Received November 14, 2021, accepted December 3, 2021, date of publication December 7, 2021,
date of current version December 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3133731

Revised Artificial Immune Recognition System

WAFA NEBILI, BRAHIM FAROU, ZINEDDINE KOUAHLA, AND HAMID SERIDI
Laboratory of Science and Information Technologies and Communication (LabSTIC), 8 Mai 1945 Guelma University, Guelma 24000, Algeria
Corresponding author: Wafa Nebili (nebili.wafa@univ-guelma.dz)

ABSTRACT Artificial Immune Recognition System is a widely used bio-inspired algorithm that describes the recognition tasks of antigen by memory cells. Despite the success of the Artificial Immune Recognition System, the basic version has some drawbacks which have a direct impact on system efficiency in terms of the quality of the results, data explosion, and calculation cost. This paper investigates these disadvantages and proposes several modifications in the original version to overcome these problems. First, the concept of weight and lifetime counter was introduced for each memory cell to improve quality; second, a new mechanism was added to eliminate inactive memory cell models to reduce data explosion, and third, the structure of the memory cells set was replaced by a binary search tree to reduce processing time. Furthermore, this paper improves some algorithm functionalities especially in the mutation function and the memory cell introduction mechanism. The experimental results conducted on eleven public datasets show that the proposed method outperforms the original version, all the revised versions, and achieved a good rank compared to the other state-of-the-art methods with an average accuracy of 93.20 % on all tested datasets.

INDEX TERMS AIRS, bio-inspired, KD-tree, kNN, supervised learning.

I. INTRODUCTION

Artificial Immune Systems (AIS) are among the new biologically motivated paradigm that has been exploited as solutions for various applications in computer science and engineering. These systems serve as metaphors for several aspects of the natural immune system. Some of these aspects, such as the self/non-self-distinction and the negative selection concept are intuitively adapted to applications of computer security, intrusion detection [1], [2], change detection, clustering [3], [4], etc. Nevertheless, the artificial immune systems are not dedicated only to these fields. Research on the natural immune system shows that the latter contains some learning properties that can be used as machine learning systems.

Artificial Immune Recognition System (AIRS) is an AIS classifier, which is dedicated initially to show that a subset from the AIS aspects could produce an efficient supervised learning system [5]. This classifier operates well and achieves good results compared to other public classifiers [6].

This paper discusses the principal steps of the original AIRS algorithm developed in [6] and proposes modifications to maintain system efficiency in terms of data reduction, search cost, and the quality of the results. For this purpose, two novel processes (**Update_Weight_DL** and **Remove_mc_Inactive**) were added to avoid the explosion of data in the memory cells set (MC). To reduce the search cost of the best memory cell mc_{match}, the MC set is restructured as a binary search tree (Kd-tree) to go from linear to binary search. Finally, slight modifications are introduced to the mutation function and to the other stages of the AIRS to improve classification accuracy.

The Artificial Immune Recognition System (AIRS) is a robust and powerful information process system that stands out with several features such as distributed control, parallel processing, and adaptation via experience. It is one of the supervised learning methods that has shown significant success in a wide range of classification problems despite some disadvantages relating to memory congestion and the high processing time. However, compared with other supervised learning methods, there is a significant lack of works to improve AIRS and get the most out of this algorithm in tackling complex problems.

The proposed modification improved the AIRS in terms of accuracy, processing time, and the amount of used memory and allowed it to get a good rank compared with other classifiers. In addition, unlike the other classifiers that require a reconfiguration for each dataset, the improved AIRS allowed to preserve the same performance on all datasets by using the same parameter's values.

To cover all sections, the paper is organized as follows. Section 2 explains in detail the steps of the original AIRS algorithm as well as the proposed modifications, obtained

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Ali Babar.

results on the public datasets are presented in Section 3, Section 4 concludes the paper.

II. RELATED WORK

With the development of the technology, the quantity of the collected and the treated data become enormous. Faced of this growing of data, the ability to automatically classify data models into categories of interest become a necessary tool that use these large collections of data as sources of information rather than just storing them.

Recently, it appears many methods for grouping individuals according to their similarities, from these methods we cite: *k* Nearest Neighbors (kNN) [7], IMVKNN [8], SVM [9], Adaptative Boosting (Adaboost) [10], etc.

The complexity of the classification task is not related only on the amount of data but also to the dimension of the latter. Reduction of the dimension of data performed with two different techniques: feature extraction and feature selection.

- Feature extraction allows creating a set of features by transforming the primary data space into a subspace of a small dimension. Many methods have been exploited as extraction features tool: Ershad and Hashemi [11] used Dispelling Classes Gradually (DCG) concept on the input data to reduce features, Siouda *et al.* [12] exploited the deep autoencoder as feature extractor tool.
- Feature selection choose the most important or the relevant features that can improve the classification quality, as an example of the feature selection methods: Adil *et al.* [13] that used Principal Component Analysis (PCA) for selecting the best features which represent more the facial expression. In the same context and for improving results quality authors of [14] exploited genetic algorithm to select the best features and to optimize SVM hyperparameters. Raj *et al.* [15] proposed an improves classifier names Optimal Deep Learning (DL) as feature selection tool for medical image.

The viability of the problems treated requires other standard classification techniques. Recently, there exist new category of algorithms called bio-inspired algorithms. This category of algorithms can serve the metaphors for computer systems such as the evolutionary computing algorithms [6]. The first research works on artificial evolution were focused on genetic algorithms (GA). The latter have managed to cope with optimization problems through the mutation and crossing mechanism [16]. In the same context, we also find the artificial immune system (AIS) [6] and the artificial neural networks, which have shown their effectiveness in the learning domain [17], [18].

In the literature, there is also another category of bio-inspired approaches, which based on the behavior of swarms in nature (swarm intelligence). This category has succeeded to attract the attention of several researchers and to generate many approaches such as ant colonies [19], particle swarm optimization algorithm (PSO) [20], bee swarms [21], artificial fish swarms [22], and so many others [23]–[30]. The artificial immune system (AIS) is among the most successful bio-inspired approaches. It includes a set of techniques that exploit the various aspects of the natural Immune System (IS), such as the concept of memorization by memory cells, recognition aspect, maturation of cells, cloning, etc. Based on these concepts and the context of the problems to be solved, several artificial immune systems have been proposed.

The first artificial immune model was proposed by the biologist Jerne in 1974 [31], this model represents the dynamic behavior of IS even in the absence of the antigen. The system is built from the idiotype-paratope relationships between B cells-antigens and the antibodies of B cells which are interconnected with each other to recognize antigens [32]. This relationship makes the cognitive capacity of IS similar to a neural network [33]. Authors in [1] suggest a negative selection algorithm inspired by the tolerance of self-mechanism defense in the production of mature *T* cells (detector cells). All cells, which able to recognize the self are eliminated and the cells that are able to detect non-self are filtered in a control phase. Works done in [34] propose a clonal selection algorithm that explains how the IS interacts with antigens (clonal selection algorithm). The algorithm exploits the principle of memory cells *mc* that keeps only the best representatives. The *B* cells that recognize the antigen and which have a high degree of affinity will be cloned and mutated in order to be selected later as memory cells. In the danger theory, the IS reacts not only to non-self cells but also to the infected self-cells. These latter send an alarm signal to allow the system to distinguish between dangerous and non-dangerous cells, which explains why the system does not react to non-self entities that are not dangerous. From this principle, authors in [35] develop an approach that characterizes this theory, where the danger is measured according to the gravity of the damage in the infected cells.

