Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.022864

ARTICLE

# Dendritic Cell Algorithm with Grouping Genetic Algorithm for Input Signal Generation

Dan Zhang[1], Yiwen Liang[1,*] and Hongbin Dong[2]

[1]School of Computer Science, Wuhan University, Wuhan, 430072, China

[2]School of Cyber Science and Engineering, Wuhan University, Wuhan, 430072, China

*Corresponding Author: Yiwen Liang. Email: zhangxiaobei125@whu.edu.cn

Received: 30 March 2022 Accepted: 15 August 2022

**ABSTRACT**

The artificial immune system, an excellent prototype for developing Machine Learning, is inspired by the function of the powerful natural immune system. As one of the prevalent classifiers, the Dendritic Cell Algorithm (DCA) has been widely used to solve binary problems in the real world. The classification of DCA depends on a data pre-processing procedure to generate input signals, where feature selection and signal categorization are the main work. However, the results of these studies also show that the signal generation of DCA is relatively weak, and all of them utilized a filter strategy to remove unimportant attributes. Ignoring filtered features and applying expertise may not produce an optimal classification result. To overcome these limitations, this study models feature selection and signal categorization into feature grouping problems. This study hybridizes Grouping Genetic Algorithm (GGA) with DCA to propose a novel DCA version, GGA-DCA, for accomplishing feature selection and signal categorization in a search process. The GGA-DCA aims to search for the optimal feature grouping scheme without expertise automatically. In this study, the data coding and operators of GGA are redefined for grouping tasks. The experimental results show that the proposed algorithm has significant advantages over the compared DCA expansion algorithms in terms of signal generation.

**KEYWORDS**

Dendritic cell algorithm; combinatorial optimization; grouping problems; grouping genetic algorithm

## 1 Introduction

The natural immune system assists in recognizing pathogens that can cause destructive infections in individuals and has certain key characteristics: diversity, distributive, dynamic, adaptivity, and robustness [1]. Inspired by its function, several researchers mixed the mechanisms of immunity and computers to propose a collection of bio-inspired algorithms. DCA, one of its prevalent paradigms, is inspired by the functioning of the dendritic cells in the natural immune system [2]. DCA has been widely applied in classification [3,4], anomaly detection [5,6], spam filtering [7], distributed and parallel operations [8], fuzzy clustering, privacy preservation, and earthquake prediction [9]. DCA is designed for low-dimensional space, and the inputs of DCA correspond to three immune signals respectively, named pathogen-associated molecular patterns (PAMP), danger signals (DS), and safe signals (SS) [2].

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Generally, the DCA appropriately maps a given problem domain to the input space of DCA in a pre-processing and initialization phase, which is crucial to obtaining reliable results. Recalling from the literature review [10], dimensionality reduction and signal categorization are the main works to generate input signals in the pre-processing and initialization phase. The task of dimensionality reduction is to create a new feature space with low dimensionality. After dimensionality reduction, each feature in the new feature space is assigned to a specific signal category, either PAMP, DS, or SS. Researchers utilized several dimensionality reduction techniques to create a new feature space with low dimensionalities, such as Principal Component Analysis (PCA) [11], Correlation Coefficient (CC) [12], Information Gain (IG) [12], Rough Set Theory (RST) [13–15], and Fuzzy Rough Set Theory (FRST) [16,17]. These approaches filtered weakly important or unimportant features and selected the most important features with their data-intrinsic methods. However, it is essential to realize that relevance/importance according to those definitions does not imply membership in the optimal feature subset, and irrelevance does not mean that a feature can not be in the optimal feature subset [18]. In addition, machine learning algorithms, such as K-Nearest Neighbors (KNN) [19] and Support Vector Machine (SVM) [20], are employed for input signal generation. Moreover, Zhou et al. [1] utilized numerical differential to extract features based on the data changes of the selected features. Among these methods, some approaches may not generate the optimal feature subset due to ignoring the effects of those filtered features; moreover, some approaches are unable to describe the relationship between the attributes and signal categorization.

After dimensionality reduction, each feature in the new feature space is assigned to a specific signal category, either PAMP, DS, or SS. For signal categorization, researchers focus on building a mapping between attributes and signal categories. An appropriate mapping relationship helps achieve good classification results. Several approaches assign a specific signal category to each selected feature by generating a mapping relationship grounded on expert knowledge; others perform signal categorization based on the importance ranking of selected features and the importance ranking of signals. However, the mapping with specialist knowledge may not assign attributes to the most suitable signal categories. In addition, the mapping based on attributes ranking could not be considered a coherent and consistent categorization procedure, leading to unsatisfactory classification results. Several researchers employed a search strategy to find an optimal mapping to overcome the problem. But the approaches with search strategy exclude the filtered features from the original feature space and ignore the effects of the filtered features on the performance of DCA. Among these methods, some algorithms rely on expertise; some establish a mapping in terms of the relationship between the importance of attributes and signals; others ignore the effects of the filtered features, resulting in unsatisfactory classification results.

Therefore, this research is motivated by the following question: how to automatically perform the feature selection and signal categorization, considering all features' effects for DCA? Inspired by the research about combinatorial optimization, this study transforms feature selection and signal categorization into a feature grouping problem. The features from the original data set are divided into four groups: $Group_{PAMP}$, $Group_{DS}$, $Group_{SS}$, and $Group_{UN}$. The $Group_{UN}$ contains all the features unselected; the $Group_{PAMP}$ contains all the features assigned to the signal PAMP; $Group_{DS}$ contains all the features assigned to the signal DS; and $Group_{SS}$ contains all the features assigned to the signal SS. The grouping problem is a special combinatorial optimization problem where a set $V$ of $n$ items is usually partitioned into a collection of mutually disjoint subsets (groups) $Group_i$. So that, the feature set $V = \cup_i^D Group_i$, and $Group_i \cap Group_j = \emptyset$, $i \neq j$, $D = \{PAMP, DS, SS, UN\}$. In literatures, there were several approaches designed for grouping problems, i.e., Moth Search (MS) [21], Artificial Bee Colony (ABC) [22], Differential Evolution (DE) [23,24], Genetic Algorithm (GA) [25], Particle Swarm

Optimization (PSO) [26], Grouping Genetic Algorithm (GGA) [27]. Over the years, those methods have been broadly used to solve grouping problems, such as parallel machine scheduling problems, job-shop scheduling problems (JSP), vehicle routing problems, and other grouping problems. The GGA is an extension of the traditional GA with a special solutions encoding. The GGA utilized a more natural or intuitive way (group-based encoding scheme) to represent the potential solutions instead of a machine-based encoding scheme and a permutation-based encoding scheme(i.e., MS, ABC, DE, GA, and PSO). In literature [28], their experiments illustrated that the GGA, with a group-based encoding scheme, can obtain better grouping results than permutation-based and machine-based encoding scheme. Due to its excellent grouping capabilities, this study employs GGA to assign each feature into a group, either $Group_{PAMP}$, $Group_{DS}$, $Group_{SS}$, or $Group_{UN}$. The GGA can simultaneously optimize consolidation on both feature selection and signal categorization. This study aims to propose an effective DCA version integrated with GGA, GGA-DCA, to automatically address the feature selection and signal categorization, taking into all the attributes. To verify the performance of GGA-DCA, 24 data sets are selected with different feature dimensions, dataset size, and imbalance rate. Compared with the several state-of-the-art DCA expansion algorithms (e.g., FLA-DCA, GA-PSM, NIDDCA, and SVM-DCA). Moreover, this study analysis the time complexity and the runtime of the GGA-DCA and the other state-of-the-art DCA expansion. The well-known classifiers, the K-Nearest Neighbor (KNN) and the Decision Tree (DT), XGboost, Random Forests (RT), and Extremely Randomized Trees (ERT), are used as the baseline comparison. Through the experiments, the GGA-DCA obtains promising results.

This paper is structured as follows: Section 2 describes related work; the DCA is introduced in Section 3; The novel model, GGA-DCA, is proposed in Section 4; our following experiment setup, results and analysis are described in Section 5; conclusions and future work are shown in Section 6.

## 2 Related Work

The classical DCA is proposed by Greensmith et al. [2] which mimics the functioning of the dendritic cells in the natural immune system. The classical DCA relies on artificial experience for feature selection and signal categorization. Aiming to overcome the limitations of the manual method, PCA, CC, IG, RST, and FRST are applied to perform those works.

Gu et al. [11] employed PCA as a feature extraction technique to project the original data onto the principal subspace. They constructed a low-dimensional space with new features. Due to destroying the underlying meaning behind the features present, feature extraction techniques are undesirable for DCA. In [12], the CC and IG were adopted to measure the relevance between attributes and the class. The features whose relevance degree exceeds a certain threshold were selected. Chelly et al. [13–15] proposed RST-DCA and RC-DCA by hybridizing the RST and DCA. RST-DCA and RC-DCA measured the importance of a feature by computing the difference between the positive region of an original data set and the positive region of the data set without this feature. Based on the importance of features, REDUCTs and CORE are achieved as candidates of optimal feature subset. Due to the expensive costs of calculating REDUCTs and CORE, QR-DCA [15] introduced the QuickReduct algorithm to generate only one REDUCT. RST-DCA assigned only one attribute randomly from a REDUCT to both PAMP and SS based on expert knowledge, as well as combined the rest features of REDUCT to represent DS. RC-DCA and QR-DCA assigned attributes of the CORE to both PAMP and SS based on expert knowledge, as well as combined the rest features of REDUCT to represent DS.

The hybrid DCA versions with RST rely upon crisp data sets. Chelly et al. [16,17] integrated FRST with DCA to provide feature reduction for both crisp and real-value attributed datasets. In [17], a greed search and fuzzy lower approximation were applied to obtain a new DCA version, FLA-DCA. The FLA-DCA calculates a feature reduction with the same fuzzy-rough dependency degree as the entire database. In [16], fuzzy boundary region-based DCA (FBR-DCA) was proposed subsequently. The FBR-DCA utilized a greed search and the fuzzy boundary region to find a feature reduction with the minimal uncertainty degree based on fuzzy rough set theory. FLA-DCA selected the feature with the most significant fuzzy-rough dependency degree to form the SS, the second attribute to form PAMP, and the rest of the reduct attributes are combined and affected to form DS. FBR-DCA selected the attributes with the slightest uncertainty degree to form the SS, as it is considered the most informative first feature added to the fuzzy-rough reduct.

The machine learning algorithms, such as KNN and SVM, are also introduced for DCA to generate input signals. Mohsin et al. [20] proposed a hybrid DCA version, SVM-DCA. They adopted SVM to generate a sparse weight matrix of attributes and selected attributes with larger weights to generate input signals. In [19], KNN was employed to filter features based on the change of the two data and was integrated with DCA to generate a hybrid KNN-DCA. Compared with C4.5, LibSVM, Hoeffding Tree, and NaiveBayes, the KNN-DCA and SVM-DCA achieved better classification results. Zhou et al. [1] proposed a NIDDCA that used numerical differential to calculate the change of the selected feature as the input signals. Through experimental results for signal extraction, they achieved a satisfactory result better than other recent DCA-derived classification algorithms on most datasets. In addition, Elisa et al. [29] utilized a two-stage approach to accomplish feature selection and signal categorization: first, they filtered some unimportant/irrelevant features; second, they utilized GA based on partial shuffle mutation (PSM) to present a new DCA version GA-PSM, to find the optimal mapping.

In summary, those approaches filtered weakly important or unimportant features and selected the most important features with their data-intrinsic methods. However, the relevance/importance according to those definitions does not imply membership in the optimal feature subset [18]. Ignoring those unimportant/irrelevant attributes may prevent DCA from being able to produce satisfactory results. For signal categorization, an appropriate mapping relationship is helpful to achieve good classification results. Those methods attempt to map the attributes ranking, in terms of relevance/importance, to the ranking of signal categories or just through expert knowledge. Thus, those approaches to establishing the mapping could not be considered a consistent procedure, leading to unsatisfactory classification results.

Inspired by solutions to grouping problems, this study transforms feature selection and signal categorization into a grouping problem of features. The grouping problem is a special combinatorial optimization problem. Several computational intelligence algorithms, i.e., MS, ABC, DE, GA, PSO, and GGA, have been applied to solve grouping problems.

