Unified Approach. MIT Press

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

[10] Storn R, Price K 1997 Differential evolution- A simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization.11:341-359

[11] Liu J, Lampinen J 2002 On setting the control parameter of the differential evolution method. In: Proc. 8th int.con. Soft Computing (MENDEL2002), pp.11-18

[12] Swagatam Das, Sankha Subhra Mullick, P.N. Suganthan 2016 Recent advances in differential evolution – An updated survey, Swarm and evolutionary computation, Elsevier, 27: 1-30

[13] Qin A K, Huang V L, and Suganthan P N 2009 Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transaction on evolutionary computation.13: 398-417

[14] Rahnamayan S, Tizhoosh H R, and Salama M M A 2008 Opposition based differential evolution. IEEE transaction on evolutionary computation.12, no.1: 64-79

[15] J.Ronkkonen J, Kukkonen S, and Price K V 2005 Real parameter optimization with differential evolution. In: Proc. IEEE Congr. Evol. Comput (CEC-2005). Piscataway. NJ: IEEE Press. 1, pp506-513. **12**

[16] Brest J, Greiner S, Boskovic B, Mernik M, and Zumor V 2006 Self-adapting control parameter in differential evolution: A comparative study on numerical benchmark problems. IEEE Transaction Evolut. Comput. 10: 646-657

[17] BingyanMao, Zhigiang Xie, Yongbo Wang, Heikki Handroos 2018 A hybrid strategy of differential evolution and modified particle swarm optimization for numerical solution of a parallel manipulator, mathematical problems in engineering, Hindawi Article ID 9815469, 9 pages

[18] Wilson D R, Martinez T R 2000 Reduction techniques for instance-based learning algorithms. Machine Learning 38 (3): 257-286

[19] Fayed H A, Hashem S R, and Atiya A F 2007 Self-generating prototypes for pattern classification. Pattern Recognition. 40 (5): 1498-1509

[20] Aha D W, Kibler D, and Albert M K 1991 Instance-based learning algorithms. Machine Learning. 6 (1): 37-66

[21] Pandey S C, Nandi G C 2014 TSD based framework for mining the induction rules. Journal of Computational Science. 5: 184-195

[22] Lam W, Keung C K, and Liu D 2002 Discovering useful concept prototypes for classification based on filtering and abstraction. IEEE Transaction on Pattern Analysis and Machine Intelligence. 14 (8): 1075-1090

[23] Hart P E 1968 The condensed nearest neighbour rule. IEEE Translations on Information Theory. 18: 515-516

[24] Wilson D L 1972 Asymptotic properties of nearest neighbour rules using edited data. IEEE Transactions on System, Man and Cybernetics 2 (3): 408-421

[25] Brighton H, Mellish C 2002 Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery. 6 (2): 153-172

[26] Kohonen T 1990 The self organizing map. In: Proceedings of the IEEE. 78 (9): 1464-1480

[27] Kim S W, Oomenn J 2003 A brief taxonomy and ranking of creative prototype reduction schemes. Pattern Analysis and Applications. 6: 232-244

[28] Lozano M, Sotoca J M, Sanchez J S, Pla F, Pekalska E, and Duin RPW 2006 Experimental study on prototype optimization algorithms for prototype-based classification in vector spaces. Pattern Recognition. 39 (10): 1827-1838

[29] Bezdek J C, Kuncheva L I 2001 Nearest prototype classifier designs: an experimental study. International Journal of Intelligent Systems. 16: 1445-1473

[30] Garcia S, Cano J R, and Herrera F 2008 A memetic algorithm for evolutionary prototype selection: a scaling up approach. Pattern Recognition. 41 (8): 2693-2709

[31] Nannin L, Lumini A 2008 Particle swarm optimization for prototype reduction. Neurocomputing. 72 (4-6): 1092-1097

[32] Cervantes A, Galvan I M, and Isasi P 2009 AMPSO: a new particle swarm method for nearest neighborhood classification. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics. 39 (5): 1082-1091

[33] Fernandez F, Isasi P 2004 Evolutionary design of nearest prototype classifiers. Journal of Heuristics. 10 (4): 431-454 **33**

[34] Garain U 2008 Prototype reduction using an artificial immune model. Pattern Analysis and Applications. 11 (3-4): 353-363