As part of supervised learning, Watkins and Boggess [6] proposed an Artificial Immune Recognition System (AIRS) inspired by learning and memorization aspects of the memory cells [36]. This algorithm measures the degree of similarity between training data (antigens) and potential solutions (antibodies or B cells). AIRS takes as input an antigen and produces a set of memory cells as output, these cells are used later in the classification phase. Authors in [37] perform a revision in some AIRS processes to increase data reduction capacity. In the literature, the AIS was also used in the context of unsupervised learning [3], [4], [38].

III. ARTIFICIAL IMMUNE RECOGNITION SYSTEM (AIRS)

When a strange element enters, the body uses a set of defense mechanisms to protect itself from these elements. The biological system has an intelligent ability to recognize self cells from non-self cells. Inspired by this biological mechanism, computer science has developed systems that can serve as a metaphor for some aspects of the natural immune system.

AIRS is a supervised artificial immune system that describes the immune response caused when an antigen is detected by measuring the degree of closeness between training data (antigens) and B cells based on antigen-antibody representation. In the AIRS, the degree of similarity is called affinity or stimulation. After recognizing the antigen, the system generates a set of memory cells which constitute later the learning model. However, the original version of AIRS [6], has some drawbacks that influence the quality of the obtained results and the processing time. In the following, we introduce some basic concepts, present and discuss the original AIRS steps, and detail the proposed modifications. Table 1 details some used keywords in AIRS.

To create a robust classification model, AIRS uses four main steps: **Initialization**, **Memory cell identification and ARB generation**, **Competition for resources and development of a candidate memory cell** and **Memory cell introduction**. Each step is detailed in the following.

A. INITIALIZATION

During this pretreatment phase, all features vectors (antigens) are normalized such that the distance between two features vectors is always in the range of [0, 1]. After that, AIRS calculate the Affinity Threshold (AT) using Equation (1). AT is used in the **Memory cell introduction** to substitute the mc_{match} with a new memory cell.

$$AT = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} Affinity(ag_i, ag_j)}{\frac{n(n-1)}{2}} \tag{1}$$

Were n is the number of antigens in the training data and $Affinity(ag_i, ag_j)$ is defined as the Euclidean distance between two pairs of antibodies or antigens. This stage ends by producing memory cells (MC) and B cells (ARB). The role of the initialization phase is to fill the MC set and the ARB set. The number of the initial memory cells in the MC set is an empirical number chosen by the user. There are two possible configurations: The first one is with an empty MC if the time of training does not matter and the second one with a set of antigens randomly selected from the training vector to speed up the time of training.

MC and ARB are composed of class groups ($MC_{ag.c}$, $AB_{ag.c}$). The MC set is very important since it constitutes the learning model. Initialized MC set with random ag can influence the quality of the obtained results. In the original initialization phase of AIRS, many groups of mc have no representatives or bad representatives.

To solve these problems we propose (Algorithm 1) a semi-random process that initializes MC set with the average vectors of ag which have the same class, then we randomly assign some $ag \in AG$ to this group. Our proposition provides a good beginning for AIRS and increases the possibility to generate a good mc_{match}. In the original version, MC set can have many inactive memory cell (mc) with a low participation rate during all the system lifetime. This situation causes a combinatorial explosion especially in large databases and increases the algorithmic complexity in all processing levels. In addition, a very high number of inactive mc can generate bad results.

TABLE 1. The basic concepts used in AIRS.

Concept	Definition
nc	The number of classes
MC	Memory cells set. Let mc represents a memory cell, such that mc $\in$ MC and MC$_i$ the ith class of MC.
mc.c	The class of mc, where $c = \{1, 2, \cdots, nc\}$
mc.f	The feature vector of mc and mc.f$_i$ represents the value of the ith feature of mc.f.
mc.w	The weight of mc, where mc.w $\in$ [0,1]. It indicates the activity degree of mc.
mc.DL	The lifetime of mc.
mc.L	The left node of mc.
mc.R	The right node of mc.
S_DL	The maximum lifetime for an mc.
S_activation	The minimum weight to active mc.
AG	Antigens set. Let ag represents an antigen, such that ag $\in$ AG and AG$_i$ the ith class of AG.
ag.c	The class of ag, where $c = \{1, 2, \cdots, nc\}$.
ag.f	The feature vector of ag and ag.f$_i$ represents the value of the ith feature of ag.f.
ARB	Artificial Recognition Ball (ARB) Also known as B cell.
AB	ARBs set. Let ab represents an ARB, such that ab $\in$ AB and AB$_i$ the ith class of AB.
ab.c	The class of ab, where $c = \{1, 2, \cdots, nc\}$.
ab.w	The weight of mc, where mc.w $\in$ [0,1]. It indicates the activity degree of ab.
ab.stim	The degree of stimulation of an ab with an antigen ag.
ab.resources	The number of resources held by ab.
Total-NumResources	The total number of resources allowed for each class of AB.
Affinity	A measure of similarity between two antibodies or antigen-antibody, this value is always between 0 and 1, and it is calculated as the Euclidean distance between two features vectors.
Stimulation	A function used to measure the response of an ARB to an antigen or to another ARB. Stimulation is the opposite of the affinity.
Affinity Threshold (AT)	It is the average affinity of all antigens of the training data items.
Affinity Threshold Scalar (ATS)	A value between 0 and 1, it is used to substitute the memory cells mc in the **Memory cell introduction** stage.
Mutation_rate	A probability value indicates if a specific feature of an ARB will be mutated or not.
Clonal_rate	An integer value fixed empirically by the user to determine the number of mutated clones that an ARB can be produced.
Hyper_mutation_rate	An integer value chosen by the user. It is used with the $Clonal\_rate$ to determine the number of mutated clones that an ARB can be produced.
Stimulation_threshold	It is a parameter between 0 and 1 used to indicate if the training of a specific antigen is achieved or not.
Dist(x,y)	The Euclidean distance between x and y

To avoid these problems, we propose for each created mc, a weight and a lifetime which will be used later in the update process of MC set. The randomly generated mc has an initial weight equal to 0. Since the ag_m are the best representatives of MC, the initial weight is set to a value