Feng et al. [21] proposed a binary MS based on self-learning to solve the multidimensional knapsack problem. They adopted a simple generic mapping method via transfer function to convert the real number vector into a binary one. Zhuang et al. [22] introduced an improved ABC to investigate shop scheduling problems with two sequence-dependent setup times. They encoded the potential schemes as numerical sequences and achieved encouraging results on the small-scale benchmark instances. Gao et al. [30] employed DE to solve the JSP with fuzzy execution time and fuzzy completion time. They encoded potential solutions with real number vectors and then computed the ranks of each element according to their descending orders. They also proposed a hybrid adaptive differential evolution algorithm (HADE) [23] to solve the multi-objective JSP with fuzzy processing time and completion time. They completed the discrete optimization problem through sort and mod operators to convert real numbers into binary values 1 or 2. Falkenauer [31] designed a variant of GA, GGA, that used a group-based solutions representation scheme and variation operators working efficiently together. Due to the excellent grouping capacity, GGA has been applied to solve various grouping problems, like bin packing, vehicle routing, JSP, and Clustering [27]. Pakzad-Moghaddam [26] applied PSO to schedule jobs on uniform parallel processors. They employed a machine-based encoding scheme as the expression of a particle to lend the PSO to adapt well to the complicated structure of the grouping problem. They suggested that the approach was efficient and could play a substantial part in directing real production. Ramos-Figueroa et al. [28] compared the GGA, GA, and PSO for the grouping problems. Their experiments illustrated that the GGA outperformed PSO and GA in most cases for grouping problems.

The MS, ABC, DE, GA, and PSO encoded their feasible solutions with integer sequences where an integer denoted an element. Among those methods, some algorithms utilized orders of elements to represent their groups; some applied operators (for instance, mod function) to compute the group of each element; others introduced real number vectors to express the feasible schemes through mapping function to convert real number vector into integer one. Those algorithms employ a permutation-based vector to express a feasible solution; thus, they require decoding work to achieve a grouping scheme corresponding to a feasible solution. However, the GGA applied a more natural or intuitive way, a group-based representation scheme, to encode their feasible solutions. The solvers' performance in tackling grouping problems can be improved by incorporating group-based representation schemes and suitable variation operators. Due to the excellent grouping capacity of GGA, this study attempts to apply GGA to synchronously perform feature selection and signal categorization by searching for the optimal feature grouping solution. The literature review reveals that the GGA is first used for DCA to generate input signals.

## 3 Preliminary

### *3.1 Basic Definition*

In algorithm, each data item of DCA contains two inputs: signals and antigens. The antigen is the identifier of the data item, in other words, the data item IDs. The input signals have only three signal categories corresponding to the three immune signals mentioned above. Each data item is transformed into the above three input signals through feature selection and signal categorization. The DCA maintains a population of detectors, namely DCs. The DCs simulate the function of dendritic cells in a tissue environment to detect whether a data item is normal or abnormal grounded by its input signals. Each data item is processed by detectors selected from the population randomly. Finally, the algorithm synthesizes the detection results generated by DCs to decide the class of each data item.

**Definition 1** An antigen is presented as $\mathbf{Ag} = \langle \mathbf{e}, t \rangle$. $\mathbf{e}$ is the identifier of a certain data item to be detected, $t$ is the timestamps.

**Definition 2** The signal is denoted as $\mathbf{Signal} = \langle PAMP, DS, SS \rangle$, a 3-dimensional real valued tuple. $SS$ is the safe signal value, $DS$ is the danger signal value, $PAMP$ is the value of pathogen-associated molecular patterns.

**Definition 3** Each DC is a detector, and is expressed as $DC = \mathbf{Ags} \times \mathbf{Signals} \times T$. $\mathbf{Signals}$ is the signal values sampled by the DC, and $T$ is a migration threshold, $\mathbf{Ags}$ is a set of antigens.

### *3.2 The DCA*

The DCA is a population-based algorithm inspired by the danger theory. It maintains a population of DCs to detect data items, whether the class of data is normal or abnormal. Greensmith and

Gale defined the algorithm DCA as a function $H = \mathbf{A} \times \mathbf{S} \times N$ where $\mathbf{A}$ is antigen set, $\mathbf{S}$ is signal set, and $N$ is the population of DCs. The algorithm contains four main phases: the pre-processing and initialization phase, the detection phase, the context assessment phase, and the classification phase.

**The pre-processing and initialization phase:** The data is used to describe the object in the real world, and is presented as $\mathbf{Data} = \{data_i | data = \langle Feature_1, Feature_2, \ldots, Feature_m \rangle\}$. The $m$, the data size, is often higher than three. However, the DCA is designed for low-dimensional space. Therefore, the works of feature selection and signal categorization are required to map the higher-dimensional data space to lower-dimensional signal space $\mathbf{Signals} = \langle PAMP, DS, SS \rangle$.

**The detection phase:** DCA utilizes a linear utility function to transform the input signals mentioned previously into detection signals, namely the costimulatory molecule signal value ($CSM$), the semimature signal value ($SEMI$), and the mature signal value ($MAT$). The three interim signals of a DC are continuous during the process of antigen processing. The linear utility function is shown in Eq. (1). The $\mathbf{w}$ of the function is a weight matrix for DCA.

$$(CSM, SEMI, MAT) = \mathbf{w} \times Signals \tag{1}$$

**The context assessment phase:** The $CSM$, $SEMI$, $MAT$ is used to assess the state of the context around a DC. As soon as the $CSM$ signal of a DC exceeds the migration threshold, the DC ceases to detect new antigens, and its context is assessed. If the cumulative $SEMI$ of the DC is more than its cumulative $MAT$, the antigens around the DC are considered normal, and vice versa.

**The classification phase:** In algorithm, a DC can detect many antigens, and an antigen can also be detected by many DCs. The class of an antigen is voted by all detectors that have caught the antigen. The abnormality of antigen, namely Mature Context Antigen Value ($MCAV$), is calculated as Eq. (2). A threshold value of $MCAV$ is introduced to represent the probability that an antigen is anomalous. If the $MCAV$ of an DC exceeds the threshold value mentioned before, the antigen is labeled as abnormal, and vice versa.

$$MCAV = \frac{DC_{MAT}}{DC_{SEMI} + DC_{MAT}} \tag{2}$$

## 4 The Hybrid Algorithm: GGA-DCA

### *4.1 Model Overview*

This study attempts to transform the work of feature selection and signal categorization into a feature grouping task. The grouping task is to divide features into four groups: $Group_{PAMP}$, $Group_{DS}$, $Group_{SS}$ and $Group_{UN}$. The features in the $Group_{UN}$ are not assigned to any signal categories; the features in $Group_{PAMP}$ are assigned to PAMP; the features in the $Group_{DS}$ are assigned to DS; the features in the $Group_{SS}$ are assigned to SS. Through grouping features, the features in $Group_{PAMP}$, $Group_{DS}$, and $Group_{SS}$ are the selected features to generate input signals. This study accomplishes the pre-processing and initialization procedure automatically for DCA through the feature grouping.

Therefore, this study uses a feature grouping process to replace the original work: feature selection and signal categorization. A novel scheme named GGA-DCA is presented, hybridizing the DCA with GGA, as shown in Fig. 1. GGA-DCA contains three components, search space, search method (GGA), and evaluation method (DCA). The search space includes all potential grouping schemes for generating the input signal of DCA. Aiming to find an optimal grouping scheme, DCA is a part of the performance evaluation function, and GGA is used as a search engine wrapping around DCA to find the optimal scheme. The key components, search space, search engine, and evaluation method, are described below.

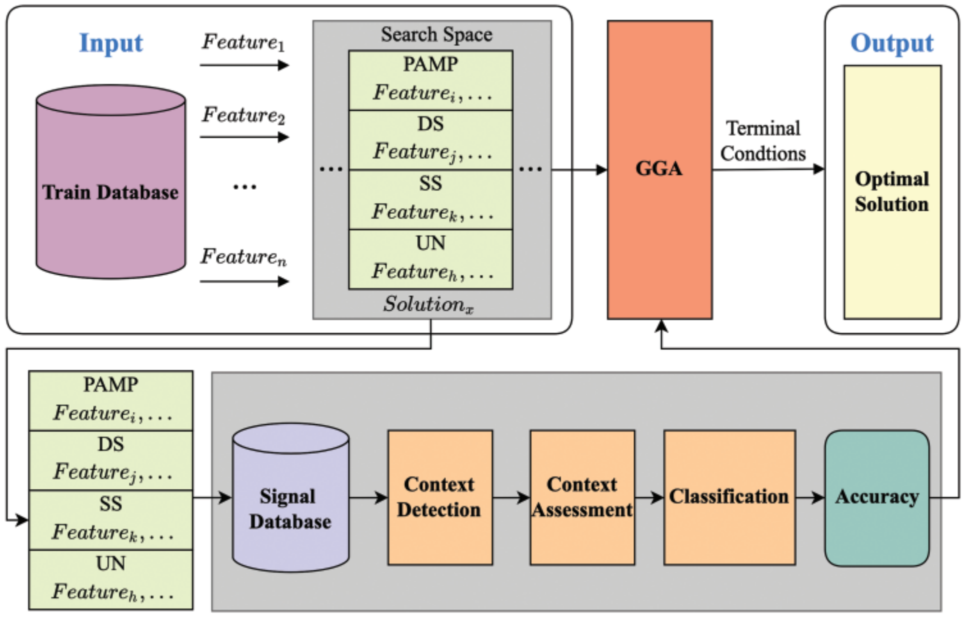


**Figure 1:** The model of GGA-DCA

### *4.2 Search Space*

As shown in Fig. 2, the work of signal categorization is to establish a mapping relationship between selected features and signal categories, offered as Eq. (3). Establishing a mapping relation can be considered as dividing the attributes into different groups.

$$\begin{pmatrix} Feature_i \\ Feature_j \\ Feature_k \\ Feature_h \\ \ldots \end{pmatrix}_f \rightarrow \begin{pmatrix} PAMP \\ SS \\ DS \\ UN \end{pmatrix} \tag{3}$$

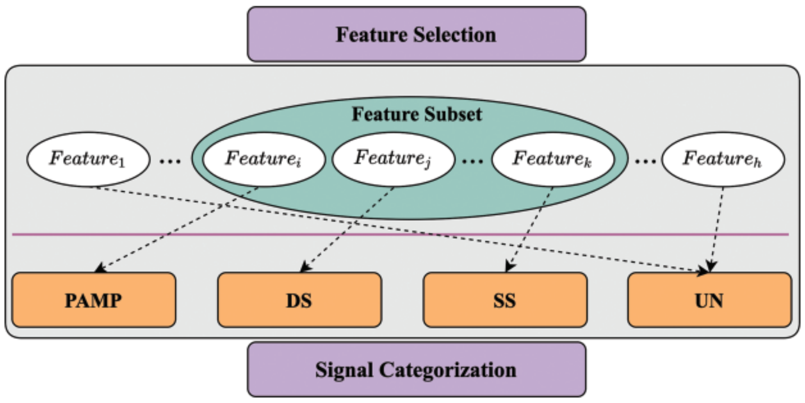


**Figure 2:** Steps of feature selection and signal categorization

The grouping scheme of the features is a solution to feature selection and signal classification, as shown in Fig. 3. In this study, the solution can be denoted as $\{Group_{PAMP}\,(Feature_i \ldots)$ $Group_{DS}, \left(Feature_j \ldots\right), Group_{SS}(Feature_k \ldots), Group_{UN}\,(Feature_h \ldots)\big\}$, the $Feature_i$, $Feature_j$, $Feature_k$, $Feature_h$ are the features form original data sets. The solutions can form a search space. Each state in the space represents a solution to generate input signals. The different grouping schemes of features represent different solutions for generating input signals. The operators, which determine the connectivity between the states, transform one state to another by swapping groups of features.

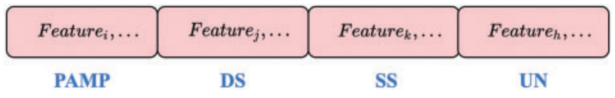


**Figure 3:** The grouping scheme: A 4-dimensional vector V

## *4.3 Evaluation Method*

In this work, the classification accuracy of DCA is adopted to evaluate the performance of states. A signal database contributed by a state is performed multiple times, and the average accuracy of multiple experiments is the performance of a state.

## *4.4 Search Method: GGA*

This study employs GGA as the search engine to find the optimal feature grouping scheme. The key steps are as described below.

**Step 1: Data encoding.** In GGA, each chromosome represents a state in the search space. The nature of the problem determines the chromosome coding [29]. This study uses integers $\{i, j, k, h, \ldots\}$ to represent independent features separately $\{Feature_i, Feature_j, Feature_k, Feature_h, \ldots\}$ from the original data sets ($i, j, k, h$ are integers, $i \neq j \neq k \neq h$, and $0 < i, j, k, h \leq m$, $m$ is the amount of the whole features). This study determines a 4-dimensional vector $\mathbf{V} = \{[i, \ldots], [j, \ldots], [k, \ldots], [h, \ldots]\}$ as a chromosome. In this study, the order of groups is fixed. The first group is the $Group_{PAMP}$, the second is the $Group_{DS}$, the third is the $Group_{SS}$, and the last is the $Group_{UN}$. The different feature grouping schemes represent different chromosomes, for exsample, $\{[i, \ldots], [j, \ldots], [k, \ldots], [h, \ldots]\} \neq \{[j, \ldots], [i, \ldots], [k, \ldots], [h, \ldots]\}$.