[35] Triguero I, Garcia S, and Herrera F 2010 A preliminary study on the use of differential evolution for adjusting the position of examples in nearest neighbour classification. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 630-637

[36] Wenzhong Y, Lei B 2013 Algorithms for chromosome classification. Engineering. 5: 400-403

[37] Tan K C, Teoh E J, Yu Q, and Goh K C 2009 A hybrid evolutionary algorithm for attribute selection in data mining. Expert system with applications. 36: 8616-8630

[38] De Jong K A, Spears W M 1991 Learning concept classification rules using genetic algorithms. In: Proceedings of the international joint conference on artificial intelligence, Morgan Kaufmann, pp. 651-656

[39] Janikow C Z 1993 A knowledge-intensive genetic algorithm for supervised learning. Machine learning. 13, No. 2-3: 189-228

[40] Aguilar-Ruiz J S, Riquelme J C, and Toro M 2003 Evolutionary learning of hierarchical decision rules. IEEE transactions on systems, man and cybernetics, Part B. 33, No. 2: 324-331

[41] Liu J, Lampinen J 2005 A fuzzy adaptive differential evolution algorithm. soft computing- A fusion of foundations, methodologies and applications. 9, no.6: 448-462

[42] Zhang J, Sanderson A C 2009 JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evolut. Comput. 13, no.5: 945-958

[43] Mallipeddi R, Suganthan P N, Pan Q K, and Tasgetiren M F 2011 Differential evolution algorithm with ensemble of parameters and mutation strategies. Applied Soft Computing. 11,issue2: 1679-1696

[44] Alc M M, Torn A 2004 Population set based global optimization algorithms: Some modifications and numerical studies. Comput.oper.Res. 31, no.10: 1703-1725

[45] Sun J, Zhang Q, and Tsang E 2004 DE/EDA: A new evolutionary algorithm for global numerical optimization, Info.sco. 169: 249-262 **51**

[46] Herrera F, Lozano M 2001 Adaptive genetic operators based on coevolution with fuzzy behaviours. IEEE Trans. Evol. Compu. l5, no.2:149-165

[47] Tu Z, Lu Y 2004 A robust stochastic genetic algorithm (StGa) for global numerical optimization. IEEE Trans. Evol. Comput.8, no.5: 456-470 **52**

[48] Yu-Xiang lei Jin Gou, Cheng Wang, Wei Luo, and Yi-Qiao 2017 Improved differential evolution with a modified orthogonal learning strategy, IEEE access 5: 9699-9716

[49] Price K, Storn R, and Lampinen J 2005 Differential Evolution- A Practical approach to global optimization, Berlin, Germany: Springer-verlog, 2005.

[50] Lampinen J, Zelinka I 2002 On stagnation of the differential evolution algorithm. In: Proc. 6th Int. Mendel conf. Soft computing, P.Osmera, Ed., pp. 76-83

[51] Fan H Y, Lampinen J 2003 A trigonometric mutation operator to differential evolution. J.Global optimization. 27, no.1: 105-129

[52] Price K V 1999 An introduction to differential equation. In: new ideas in optimization, D.Corne, M.Dorigo, and F.Glover,Eds, London, U.K, McGraw-Hill, pp. 79-108

[53] Gamperle R, Muller S D, and Koumoutsakos P 2002 A parameter study for differential evolution. In: Advances in intelligent systems, Fuzzy Systems,Evolutionary computation, A Grmela and N.E.Mastorakis, Eds. Interlaken, Switzerland: WSEAS Press, pp.293-298

[54] Ronkkonen J, Kukkonen S, and Price K V 2005 Real- parameter optimization with differential evolution. In: Proc.IEEE congr. Evolutionary computing, Edinburgh, Scotland, pp.506-513

[55] Rutovitz D 1968 Automated chromosome analysis. Br.Med. Bull, pp.260-267

[56] Neurath P W, Bablouzian B L, Warms T H, Serbagi R C, and Falek A 1966 Human chromosome analysis by computer –an optical pattern recognition problem. Annal. N.Y. Acad. Sci, 128: 1013-1028.