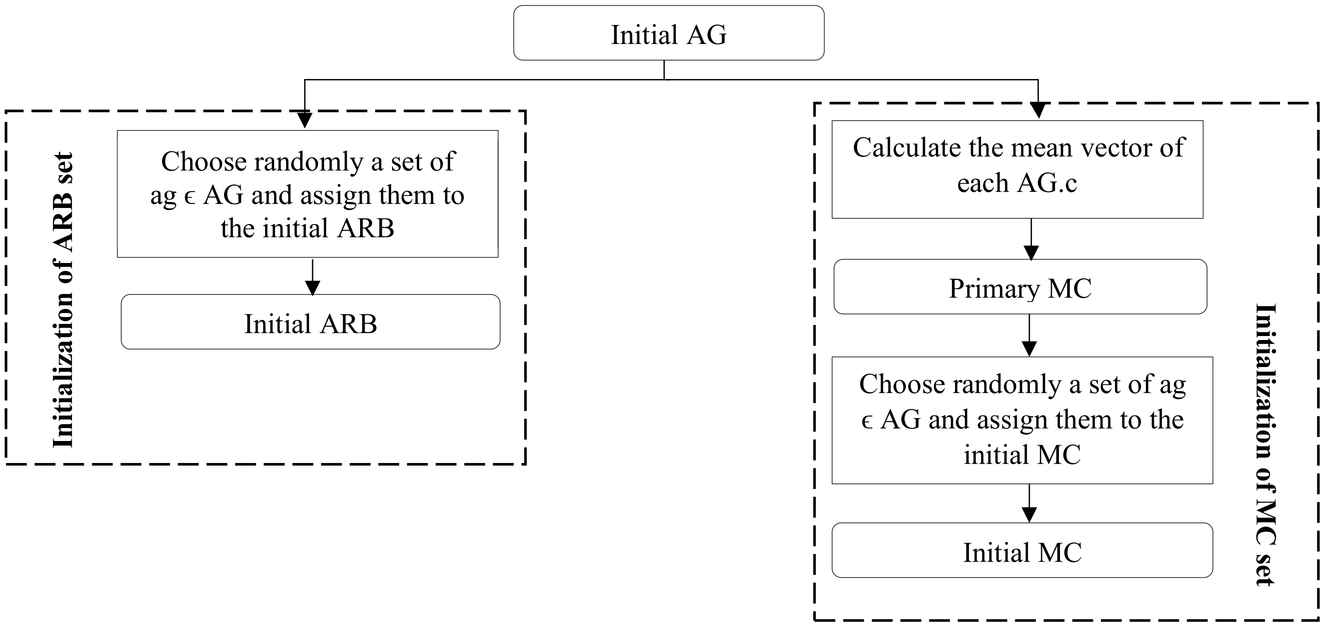
Initial AG
Initialization of ARB set
Choose randomly a set of ag ϵ AG and assign them to the initial ARB
Initial ARB
Calculate the mean vector of each AG.c
Primary MC
Choose randomly a set of ag ϵ AG and assign them to the initial MC
Initial MC
Initialization of MC set

FIGURE 1. **Flowchart of the proposed initialization of *MC* and *ARB* sets.**

greater than 0 according to the Equation (2). The lifetime of the created *mc* is set to 0. All Artificial Recognition Ball (*ab*) weights are initialized with 0. The concept of weight in *ARB* replaces the notion of resources in the original version of AIRS.

$$\mathrm{ag}_m.w = \frac{1}{\mathrm{nc}} \quad (2)$$

B. MEMORY CELL IDENTIFICATION AND ARB GENERATION

This phase contains two essential steps: the first step consists to identify the memory cell mc_{match} from *MC* set that belongs to the same class of ag and which has a high stimulation with it(Equation 3). If $MC_{ag.c}$ is empty so $mc_{match} \leftarrow ag$ and $MC_{ag.c} \leftarrow MC_{ag.c} \cup ag$.

$$\mathrm{mc}_{match} = argmax_{\mathrm{mc}\in\mathrm{MC}_{ag.c}} Stimulation(\mathrm{ag}, \mathrm{mc}) \quad (3)$$

With:

$$Stimulation(\mathrm{ag}, \mathrm{mc}) = 1 - Affinity(\mathrm{ag}, \mathrm{mc}) \quad (4)$$

The structure of the *MC* set as it is proposed in [6] requires a sequential research method to find the mc_{match}. The use of linear search gives rise to a higher search cost and more powerful machines in the case of large datasets. To solve this problem, we propose to organize the *MC* set as a tree (kd-Tree [39]) where the most similar *mc* are stored as a bag in its leaves.

The capacity of the bags is fixed empirically by the user. Once the bag is saturated, it will be exploded into two new bags, and the two most dissimilar *mc* are chosen to represent each new bag (left node, right node). Searching for mc_{match} takes only one side of the tree and depending on the distance of ag with the left and right node, the bag of mc_{match} can be easily located. After determining the bag, a local search is made in the latter to find the most similar *mc* having the same class of *ag*. In the case of an empty search result, *ag* is chosen as mc_{match}. The substitution of the sequential structure used in the original version with a tree structure allowed to reduce the search complexity from sequential to logarithmic. Algorithm 2 describes in detail the proposed Memory cell identification step. As mentioned previously, we introduced the concepts of weight and lifetime for each *mc* to avoid quality deterioration and data explosion caused by inactive *mc* with a low participation rate. Inspired from the biological mechanism of elimination of inactive memory cells, we propose two functions: **Update_Weight_DL** to update the parameters of *mc* (Algorithm 3), and **Remove_mc_Inactive** to remove the inactive *mc* from *MC* set(Algorithm 4).

Each memory cell *mc* has a life counter *mc.DL* which increases each time when it encounters the antigens. Memory cells that consumed their *DL* and which have a low activity are removed from the *MC* set. Unlike, if a memory cell is very active and it has consumed its lifetime, it benefits from a new lifetime ($mc.DL = 0$). In both cases, the activity rate is measured according to the weight. Note that **Update_Weight_DL** and **Remove_mc_Inactive** are applied only for the bag that contains mc_{match}.

After identifying the memory cell mc_{match}, the second step consists of generating new *ARB*'s clones by the mutation of the feature vector of mc_{match} with an empirically fixed rate between 0 and 1. The number of clones generated is calculated by (III-B).

$$NumClones = hyper\_mutation\_rate \times clonal\_rate \times stimulation(\mathrm{ag}, \mathrm{mc}_{match}) \quad (5)$$

The **hyper_mutation_rate** and the **clonal_rate** are two integer values empirically chosen by the user. We kept the

Algorithm 1 Initialization

```
Require: $AG$
Ensure: the initial $MC$ set, the initial $ARB$ set
  Variables
  Begin
  $nbrMC$, $nbrARB$: They are respectively the initial number of $mc$ and $ARBs$
  $minf_i$: The ith minimum value in the feature vectors
  $maxf_i$: The ith maximum value in the feature vectors
  $ag_m$: The mean vector of the antigens belonging to the same class
  End
  for each $ag \in AG$ do
    for each $ag.f_i \in ag.f$ do
      $ag.f_i \leftarrow \frac{\frac{ag.f_i - minf_i}{maxf_i - minf_i}}{|ag.f|}$
    end for
  end for
  $AT \leftarrow \frac{\sum_{i=1}^{|AG|-1} \sum_{j=i+1}^{|AG|} Affinity(ag_i, ag_j)}{\frac{|AG|(|AG|-1)}{2}}$
  $nbrMC \leftarrow value$
  for $c = 1$ to $nc$ do
    $ag_m \leftarrow mean\_vector(AG_c)$
    $mc \leftarrow ag_m$
    $mc.w \leftarrow \frac{1}{nc}$
    $mc.DL \leftarrow 0$
    $MC_c \leftarrow MC_c \cup mc$
  end for
  while $|MC| \leq nbrMC$ do
    Choose randomly an $ag \in AG$
    $mc \leftarrow ag$
    $mc.w \leftarrow 0$
    $mc.DL \leftarrow 0$
    $MC_{mc.c} \leftarrow MC_{mc.c} \cup mc$
  end while
  $nbrARB \leftarrow value$
  while $|AB| \leq nbrARB$ do
    Choose randomly an $ag \in AG$
    $ab \leftarrow ag$
    $ab.w \leftarrow 0$
    $AB_{ab.c} \leftarrow AB_{ab.c} \cup ab$
  end while
  Create a KD-Tree with the initial $MC$
```

same principle of the mutation function proposed in [37] with some modification. So we added a weight to each new clone according to their stimulation with the current antigen in order to increase their opportunity to be selected as a candidate memory cell (6).

$$\text{mc}_{clone}.\text{w} = \text{stimulation}(\text{ag}, \text{mc}_{clone}) \qquad (6)$$

In the original version of AIRS, the feature vector produced after the mutation is assigned randomly to a class which causes an increase in false classifications. In order to overcome this problem, we propose to label the new clones with the class of the nearest mean vector of the antigens generated