**Step 2: Fitness function.** The fitness function of a given chromosome determines its probability of being chosen to create the next generation [29]. This study adopts the classification accuracy of DCA, that described in Section 3.2, to evaluate the performance of a chromosome.

$$fitness(\mathbf{V}) = Accuracy(DCA(\mathbf{V})), \tag{4}$$

where $\mathbf{V}$ is a chromosome, $fitness\,(\mathbf{V})$ is the fitness value of chromosome $\mathbf{V}$. $Accuracy(DCA(\mathbf{V}))$ is the mean of classification accuracy achieved by DCA running multi-times on a signal database generated through the chromosome $\mathbf{V}$.

**Step 3: Selection function.** After computing the fitness for each chromosome in the current population, the better ones need to be selected as the next generation's parents by utilizing the roulette wheel. The cumulative fitness values of all the chromosomes in each iteration are calculated. A chromosome is selected with a probability equaling the proportion of its fitness value in the cumulative, shown in Eq. (5).

$$R_{\mathrm{V}_i} = \frac{fitness\left(\mathbf{V}_i\right)}{\sum_{j=0}^{n} fitness\left(\mathbf{V}_j\right)}, \tag{5}$$

where $\sum_{j=0}^{n} fitness\left(\mathbf{V}_j\right)$ is the cumulative fitness value of all the chromosomes in each iteration, $R_{\mathrm{V}_i}$ is probability of a chromosome $\mathbf{V}_i$.

**Step 4: Crossover operator.** In this work, the chromosomes with different orders of groups are unique. To explore the more possible chromosomes around the current one, 1PX (One-point Crossover) [27] is utilized to swap the groups of two chromosomes randomly. The crossover is performed on a randomly selected chromosome with a probability $P_{cross}$. In this operator, as shown in Fig. 4, one crossing points (cp1) between 0 and (n−1) are selected randomly to divide both two parents ($p1$ and $p2$) into two segments. The segments on the right of both two parents are the crossing segments ($cs$). This study swap $cs$ of $p1$ and $p2$ to generate two children($c1$ and $c2$). Because the children $c1$ and $c2$ have repeated and missed items ($MI$), 1PX may generate infeasible solutions. Therefore, using efficient problem-domain heuristics is crucial to handle such infeasibility. The parents are the chromosomes with the higher accuracy in the previous generation, and preserving the parents' grouping is important to produce good results. Therefore, this study removes the repeated genes in $cs$. As shown in Fig. 4, the feature {5} of $Group_{SS}$ in $c1$ is repeated with the one in $Group_{DS}$, and so is {10} in $Group_{UN}$ of $c2$. This study keeps the duplicate part in $c1$ and $c2$ and removes them in $cs$. Through the crossover operator, the newly generated children may miss some genes. As shown in Fig. 4, the missing genes($MI$) in $c1$ is {10}, and $MI$ in $c2$ is {5}. Based on the group of $MI$ in the parents $p1$ or $p2$, the $MI$ is inserted into a group of children. Suppose that the fitness of $p1$ is bigger than $p2$. This study determines the $Group$ of $MI$ in $c1$ and $c2$ based on the original $Group$ in $p1$. This study inserts $MI$ of $c1$ into $Group_{UN}$, and the one of $c2$ into $Group_{DS}$.

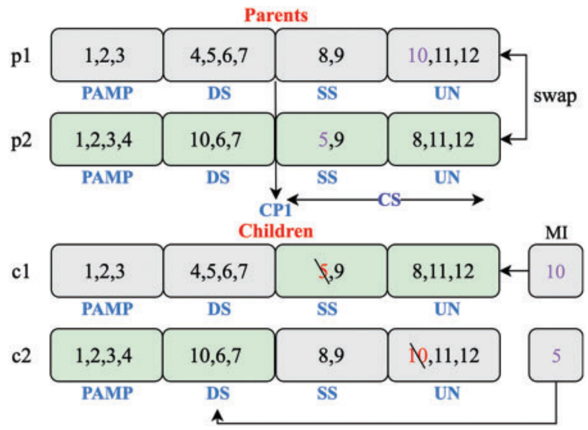


**Figure 4:** The crossover operators of GGA-DCA

**Step 5: Mutation operator.** The operator transforms one chromosome into another by swapping genes of two existing groups in the chromosome. This study performs mutation with a probability $P_{mutation}$.

**Step 6: Termination conditions.** The search process is running iteratively until satisfying one of the two conditions. The first condition is when the fitness value of a chromosome is more significant than a threshold. The second condition is when the generations of populations execute up to a certain number *G*.

**Step 7: Algorithm of GGA-DCA.** The search process of GGA for an optimal solution is illustrated in Algorithm 1. In algorithm, the first step is to initialize a population. For initializing, the data set features are randomly mapped to the group schemes of *P* chromosomes in a population to generate *P* input signal data sets. After initializing, the fitness values of all chromosomes in the population are computed and stored in memory. To expand the searching scope, the genetic operators are applied to the current population to generate the next generation population $P_{new}$. In the third step, chromosomes are selected from the current population as parents utilizing the roulette wheel method. The crossover operator is applied to generate children with a probability $P_{cross}$, and the children mutate under the influence of probability $P_{mutation}$ before appending them to population $P_{new}$. Let $P_{new}$ replace the current population. Repeat the second step and third step until satisfying the termination conditions.

**Algorithm 1:** GGA-DCA

```
Input: Train_data
Output: Optimal feature sequence
1:  Population = Initial_Population(P)
2:  while G > 0 do:
3:      for chromosome in Population do:
4:          Dataset = Generat_Newdata(Train_data, chromosome)
5:          Fitness_value = Accuracy(DCA(Dataset, chromosome))
6:      end for
7:      P_new = Null
8:      Better_ones_p1 = Selection(Population)
9:      Better_ones_p2 = Selection(Population)
10:     while len(Pnew) < len(Population) do:
11:         if rate_random < Pcross then:
12:             c1, c2 = Cross(Better_ones_p1,Better_ones_p2)
13:         end if
14:         if rate_random < Pmutaion then:
15:             c1, c2 = Mutation(c1, c2)
16:         end if
17:         Add_To_Population((c1, c2), Pnew)
18:     end while
19:     Population = Pnew
20:     G = G−1
21: end while
```

## 5 Experimentation

### *5.1 Data Sets*

To verify the performance of the proposed GGA-DCA, twenty-four different classification problems from UCI Machine Learning Repository [32], Keel [33], are used for experiments. Those data sets, shown in Table 1, are selected according to the level of feature dimensions, the size of the dataset, and the level of the imbalance rate. Those data sets are divided into eight categories; Category 1: balanced small-sized low-dimensional data sets; Category 2: imbalanced small-sized low-dimensional data sets; Category 3: balanced large-sized low-dimensional data sets; Category 4: imbalanced large-sized low-dimensional data sets; Category 5: balanced small-sized high-dimensional data sets; Category 6: imbalanced small-sized high-dimensional data sets; Category 7: balanced large-sized high-dimensional data sets; Category 8: imbalanced large-sized high-dimensional data sets. Each category contains three data sets so that the performance of GGA-DCA can be fully tested by performing classification tasks on these 24 data sets. In this work, non-numerical features are transformed into numerical features. Before experiments, this study filters the features with a high percentage of missing values and the features with a single unique value. The data imputation techniques may not work well for data sets with a high rate of missing values. Thus, this study filters those features which are missing more than 40% of data. Those features with a single unique value cannot be useful for machine learning because of their zero variance. Thus, this study counts unique values for each attribute in a data set. The features with one unique value are filtered. Due to the significant effect of influential points on the algorithm, this study replaces these influential points with the mean. Thus, this study utilizes the Z-score method to look for influential points, shown in Eq. (6). This study filters those features whose $z_{i,j}$ exceeds 2.5.

$$z_{i,j} = (x_{i,j} - \boldsymbol{\mu}_j)/\boldsymbol{\sigma}_j, \tag{6}$$

where $x_{i,j}$ is the value of $i_{th}$ data item in $j^{th}$ feature, $\boldsymbol{\mu}_j$ is the mean of the $j^{th}$ feaure, and $\boldsymbol{\sigma}_j$ is the standard deviation of the $j^{th}$ feature.

**Table 1:** Description of data sets

| Category | Data set | Ref. | Attributes | Instances | Imbalance rate | Source |
|---|---|---|---|---|---|---|
| Category 1 | Breast Cancer Wisconsin | BCW | 11 | 700 | 1.9 | UCI |
| | Cervical Cancer Behavior Risk | CCBR | 19 | 72 | 2.4 | UCI |
| | Hepatitis Domain | HE | 20 | 155 | 1.2 | UCI |
| Category 2 | Yeast (Imbalanced: 2 *vs.* 4) | Yeast2 | 8 | 514 | 9.1 | Keel |
| | Abalone (Imbalanced: 18 *vs.* 9) | Abalone18v9 | 8 | 731 | 16.4 | UCI |
| | Glass Identification | Glass | 9 | 214 | 11.6 | Keel |
| Category 3 | Titanic | Titanic | 3 | 2201 | 2.1 | Keel |
| | German Credit Data | GCD | 20 | 1000 | 2.3 | UCI |
| | Mushroom | Mushroom | 23 | 5644 | 1.1 | Keel |
| Category 4 | Yeast (Imbalanced: 3) | Yeast3 | 8 | 1484 | 8.1 | Keel |
| | Abalone (Imbalanced: 19) | Abalone19 | 8 | 4174 | 129.4 | Keel |
| | Page Blocks Classification (Imbalanced: 0) | PBC0 | 10 | 5472 | 8.8 | Keel |
| Category 5 | Divorce Predictors | DP | 54 | 170 | 1 | UCI |
| | Sonar | Sonar | 60 | 208 | 1.1 | UCI |
| | Musk1 | Musk1 | 168 | 476 | 1.3 | Keel |
| Category 6 | KDD Cup (Imbalanced: land *vs.* satan) | KDDls | 41 | 805 | 75.7 | Keel |

(Continued)

**Table 1 (continued)**

| Category | Data set | Ref. | Attributes | Instances | Imbalance rate | Source |
|---|---|---|---|---|---|---|
| | KDD Cup (Imbalanced: guess passwd *vs.* satan) | KDDgps | 41 | 821 | 30 | Keel |
| | KDD Cup (Imbalanced: land *vs.* portsweep) | KDDlp | 41 | 1061 | 49.5 | Keel |
| Category 7 | Spambase | SP | 57 | 4601 | 1.5 | UCI |
| | Elephant | Elephant | 231 | 1391 | 1.2 | Keel |
| | Tiger | Tiger | 231 | 1220 | 1.2 | Keel |
| Category 8 | KDD Cup (Imbalanced: rootkit-imap *vs.* back) | KDDrvb | 41 | 2225 | 100.1 | Keel |
| | Insurance Company Benchmark (COIL 2000) | ICB2000 | 85 | 9822 | 15.8 | Keel |
| | Musk2 | Musk2 | 168 | 6598 | 5.5 | Keel |

### *5.2 Experiment Setup*

In this work, two experiments are performed to study the feasibility and superiority of the proposed approach. In the first experiment, GGA-DCA and state-of-the-art DCA expansion algorithms (NIDDCA [1], FLA-DCA [17], GA-PSM [29], and the SVM-DCA [20]) perform classification tasks on the 24 data sets. For the purpose of baseline comparison, the well-known classifiers, the KNN, the DT, the XGboost, the RF, and the ERT, also perform classification tasks on the 24 data sets. Each data set is divided into two disjoint sets: training (80%) and testing (20%). In all experiments, ten independent runs of each algorithm are conducted based on the 10-fold cross-validation method. To evaluate the performance of the above approaches, the accuracy, precision, specificity, F-measure, and the area under the curve (AUC), are calculated for all the data sets. Accuracy is a proportion of the correct predictions on all items. Precision is a proportion of positive items on all cases labeled normal. The specificity is the percentage of positive cases on all correct cases. F-measure is a harmonic mean of precision and recall. All experiments are performed on a laptop with Intel Core i7-5600U 2.6 GHz-8 GB RAM-HP running Windows 10. Those approaches are implemented in Python using PyCharm.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$specificity = \frac{TP}{TP + FN} \tag{9}$$

$$F - measure = \frac{2 \times Precision \times reacall}{Precision + reacall} \tag{10}$$

### *5.3 Parameters Description*

In this work, the size of the DC poll is 100, and each antigen samples up to 10 data items. The migration threshold is the combination of the weight values and the max signal values using

Eq. (1). For classification, this study adopts the proportion of the abnormal items in a data set as the threshold of MCAV. The weights of DCA are adjusted through the feedback adjustment method of NIDDCA [1]. The iterations ($G$) and the population size of GGA-DCA using GGA are both 10 in each experiment. The probability $P_{cross}$ is set in [0.5, 0.8], and the probability $P_{mutation}$ is set in [0.1, 0.2]. The parameters of GA-PSM are the same as GGA-DCA.