[57] Ledley.R.S, Rotolo L S, Golab T J, Jacobson J D, Ginsberg M D, and Wilson J B 1965 Fidac- Film input to digital automatic computer and associated syntax directed pattern recognition programming system. In:optical and electro-optical information processing, MIT Press, Cambridge, M.A.,pp.591-613

[58] .Levan A, K.Fredga , and Sandberg A 1964 Nomenclature for centromeric position on chromosomes, pp.1-20 http:onlinelibrary.wiley.com/doi/10.1111.j.1601-5223.1964.tb01953.x/pdf,
[59] Denver study group 1960 A proposed standard system of nomenclature of human mitotic chromosomes. Acta Gery.10: 322-328
[60] Murphy P M, Aha D W 1995 UCI repository of machine learning databases, http://www.ics.uci.edu/~learn/MLRepository.html (for information contact ml-repository@ics.uci.edu)
[61] Eiben A E, Rudolph G 1999 Theory of evolutionary algorithms: a bird's eye view. Theoretical Computer Science. 229, no. 1-2: 3-9
[62] Hu Z, Xiong S, Su Q, and Zhang X 2013 Sufficient conditions for global convergence of differential evolution algorithm. Journal of Applied Mathematics. pages 14
[63] Lampinen J, Elinka I 2000 On stagnation of the differential evolution algorithm. In: Proc. 6th Int. Mendel Conf. Soft computing, Brno, Czech Republic, Jun, pp. 76-83
[64] Zielinski K, Peters D, and Laur R 2005 Run time analysis regarding stopping criteria for differential evolution and particle swarm optimization. In: Proc. 1st Int. Conf. Experiments/Process/System. Modelling/Simulation/Optimization, Athens, Greece
[65] Das S, Abraham A, Chakraborty U K, and Konar A 2009 Differential evolution using a neighbourhood-based mutation operator. IEEE transaction on evolutionary computation. 13, No. 3: 526-553
[66] Das S, Abraham A, Chakraborty U K, and Konar A 2009 Differential evolution using a neighbourhood-based mutation operator. IEEE Transactions on Evolutionary Computation. 13 (3): 5216-553
[67] Neri F, Tirronen V 2009 Scale factor local search in differential evolution. Memetic Computing. 1 (2): 153-171
[68] Isaac T, Salvador G, and Francisco H 2011 Differential evolution for optimizing the positioning of prototypes in nearest neighbour classification. Pattern recognition. 44: 901-916
[69] Derrac J, Garcia S, Molina D, and Herrera F 2011 A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation. 1 (1): 3–18
[70] Iman R L, Davenport J M 1980 Approximations of the critical region of the Friedman statistic. Communications in Statistics. A9 (6): 571-595.
[71] Demsar J 2006 Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research. 7: 1-30
[72] S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," Journal of King Saud University-Computer and Information Sciences, vol. 29, pp. 462-472, 2017.
[73] A. Chellam, et al., "Intrusion Detection in Computer Networks using Lazy learning Algorithm," Procedia computer science, vol. 132, pp. 928-936, 2018.
[74] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," Computers & Security, vol. 81, pp. 148-155, 2019.
[75] S. Aljawarneh, et al., "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," Journal of Computational Science, vol. 25, pp. 152-160, 2018.
[76] Ankit Thakkar, Ritika Lohiya, Role of swarm and evolutionary algorithms for intrusion detection system: A survey, Swarm and evolutionary computation, Vol. 53, March, 2020. DOI: 10.1016/j.swevo.2019.100631
[77] KDDCup 1999 data. http://kdd.ics.uci.edu/ Databases/kddcup99/10 percent.gz.
[78] P Amudha, H Abdul Rauf, "Performance Analysis of Data Mining Approaches in Intrusion Detection", In Proc. of IEEE Conference on Process Automation Control and computing, pp.9-16, 2011.