Algorithm 2 Memory_cell_identification

```
Require: ag, MC
Ensure: $mc_{match}$
  maxStim $\leftarrow -1$
  Find $\leftarrow$ False
  if MC = null then
    $mc_{match} \leftarrow$ ag
    MC.bag $\leftarrow$ MC.bag $\cup$ ag
  else
    if ( MC.L = null) and (MC.R = null) then
      Update_Weight_DL (MC.bag, ag)
      Remove_mc_Inactif (MC.bag)
      for j = 1 to |MC.bag| do
        if ag.c = $mc_j$.c then
          Find $\leftarrow$ True
          valStim $\leftarrow$ stimulation (ag,$mc_j$)
          if valStim > maxStim then
            $mc_{match} \leftarrow mc_j$
            maxStim $\leftarrow$ valStim
          end if
        end if
      end for
      if Find=False then
        $mc_{match} \leftarrow$ ag
        MC.bag $\leftarrow$ MC.bag $\cup$ ag
      end if
    else
      if dist(ag, MC.L) $\leq$ dist(ag, MC.R) then
        $mc_{match} \leftarrow$ Memory_cell_identification(ag, MC.L)
      else
        $mc_{match} \leftarrow$ Memory_cell_identification(ag, MC.R)

      end if
    end if
  end if
```

in the initialization phase. The proposed mutation function is defined in the Algorithm 6. with drandom(): A function that returns a random number between 0 and 1. Note that *AB* set contains *ARBs* generated during the previous iterations and the new clones created from mc_{match}. The *ab* within *AB* set are also grouped into classes.

C. COMPETITION FOR RESOURCES AND DEVELOPMENT OF A CANDIDATE MEMORY CELL

The aim of this phase is to filter the *AB* set in order to keep only the most representative *ab* and the most similar to the antigen *ag*. For this purpose authors in [6] propose a technique that organizes the survival of individuals within the *AB* population. This technique suggests a cumulative resource sharing process based on the nature of the *ag* class. Half of the cumulative resources is allocated to the *ab* that has the same class of *ag*, the rest is shared between the other *ab*. Each

Algorithm 3 Remove_mc_Inactive

```
Require: MC.bag
Ensure: MC.bag
  for $i = 1$ to |MC.bag| do
    if $mc_i$.DL $\geq$ S_DL then
      if $mc_i$.w $\leq$ S_activation then
        MC.bag $\leftarrow$ MC.bag $-$ $mc_i$
      end if
    else
      $mc_i$.DL $\leftarrow 0$
    end if
  end for
```

Algorithm 4 Update_Weight_DL

```
Require: MC.bag, ag
Ensure: MC.bag
  for $i = 1$ to |MC.bag| do
    if $mc_i$.c = ag.c then
      $mc_i$.DL $\leftarrow$ $mc_i$.DL+1
      $mc_i.w \leftarrow mc_i.w + \frac{stimulation(ag, mc_i)}{sum\_stimulation(\text{MC.bag})}$
    end if
  end for
  for $i = 1$ to |MC.bag| do
    if $mc_i$.c = ag.c then
      Normalized the weight of $mc_i$ by:
      $mc_i.w \leftarrow \frac{mc_i.w}{sum\_weigth(\text{MC.bag})}$
    end if
  end for
```

Algorithm 5 ARB Generation

```
Require: $mc_{match}$, ag,hyper_mutation_rate, clonal_rate
Ensure: ARB (AB)
  Variables
  Begin
  MU: The mutated ARB clones.
  mut: A boolean value
  End
  MU $\leftarrow \emptyset$
  MU $\leftarrow$ MU $\cup$ makeARB($mc_{match}$)
  stim $\leftarrow$ stimulation (ag, $mc_{match}$ )
  NumClones $\leftarrow$ hyper_mutation_rate $\times$ clonal_rate $\times$ stim

  while |MU| <NumClones do
    mut $\leftarrow$ false
    $mc_{clone} \leftarrow mc_{match}$
    $mc_{clone}$, mut $\leftarrow$ mutation ($mc_{clone}$, mut)
    if mut = true then
      $mc_{clone}$.w $\leftarrow$ stimulation (ag,$mc_{clone}$)
      MU $\leftarrow$ MU $\cup$ makeARB ($mc_{clone}$)
    end if
  end while
  AB $\leftarrow$ AB $\cup$ MU
```

class in the *AB* set has a maximum resource allocation, if the sum of resources allocated in a class exceeds their maximum

Algorithm 6 Mutation Function

```
Require: x, b, mutation_rate
Ensure: x, b
  for each x.$f_i \in$ x.f do
    Change $\leftarrow$ drandom()
    Changeto $\leftarrow$ drandom()
    if Change < $mutation\_rate$ then
      $x.f_i$ $\leftarrow$Changeto $\times$ normalization_value
      b $\leftarrow$ True
    end if
  end for
  if b=True then
    Change $\leftarrow$ drandom()
    if Change < mutation_rate then
      x.c $\leftarrow$ $\text{argmin}_{c_i \in 1,2,..,nc}$dist(x,mean(AG.$c_i$))
    end if
  end if
```

allocation, the additional resources will be removed from the least stimulated *ab* of this class. In addition, to eliminate the overlap between classes created by the *ab* having maximum stimulation with ag from other classes, any ab that has a number of resources equal to zero will also be removed from *AB* population (see Algorithm 7).

After this process, The *AB* set has the most stimulated *ARBs* with the antigen ag and which succeeded in the acquisition of resources. Once this stage is finished, a stop criterion is calculated to verify if the stimulation of the *ARBs* with *ag* is sufficient. The stopping criterion is reached if and only if $s_i \geq$ stimulation_threshold where $i = ag.c$

$$S_i = \frac{\sum_{j=1}^{|AB_i|} \text{ab}_j.\text{stim}}{|\text{AB}_i|}, \quad \text{ab}_j \in \text{AB}_i \tag{7}$$

During this time, each $ab \in AB$ has a chance to produce mutated offspring, in order to increase the diversity of members within the *AB* set.

To reach this goal, the original mutation function was substituted in **Mutation of surviving ARB** process by the mutation function proposed in the algorithm 6.

The mutation of *ARBs* process passes to the next step only if the stopping criterion is met; otherwise, it will be repeated until the stopping criterion is reached. Note that the mutation of the *ARBs* is performed at least once even if the stopping criterion is reached.

In the original version, when the mutation of *ARBs* is finished, the AIRS selects a candidate memory cell ($mc_{candidate}$) from the remaining *ab*. The $mc_{candidate}$ is the most stimulated *ab* with *ag* having the same class as the latter. However, the choice of a single memory cell candidate is unjust, since the rest *ab* that have the same class of the antigen *ag* are also among the best representatives of the latter. For this purpose we propose to build a population of $mc_{candidates}$ ($MC_{candidates}$) with the remains *ab* instead of choosing a single $mc_{candidate}$. This proposition allows increasing the possibility of introducing new memory cells that can give a