## *5.4 Complexity Analysis*

The GGA-DCA wraps a search task around the DCA. The runtime of DA-DCA depends on the iteration number of GGA, the chromosome number of each generation, and the runtime of DCA. The calculation of the runtime is performed phase by phase. According to work in Gu et al. [34], the runtime complexity of DCA is bounded by $O(n^2)$, the $n$ is the data size. Thus, this study focus on the runtime of the whole GGA-DCA. The Table 2 shows the detail of all the primitive operations of the GGA-DCA. In Table 2, each line contains one operation and the number of times that operation is executed corresponding to Algorithm 1. The runtime complexity of the GGA-DCA is calculated as follows:

$$T(n) = P + G + P \times G \times (1 + n + n \times n + P + P + P \times 2 + P_{cross} \times P + P_{mutation} \times P + P \times 2) \quad (11)$$

where $P$ is the population's size, $G$ is the number of iterations, $n$ is the data size, $P_{cross}$ is the probability of two chromosomes using a crossover operator, $P_{mutation}$ is the probability of two chromosomes using a mutation operator. In this study, the $P$ and the $G$ are in the same magnitude. This study utilizes $P$ instead of $G$. The GGA performs crossover operator and mutation operator with probability $P_{cross}$ and $P_{mutation}$, respectively. This study merely focus on the worst-case scenario, which occurs if $P_{cross} = 1$ and $P_{mutation} = 1$. Therefore, the runtime is calculated as follows:

$$\begin{aligned}
T(n) &= 2 \times P + P^2(1 + P + n + n \times n + P + P + P \times 2 + P_{cross} \times P + P_{mutation} \times P + P \times 2) \\
&\Rightarrow P = G, 2 \times P \ll P^2 \\
T(n) &= P^2(1 + 9 \times P + n + n^2) \\
&\Rightarrow n^2 \gg 1 + 9 \times P + n \\
T(n) &= O(P^2 \times n^2) \quad (12)
\end{aligned}$$

**Table 2:** Details of primitive operations of Algorithm 1, where $P$ is the size of the population, and $G$ is the number of iterations

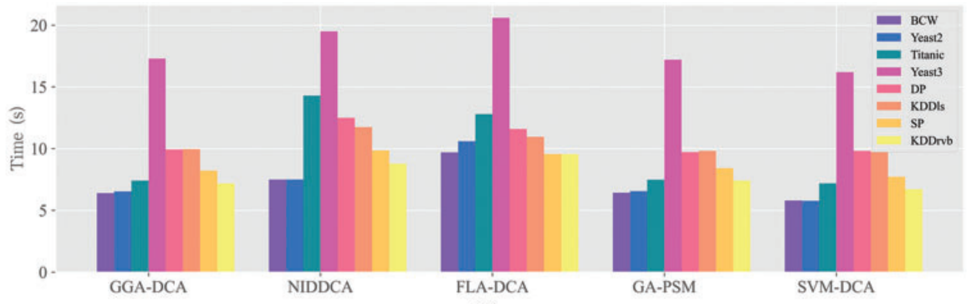
| Line No. | Description | Times |
|---|---|---|
| 1 | P = Initial_Population(m) | $P$ |
| 2 | **While** loop | $G$ |
| 3 | **For** loop | $P \times G$ |
| 4 | Generat_Newdata(Train_data, chromosome) | $n \times G \times P$ |
| 5 | Fitness_value = Accuracy(DCA(Dataset, chromosome)) | $n \times n \times G \times P$ |
| 6 | Save $Fitness_i$ | $P \times G \times P$ |
| 7 | **For** loop | $P \times G \times P$ |
| 8 | **Selection** $p1$ and $p2$ | $P \times 2 \times G \times P$ |

(Continued)

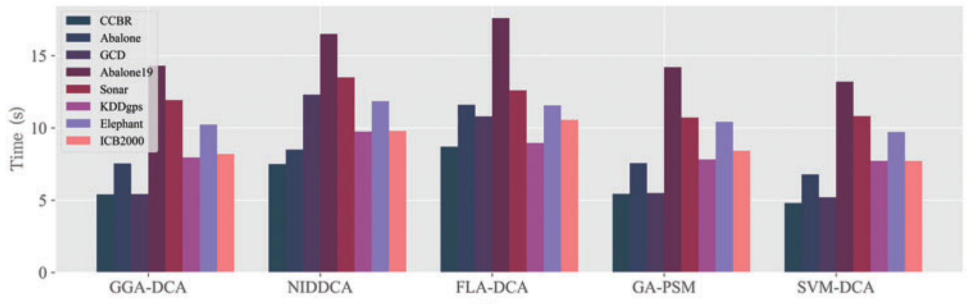
**Table 2** (continued)

| Line No. | Description | Times |
|---|---|---|
| 9 | $c1, c2 = \textbf{Cross}(p1, p2)$ with probability $P_{cross}$ | $P_{cross} \times P \times G \times P$ |
| 10 | $\textbf{Mutation}(c1, c2)$ with probability $P_{mutation}$ | $P_{mutation} \times P \times G \times P$ |
| 11 | **Add** $c1$, $c2$ to $P$ | $P \times 2 \times G \times P$ |

As shown in Eq. (12), GGA-DCA has a worse case runtime complexity of $O(P^2 \times N^2)$. To further verify the performance of GGA-DCA, we calculated the running time of the DCA versions performing classification on the 24 data sets. In this study, three experiments are performed to calculate the running time of the DCA version on 24 data sets. In each experiment, a data set is selected from eight data categories. Fig. 5 illustrates that the GGA-DCA does not maintain the runtime advantage on all the data sets. The reason is that the GGA-DCA performs a search task, and this study focuses on how DCA can automatically find the optimal signal generation scheme. GGA-DCA does not have an advantage over SVM-DAC in terms of runtime. However, when performing classification on the data sets with large data size and high-dimensional feature space, e.g., Mushroom, Elephant, Tiger, ICB2000, and Musk2, the proposed GGA-DCA is superior to other algorithms (NIDDCA, FLA-DCA, and GA-PSM) in terms of runtime. In light of the performance improvement offered by the GGA-DCA, the additional time consumption is deemed acceptable.

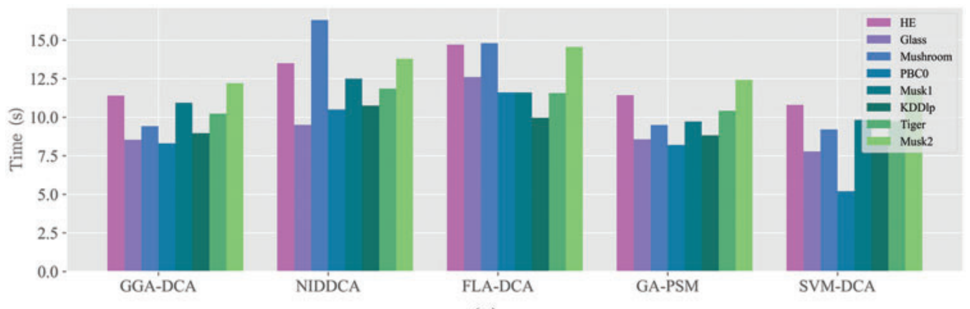


(a)



(b)

**Figure 5:** (Continued)



(c)

**Figure 5:** Mean execution time (in seconds) acquired by DCA versions (e.g., GGA-DCA, NIDDCA, FLADCA, GA-PSM, SVM-DCA)

### *5.5 Results Analysis and Comparison*

For all the data sets, this study computes the mean and standard deviations of the accuracy obtained by the ten algorithms (GGA-DCA, NIDDCA, FLA-DCA, SVM-DCA, GA-PSM, KNN, DT, XGboost, RF, and ERT), and the results are shown in Table 3. The Table 4 shows the mean and standard deviations of precision obtained by those ten algorithms. Moreover, the specificity, F-Measure, and AUC of the ten algorithms are also calculated based on the 24 data sets, and the results are respectively presented in Tables 5, 6 and 7. The number in bold represents the best result among the then algorithms in all the tables.

**Table 3:** Mean accuracy with standard deviation acquired by the ten algorithms

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA | KNN | DT | XGboost | RF | ERT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | BCW | **99.3 ± 0.4** | 90.3 ± 0.7 | 89.7 ± 1.5 | 90.8 ± 1.0 | 83.7 ± 1.4 | 93.4 ± 2.6 | 92.8 ± 3.2 | 95.8 ± 2.1 | 96.0 ± 2.2 | 95.5 ± 2.2 |
| 1 | CCBR | **84.3 ± 1.5** | 72.9 ± 4.0 | 68.0 ± 5.2 | 72.5 ± 4.6 | 68.7 ± 4.0 | 86.2 ± 10.7 | 83.5 ± 15.0 | 89.1 ± 10.1 | 82.0 ± 8.9 | 80.7 ± 12.3 |
| | HE | **88.9 ± 1.0** | 58.4 ± 1.8 | 55.2 ± 1.8 | 57.9 ± 3.4 | 55.2 ± 1.9 | 58.7 ± 10.5 | 58.1 ± 10.0 | 51.7 ± 9.9 | 63.2 ± 11.9 | 58.8 ± 1.0 |
| Category | Yeast2 | **99.1 ± 0.3** | 77.6 ± 1.7 | 73.7 ± 1.8 | 77.9 ± 1.9 | 73.7 ± 1.8 | 94.5 ± 1.9 | 95.1 ± 3.0 | 95.9 ± 2.0 | 96.1 ± 3.0 | 95.1 ± 2.6 |
| 2 | Abalone | **90.4 ± 0.1** | 89.7 ± 0.6 | 89.1 ± 0.4 | 89.4 ± 0.4 | 87.7 ± 0.9 | 94.4 ± 18 | 91.9 ± 2.0 | 94.6 ± 9.0 | 94.9 ± 1.7 | 94.8 ± 1.6 |
| | Glass | **85.9 ± 3.0** | 83.8 ± 1.3 | 82.0 ± 1.5 | 83.8 ± 1.3 | 76.8 ± 1.9 | 81.3 ± 3.9 | 72.8 ± 8.9 | 81.7 ± 9.5 | 83.6 ± 8.1 | 80.8 ± 6.6 |
| Category | Titanic | **79.1 ± 0.3** | 75.6 ± 0.4 | 75.1 ± 0.2 | 75.6 ± 0.2 | 74.9 ± 0.4 | 74.1 ± 3.9 | 79.0 ± 1.2 | 78.7 ± 1.2 | 78.6 ± 2.0 | 79.0 ± 1.2 |
| 3 | GCD | **78.6 ± 0.9** | 53.3 ± 2.4 | 53.5 ± 2.0 | 57.7 ± 1.5 | 50.8 ± 3.2 | 58.5 ± 7.1 | 71.0 ± 2.8 | 75.6 ± 2.4 | 73.7 ± 2.7 | 72.4 ± 3.9 |
| | Mushroom | **99.5 ± 0.0** | 82.5 ± 2.0 | 84.1 ± 0.8 | 87.6 ± 0.9 | 78.6 ± 2.7 | 94.2 ± 0.1 | 94.3 ± 0.0 | 94.2 ± 0.0 | 94.1 ± 0.0 | 93.5 ± 0.0 |
| Category | Yeast3 | **96.0 ± 0.2** | 78.3 ± 0.7 | 77.1 ± 0.4 | 80.6 ± 0.4 | 74.8 ± 0.9 | 93.2 ± 1.1 | 92.5 ± 1.5 | 94.3 ± 1.3 | 94.6 ± 1.6 | 94.4 ± 1.0 |
| 4 | Abalone19 | 98.6 ± 0.1 | 90.0 ± 0.2 | 90.5 ± 0.0 | 90.9 ± 0.1 | 89.4 ± 0.2 | 99.2 ± 0.3 | 98.4 ± 3.0 | **99.2 ± 3.0** | 99.2 ± 4.0 | 99.2 ± 0.4 |
| | PBC0 | **98.2 ± 0.1** | 82.2 ± 1.5 | 83.4 ± 0.8 | 86.5 ± 0.8 | 78.3 ± 2.2 | 95.6 ± 0.6 | 96.6 ± 0.6 | 97.5 ± 0.2 | 97.3 ± 0.4 | 97.2 ± 0.6 |
| Category | DP | 96.0 ± 0.4 | 86.8 ± 1.9 | 86.8 ± 2.5 | 86.9 ± 2.5 | 85.5 ± 1.7 | 97.6 ± 3.9 | 96.4 ± 3.9 | 97.6 ± 3.9 | 97.6 ± 3.9 | **98.2 ± 2.7** |
| 5 | Sonar | **83.4 ± 1.3** | 81.7 ± 3.0 | 76.6 ± 2.6 | 81.6 ± 2.7 | 69.5 ± 3.3 | 84.1 ± 4.3 | 71.7 ± 10.2 | 83.6 ± 9.3 | 81.3 ± 7.7 | 83.2 ± 7.4 |
| | Musk1 | **89.6 ± 0.5** | 80.7 ± 1.0 | 80.1 ± 0.6 | 82.2 ± 1.7 | 78.1 ± 2.0 | 88.0 ± 2.73 | 78.3 ± 4.0 | 89.8 ± 5.0 | 86.7 ± 6.4 | 89.4 ± 4.3 |
| Category | KDDls | 97.3 ± 0.0 | 79.3 ± 0.4 | 78.6 ± 0.2 | 80.2 ± 0.3 | 80.4 ± 0.2 | 99.8 ± 0.2 | **100.0 ± 0.0** | 100.0 ± 0.0 | 99.9 ± 0.2 | 100.0 ± 0.0 |
| 6 | KDDgps | 98.0 ± 0.1 | 89.2 ± 0.8 | 79.8 ± 0.4 | 81.1 ± 0.4 | 79.3 ± 0.3 | 99.9 ± 0.2 | **100.0 ± 0** | 99.8 ± 0.2 | 100.0 ± 0.0 | 99.9 ± 0.2 |
| | KDDlp | 99.2 ± 0.1 | 83.2 ± 1.5 | 82.0 ± 0.5 | 85.3 ± 1.1 | 76.4 ± 3.0 | 99.9 ± 0.3 | **100.0 ± 0.0** | 100.0 ± 0.0 | 99.9 ± 0.3 | 100.0 ± 0.0 |
| Category | SP | **94.7 ± 1.5** | 89.6 ± 0.3 | 89.3 ± 0.1 | 90.8 ± 0.3 | 88.2 ± 0.7 | 80.6 ± 3.0 | 91.6 ± 11 | 95.4 ± 0.9 | 90.3 ± 1.2 | 94.5 ± 1.3 |
| 7 | Elephant | 97.6 ± 0.2 | 84.7 ± 1.3 | 83.8 ± 0.7 | 87.2 ± 1.0 | 84.1 ± 1.0 | 99.7 ± 0.5 | **100.0 ± 0.0** | 100.0 ± 0.0 | 98.2 ± 1.6 | 97.5 ± 1.5 |
| | Tiger | 95.9 ± 0.6 | 86.4 ± 0.8 | 85.7 ± 0.2 | 88.1 ± 0.2 | 85.0 ± 0.6 | 99.8 ± 0.3 | **100.0 ± 0** | 100.0 ± 0 | 96.4 ± 1.5 | 94.9 ± 3.0 |
| Category | KDDrvb | 99.4 ± 0.0 | 85.2 ± 0.6 | 84.7 ± 0.5 | 86.6 ± 1.5 | 83.7 ± 0.6 | **100.0 ± 0.0** | **100.0 ± 0.0** | 100.0 ± 0.0 | 99.9 ± 0.1 | 99.9 ± 0.1 |
| 8 | ICB2000 | **93.93 ± 0.0** | 90.5 ± 0.3 | 90.2 ± 0.3 | 91.6 ± 0.4 | 89.1 ± 0.5 | 93.7 ± 0.3 | 89.6 ± 7.8 | 93.2 ± 5.0 | 92.7 ± 5.3 | 92.4 ± 0.3 |
| | Musk2 | 90.9 ± 0.6 | 83.1 ± 1.0 | 81.0 ± 2.0 | 86.4 ± 0.7 | 80.0 ± 1.0 | **96.9 ± 0.53** | 88.2 ± 0.7 | 93.3 ± 1.1 | 95.5 ± 1.3 | 96.1 ± 0.6 |