Figures

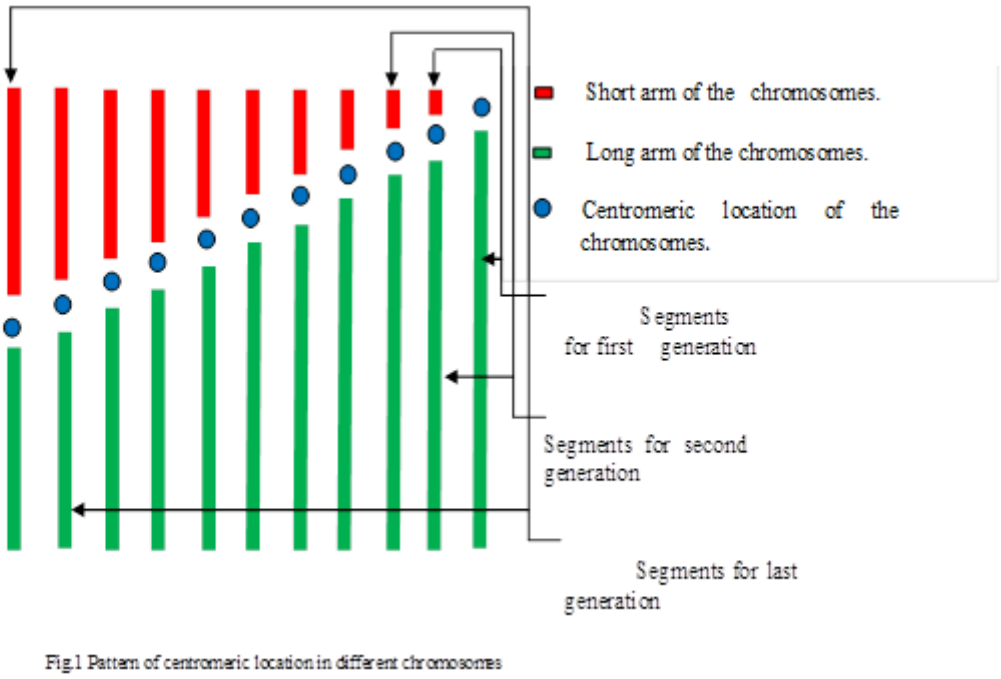

Fig.1 Pattern of centromeric location in different chromosomes

Figure 1

See image above for figure legend

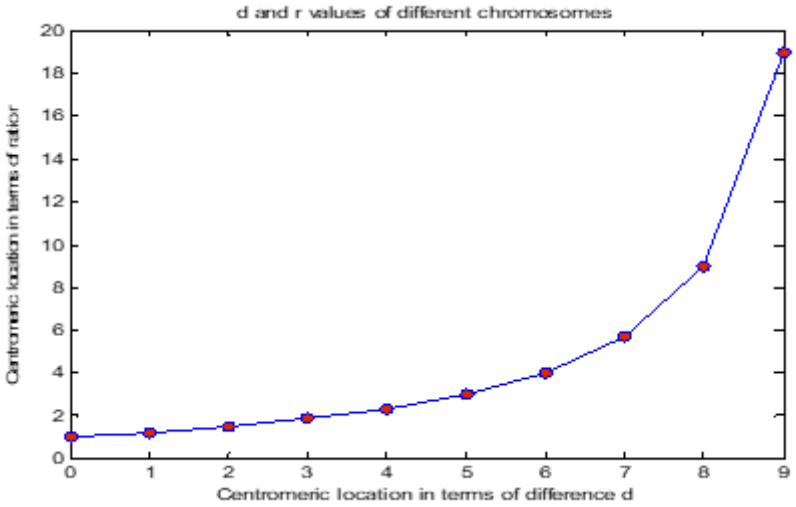

Figure 2

Interrelation of *d*, and *r* for different chromosomes

Step 1 Set the generation number ?? = 1 and initialize a population of ???? individuals. This ???? individuals include longer arm of the chromosomes(??) and shorter arm of the chromosomes (??) ?? = 1,2, … , ????.

Step 2 WHILE stopping criterion is not satisfied.

DO

Step 2.1 ***Mutation step***

/*Generate a mutated vector ($d_{i,G}$) for each target vector*/

FOR ?? = 1 ???? ????

Generate mutated vector $d_{i,G}$ *corresponding to the target vector via equation(11)*

END FOR

Step 2.2 ***Crossover step***

/*Generate a trial vector $r_{i,G}$ *and* $i_{i,G}$ for each target vector*/

FOR ?? = 1 ???? ????

Generate trial vector $r_{i,G}$ *via equation(12)*

Generate trial vector $i_{i,G}$ *via equation(13)*

END FOR

Step 2.3 ***Selection step***

FOR ?? = 1 ???? ????

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 ?? = ?? + 1 and repeat the step 2.