Algorithm 7 Stimulation, Resource Allocation, and ARB Removal

```
Require: AB
Ensure: AB
  Variables
  Begin
  MAX, MIN: Two real numbers.
  TotalNumResources: The number of resources in the system.
  End
  minStim ← MAX
  maxStim ← MIN
  for each ab ∈ AB do
    stim ← stimulation (ag, ab)
    if stim < minStim then
      minStim ← stim
    end if
    if stim > maxStim then
      maxStim ← stim
    end if
    ab.stim ← stim
  end for
  for each ab ∈ AB do
    if ab.c = ag.c then
      ab.stim ← $\frac{\text{ab.stim-minStim}}{\text{maxStim-minStim}}$
    else
      ab.stim ← 1 − $\frac{\text{ab.stim-minStim}}{\text{maxStim-minStim}}$
    end if
    ab.resources ← ab.stim × clonal_rate
  end for
  i ← 1
  while i<nc do
    resAlloc ← $\sum_{j=1}^{|AB_i|}$ ab_j.resources, ab_j ∈ AB_i
    if i=ag.c then
      numResAllowed ← $\frac{\text{TotalNumResources}}{2}$
    else
      numResAllowed ← $\frac{\text{TotalNumResources}}{2\times(nc-1)}$
    end if
    while resAlloc > numResAllowed do
      numResRemoved ← resAlloc − numResAllowed
      ab_removed ← $argmin_{ab \in AB_i}$ (ab.stim)
      if ab_removed.resources ≤ numResRemoved then
        AB_i ← AB_i − ab_removed
        resAlloc ← resAlloc − ab_removed.resources
      else
        ab_removed.resources ← ab_removed.resources − numResRemoved
        resAlloc ← resAlloc − numResRemoved
      end if
    end while
    i ← i+1
  end while
```

more correct and more complete classification model for the antigen *ag*.

Algorithm 8 Mutation of Surviving ARB Function

```
Require: AB
Ensure: AB
  MU ← Ø
  for each ab ∈ AB do
    rd ← drandom()
    if ab.stim > rd then
      NumClones ← ab.stim × clonal_rate
      i ← 1
      while i ≤ NumClones do
        mut ← false
        $ab_{clone}$ ← ab
        $ab_{clone}$ ← mutation ($ab_{clone}$, mut)
        if mut = True then
          MU ← MU ∪ $ab_{clone}$
        end if
        i ← i+1
      end while
    end if
  end for
  AB ← AB ∪ MU
```

Algorithm 9 Memory Cells Introduction

```
Require: $MC_{candidates}$, ag, AT, ATS, $mc_{match}$
Ensure: MC
  Find ← False
  MatchStim ← stimulation (ag, $mc_{match}$)
  for each $mc_{candidate}$ ∈ $MC_{candidates}$ do
    CandStim ← stimulation (ag,$mc_{candidate}$)
    CellAff ← affinity ($mc_{candidate}$, $mc_{match}$)
    if CandStim > MatchStim then
      if (CellAff < AT × ATS) and (Find =False) then
        Remove $mc_{match}$ from the MC tree
        Find ← True
      end if
      $mc_{candidate}$.DL ← 0
      Add $mc_{candidate}$ to MC tree
    end if
  end for
```

D. MEMORY CELL INTRODUCTION

The last stage in the training process of one antigen consists to introduce the memory cell candidate $mc_{candidate}$ in the *MC* set. The $mc_{candidate}$ which has a stimulation with *ag* greater than the stimulation of mc_{match} with *ag* will be added to the *MC* set as a new memory cell (*mc*). If the affinity of $mc_{candidate}$ is lower than ($AT \times ATS$), $mc_{candidate}$ replaces mc_{match} in the *MC* set.

In the proposed Memory cell introduction algorithm (Algorithm 9), the same principle of the original AIRS is kept except for the number of introduced memory cells in the updating process. Indeed, the proposed mechanism improves the system and provides a high opportunity to produce a more representative model by adding all $mc_{candidate}$ with higher stimulation than the mc_{match}.

TABLE 2. Description of the used datasets.

Datasets	Size	# attributs	Missing values	# classes
Iris	150	4	No	3
Pima-Diabetes	768	8	No	2
Ionosphere	351	34	No	2
Sonar	208	60	No	2
Heart-Statlog	270	13	No	2
Hepatitis	155	19	Yes	2
Wisconsin-Breast Cancer	699	9	Yes	2
Dry Bean Dataset	13611	17	No	7
Steel Plates Faults	1941	33	No	2
HTRU2	17898	8	No	2
Yeast	1484	8	No	10

TABLE 3. Used parameters for RAIRS.

Parameters	Value
Initial number of memory cells and ARBs	10
Stimulation threshold	0.8
$mutation\_rate$	0.1
$clonal\_rate$	10
$hyper\_mutation\_rate$	2
Affinity threshold scalar	0.1
k of kNN	1

The lifetime counter of each introduced $mc_{candidate}$ is initialized with 0. Then, the Remove_mc_Inactive algorithm is applied in order to avoid the explosion of the MC set. This mechanism allows removing all mc which participates infrequently in the system (a very lower lifetime).

Once the evaluation of the $mc_{candidates}$ is completed, we proceed to the training of the next antigen in the training set starting from **Memory cell identification and** ***ARB*** **generation** stage until this stage. The training is finished when all antigens are presented to the system.

E. CLASSIFICATION

After the training stage, the MC set is ready for the classification of new elements. The classification is achieved by kNN (k nearest neighbor) classifier [7] according to a majority vote of k memory cells which is closest to the input element. However, research on the k nearest neighbor requires an iterative process of all model elements, which increases the search cost. The new structure proposed for the MC set in the form of a kd-Tree allows migrating from sequential to binary search, which facilitates finding the k nearest neighbor and accelerates the time spend for this task since the complexity is reduced from sequential to logarithmic.

IV. TESTS AND RESULTS

This section presents the obtained results of the proposed RAIRS on public datasets selected from the Irvine automatic learning repository of the University of California (UCI) [40]. All tests were performed in a Python environment on an i7 PC, with a memory of 8 GB.

Recall that the objective of this paper is to improve the original AIRS. For this purpose, we have chosen the same datasets used in [6] (AIRS1) for the original version

TABLE 4. Comparison of RAIRS accuracy on iris, ionosphere, sonar and pima-diabet datasets. (The other results are gathered from [41] and [42]).

Accuracy(%)							
Iris		**Ionosphere**		**Sonar**		**Pima-Diabetes**	
Grobian (rough)	100	3-NN+ simplex	98.7	**RAIRS**	**94.11**	**RAIRS**	**79.22**
RAIRS	**100**	**RAIRS**	**97.35**	TAP MFT Bayesian	92.3	Logdisc	77.77
SSV	98.0	IB3	96.7	SVM	90.4	IncNet	77.6
C-MLP2LN	98.0	3-NN	96.7	Best 2-layer MLP+ BP, 12 hidden	90.4	DIPOL92	77.6
PVM 2 rules	98.0	MLP+ BP	96.0	Naive MFT Bayesian	90.4	Linear Disc. Anala.	77.5
PVM 1 rule	97.3	AIRS2	95.6	AIRS2	84.9	SMART	76.8
FuNe-I	96.7	AIRS1	94.9	MLP+ BP, 12 hidden	84.7	GTO DT (5 × CV)	76.8
NEFCLASS	96.7	C4.5	94.9	MLP+ BP, 24 hidden	84.5	ASI	76.6
AIRS1	96.7	RIAC	94.6	1-NN, Manhattan	84.2	Fischer Discr. Anal.	76.5
AIRS2	96.0	SVM	93.2	AIRS1	84.0	MLP+ BP	76.4
CART	96.0	FSM+ rotation	92.8	FSM	83.6	LVQ	75.8
FUNN	95.7	Non-linear perceptron	92.0	MLP+ BP, 6 hidden	83.5	LFC	75.8
		1-NN	92.1	1-NN Euclidean	82.2	RBF	75.7
		DB-CART	91.3	DB-CART, (10 × CV)	81.8	NB	75.5
		Linear perceptron	90.7	CART, (10 × CV)	67.9	kNN, k=22, Manh	75.5
						MML	75.5
						SNB	75.4
						BP	75.2
						SSV DT	75
						kNN, k= 18, Euclid	74.8
						CART DT	74.7
						DB-CART	74.4
						ASR	74.3
						AIRS2	74.2
						AIRS1	74.1
						SSV DT	73.7
						C4.5 DT	73.0
						Bayes	72.2
						CART	72.8
						Kohonen	72.7
						ID3	71.7
						IB3	71.7
						IB1	70.4
						C4.5 rules	67.0
						OCN2	65.1

TABLE 5. RAIRS and AIRS1 results (accuracy) on iris, pima-diabets, ionosphere, sonar, heart–statlog, hepatitis, Wisconsin-breast cancer, dry bean dataset, steel plates faults, HTRU2 and yeast datasets.

