A Mathematical Analysis of the Genetic-AIRS Classification Algorithm

Dimitrios Mathioudakis, Dionysios Sotiropoulos and George A. Tsihrintzis
Department of Informatics, University of Piraeus
Piraeus 185 34, GREECE
Email: mathiou0@gmail.com, dsotirop@gmail.com, geoatsi@unipi.gr

***Abstract*—This paper presents the inception and the basic concepts of a hybrid classification algorithm called Genetic-AIRS [1]. Genetic-AIRS, is a combination of the Artificial Immune Resource System (AIRS) algorithm witch uses evolutionary computation techniques. An analysis is presented to determine the final algorithm architecture and parameters. The paper also includes an experimental evaluation on various publicly available datasets of Genetic-AIRS vs AIRS.**

***Index Terms*—Artificial immune system, Genetic algorithm, Evolutionary computation, Machine learning, Classification**

I. Introduction

Machine learning focuses on the construction of algorithms that can optimize a performance criterion using samples or past experience [2]. Given a model which is defined up to some parameters, learning then consists of execution of a computer program that can optimize the parameters of this model and generalize this methods using the training data or past experience. The model may be either predictive or descriptive to export knowledge from data or both predictive and descriptive [3]. As a field, machine learning has seen tremendous growth over the recent years, both in terms of theoretical advances and in range of applications [4].

Depending on the type of learning strategy and outcome produced, machine learning is categorized into two main categories, which are called *supervised* and *unsupervised* learning [5].

More specifically, in supervised learning, the machine learns inductively via generalization of a set of training samples. For each sample, its class of origin is known and the task of the supervised learning algorithm is to learn to classify samples other than the ones used during training. Cases, such as for example digit recognition, in which the task is to assign each input feature vector to one of a finite number of categories, are called *classification* problems. If, on the other hand, the output consists of one or more continuous variables, then the corresponding learning tasks are referred to as *regression* problems [6].

In unsupervised learning, on the other hand, the class of origin of the available samples is unknown. The number of classes of origin itself may be either known or unknown. The goal in such unsupervised learning problems may be (i) to find similarities in a data set, which corresponds to a *clustering* problem, or (ii) to estimate the distribution of the data within the input space, which is referred to as a *density estimation* problem, or (iii) to visualize the data from a high-dimensional space down to lower dimensions for the purpose of *visualization* [6].

Clearly, the goals of both supervised and unsupervised learning are difficult to achieve. At the same time, evolution has equipped biological organisms with the ability to address similar problems in efficient ways. The sophistication, the ability to overcome abnormalities in input and the adaptability of biological systems represent a strong motivation for mimicking the mechanisms of natural evolution in an attempt to build algorithms with characteristics similar to those of biological systems. Many decades ago, computer scientists began developing methods inspired by natural evolution to solve problems that were too difficult to address with use of analytical methods. Mimicking biological systems in an artificial sense gives rise to a bundle of machine learning and computational intelligence paradigms, collectively referred to as *biologically-inspired computing*. *Artificial immune systems* are such an example of biologically-inspired systems that are capable of learning, memorizing and recognizing patterns in signals in a efficient way.

Biological immune systems are the outcome of a lengthy natural process, called evolution [7]. They are exceedingly complex and are typically formed by several components that work in coordination. Their task is to protect the organism that carries them from external pathogens, such as bacteria and viruses which have entered the host body and to eliminate them with minimal cost to the host. Biological immune systems perform a self/non-self discrimination process that allows them to identify cells of the organism and distinguish them from intruders.

On the other hand, *genetic algorithms* constitute another biologically-inspired learning methodology motivated by an analogy to the biological evolution process [2]. They are search algorithms based on the principles of *random mutations*, *random crossovers*, *natural selection* and *survival of the fittest*. As such, genetic algorithms aim at mimicking the natural evolution process and provide efficient solutions in search and learning optimization problems.

More specifically, genetic algorithms are essentially stochastic algorithms. They begin with a randomly initial population of solutions and compute generations of evolved solutions based on random mutations and random crossovers of existing solutions. At each step, the current population is updated by

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

replacing some fraction of the population by offsprings of the fittest. The efficiency of each solution is ranked according to an objective fitness function and the algorithmic loop is repeated either a pre-specified number of times defined by the user or until a termination criterion is met.

Genetic algorithms are conceptually simple as they only require the ability to compute an objective fitness function and not its derivatives [8]. They provide a broad paradigm to solving extensive classes of search and optimization problems and have proven particularly useful when the objective fitness function is such that classical optimization methods are not applicable.