**Table 4:** Mean precision with standard deviation acquired by the ten algorithms

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA | KNN | DT | XGboost | RF | ERT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | BCW | **97.9 ± 1.3** | 89.5 ± 2.0 | 88.2 ± 2.5 | 90.5 ± 1.2 | 85.5 ± 2.1 | 94.2 ± 6.4 | 90.5 ± 8.3 | 92.6 ± 7.5 | 93.7 ± 7.3 | 92.5 ± 7.5 |
| 1 | CCBR | 67.8 ± 2.4 | 72.9 ± 4.0 | 68.1 ± 5.2 | **72.5 ± 4.6** | 71.4 ± 9.8 | 69.5 ± 45.5 | 71.9 ± 32.4 | 81.1 ± 32 | 61.2 ± 43.1 | 66.1 ± 44.3 |
| | HE | **98.2 ± 0.0** | 59.4 ± 1.6 | 56.8 ± 1.4 | 59.1 ± 2.8 | 56.8 ± 1.4 | 70.8 ± 14.9 | 63.3 ± 10.7 | 55.4 ± 7.9 | 65.5 ± 14.5 | 63.5 ± 11.3 |
| Category | Yeast2 | **97.39 ± 1.5** | 74.9 ± 7.1 | 72.7 ± 6.9 | 79.2 ± 4.6 | 69.7 ± 8.7 | 91.1 ± 13.6 | 74.32 ± 16.4 | 82.2 ± 13.3 | 92.2 ± 12.0 | 85.2 ± 16.7 |
| 2 | Abalone | 7.4 ± 0.1 | 5.4 ± 0.7 | 4.8 ± 0.4 | 5.1 ± 0.4 | 3.9 ± 0.5 | 6.6 ± 19.9 | 27.9 ± 19.3 | **54.6 ± 34.1** | 39.6 ± 48.6 | 19.9 ± 39.8 |
| | Glass | **86.4 ± 0.02** | 84.4 ± 4.8 | 82.2 ± 4.0 | 83.5 ± 6.1 | 76.4 ± 4.5 | 89.2 ± 11.2 | 60.3 ± 11.9 | 74.6 ± 13.7 | 84.5 ± 11.7 | 84.6 ± 14.8 |
| Category | Titanic | 60.1 ± 0.3 | 57.1 ± 0.3 | 59.4 ± 0.4 | 60.3 ± 0.4 | 59.2 ± 0.6 | 77.8 ± 18.5 | 93.1 ± 4.0 | 90.7 ± 4.1 | 89.7 ± 8.8 | **93.1 ± 4.0** |
| 3 | GCD | 76.6 ± 0.7 | 53.2 ± 2.5 | 54.1 ± 1.9 | 57.4 ± 1.9 | 50.5 ± 3.6 | 77.0 ± 5.5 | 78.5 ± 2.2 | **80.2 ± 1.6** | 80.0 ± 1.9 | 79.3 ± 2.1 |
| | Mushroom | **99.93 ± 0.0** | 82.5 ± 2.1 | 83.5 ± 0.6 | 87.3 ± 1.1 | 78.4 ± 2.4 | 89.9 ± 0.15 | 89.7 ± 0.0 | 89.6 ± 0.0 | 86.8 ± 0.0 | 90.1 ± 0.0 |
| Category | Yeast3 | 73.2 ± 0.9 | 33.7 ± 0.8 | 32.3 ± 0.4 | 36.2 ± 0.4 | 30.4 ± 0.8 | 84.6 ± 10.5 | 67.6 ± 9.7 | 76.4 ± 6.8 | 85.1 ± 9.8 | **87.1 ± 9.1** |
| 4 | Abalone19 | 0.0 ± 0.0 | 5.2 ± 0 | 5.4 ± 0 | **5.6 ± 0.1** | 4.8 ± 0.1 | 0.0 ± 0.0 | 4.49 ± 9.1 | 4.9 ± 14.9 | 0.0 ± 0.0 | 0.0 ± 0.0 |
| | PBC0 | 84.9 ± 0.4 | 82.1 ± 2.0 | 83.7 ± 2.0 | 86.3 ± 1.6 | 79.0 ± 3.0 | **89.8 ± 2.4** | 83.8 ± 4.8 | 89.2 ± 2.6 | 89.5 ± 3.7 | 89.7 ± 3.3 |
| Category | DP | 94.8 ± 0.5 | 87.5 ± 2.3 | 86.1 ± 4.2 | 87.0 ± 2.3 | 85.0 ± 3.4 | 99.8 ± 0.3 | 95.3 ± 5.6 | 99.8 ± 0.1 | 99.8 ± 0.1 | **99.9 ± 0.0** |
| 5 | Sonar | 76.3 ± 1.5 | 80.7 ± 4.0 | 76.6 ± 3.0 | 83.1 ± 2.6 | 69.9 ± 3.9 | **88.0 ± 6.9** | 73.4 ± 12.4 | 84.6 ± 11.9 | 86.6 ± 10.4 | 87.9 ± 10.7 |
| | Musk1 | 80.7 ± 0.9 | 80.3 ± 2.3 | 80.5 ± 2.0 | 82.3 ± 2.3 | 79.1 ± 2.3 | **90.9 ± 5.1** | 74.2 ± 6.1 | 89.7 ± 6.8 | 90.9 ± 7.2 | 86.6 ± 8.0 |
| Category | KDDls | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | **96.1 ± 0.1** | 99.5 ± 0.1 | 99.5 ± 0.1 | 99.4 ± 0.2 | 99.5 ± 0.1 |
| 6 | KDDgps | 62.3 ± 1.8 | 85.8 ± 4.5 | 7.1 ± 0.0 | 7.9 ± 0.0 | 11.5 ± 0.2 | 98.1 ± 5 | **99.7 ± 0.1** | 98.1 ± 4.9 | 99.7 ± 0.1 | 99.7 ± 0.1 |
| | KDDlp | 71.5 ± 2.2 | 86.6 ± 7.6 | 79.4 ± 8.5 | 85.6 ± 6.7 | 74.7 ± 7.4 | 96.2 ± 10.0 | 99.5 ± 0.1 | **99.52 ± 0.1** | 99.46 ± 0.2 | 99.52 ± 0.1 |
| Category | SP | 89.4 ± 1.3 | 89.6 ± 0.7 | 89.2 ± 0.6 | 90.8 ± 0.5 | 88.3 ± 1.1 | 83.8 ± 3.5 | 89.5 ± 1.6 | 94.3 ± 1.3 | **94.9 ± 1.5** | 94.7 ± 1.3 |
| 7 | Elephant | 95.9 ± 0.3 | 84.1 ± 1.3 | 84.1 ± 1.4 | 87.4 ± 1.6 | 84.6 ± 1.2 | 99.7 ± 0.5 | **100.0 ± 0.0** | 100.0 ± 0.0 | 99.4 ± 0.9 | 98.2 ± 1.5 |
| | Tiger | 95.6 ± 1.1 | 86.7 ± 1.0 | 85.9 ± 0.8 | 87.5 ± 0.9 | 84.7 ± 1.1 | **100.0 ± 0.0** | 100.0 ± 0.0 | 100.0 ± 0.0 | 96.6 ± 2.1 | 96.1 ± 3.1 |
| Category | KDDrvb | 66.0 ± 2.8 | 80.8 ± 5.7 | 88.6 ± 6.5 | 88.1 ± 9.8 | 85.4 ± 10.1 | **99.5 ± 0.1** | 99.5 ± 0.1 | 99.5 ± 0.1 | 99.4 ± 0.2 | 99.4 ± 0.2 |
| 8 | ICB2000 | 12.5 ± 3.4 | 38.7 ± 0.9 | 30.4 ± 1.0 | **40.3 ± 1.4** | 27.3 ± 1.4 | 31.9 ± 11.9 | 13.4 ± 3.8 | 24.8 ± 12.3 | 17.8 ± 7.9 | 17.5 ± 7.7 |
| | Musk2 | 90.93 ± 5.4 | 82.9 ± 2.0 | 80.0 ± 2.0 | 87.0 ± 1.3 | 79.7 ± 2.0 | **97.6 ± 2.0** | 65.8 ± 3.3 | 89.3 ± 3.5 | 96.3 ± 1.9 | 96.3 ± 1.6 |