Step 3 END WHILE

Figure 3

Algorithmic description of CIDE

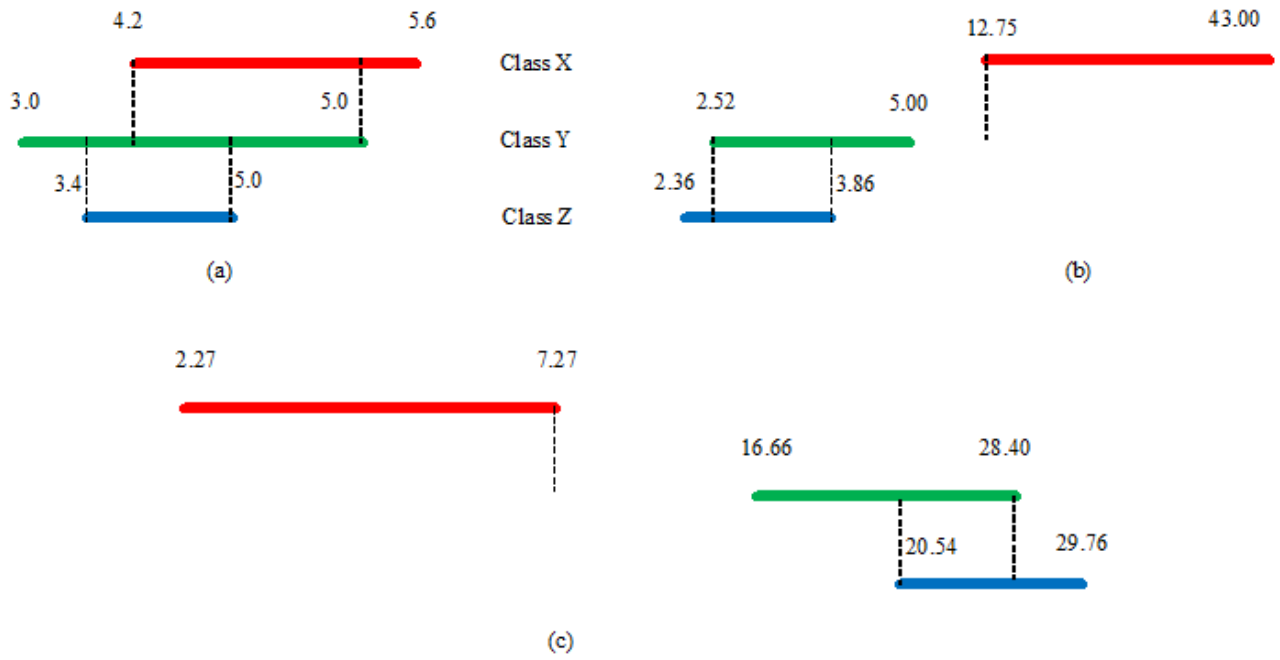

Figure 4

Rule extraction from sample data after the first generation of CIDE (a) from (b) from (c) from *i*

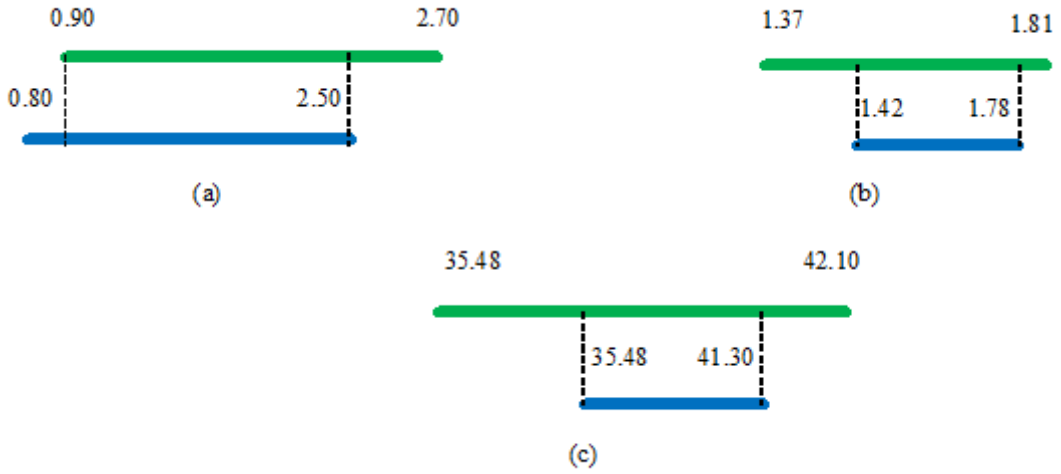

Figure 5

Rule extraction from sample data after the first generation of CIDE (a) from (b) from (c) from *i*