In this paper, we combine the two biologically-inspired paradigms mentioned previously, namely artificial immune systems and genetic algorithms. The outcome is a new classification algorithm, called *Genetic-AIRS*. The remainder of the paper is organized as follows: Section 2 presents a brief review of the basic concepts of Artificial Immune Recognition System (AIRS) algorithm. In Section 3, we present a mathematical analysis of Genetic-AIRS, compare it to the known AIRS algorithm and identify advantages and disadvantages. In Section 4, we perform an experimental evaluation of Genetic-AIRS and, finally, in Section 5, we summarize the paper and point to future related research.

II. Artificial Immune Recognition System (AIRS)

A. Immune systems

Artificial immune systems (AIS) constitute a machine learning/computational intelligence paradigm based on immunology and have received a lot of research interest since the 1980's. Biological immune systems in vertebrates are capable of efficiently performing complex computations, learning and memorizing in a parallel and distributed manner. Their task is to protect the organism that carries them from external pathogens, such as bacteria and viruses. They achieve this task via a *self/non-self discrimination* process that allows them to identify cells of the organism and distinguish them from intruders. Biological immune systems consist of an *innate* and an *adaptive* subsystem. The former is reprogrammed to recognize specific antigens, while the latter is capable of learning.

Artificial immune systems attempt to mimic the functions of the adaptive subsystem of biological immune systems. One of the best known and efficient classification algorithms based on artificial immune systems is the *Artificial Immune Recognition System (AIRS)* [9]. AIRS does not attempt to model the mechanisms of the immune system, but rather borrows some of its features, namely clone selection, maturation, and hypermutation [10]. AIRS is characterized by:

1) self-regulation, i.e., the ability to learn and adapt its architecture depending on the problem,
2) competitive performance, i.e., AIRS performance ranks within the top five state-of-the-art classifiers,
3) generalization via data reduction, i.e., AIRS outperform other classifiers while using less data reduction, and
4) parameter stability, i.e., the methods uses to auto-tune its parameters to achieve different accuracy on different data.

The AIRS learning algorithm was originally conceived in an attempt to demonstrate that artificial immune systems are capable for the task of classification. More specifically, the goal of the AIRS algorithm is, given a training set of samples from a data class (*antigens*), to produce a set of *memory antibodies*, which are used to recognize this class. For a complete presentation of AIRS, the reader is referred to [9].

III. Genetic AIRS

A. The basic concept

Despite the fact that significant research efforts have been put on methods and techniques of both genetic algorithms and artificial immune systems, cross-fertilization from the two disciplines has been only limited so far. In this paper, we present a new classification algorithm, called *Genetic-AIRS* which combines genetic algorithms with artificial immune systems [1]. Genetic-AIRS addresses the classification problem globally in contrast to the AIRS algorithm which addresses the classification task locally. This is due to the fact that genetic algorithms perform a global search in the search space. The new algorithm achieves higher classification accuracy when compared to AIRS. More specifically, Genetic-AIRS [1] utilizes detectors, which play the role of memory antibodies, and locate the samples in the training set. Thus, the set of detectors forms a representation of the spatial distribution of the training set.

Each detector x is defined by a sphere $B_r p = \{d(x,p) \leq r\}$ where p is a sample in the training set and r is the radius of the sphere. The radius r specifies how close the detector x can be to the object p that will be detected. Thus, we can model the problem of selecting appropriate detectors as a problem of optimizing the distance of the detector set from the samples in the training set. In this paper, we address this optimization problem with a genetic algorithm.

B. Optimization problem definition

Let us describe the classification problem formally. Let $Data$ be the set we want to classify which consists of N feature vectors. If C is a class of $Data$ then $C_i \subseteq Data$ with $i \in \{1, 2 \cdots S\}$ and $\{C_1, C_2, \cdots, C_S\} = Data$

We can define the training sets $\widehat{C_1}, \widehat{C_2}, \cdots \widehat{C_S}$ which are subsets of classes $C_1, C_2, \cdots C_S$, respectively, i.e. $\widehat{C_i} \subseteq C_i$ for $i = 1, 2, \cdots S$, and

$$\begin{aligned} \widehat{C_1} &= \{\overrightarrow{x_1}^1, \overrightarrow{x_2}^1, \cdots, \overrightarrow{x_n}^1\} \\ \widehat{C_2} &= \{\overrightarrow{x_1}^2, \overrightarrow{x_2}^2, \cdots, \overrightarrow{x_n}^2\} \\ &\vdots \\ \widehat{C_S} &= \{\overrightarrow{x_1}^S, \overrightarrow{x_2}^S, \cdots, \overrightarrow{x_n}^S\} \end{aligned}$$

where

$$\begin{aligned} &\overrightarrow{x_j}^\kappa \in U_n = [0,1]^L, \\ &\forall \kappa \in \{1, 2, \cdots, S\}, \end{aligned}$$

$$\forall j \in [1, 2, \cdots n].$$

The variable $\overrightarrow{x_j}^{\kappa}$ is the j-th element of the κ-st class with value in [0,1]. Moreover, to form the detector set, we take a subset of the training set.