Datasets	**Validation methods**	**RAIRS Accuracy (%)**	**AIRS1 Accuracy (%)**
Iris	Cross validation 5 folders	100	96.7
Pima-Diabetes	Cross validation 10 folders	79.22	74.1
Ionosphere	200 : Training, 151 : Test	97.35	94.9
Sonar	Cross validation 13 folders	94.11	84.0
Heart-Statlog	Cross validation 10 folders	92.59	85.19
Hepatitis	Cross validation 10 folders	90.67	88.89
Wisconsin-Breast Cancer	Cross validation 10 folders	98.52	97.2
Dry Bean Dataset	Cross validation 10 folders	94.71	92.45
Steel Plates Faults	Cross validation 10 folders	99.31	98.97
HTRU2	Cross validation 10 folders	96.87	96.64
Yeast	Cross validation 10 folders	57.33	52.05

TABLE 6. Running time, execution time and search time on mc_{match} of RAIRS and AIRS1.

Datasets	**RAIRS**			**AIRS1**		
	Search time on mc_{match} (s)	Training time (s)	Execution time (s)	Search time on mc_{match} (s)	Training Time (s)	Execution time (s)
Iris	0.0051	0.7535	0.0030	0.0043	0.4780	0.0061
Pima-Diabetes	0.111	15.5118	0.0091	0.2960	8.0345	0.3063
Ionosphere	0.0795	64.5060	0.0442	0.2087	11.8576	0.4203
Sonar	0.0718	33.4264	0.0085	0.0804	12.9322	0.0190
Heart-Statlog	0.03795	11.8483	0.0101	0.0538	4.3603	0.0366
Hepatitis	0.0082	4.0762	0.0010	0.0085	1.2475	0.0033
Wisconsin-Breast Cancer	0.1082	12.8517	0.0089	0.1733	7.9575	0.1510
Dry Bean Dataset	2.9526	655.3306	0.1904	8.4879	521.7762	56.7468
Steel Plates Faults	1.1883	90.9360	0.0651	3.0731	62.6659	2.2684
HTRU2	3.4724	2145.8260	0.1860	38.4828	569.8125	80.4169
Yeast	0.1821	27.9603	0.0126	0.2954	14.7843	0.8629

TABLE 7. Comparison of RAIRS accuracy with other state-of-the-art methods on heart–statlog, hepatitis and Wisconsin-breast cancer datasets.

Accuracy(%)					
Heart-Statlog		Hepatitis		Wisconsin-Breast Cancer	
Naive Bayes and Multi-layer perceptron [43]	96.66	Back Propagation Neural Network [44]	94.0	XGBoost+ RFE [45]	99.02
RAIRS	**92.59**	LSSVM with IACA algorithm [46]	93.7	KE's algorithm [47]	99.01
RFRS Heart Diseases Prediction Scheme [48]	92.59	**RAIRS**	**90.67**	**RAIRS**	**98.52**
MLMNB [49]	90.6	Regression Logic [50]	83.33	AR-SVM [51]	98.0
SCC [52]	88.8	A Radial Basis Function Network (RBFN) [53]	88.04	DL+DA [51]	97.91
PSO with NB, Dulhare [54]	87.91	SDD,SVD and Multilayer Perceptron [55]	87.7	PSO+Kmeans+BP [56]	97.82
PSO with NB, Wijaya [57]	86.67	WLS-TSVM + Gaussian Kernel Function [58]	85.61	BK with IVFS [59]	95.26
LVQ [60]	85.07	Decision Table [61]	85.3	BPNN optimization + Nelder Mead [62]	89.8

and in [37]) (AIRS2) which also proposes an improvement of AIRS1. In addition, the proposed algorithm is compared with other state-of-the-art methods such as RBF, C4.5 decision trees, SVM and MLP+BP, etc. on the same datasets. A brief description of the used datasets is shown in Table 2. The presented results are the outputs of several cross-validation based tests varying from 5 to 13 folder depending on the nature of the dataset. Except for the Ionosphere dataset in which 200 and 151 samples are selected respectively for training and testing.

TABLE 8. The search complexity on mc_{match} and the k nearest neighbors.

AIRS1	AIRS2	RAIRS
$O(n)$	$O(n)$	$O(log_2(n))$

After several tests, the best results were obtained with the parameters presented in Table 3. The mutation rate was fixed empirically after several tests as 0.1. The AIRS algorithm, like many other classification tools, needs to fix its parameters empirically, and we found no work in the state of the art which can give the way to choose correctly these values.

We noticed that the mutation rate should be kept very low, otherwise convergence may be delayed unnecessarily.

Note that the same parameters were used for all tests of all datasets. The proposed system has achieved better results when the parameters were tuned separately for each dataset.

The comparative study presented in Tables 4, 5, and 7, shows clearly that the Revised AIRS has obtained good results and achieved good rank in the most of datasets. The proposed RAIRS have surpassed the basic AIRS (AIRS1) in all used datasets and have reached the best accuracy rate in Iris, Ionosphere, Sonar, Pima-Diabetes, Heart-Statlog, Dry Bean Dataset, Steel Plates Faults, HTRU2 and Yeast datasets. Moreover, we have reached an accuracy rank better than the improved AIRS (AIRS2) in the datasets: Iris, Ionosphere, Sonar and Pima-Diabetes. This is due to the corrections made to the basic algorithm, which allowed on the one hand a better start of the system thanks to the new initialization process and on the other hand to the improvements made in almost all the levels of the basic AIRS. However, the proposed method has ranked second in Ionosphere dataset and third in: Wisconsin-Breast Cancer and Hepatitis datasets.

Table 6 shows clearly that the Revised Artificial Immune Recognition System (RAIRS) achieved a good search time on mc_{match} and a better execution time compared to the AIRS1, especially in the big datasets. This is due to the use of a kd-tree to structure the MC set.

It is really that the training times of RAIRS are not improved, due to the added functions **Update_Weight_DL** and **Remove_mc_Inactive** and the calculation of the mean vectors. All these modifications have improved the quality of the results.

From the Table 8, we can see that the search complexity of RAIRS has been significantly improved. Indeed, the reformulation of the MC set as a binary search tree decreased the search complexity of the mc_{match} and the k nearest neighbors from sequential to logarithmic.

To get a more accurate idea on the obtained results, the statistical P-value test was used under the null hypothesis ($H0$) that there are no significant performance differences on the used datasets. The P-value was calculated using the Friedman aligned ranks test and compared with the significance level 0.05. The obtained P-value is 0.0035 which means that the proposed RAIRS gave significant results on all datasets since the calculated P-value is smaller than the significance threshold.