**Table 5:** Mean specificity with standard deviation acquired by the ten algorithms

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA | KNN | DT | XGboost | RF | ERT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | BCW | **98.9 ± 0.7** | 94.3 ± 1.0 | 93.6 ± 1.3 | 94.8 ± 0.6 | 91.6 ± 1.2 | 97.7 ± 2.3 | 95.5 ± 3.5 | 96.6 ± 2.9 | 97.3 ± 2.6 | 96.6 ± 2.8 |
| 1 | CCBR | 82.6 ± 1.8 | 87.6 ± 3.0 | 85.0 ± 4.0 | 87.3 ± 4.0 | 85.3 ± 4.0 | **99.7 ± 0.1** | 86.1 ± 18.0 | 95.7 ± 8.0 | 95.8 ± 8.0 | 95.7 ± 8.0 |
| | HE | **98.8 ± 0.0** | 36.4 ± 0.4 | 29.3 ± 4.0 | 35.3 ± 7.5 | 29.2 ± 4.1 | 79.8 ± 11.4 | 54.2 ± 18.9 | 41.3 ± 16.2 | 62.7 ± 11.3 | 54.2 ± 17.8 |
| Category | Yeast2 | 99.72 ± 0.2 | 96.5 ± 1 | 96.1 ± 0.9 | 97.1 ± 0.7 | 95.6 ± 1.2 | 99.1 ± 1.5 | 97.1 ± 1.4 | 97.8 ± 1.7 | **99.1 ± 1.4** | 98.4 ± 1.9 |
| 2 | Abalone | 90.3 ± 0.1 | 94.8 ± 0.7 | 94.2 ± 0.4 | 94.5 ± 0.4 | 92.7 ± 1.0 | 99.8 ± 0.5 | 95.1 ± 2.2 | 98.2 ± 1.3 | **100.0 ± 0.0** | 100.0 ± 0.0 |
| | Glass | 92.0 ± 1.1 | 90.8 ± 2.3 | 89.4 ± 2 | 90.4 ± 3.1 | 85.6 ± 2.2 | **95.5 ± 4.7** | 75.9 ± 8.9 | 86.1 ± 8.2 | 93.4 ± 5 | 92.6 ± 7.1 |
| Category | Titanic | 69.1 ± 0.4 | 64.0 ± 0.5 | 76.5 ± 0.3 | 77.4 ± 0.4 | 76.4 ± 0.6 | 96.2 ± 3.3 | 98.6 ± 0.8 | 98.1 ± 0.8 | 97.7 ± 2.3 | **98.6 ± 0.8** |
| 3 | GCD | 28.6 ± 3.0 | 33.0 ± 2.2 | 32.8 ± 2 | 37.0 ± 1.5 | 30.9 ± 2.6 | **60.3 ± 8.6** | 48.3 ± 8.1 | 49.9 ± 5.9 | 51.3 ± 6.5 | 50.3 ± 4.8 |
| | Mushroom | **99.93 ± 0.0** | 88.4 ± 1.5 | 89.2 ± 0.4 | 91.8 ± 0.7 | 85.4 ± 1.8 | 92.7 ± 0.1 | 93.5 ± 0.0 | 93.5 ± 0.0 | 95.4 ± 0.0 | 95.6 ± 0.0 |
| Category | Yeast3 | 95.5 ± 0.2 | 75.6 ± 1.0 | 74.2 ± 0.5 | 78.2 ± 0.5 | 71.8 ± 1.0 | **98.7 ± 8.4** | 95.9 ± 1.7 | 97.1 ± 1.2 | 98.5 ± 1.0 | 98.6 ± 1.3 |
| 4 | Abalone19 | 99.37 ± 0.0 | 90.1 ± 0.1 | 90.7 ± 0.0 | 91.0 ± 0.1 | 89.5 ± 0.2 | **100.0 ± 0.0** | 99.1 ± 0.3 | 99.9 ± 0.0 | 100.0 ± 0.0 | 99.9 ± 0.1 |
| | PBC0 | 97.9 ± 0.1 | 97.5 ± 0.3 | 97.8 ± 0.2 | 98.2 ± 0.2 | 97.3 ± 0.4 | 99.1 ± 0.2 | 98.1 ± 0.8 | 98.8 ± 0.4 | 98.8 ± 0.0 | **98.9 ± 0.4** |
| Category | DP | 95.06 ± 0.4 | 87.6 ± 1.8 | 86.7 ± 3.5 | 87.2 ± 2.2 | 85.5 ± 2.7 | **99.9 ± 0.3** | 95.4 ± 5.4 | 99.9 ± 0 | 99.9 ± 0 | 99.9 ± 0 |
| 5 | Sonar | 64.2 ± 2.9 | 79.0 ± 3.4 | 74.1 ± 2.6 | 80.5 ± 3.0 | 66.7 ± 3.4 | 86.2 ± 4.1 | 66.1 ± 17.5 | 79.3 ± 18.9 | 84.5 ± 12.9 | **85.7 ± 12.8** |
| | Musk1 | 81.6 ± 1.0 | 84.3 ± 1.5 | 84.2 ± 1.1 | 85.8 ± 1.6 | 82.8 ± 1.9 | **93.6 ± 3.7** | 78.7 ± 7.0 | 92.2 ± 5.3 | 93.6 ± 4.5 | 89.2 ± 7.1 |
| Category | KDDls | **98.6 ± 0.0** | 80.3 ± 0.4 | 79.6 ± 0.2 | 81.2 ± 0.3 | 81.5 ± 0.2 | 99.9 ± 0.2 | **100.0 ± 0.0** | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| 6 | KDDgps | 98.0 ± 0.1 | 99.5 ± 0.1 | 80.9 ± 0.3 | 82.3 ± 0.3 | 79.2 ± 0.3 | 99.9 ± 0.0 | **100.0 ± 0.0** | 99.94 ± 0.2 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| | KDDlp | 99.2 ± 0.01 | 99.6 ± 0.2 | 99.4 ± 0.2 | 99.6 ± 0.2 | 99.3 ± 0.2 | 99.9 ± 0.3 | **100.0 ± 0.0** | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| Category | SP | 92.4 ± 1.3 | 93.0 ± 0.4 | 92.7 ± 0.3 | 93.8 ± 0.2 | 92.0 ± 0.7 | 92.1 ± 4.4 | 93.2 ± 1.1 | 96.3 ± 1.0 | **97.2 ± 1.0** | 96.7 ± 0.8 |
| 7 | Elephant | 94.9 ± 0.4 | 81.6 ± 1.4 | 81.3 ± 1.3 | 85.1 ± 1.6 | 81.7 ± 1.2 | 99.6 ± 0.6 | **100.0 ± 0.0** | 100.0 ± 0.0 | 99.3 ± 1.1 | 98.5 ± 1.8 |
| | Tiger | 96.47 ± 0.9 | 89.0 ± 0.8 | 88.3 ± 0.5 | 89.8 ± 0.6 | 87.4 ± 0.7 | **100.0 ± 0.0** | 100.0 ± 0.0 | 100.0 ± 0.0 | 97.3 ± 1.7 | 97.0 ± 2.6 |
| Category | KDDrvb | 99.48 ± 0.0 | 99.7 ± 0.0 | 99.8 ± 0.0 | 99.8 ± 0.1 | 99.8 ± 0.1 | **100.0 ± 0.0** | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| 8 | ICB2000 | **99.9 ± 0.0** | 89.9 ± 0.4 | 92.9 ± 0.3 | 92.0 ± 0.4 | 91.7 ± 0.6 | 99.3 ± 0.3 | 94.5 ± 0.8 | 98.6 ± 0.4 | 98.1 ± 0.4 | 97.7 ± 0.4 |
| | Musk2 | 98.2 ± 0.0 | 96.3 ± 0.4 | 95.6 ± 0.4 | 97.3 ± 0.3 | 95.6 ± 0.5 | 99.6 ± 0.3 | 95.3 ± 0.6 | 98.6 ± 0.5 | **99.4 ± 0.2** | 99.4 ± 0.2 |

**Table 6:** Mean F-Measure with standard deviation acquired by the ten algorithms

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA | KNN | DT | XGboost | RF | ERT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category 1 | BCW | **98.5 ± 0.6** | 86.5 ± 1.9 | 85.0 ± 2.2 | 86.6 ± 1.3 | 77.7 ± 2.0 | 89.0 ± 4.1 | 88.5 ± 4.9 | 93.0 ± 4.2 | 93.1 ± 4.3 | 92.5 ± 4.4 |
| | CCBR | **76.72 ± 2** | 61.4 ± 4.5 | 56.1 ± 6.7 | 60.8 ± 5.7 | 56.4 ± 6.3 | 57.3 ± 39.7 | 70.6 ± 28.8 | 73.5 ± 29.6 | 50.4 ± 35.1 | 50.7 ± 34.6 |
| | HE | **96.2 ± 1.0** | 66.4 ± 1.0 | 64.7 ± 1.0 | 66.2 ± 1.8 | 64.7 ± 1.0 | 50.8 ± 14.4 | 61.2 ± 7.3 | 56.6 ± 10.0 | 63.2 ± 17.1 | 61.5 ± 10.0 |
| Category 2 | Yeast2 | **95.7 ± 1.7** | 84.1 ± 0.6 | 50.0 ± 0.9 | 48.6 ± 0.5 | 52.8 ± 0.5 | 63.7 ± 15.3 | 73.5 ± 17.9 | 77.5 ± 13.2 | 75.5 ± 18.5 | 71.3 ± 15.2 |
| | Abalone | 13.7 ± 1.7 | 4.6 ± 0.3 | 4.3 ± 0.2 | 4.5 ± 0.2 | 3.8 ± 0.3 | 4.9 ± 14.8 | 29.27 ± 21.1 | **36.4 ± 21.8** | 32.4 ± 27.8 | 10.6 ± 21.3 |
| | Glass | **80.9 ± 0.7** | 78.2 ± 2.1 | 75.9 ± 2.0 | 78.0 ± 2.5 | 69.6 ± 2.4 | 66.9 ± 7.3 | 62.1 ± 14 | 73.0 ± 13.7 | 72.4 ± 13.6 | 66.6 ± 13.1 |
| Category 3 | Titanic | **75.1 ± 1.2** | 72.1 ± 0.1 | 64.6 ± 0.2 | 65.0 ± 0.2 | 64.4 ± 0.4 | 38.9 ± 15.1 | 53.4 ± 3.5 | 53.3 ± 3.4 | 53.5 ± 3.4 | 53.4 ± 3.5 |
| | GCD | **86.2 ± 0.5** | 60.9 ± 2.3 | 61.5 ± 1.8 | 65.0 ± 1.5 | 58.4 ± 3.2 | 65.4 ± 6.5 | 79.0 ± 2.3 | 82.6 ± 2.1 | 81.0 ± 2.1 | 80.0 ± 3.1 |
| | Mushroom | **98.9 ± 0.0** | 77.8 ± 2.5 | 79.4 ± 0.9 | 83.8 ± 1.1 | 73.2 ± 3.1 | 91.4 ± 0.1 | 92.5 ± 0.0 | 91.6 ± 0.0 | 92.3 ± 0.0 | 92.6 ± 0.0 |
| Category 4 | Yeast3 | 11.0 ± 0.4 | 50.0 ± 0.9 | 48.6 ± 0.5 | 52.7 ± 0.5 | 46.3 ± 0.9 | 59.9 ± 9.6 | 65.3 ± 7.1 | **72.7 ± 6.2** | 71.37 ± 9.5 | 68.9 ± 6.8 |
| | Abalone19 | 0.5 ± 0.0 | 9.4 ± 0.1 | 10.0 ± 0.7 | 10.2 ± 0.1 | 8.9 ± 0.2 | **16.7 ± 21.7** | 0.0 ± 0.0 | 4.3 ± 8.7 | 4.9 ± 14.5 | 0.0 ± 0.0 |
| | PBC0 | **91.3 ± 0.2** | 48.1 ± 2.5 | 50.3 ± 1.4 | 56.3 ± 1.9 | 42.4 ± 3.1 | 74.7 ± 4.0 | 82.7 ± 3.0 | 87.0 ± 1.4 | 86.3 ± 2.1 | 85.3 ± 4.0 |
| Category 5 | DP | 95.5 ± 0.2 | 86.3 ± 1.8 | 86.0 ± 2.7 | 86.3 ± 2.4 | 84.8 ± 1.8 | 96.7 ± 4.6 | 95.8 ± 4.1 | 96.7 ± 4.6 | 96.7 ± 4.6 | **97.4 ± 2.9** |
| | Sonar | **86.1 ± 0.9** | 82.0 ± 2.9 | 77.2 ± 2.3 | 82.3 ± 2.5 | 70.5 ± 3.1 | 84.0 ± 4.1 | 73.7 ± 8.7 | 84.9 ± 7.4 | 81.0 ± 7.5 | 82.9 ± 7.9 |
| | Musk1 | **88.8 ± 0.5** | 77.9 ± 1.2 | 77.4 ± 0.6 | 79.6 ± 1.8 | 75.4 ± 2.1 | 84.8 ± 3.6 | 75.1 ± 44.9 | 87.5 ± 6.0 | 83.4 ± 9.1 | 86.5 ± 6.0 |
| Category 6 | KDDls | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 0.0 ± 0.0 | 93.7 ± 11 | **99.0 ± 0.1** | 99.0 ± 0.1 | 95.7 ± 10 | 99.0 ± 0.1 |
| | KDDgps | 76.3 ± 1.5 | 33.8 ± 2.6 | 11.9 ± 0.2 | 12.7 ± 0.2 | 20.0 ± 0.3 | 98.3 ± 2.7 | 99.3 ± 0.0 | 97.4 ± 3.6 | **99.3 ± 0** | 98.1 ± 3.3 |
| | KDDlp | 83.4 ± 1.5 | 16.8 ± 1.6 | 14.7 ± 1.2 | 18.7 ± 2.1 | 11.1 ± 1.8 | 97.0 ± 6 | **99.0 ± 0.1** | 99.0 ± 0.1 | 95.7 ± 10 | 99.0 ± 0.1 |
| Category 7 | SP | **93.6 ± 1.2** | 86.7 ± 0.4 | 86.3 ± 0.2 | 88.1 ± 0.3 | 85.0 ± 0.8 | 71.2 ± 4.3 | 88.8 ± 1.4 | 93.6 ± 1.1 | 86.2 ± 1.8 | 93.3 ± 1.5 |
| | Elephant | 97.9 ± 0.1 | 85.2 ± 1.2 | 84.5 ± 0.8 | 87.7 ± 1.0 | 84.9 ± 0.9 | 99.2 ± 0.4 | 99.5 ± 0.1 | **99.5 ± 0.0** | 97.8 ± 1.5 | 97.1 ± 1.4 |
| | Tiger | 94.9 ± 0.7 | 84.6 ± 8.2 | 83.7 ± 2.7 | 86.3 ± 0.3 | 83.0 ± 0.7 | 99.3 ± 0.4 | **100.0 ± 0.0** | 100.0 ± 0.0 | 95.4 ± 1.7 | 93.1 ± 3.6 |
| Category 8 | KDDrvb | 79.5 ± 2 | 9.7 ± 0.9 | 10.2 ± 0.8 | 11.5 ± 1.8 | 9.3 ± 1.2 | 99.0 ± 0.1 | **99.0 ± 0.1** | 99.0 ± 0.1 | 95.7 ± 10.0 | 95.7 ± 10 |
| | ICB2000 | **56 ± 0.2** | 55.4 ± 0.95 | 37.0 ± 0.6 | 54.2 ± 1.3 | 34.5 ± 1.1 | 7.2 ± 2.4 | 12.7 ± 3.8 | 10.2 ± 6.3 | 8.9 ± 4.8 | 10.6 ± 5.8 |
| | Musk2 | 75.0 ± 1.3 | 59.8 ± 1.8 | 56.1 ± 2.7 | 65.9 ± 1.2 | 54.7 ± 1.9 | 88.8 ± 2.0 | 55.3 ± 4.1 | 74.1 ± 5.3 | **82.8 ± 5.6** | 85.7 ± 2.7 |