$$\begin{aligned}\widehat{D_1} &= \{\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1\}\\ \widehat{D_2} &= \{\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2\}\\ &\vdots\\ \widehat{D_S} &= \{\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S\}\end{aligned}$$

where

$$\overrightarrow{d_j}^{\kappa} \in U_n = [0,1]^L,$$

$$\forall \kappa \in \{1, 2, \cdots, S\},$$

$$\forall j \in [1, 2, \cdots m].$$

With $m \leq n$ so the detector set has less attributes than the training set. Every detector $\overrightarrow{d_j}^i \in \widehat{D_i}$ with $i = 1, 2, \cdots, S$ and $j = 1, 2, \cdots, m$ defines a sphere of radius $emax$ which is further defined by the user. This radius defines the distance between the detector and the element that needs to be detected $\|\overrightarrow{x_i}^{\kappa} - \overrightarrow{d_j}^{\kappa}\|_{norm} - emax$. When a detector is $emax$ close to the element $\|\overrightarrow{x_i}^{\kappa} - \overrightarrow{d_j}^{\kappa}\|_{norm} \leq emax$ then the detector is considered to be close enough to the element and thus is excluded from the detector set. A minimization problem of the total distance is faced between the detector and the training set. For this reason we define an objective function f_κ that represent the total distance between the detector and the training set.

C. Objective function

`Minimize`

$$f_\kappa(\overrightarrow{d_1}^{\kappa}, \overrightarrow{d_2}^{\kappa}, \cdots, \overrightarrow{d_m}^{\kappa}) = \sum_{i=1}^{n} {\xi_i}^{\kappa}$$

`such that`

$${\xi_i}^{\kappa} = \min_{j\in[m]}\{{\phi_{ij}}^{\kappa}\}, \quad \forall i \in [n]$$

`with`

$${\phi_{ij}}^{\kappa} = \max\{0, \|\overrightarrow{x_i}^{\kappa} - \overrightarrow{d_j}^{\kappa}\|_{norm} - emax\}$$

$$\overrightarrow{d_j}^{\kappa} \in U_n = [0,1]^L, \ \ \forall j \in [m]$$

`and`

$$\kappa \in [1, 2, \cdots, S].$$

Here, κ the κ-th class; thus, for all of the S classes we will have a total of S optimization problems.

$$\begin{aligned}&\texttt{Minimize} \quad f_1(\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1) = \sum_{i=1}^{n} {\xi_i}^1\\ &\texttt{Minimize} \quad f_2(\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2) = \sum_{i=1}^{n} {\xi_i}^2\\ &\qquad\vdots\\ &\texttt{Minimize} \quad f_S(\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S) = \sum_{i=1}^{n} {\xi_i}^S\end{aligned}$$

`such that`

$$\begin{aligned}{\xi_i}^1 &= \min_{j\in[m]}\{{\phi_{ij}}^1\}, \quad \forall i \in [n]\\ {\xi_i}^2 &= \min_{j\in[m]}\{{\phi_{ij}}^2\}, \quad \forall i \in [n]\\ &\vdots\\ {\xi_i}^S &= \min_{j\in[m]}\{{\phi_{ij}}^S\}, \quad \forall i \in [n]\end{aligned}$$

`with`

$$\begin{aligned}{\phi_{ij}}^1 &= \max\{0, \|\overrightarrow{x_i}^1 - \overrightarrow{d_j}^1\|_{norm} - emax\}\\ {\phi_{ij}}^2 &= \max\{0, \|\overrightarrow{x_i}^2 - \overrightarrow{d_j}^2\|_{norm} - emax\}\\ &\vdots\\ {\phi_{ij}}^S &= \max\{0, \|\overrightarrow{x_i}^S - \overrightarrow{d_j}^S\|_{norm} - emax\}\\ \overrightarrow{d_j}^{\kappa} &\in U_n = [0,1]^L, \ \ \forall j \in [m]\end{aligned}$$