V. CONCLUSION

The Artificial Immune Recognition System (AIRS) has achieved great success in solving optimization and classification problems. However, the initial version of AIRS suffers from a high computation cost, an exponential growth for the generated data, and the algorithm's complexity.

For this purpose, a revised version of AIRS was proposed in this article. The RAIRS introduces some new mechanisms: deleting inactive mc to avoid data explosion, adding the concept of weight and lifetime counter for each mc to improve quality, and selecting only the best representative cells. In addition, slight modifications in the AIRS functionalities were made like the mutation function and the memory cell introduction mechanism. We also proposed to replace the structure of the MC set by a binary tree structure (kd-tree) to reduce the search complexity and calculation time.

The evaluations of the proposed RAIRS on some UCI datasets show the effectiveness and the improvements that we have obtained compared with: the original version of AIRS, an improvement of the AIRS, and some public methods on the same datasets.

As future work, the RAIRS can be used to resolve the immunological problems for predicting the binding or non-binding of T-cell receptors. The AIRS can be improved even further by using a multi-view learning approach. Also, we investigate the possible use of deep learning for the AIRS.

REFERENCES

[1] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, May 1994, pp. 202–212.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[3] J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," *Biosystems*, vol. 55, nos. 1–3, pp. 143–150, Feb. 2000.

[4] J. Timmis, M. Neal, and J. Hunt, "Data analysis using artificial immune systems, cluster analysis and Kohonen networks: Some comparisons," in *Proc. IEEE SMC Conf. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1999, pp. 922–927.

[5] A. Watkins, "A resource limited artificial immune classifier," Ph.D. dissertation, Dept. Comput. Sci., Mississippi State Univ., Starkville, MS, USA, 2001.

[6] A. Watkins and L. Boggess, "A new classifier based on resource limited artificial immune systems," in *Proc. Congr. Evol. Comput. (CEC)*, May 2002, pp. 1546–1551.

[7] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[8] E. Ozturk Kiyak, D. Birant, and K. U. Birant, "An improved version of multi-view k-nearest neighbors (MVKNN) for multipleview learning," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 29, no. 3, pp. 1401–1428, 2021.

[9] V. Vapnik, "Pattern recognition using generalized portrait method," *Automat. Remote Control*, vol. 24, no. 6, pp. 774–780, Jan. 1963.

[10] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, Aug. 1997.

[11] S. F. Ershad and S. Hashemi, "To increase quality of feature reduction approaches based on processing input datasets," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, May 2011, pp. 367–371.

[12] R. Siouda, M. Nemissi, and H. Seridi, "ECG beat classification using neural classifier based on deep autoencoder and decomposition techniques," *Prog. Artif. Intell.*, vol. 10, pp. 333–347, Apr. 2021.

[13] B. Adil, K. M. Nadjib, and L. Yacine, "A novel approach for facial expression recognition," in *Proc. Int. Conf. Netw. Adv. Syst. (ICNAS)*, Jun. 2019, pp. 1–5.

[14] A. Boughida, M. N. Kouahla, and Y. Lafifi, "A novel approach for facial expression recognition based on Gabor filters and genetic algorithm," *Evolving Syst.*, pp. 1–15, Jul. 2021.

[15] R. J. S. Raj, S. J. Shobana, I. V. Pustokhina, D. A. Pustokhin, D. Gupta, and K. Shankar, "Optimal feature selection-based medical image classification using deep learning model in Internet of Medical Things," *IEEE Access*, vol. 8, pp. 58006–58017, 2020.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[17] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.

[18] E. Bonabeau, M. Dorigo, D. d. R. D. F. Marco, G. Theraulaz, and G. Théraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, no. 1. London, U.K.: Oxford Univ. Press, 1999.

[19] M. Dorigo and L. M. Gambardella, "A study of some properties of Ant-Q," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 1996, pp. 656–665.

[20] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. MHS 6th Int. Symp. Micro Mach. Hum. Sci.*, Oct. 1995, pp. 39–43.

[21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes Univ., Eng. Fac., Comput., Kayseri, Turkey, Tech. Rep. tr06, 2005.

[22] X. Li, Z. Shao, and J. Qian, "An optimizing method based on autonomous animats: Fish-swarm algorithm," *Syst. Eng. Theory Pract.*, vol. 22, no. 11, pp. 32–38, Nov. 2002.

[23] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*. Berlin, Germany: Springer, 2010, pp. 65–74.

[24] B. Bullnheimer, R. F. Hartl, and C. Strauß, "A new rank based version of the ant system. A computational study," SFB Adapt. Inf. Syst. Model. Econ. Manage. Sci., Univ. Vienna, Vienna, Austria, Comput. Study Work. Paper, 1997, pp. 25–38, vol. 7, no. 1.

[25] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2010.

[26] Y. Gheraibia and A. Moussaoui, "Penguins search optimization algorithm (PeSOA)," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.* Berlin, Germany: Springer, 2013, pp. 222–231.

[27] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[28] Y.-W. Zhang, L. Wang, and Q. D. Wu, "Dynamic adaptation cuckoo search algorithm," *Control Decis.*, vol. 29, no. 4, pp. 617–622, 2014.

[29] W. Li and W. Zhang, "Method of tasks allocation of multi-UAVs based on particles swarm optimization," *Control Decis.*, vol. 25, no. 9, pp. 1359–1363, 2010.

[30] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization," *Eng. Optim.*, vol. 38, no. 2, pp. 129–154, 2006.

[31] N. K. Jerne, "Towards a network theory of the immune system," *Annu. Immunol.*, vol. 125, no. 3, pp. 373–389, 1974.

[32] J. Timmis, E. Hart, A. Hone, M. Neal, A. Robins, S. Stepney, and A. Tyrrell, *Immuno-Engineering*, vol. 268. Berlin, Germany: Springer, 2008.

[33] S. J. Nanda, "Artificial immune systems: Principle, algorithms and applications," M.S. thesis, Dept. Electron. Commun. Eng., Nat. Inst. Technol., Rourkela, India, 2009.

[34] L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 239–251, Jun. 2002.

[35] P. Matzinger, "The danger model: A renewed sense of self," *Science*, vol. 296, no. 5566, pp. 301–305, 2002.

[36] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Phys. D, Nonlinear Phenomena*, vol. 22, nos. 1–3, pp. 187–204, Oct./Nov. 1986.

[37] A. Watkins, J. Timmis, and L. Boggess, "Artificial immune recognition system (AIRS): An immune-inspired supervised learning algorithm," *Genetic Program. Evolvable Mach.*, vol. 5, no. 3, pp. 291–317, Sep. 2004.

[38] L. N. De Casto and F. J. Von Zuben, "An evolutionary immune network for data clustering," in *Proc. 6th Brazilian Symp. Neural Netw.*, vol. 1, Nov. 2000, pp. 84–89.

[39] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time KD-tree construction on graphics hardware," *ACM Trans. Graph.*, vol. 27, no. 5, p. 126, 2008.

[40] *UCI Machine Learning Repository*. Accessed: Dec. 20, 2018. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.php

[41] W. Duch. (2000). *Datasets Used for Classification: Comparison of Results*. [Online]. Available: http://158.75.5.90/kmk/projects/datasets.html

[42] W. Duch. (2000). *Logical Rules Extracted From Data*. [Online]. Available: http://www.phys.uni.torun.pl/kmk/projects/rules.html

[43] M. Raman, V. K. Sharma, S. Hiranwal, and A. K. Bairwa, "Efficient method for prediction accuracy of heart diseases using machine learning," in *Proc. Int. Conf. Commun. Comput. Technol.* Singapore: Springer, 2021, pp. 113–121.