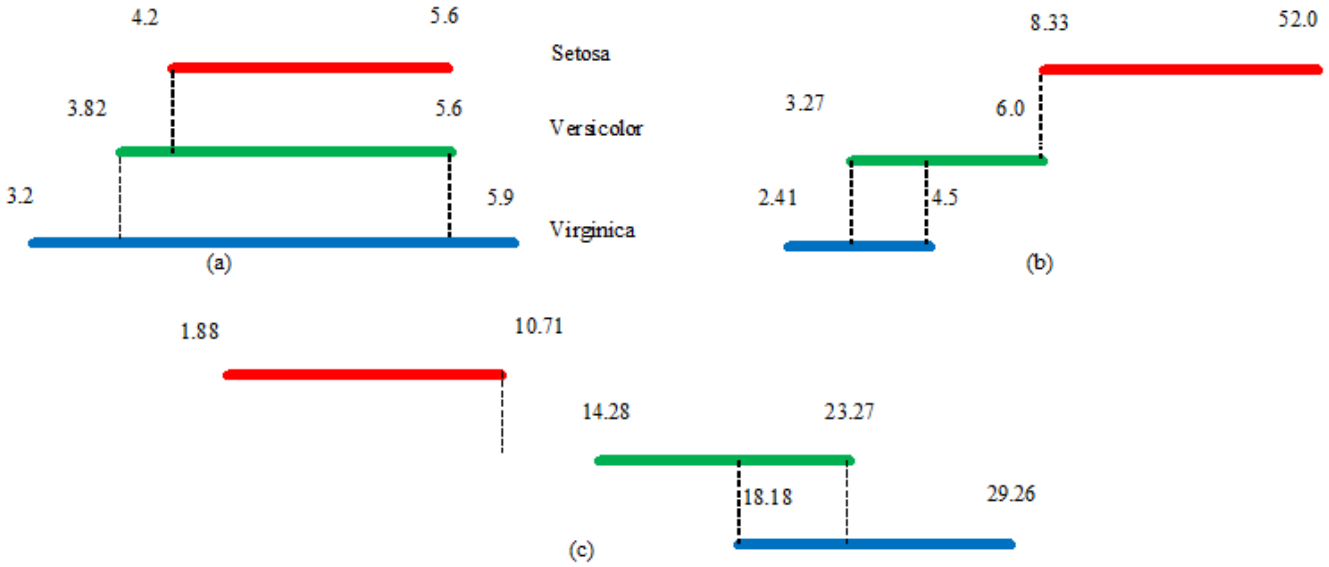

Figure 6

Rule extraction from iris data set after the first generation of CIDE algorithm (a) rule extraction on the basis of (b) rule extraction on the basis of (c) rule extraction on the basis of *i*

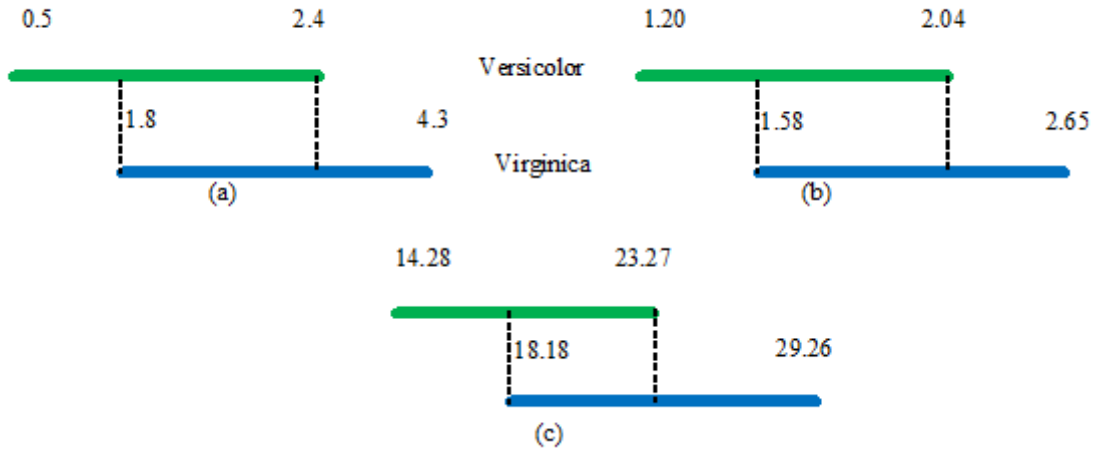

Figure 7

Rule extraction from iris dataset after the second generation of CIDE algorithm (a) rule extraction on the basis of (b) rule extraction on the basis of (c) rule extraction on the basis of *i*