`and`

$$\kappa \in [1, 2, \cdots, S].$$

Since $\overrightarrow{d}, \overrightarrow{x} \in U_n = [0,1]^L$, the normalized Euclidean distance is given by :

$$\|\overrightarrow{d} - \overrightarrow{x}\|_{norm} = \frac{1}{\sqrt{L}}\|\overrightarrow{d} - \overrightarrow{x}\|_{norm}$$

$f_1, f_2, \cdots, f_s$ take as an input:

$$\begin{aligned}\widehat{D_1} &= \{\overrightarrow{d_1}^1, \overrightarrow{d_2}^1, \cdots, \overrightarrow{d_m}^1\}\\ \widehat{D_2} &= \{\overrightarrow{d_1}^2, \overrightarrow{d_2}^2, \cdots, \overrightarrow{d_m}^2\}\\ &\vdots\\ \widehat{D_S} &= \{\overrightarrow{d_1}^S, \overrightarrow{d_2}^S, \cdots, \overrightarrow{d_m}^S\}\end{aligned}$$

as random detectors that initialize the optimization problem and produce the training detectors.

$$\begin{aligned}\widehat{D_1{}^*} &= \{\overrightarrow{d_1}^{1^*}, \overrightarrow{d_2}^{1^*}, \cdots, \overrightarrow{d_m}^{1^*}\}\\ \widehat{D_2{}^*} &= \{\overrightarrow{d_1}^{2^*}, \overrightarrow{d_2}^{2^*}, \cdots, \overrightarrow{d_m}^{2^*}\}\\ &\vdots\\ \widehat{D_S{}^*} &= \{\overrightarrow{d_1}^{S^*}, \overrightarrow{d_2}^{S^*}, \cdots, \overrightarrow{d_m}^{S^*}\}\end{aligned}$$

As a result the detector set $D_i{}^* \in [1, 2, \cdots S]$ is the output of the genetic algorithm. This output has different

distribution and contains better features than the variables in $\widehat{C} = \{\widehat{C_1}, \widehat{C_2}, \cdots, \widehat{C_S}\}$ because of the genetic algorithm optimization techniques. We use the training detector set ${D_i}^* \in [1, 2, \cdots S]$ to classify our dataset. Classification is computed with a simple majority vote of the nearest neighbors to each point $NN(D_i^*, \overrightarrow{x_i^*})$. A query point is assigned to the data class which has the most representatives within the nearest neighbors of the point. Nearest neighbors algorithm classifies the input dataset into groups, based on the grouping of the rows of the training detector set ${D_i}^* \in [1, 2, \cdots S]$.

This way, Genetic-AIRS is constructed as a hybrid optimization algorithm that uses immune system concepts for data representation, a genetic approach to find the best solution to the optimization problem and, at the end, the k-nearest neighbor algorithm as a classification method.

D. Similarities and differences between AIRS and Genetic-AIRS

AIRS and Genetic-AIRS are both based on the artificial immune systems. Thus, they exhibit several resemblances.

- In Genetic-AIRS, we introduce detectors which correspond to antibodies in AIRS. Detectors and antibodies serve the same purpose, that is, to discover feature vectors in the search space.
- The Euclidean metric is used in both Genetic-AIRS and AIRS to compute the distance in the search space.
- Mutation is used in both Genetic-AIRS and AIRS to genetically vary detectors and antibodies, respectively. Genetic-AIRS utilizes the mutation method from the genetic approach.
- Selection of both detectors and antibodies is based on highest affinity as a realization of the "Survival of the fittest" evolution process.
- Both Genetic-AIRS and AIRS data are normalized in [0,1].

There are, however, several differences between Genetic-AIRS and AIRS.

- The main difference among them lies in their corresponding data reduction methods. Specifically, AIRS uses an automated data reduction technique which achieves higher classification accuracy at the cost of requiring a higher number of training samples. On the other hand, Genetic-AIRS uses a standard, user-defined data reduction technique which allows for requiring fewer training samples without cost on classification accuracy.
- Another difference between Genetic-AIRS and AIRS lies in their execution process. The latter is executed locally, examining one antigen at a time . Genetic-AIRS, on the other hand, is executed globally and trains all detectors simultaneously. This results in higher execution time than required by AIRS.
- Genetic-AIRS and AIRS are differently initialized. Genetic-AIRS is initialized by random selection of an initial population of detectors near the mean value of the training samples, while AIRS is initialized by putting antigens to the set of memory antibodies.
- Each detector in Genetic-AIRS is characterized by a radius, which determines how close a sample can be to the detector in order to be detected.
- Besides a mutation process, a detector in Genetic-AIRS is genetically varied by a crossover process, as well.
- Genetic-AIRS, being based on a genetic algorithm, applies a survival-of-the-fittest strategy to select its detectors, while AIRS employs a resource allocation strategy to select memory antibodies.