[44] M. Mitra and R. Samanta, "Hepatitis disease diagnosis using multiple imputation and neural network with rough set feature reduction," in *Proc. 3rd Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*. Cham, Switzerland: Springer, 2015, pp. 285–293.

[45] S. A. Abdulkareem and Z. O. Abdulkareem, "An evaluation of the Wisconsin breast cancer dataset using ensemble classifiers and RFE feature selection," *Int. J. Sci., Basic Appl. Res.*, vol. 55, no. 2, pp. 67–80, 2021.

[46] N. P. Husain, N. N. Arisa, P. N. Rahayu, A. Z. Arifin, and D. Herumurti, "Least squares support vector machines parameter optimization based on improved ant colony algorithm for hepatitis diagnosis," *Jurnal Ilmu Komputer Dan Informasi*, vol. 10, no. 1, pp. 43–49, 2017.

[47] G. Priyanka, P. K. Sahoo, V. Rohith, and K. Eswaran, "Breast cancer prediction system using KE Sieve algorithm," *Int. J. Sci. Eng. Res.*, vol. 10, no. 1, pp. 19–21, 2019.

[48] X. Liu, X. Wang, Q. Su, M. Zhang, Y. Zhu, Q. Wang, and Q. Wang, "A hybrid classification system for heart disease diagnosis based on the RFRS method," *Comput. Math. Methods Med.*, vol. 2017, pp. 1–11, Jan. 2017.

[49] N. S. Harzevili and S. H. Alizadeh, "Mixture of latent multinomial naive Bayes classifier," *Appl. Soft Comput.*, vol. 69, pp. 516–527, Aug. 2018.

[50] G. V. Nivaan and A. W. Emanuel, "Analytic predictive of hepatitis using the regression logic algorithm," in *Proc. 3rd Int. Seminar Res. Inf. Technol. Intell. Syst. (ISRITI)*, Dec. 2020, pp. 106–110.

[51] A. Ed-daoudy and K. Maalmi, "Breast cancer classification with reduced feature set using association rules and support vector machine," *Netw. Model. Anal. Health Informat. Bioinf.*, vol. 9, no. 1, pp. 1–10, Dec. 2020.

[52] T. Karadeniz, G. Tokdemir, and H. H. Maraş, "Ensemble methods for heart disease prediction," *New Gener. Comput.*, vol. 39, no. 3, pp. 1–13, 2021.

[53] B. S. Alshamrani and A. H. Osman, "Investigation of hepatitis disease diagnosis using different types of neural network algorithms," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 2, p. 242, 2017.

[54] U. N. Dulhare, "Prediction system for heart disease using naive Bayes and particle swarm optimization," *Biomed. Res.*, vol. 29, no. 12, pp. 2646–2649, 2018.

[55] I. Hussien, S. Omer, N. E. Oweis, and V. Snášel, "Feature selection using semi discrete decomposition and singular value decompositions," in *Proc. 1st Int. Sci. Conf. Intell. Inf. Technol. Ind. (IITI)*. Russia: Springer, 2016, pp. 87–97.

[56] S. Roguia and N. Mohamed, "An optimized RBF-neural network for breast cancer classification," *Int. J. Informat. Appl. Math.*, vol. 1, no. 1, pp. 24–34, 2019.

[57] S. H. Wijaya, G. T. Pamungkas, and M. B. Sulthan, "Improving classifier performance using particle swarm optimization on heart disease detection," in *Proc. Int. Seminar Appl. Technol. Inf. Commun.*, Sep. 2018, pp. 603–608.

[58] D. Tomar, S. Singhal, and S. Agarwal, "Weighted least square twin support vector machine for imbalanced dataset," *Int. J. Database Theory Appl.*, vol. 7, no. 2, pp. 25–36, Apr. 2014.

[59] C. K. Lim and C. S. Chan, "A weighted inference engine based on interval-valued fuzzy relational theory," *Expert Syst. Appl.*, vol. 42, no. 7, pp. 3410–3419, May 2015.

[60] A. L. Brun, A. S. Britto, Jr., L. S. Oliveira, F. Enembreck, and R. Sabourin, "A framework for dynamic classifier selection oriented by the classification problem difficulty," *Pattern Recognit.*, vol. 76, pp. 175–190, Apr. 2018.

[61] S. O. Hussien, S. S. Elkhatem, N. Osman, and A. O. Ibrahim, "A review of data mining techniques for diagnosing hepatitis," in *Proc. Sudan Conf. Comput. Sci. Inf. Technol. (SCCSIT)*, Nov. 2017, pp. 1–6.

[62] E. Kusuma, U. D. Nuswantoro, G. Shidik, R. Pramunendar, U. D. Nuswantoro, and U. D. Nuswantoro, "Optimization of neural network using Nelder Mead in breast cancer classification," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 6, pp. 330–337, Dec. 2020.

WAFA NEBILI received the bachelor's and master's degrees (Hons.) in computer science from 8 Mai 1945 Guelma University, Algeria, in 2013 and 2015, respectively, where she is currently pursuing the Ph.D. degree in computer science. She is also a member of the Laboratory of Science and Information Technologies and Communication (LabSTIC, http://labstic.univ-guelma.dz/fr). She is interested in research on video surveillance, data mining, big data, and artificial intelligence.

BRAHIM FAROU received the State Engineer degree in computer science systems from the National School of Computer Science, Algiers, Algeria, in 2006, the Magister degree in sciences and technologies of information and communication from Guelma University, in 2009, and the D.Sc. degree (Hons.) in computer science and the H.D.R. degree (Hons.) from the University of Annaba, Algeria, in 2016 and 2018, respectively. He also occupied many administrative positions, he was the Deputy Head of the Department responsible for teaching, from 2010 to 2012, and is currently the Deputy Head of the Department responsible for post-graduation. He is currently an Associate Professor with the Department of Computer Science, Guelma University, and a GADM Team Member of LabSTIC Laboratory. His research interests include color constancy, video mining, object recognition, object tracking, surveillance systems, and computer vision.

ZINEDDINE KOUAHLA graduated from the University of Burgundy, in 2008. He defended the Ph.D. thesis at the University of Nantes, in February 2013. The thesis, with indexation in the metric spaces index tree and parallelization, was prepared at the Laboratoire des Sciences du Numérique de Nantes. He is currently a Teacher-Researcher at 8 Mai 1945 Guelma University, more precisely at the Faculty of Mathematics, Computer Science and Sciences of the Material. His current research interests include multimedia databases: multimedia information modeling and structuring, content search, hypermedia navigation implement systems for recognizing, indexing, and searching and classifying multimedia documents automatic management of multimedia documents (text, image, and video).

HAMID SERIDI received the bachelor's degree (Hons.) in electrical engineering from the University of Annaba, Algeria, in 1981, the master's degree in electrical engineering from the Polytechnic Institute of New York, USA, in 1984, and the Ph.D. degree (Hons.) in computer science from the University of Reims Champagne-Ardenne, France, in 2001. He was the Vice Dean of the post-graduation, scientific research, and external relations with the University of Guelma. He is currently a Professor and the Director of the Laboratory of Science and Information Technologies and Communication (LabSTIC, http://labstic.univ-guelma.dz/fr). His research interests include approximate knowledge management, pattern recognition and artificial intelligence, data mining, video mining, machine learning, and cryptography. He is an Expert Member of the National Committee for Evaluation and Accreditation National Projects Research. He is also the Chairperson of the Scientific Council of the Faculty of Mathematics and Computing and Material Sciences.