**Table 7:** Mean AUC with standard deviation acquired by the ten algorithms

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA | KNN | DT | XGboost | RF | ERT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category 1 | BCW | **0.98 ± 0.006** | 0.85 ± 0.064 | 0.88 ± 0.016 | 0.89 ± 0.011 | 0.81 ± 0.015 | 0.91 ± 0.03 | 0.91 ± 0.033 | 0.95 ± 0.02 | 0.95 ± 0.021 | 0.95 ± 0.022 |
| | CCBR | **0.81 ± 0.015** | 0.70 ± 0.034 | 0.66 ± 0.050 | 0.69 ± 0.043 | 0.66 ± 0.044 | 0.75 ± 0.195 | 0.81 ± 0.18 | 0.83 ± 0.166 | 0.71 ± 0.166 | 0.72 ± 0.176 |
| | HE | **0.89 ± 0.100** | 0.57 ± 0.022 | 0.54 ± 0.025 | 0.57 ± 0.041 | 0.53 ± 0.024 | 0.61 ± 0.103 | 0.57 ± 0.107 | 0.51 ± 0.101 | 0.63 ± 0.115 | 0.58 ± 0.103 |
| Category 2 | Yeast2 | **0.97 ± 0.015** | 0.62 ± 0.017 | 0.59 ± 0.012 | 0.62 ± 0.015 | 0.59 ± 0.020 | 0.76 ± 0.091 | 0.87 ± 0.12 | 0.88 ± 0.103 | 0.84 ± 0.114 | 0.81 ± 0.096 |
| | Abalone | 0.53 ± 0.000 | 0.49 ± 0.004 | 0.49 ± 0.002 | 0.49 ± 0.002 | 0.49 ± 0.003 | 0.51 ± 0.057 | 0.65 ± 0.138 | **0.64 ± 0.109** | 0.55 ± 0.074 | 0.54 ± 0.074 |
| | Glass | **0.84 ± 0.004** | 0.82 ± 0.014 | 0.80 ± 0.016 | 0.82 ± 0.014 | 0.75 ± 0.019 | 0.75 ± 0.043 | 0.71 ± 0.105 | 0.79 ± 0.105 | 0.79 ± 0.098 | 0.75 ± 0.084 |
| Category 3 | Titanic | **0.80 ± 0.002** | 0.78 ± 0.005 | 0.77 ± 0.001 | 0.72 ± 0.002 | 0.72 ± 0.004 | 0.61 ± 0.06 | 0.68 ± 0.017 | 0.68 ± 0.017 | 0.68 ± 0.019 | 0.68 ± 0.017 |
| | GCD | **0.88 ± 0.003** | 0.53 ± 0.022 | 0.52 ± 0.019 | 0.57 ± 0.013 | 0.51 ± 0.026 | 0.59 ± 0.074 | 0.64 ± 0.033 | 0.68 ± 0.023 | 0.67 ± 0.031 | 0.66 ± 0.037 |
| | Mushroom | **0.99 ± 0.000** | 0.81 ± 0.021 | 0.83 ± 0.008 | 0.86 ± 0.009 | 0.77 ± 0.027 | 0.93 ± 0.04 | 0.93 ± 0.000 | 0.92 ± 0.000 | 0.94 ± 0.000 | 0.95 ± 0.000 |
| Category 4 | Yeast3 | **0.86 ± 0.006** | 0.66 ± 0.004 | 0.66 ± 0.002 | 0.68 ± 0.002 | 0.65 ± 0.004 | 0.73 ± 0.056 | 0.80 ± 0.051 | 0.84 ± 0.046 | 0.80 ± 0.059 | 0.79 ± 0.054 |
| | Abalone19 | 0.50 ± 0.001 | **0.52 ± 0.001** | 0.52 ± 0.001 | 0.52 ± 0.001 | 0.52 ± 0.001 | 0.50 ± 0.000 | 0.52 ± 0.045 | 0.52 ± 0.075 | 0.50 ± 0.000 | 0.50 ± 0.001 |
| | PBC0 | **0.92 ± 0.018** | 0.66 ± 0.012 | 0.67 ± 0.007 | 0.70 ± 0.010 | 0.63 ± 0.015 | 0.82 ± 0.028 | 0.90 ± 0.030 | 0.92 ± 0.020 | 0.91 ± 0.030 | 0.90 ± 0.030 |
| Category 5 | DP | 0.96 ± 0.002 | 0.86 ± 0.018 | 0.86 ± 0.024 | 0.87 ± 0.025 | 0.85 ± 0.017 | 0.97 ± 0.041 | 0.96 ± 0.038 | 97.56 ± 0.041 | 0.97*pm*0.041 | **0.98 ± 0.028** |
| | Sonar | **0.88 ± 0.013** | 0.81 ± 0.029 | 0.76 ± 0.027 | 0.81 ± 0.028 | 0.69 ± 0.034 | 0.84 ± 0.0447 | 0.71 ± 0.104 | 0.83 ± 0.097 | 0.81 ± 0.076 | 0.83 ± 0.075 |
| | Musk1 | **0.90 ± 0.004** | 0.80 ± 0.010 | 0.79 ± 0.006 | 0.81 ± 0.017 | 0.77 ± 0.020 | 0.87 ± 0.028 | 0.78 ± 0.041 | 0.89 ± 0.051 | 0.85 ± 0.072 | 0.88 ± 0.043 |
| Category 6 | KDDls | 0.49 ± 0.000 | 0.49 ± 0.000 | 0.49 ± 0.000 | 0.49 ± 0.006 | 0.49 ± 0.003 | 0.97 ± 0.074 | **1.00 ± 0.000** | 1.00 ± 0.000 | 0.97 ± 0.075 | 1.00 ± 0.000 |
| | KDDgps | 0.98 ± 0.007 | 0.60 ± 0.009 | 0.52 ± 0.000 | 0.52 ± 0.000 | 0.55 ± 0.001 | 0.99 ± 0.001 | **1.00 ± 0.000** | 0.99 ± 0.025 | 1.00 ± 0.000 | 0.99 ± 0.030 |
| | KDDlp | 0.99 ± 0.000 | 0.55 ± 0.005 | 0.54 ± 0.004 | 0.55 ± 0.007 | 0.52 ± 0.006 | 0.99 ± 0.001 | **1.00 ± 0.000** | 1.00 ± 0.000 | 97.5 ± 0.075 | 1.00 ± 0.000 |
| Category 7 | SP | 0.94 ± 0.010 | 0.88 ± 0.003 | 0.88 ± 0.002 | 0.90 ± 0.003 | 0.87 ± 0.007 | 0.77 ± 0.030 | 0.91 ± 0.01 | **0.95 ± 0.010** | 0.88 ± 0.015 | 0.93 ± 0.016 |
| | Elephant | 0.97 ± 0.001 | 0.84 ± 0.012 | 0.83 ± 0.008 | 0.87 ± 0.010 | 0.84 ± 0.009 | 0.99 ± 0.005 | **1.00 ± 0.000** | 1.00 ± 0.00 | 0.98 ± 0.016 | 0.97 ± 0.015 |
| | Tiger | 0.96 ± 0.600 | 0.86 ± 0.007 | 0.85 ± 0.002 | 0.87 ± 0.002 | 0.84 ± 0.007 | 0.99 ± 0.004 | **1.00 ± 0.000** | 1.00 ± 0.000 | 0.96 ± 0.015 | 0.94 ± 0.032 |
| Category 8 | KDDrvb | 0.99 ± 0.000 | 0.52 ± 0.003 | 0.52 ± 0.002 | 0.53 ± 0.006 | 0.52 ± 0.004 | **1.00 ± 0.000** | 1.00 ± 0.000 | 1.00 ± 0.000 | 0.97 ± 0.075 | 0.97 ± 0.075 |
| | ICB2000 | 0.53 ± 0.017 | **0.69 ± 0.005** | 0.63 ± 0.004 | 0.69 ± 0.007 | 0.62 ± 0.007 | 0.51 ± 0.007 | 0.53 ± 0.020 | 0.52 ± 0.021 | 0.52 ± 0.017 | 0.52 ± 0.022 |
| | Musk2 | 0.84 ± 0.038 | 0.71 ± 0.010 | 0.69 ± 0.016 | 0.75 ± 0.007 | 0.69 ± 0.011 | 0.91 ± 0.016 | 0.72 ± 0.024 | 0.81 ± 0.035 | 0.86 ± 0.038 | **0.88 ± 0.022** |

The Tables 3–7 show that the proposed GGA-DCA consistently achieves better performance on the 24 data sets compared with DCA versions (e.g., NIDDCA, FLA-DCA, SVM-DCA, and GA-PSM). We can conclude that the proposed GGA-DCA is superior to the state-of-the-art DCA versions (e.g., NIDDCA, FLA-DCA, SVM-DCA, and GA-PSM) over all the UCI and Keel data sets in a statistically significant manner. Moreover, the Table 3 also shows that the GGA-DCA obtains better classification accuracy compared with those machine learning algorithms, e.g., KNN, DT, XGboost, RF, and ERT, on those data sets with low-dimensional feature space (e.g., BCW, CCBR, HE, Yeast2, Abalone, Glass, Titanic, GDD, Yeast3, and PBC0). When performing classification on those data sets with high-dimensional feature space (e.g., Musk2, Tiger, and Elephant), the proposed GGA-DCA has not obtained the same satisfactory results as machine learning algorithms. However, our method maintains the same advantage as those machine learning algorithms in terms of AUC.