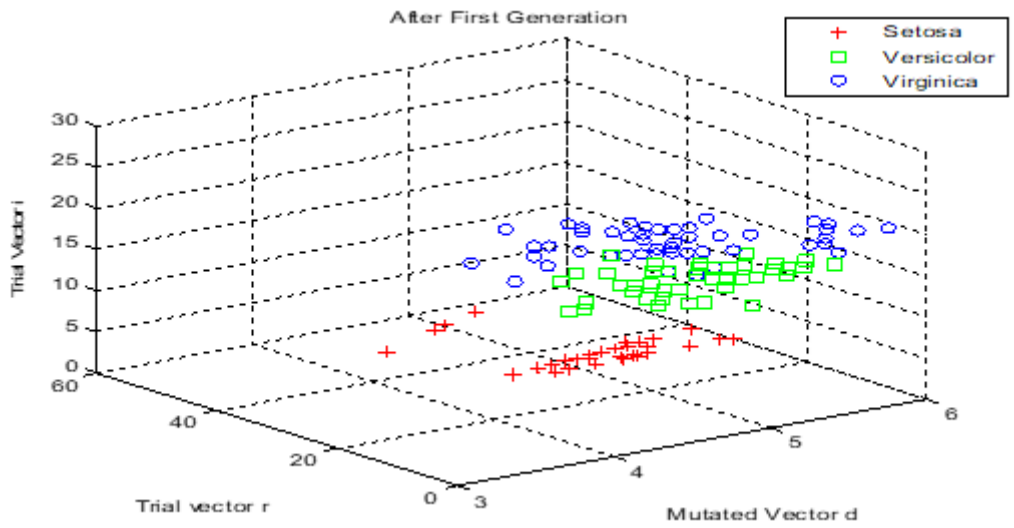

Figure 8

Scatter plot for the distribution of mutated vector *d*, trial vector *r*, and trial vector after the first generation of CIDE for iris dataset

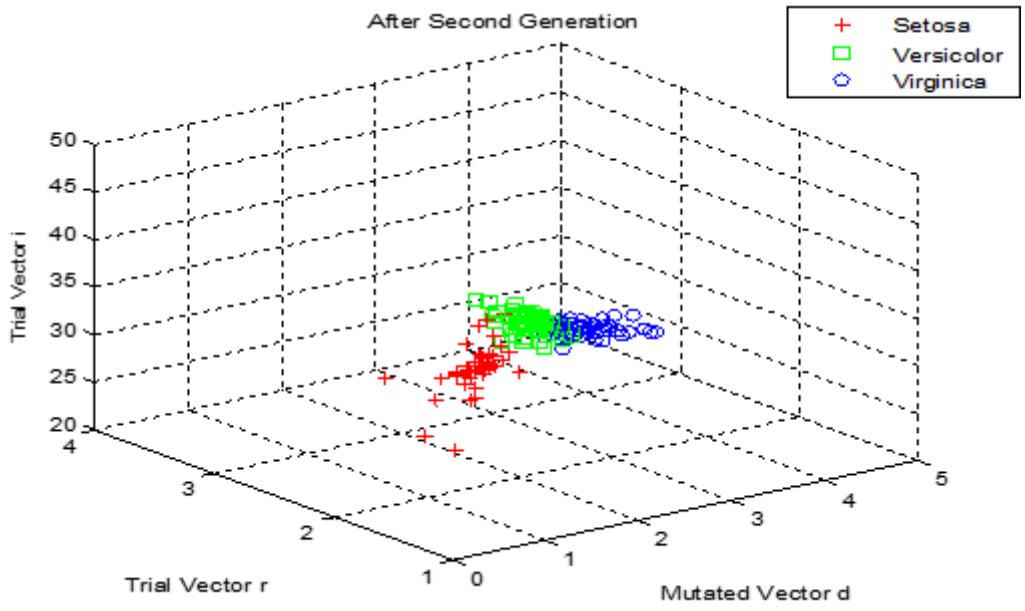

Figure 9

Scatter plot for the distribution of mutated vector *d*, trial vector *r*, and trial vector after the second generation of CIDE for iris dataset