To better analyze the results, this study tests the following hypotheses using the *t*-test to analyze whether significant differences exist in the experiments between the GGA-DCA and other signal acquisition algorithms of DCA (called "Comparisons") under the condition $z = 0.05$.

$$\begin{aligned} &H_0 : \mu^{GGA-DCA} = \mu^{Comparisons} \\ &H_0 : \mu^{GGA-DCA} \pm \mu^{Comparisons} \end{aligned} \tag{14}$$

The Table 8 shows the *t*-test results on accuracy. When the degree of freedom is nine, and the significance level of the *t*-test is 0.05, the critical *t*-value is 2.262. Therefore, if the result is below 2.262, we can conclude that $H_0$; otherwise, we can determine that significant differences exist. The Table 8 illuminates all the *t*-value exceeds 2.262. We can conclude that in terms of classification accuracy, our algorithm GGA-DCA and the other signal acqusition algorithms of DCA exhibit significant differences on all the test problems. This accordingly proves once again, from a statistical point of view, that our algorithm is the best in all test problems compared with the DCA expansion algorithms.

**Table 8:** *t*-test results of the signal acquisition algorithms of DCA on accuracy

| Category | Data set | GGA-DCA | NIDDCA | FLA-DCA | GA-PSM | SVM-DCA |
|---|---|---|---|---|---|---|
| Category 1 | BCW | - | 47.9157 | 60.0471 | 52.8680 | 25.8835 |
| | CCBR | - | 6.50704 | 7.0018 | 7.0719 | 8.7406 |
| | HE | - | 78.2049 | 34.3254 | 24.8921 | 33.3283 |
| Category 2 | Yeast2 | - | 32.2691 | 49.3114 | 32.7919 | 47.7141 |
| | Abalone | - | 3.7366 | 7.5889 | 5.7975 | 9.8027 |
| | Glass | - | 2.6940 | 8.6605 | 2.9715 | 10.8121 |
| Category 3 | Titanic | - | 26.2637 | 48.1639 | 40.1102 | 27.7611 |
| | GCD | - | 30.2491 | 34.9082 | 38.2404 | 25.7767 |
| | Mushroom | - | 23.8558 | 55.7381 | 41.2815 | 23.1552 |
| Category 4 | Yeast3 | - | 65.1644 | 126.8233 | 112.3292 | 63.1638 |
| | Abalone19 | - | 159.4095 | 238.6075 | 175.9318 | 109.3993 |
| | PBC0 | - | 31.9973 | 54.3658 | 42.3448 | 27.6673 |
| Category 5 | DP | - | 11.2283 | 8.8496 | 8.9713 | 15.8397 |
| | Sonar | - | 2.0293 | 9.3825 | 1.7648 | 12.2132 |
| | Musk1 | - | 21.9602 | 30.3061 | 14.3279 | 15.8293 |
| Category 6 | KDDls | - | 204.5178 | 234.7499 | 116.5525 | 182.2177 |
| | KDDgps | - | 36.8735 | 158.6763 | 127.6900 | 134.5714 |
| | KDDlp | - | 30.8391 | 74.9685 | 32.7716 | 21.5856 |
| Category 7 | SP | - | 44.0675 | 107.0000 | 48.1708 | 27.9601 |
| | Elephant | - | 29.4309 | 53.1809 | 29.5649 | 38.6331 |
| | Tiger | - | 38.0717 | 93.3053 | 61.5919 | 47.6568 |
| Category 8 | KDDrvb | - | 70.8401 | 80.9871 | 27.0485 | 75.9500 |
| | ICB2000 | - | 26.7584 | 41.9830 | 15.6973 | 26.2586 |
| | Musk2 | - | 25.2447 | 18.2440 | 21.6173 | 27.8823 |

## 6 Conclusion

This study firstly provides formal definitions of the DCA, such as DCs, inputs, and outputs, to make researchers understand the algorithm better. The works of feature selection and signal categorization are analyzed. Inspired by the grouping problems, this study models those works into a feature grouping problem. By searching for the optimal grouping scheme, this study automatically accomplishes feature selection and signal categorization without any expertise. The GGA is introduced to perform the searching task without expertise, making the algorithm more adaptive to apply to more fields. This study mixes the GGA and DCA to form a novel DCA version, GGA-DCA, to automatically accomplish feature selection and signal categorization. This study redefined crossover and mutation in GGA and let it find the optimal grouping with the most less time for DCA. The classification results from experiments indicate that GGA-DCA generally finds the optimal grouping schemes and keeps the algorithm lightweight in terms of running time without expertise.

The future works include two parts. Firstly, more search methods will be explored to perform feature selection and signal categorization, such as Monarch Butterfly Optimization (MBO) [35], Elephant Herding Optimization (EHO) [36], Krill Herd (KH) [37], and three-phase search approach with dynamic population size (TPSDP) [38]. Secondly, other metrics will be attempted to measure the performance of a feature grouping. The F-measure is the weighted harmonic average of precision and recall, which can measure the performance better. Hence, the F-measure is the next evaluation indicator for GGA-DCA.

**Funding Statement:** The authors want to thank NSFC http://www.nsfc.gov.cn/ for the support through Grants No. 61877045, and Fundamental Research Project of Shenzhen Science and Technology Program for the support through Grants No. JCYJ2016042815-3956266.

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

## References

1. Zhou, W., Liang, Y. (2021). A new version of the deterministic dendritic cell algorithm based on numerical differential and immune response. *Applied Soft Computing, 102,* 107055. DOI 10.1016/j.asoc.2020.107055.
2. Greensmith, J., Aickelin, U. (2008). The deterministic dendritic cell algorithm. In: *Artificial immune systems*, vol. 5132. Phuket, Thailand: lNCS.
3. Dagdia, Z. C. (2019). A scalable and distributed dendritic cell algorithm for big data classification. *Swarm and Evolutionary Computation, 50,* 100432. DOI 10.1016/j.swevo.2018.08.009.
4. Zhang, C., Yi, Z. (2010). A danger theory inspired artificial immune algorithm for on-line supervised two-class classification problem. *Neurocomputing, 73(7),* 1244–1255. DOI 10.1016/j.neucom.2010.01.005.
5. Abdelhaq, M., Hassan, R., Alsaqour, R. (2011). Using dendritic cell algorithm to detect the resource consumption attack over manet. *International Conference on Software Engineering and Computer Systems*, pp. 429–442. Berlin, Heidelberg, Springer.
6. Farzadnia, E., Shirazi, H., Nowroozi, A. (2021). A novel sophisticated hybrid method for intrusion detection using the artificial immune system. *Journal of Information Security and Applications, 58,* 102721. DOI 10.1016/j.jisa.2020.102721.
7. El-Alfy, E. S. M., Al-Hasan, A. A. (2014). A novel bio-inspired predictive model for spam filtering based on dendritic cell algorithm. *2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, Florida, USA, IEEE.

8. Chen, W., Zhao, H., Li, T., Liu, Y. (2018). Optimal probabilistic encryption for distributed detection in wireless sensor networks based on immune differential evolution algorithm. *Wireless Networks, 24(7),* 2497–2507. DOI 10.1007/s11276-017-1484-3.
9. Zhou, W., Liang, Y., Ming, Z., Dong, H. (2020). Earthquake prediction model based on danger theory in artificial immunity. *Neural Network World, 30(4),* 231–247. DOI 10.14311/NNW.2020.30.016.
10. Chelly, Z., Elouedi, Z. (2016). A survey of the dendritic cell algorithm. *Knowledge and Information Systems, 48(3),* 505–535. DOI 10.1007/s10115-015-0891-y.
11. Gu, F., Greensmith, J., Oates, R., Aickelin, U. (2009). PCA 4 DCA: The application of principal component analysis to the dendritic cell algorithm. *9th Annual Workshop on Computational Intelligence (UKCI 2009)*, Nottingham, UK.
12. Gu, F. (2011). *Theoretical and empirical extensions of the dendritic cell algorithm* (*Ph.D. Thesis*). University of Nottingham.
13. Chelly, Z., Elouedi, Z. (2012). RST-DCA: A dendritic cell algorithm based on rough set. *International Conference on Neural Information Processing*, pp. 480–487. Berlin, Heidelberg: Springer.
14. Chelly, Z., Elouedi, Z. (2012). RC-DCA: A new feature selection and signal categorization technique for the dendritic cell algorithm based on rough set theory. *Artificial Immune Systems. 11th International Conference*, vol. 7597. Taormina, Italy, LNCS.
15. Chelly, Z., Elouedi, Z. (2013). QR-DCA: A new rough data pre-processing approach for the dendritic cell algorithm. In: *Adaptive and natural computing algorithms*, vol. 7824. Lausanne, Switzerland, LNCS.
16. Chelly, Z., Elouedi, Z. (2013). A fuzzy-rough data pre-processing approach for the dendritic cell classifier. In: *Symbolic and quantitative approaches to reasoning with uncertainty*, pp. 109–120. Berlin, Heidelberg: LNAI, Springer.
17. Chelly, Z., Elouedi, Z. (2013). Supporting fuzzy-rough sets in the dendritic cell algorithm data pre-processing phase. In: *Neural information processing*, vol. 8227. Berlin, Heidelberg: LNCS, Springer.
18. Kohavi, R., John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97(1–2),* 273–324. DOI 10.1016/S0004-3702(97)00043-X.
19. Ali, K. B., Chelly, Z., Elouedi, Z. (2015). A new version of the dendritic cell immune algorithm based on the k-nearest neighbors. *Neural Information Processing. 22nd International Conference (ICONIP 2015)*, vol. 9489. Cham, LNCS, Springer International Publishing.
20. Mohsin, M. F., Hamdan, A. R., Bakar, A. A. (2014). An evaluation of feature selection technique for dendrite cell algorithm. *2014 International Conference on IT Convergence and Security (ICITCS)*, Beijing, China.
21. Feng, Y., Wang, G. G. (2022). A binary moth search algorithm based on self-learning for multidimensional knapsack problems. *Future Generation Computer Systems, 126,* 48–64. DOI 10.1016/j.future.2021.07.033.
22. Zhuang, Z., Huang, Z., Lu, Z., Guo, L., Cao, Q. et al. (2019). An improved artificial bee colony algorithm for solving open shop scheduling problem with two sequence-dependent setup times. *Procedia CIRP, 83,* 563–568. DOI 10.1016/j.procir.2019.04.119.
23. Wang, G. G., Gao, D., Pedrycz, W. (2022). Solving multi-objective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Transactions on Industrial Informatics,* DOI 10.1109/TII.2022.3165636.
24. Gao, S., Wang, K., Tao, S., Jin, T., Dai, H. et al. (2021). A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models. *Energy Conversion and Management, 230,* 113784. DOI 10.1016/j.enconman.2020.113784.
25. Baniamerian, A., Bashiri, M., Tavakkoli-Moghaddam, R. (2019). Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking. *Applied Soft Computing, 75,* 441–460. DOI 10.1016/j.asoc.2018.11.029.

26. Pakzad-Moghaddam, S. (2016). A levy flight embedded particle swarm optimization for multi-objective parallel-machine scheduling with learning and adapting considerations. *Computers & Industrial Engineering, 91,* 109–128. DOI 10.1016/j.cie.2015.10.019.
27. Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., Kharel, R. (2021). Variation operators for grouping genetic algorithms: A review. *Swarm and Evolutionary Computation, 60,* 100796. DOI 10.1016/j.swevo.2020.100796.
28. Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., Schutze, O. (2020). Metaheuristics to solve grouping problems: A review and a case study. *Swarm and Evolutionary Computation, 53,* 100643. DOI 10.1016/j.swevo.2019.100643.
29. Elisa, N., Yang, L., Chao, F. (2020). Signal categorisation for dendritic cell algorithm using ga with partial shuffle mutation. In: *Advances in computational intelligence systems*, vol. 1043. Germany: AISC, Springer.
30. Gao, D., Wang, G. G., Pedrycz, W. (2020). Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems, 28(12),* 3265–3275. DOI 10.1109/TFUZZ.91.
31. Falkenauer, E. (1993). The grouping genetic algorithms: Widening the scope of the GAs. *Belgian Journal of Operations Research, Statistics, and Computer Science, 33(1–2),* 79–102.
32. Asuncion, A., Newman, D. (2007). *UCI machine learning repository*. University of California, Irvine, USA: Dua, Dherru and Graff, Casey.
33. Alcala-Fdez, J., Fernández, A., Luengo, J., Derrac, J., Garc'ia, S. et al. (2010). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing, 17,* 255–287.
34. Gu, F., Greensmith, J., Aickelin, U. (2013). Theoretical formulation and analysis of the deterministic dendritic cell algorithm. *Biosystems, 111(2),* 127–135.
35. Wang, G. G., Deb, S., Cui, Z. (2019). Monarch butterfly optimization. *Neural Computing & Applications, 31(7),* 1995–2014.
36. Li, W., Wang, G. G., Alavi, A. H. (2020). Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowledge-Based Systems, 195,* 105675.
37. Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. S., Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences, 274,* 17–34.
38. Yang, X., Cai, Z., Jin, T., Tang, Z., Gao, S. (2022). A Three-phase search approach with dynamic population size for solving the maximally diverse grouping problem. *European Journal of Operational Research, 302(3),* 925–953.