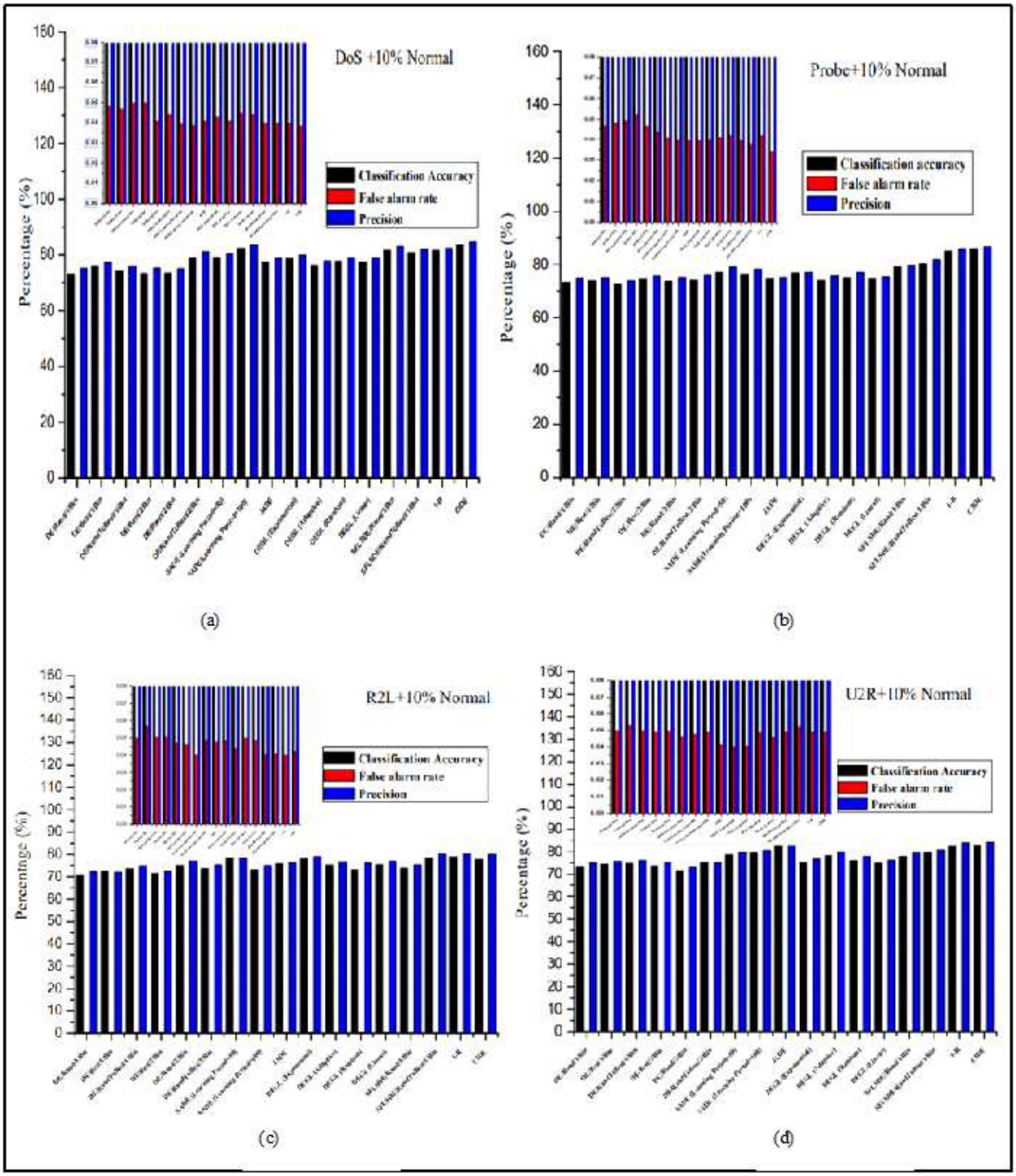

Figure 10

Bar graph of classification accuracy, false alarm rate and precision (a) Dos+10% Normal (b) Probe+10% Normal (c) R2L+10% Normal (d) U2R+10% Normal