IV. Experimental Evaluation

A. Description of test data

We apply a 10-fold cross validation technique to test the performance of Genetic-AIRS in various datasets including the dataset described in [1]. The classes were tested in groups of different size, including pairwise comparison for two, three, four, up to ten classes . In order to test and well-tune the Genetic-AIRS parameters, the algorithm was executed for 350 consecutive iterations using random values for each parameter. The Genetic-AIRS parameters include the following:

- Detectors set: is the set that detects the variables of our initial set. This set is a subset of our training set that will be trained to recognize our initial set. The values of this parameter in tests was 80% of the training set .
- Population Size: This parameter determines the size of the population at each generation. Increasing the population size enables the genetic algorithm to search more points in search space and obtain better results. On the other hand the larger the population size, the longer the genetic algorithm takes to compute each generation. The values of this parameter in tests was randomly selected from [20,150].
- Elite Count: The number of individuals with the best fitness values in the current generation that are guaranteed to survive to the next generation.
- Crossover Fraction: The fraction of individuals in the next generation that are created by crossover. The values of this parameter are a single uniformly random number between 0 and 1.
- Migration Fraction: Specifies how many individuals move between sub populations. The values of this parameter are a single uniformly random number between 0 and 1.
- Migration Interval: Specifies how many generation pass between migrations. The values are randomly selected from 1 to Population Size, [1,Population_Size].
- k : Specifies the k on the k nearest neighbour algorithm. The values are between [1,15].
- Emax: Is the variable that defines the distance between the detector and element that needs to be detected. The values are randomly selected from [0.1,1].
- Ranges : Specifies the range of the initial population. This parameter is a range of values between the mean value of the Dataset and a randomly selected variable. More formally, Initial Population=$(mean - Ranges, mean + Ranges)$. The values of this variable are a single uniformly random number between 0 and 1.

- The genetic algorithm parameters like Creation Function, Migration Function, Selection Function, Crossover Function are set to default value.

B. Evaluation results

In Table I we present the evaluation results. We evaluate these results testing our MatLab implementation of Genetic Algorithm compared to WEKA implementation of AIRS2. Both algorithms were tested with data reduction of 20% so we could have comparable values. For the first data set, namely Iris Plant, we observe that Genetic-AIRS slightly outperforms AIRS2 with 96.4% accuracy vs. 95% accuracy of AIRS2. For the second data set, namely Prima Indians Diabetes, Genetic-AIRS outperforms AIRS2 with 74.4% accuracy vs. 70.8% accuracy of AIRS2. Genetic-AIRS classifies correctly 565 out of 700 instances and for this reason it is considered to be in the top ten best classification algorithms for this data set. Genetic-AIRS performs worse than AIRS2 with the Sonar data set, achieving 63.52% accuracy vs. 66.01% accuracy of AIRS2. Finally, with the 10-Class Western Music data set, Genetic-AIRS outperforms AIRS2 by a margin of 1% to 6% in various tests.

TABLE I
EVALUATION RESULTS

DataSet	Genetic-AIRS Accuracy %	AIRS2 Accuracy %
Iris Plant	96.4	95
Prima Indians Diabetes	74.4	70.872
Sonar	63.52	66.01
C1vsC2	92	87
C1vsC2vsC3	67.7	61.6
C1vsC2vsC3vsC4	62.775	52.5
C1vsC2vs...vsC5	57.48	50.8
C1vsC2vs...vsC6	55.15	45.16
C1vsC2vs...vsC7	53.55	44.85
C1vsC2vs...vsC8	50.03	37.62
C1vsC2vs...vsC9	46.6	37.11
C1vsC2vs...vsC10	41.99	34.5

V. DISCUSSION

From our comparative evaluation, we can draw the following conclusions some of which were presented preliminary form in [1]:

- Genetic-AIRS uses fewer training samples (about 40-20% of those available), while AIRS uses all of the available training set.
- Genetic-AIRS is efficient and performs well independently of the size of the data set and may exceed the accuracy of AIRS. On the other hand, execution time may highly depend on the size of the input data.
- While AIRS uses samples from the training set, Genetic-AIRS uses the range parameter, which involves the mean value of the training set, in order to initialize the population.
- Because it is based on a genetic algorithm, the execution time of Genetic-AIRS is higher than that of AIRS.
- Genetic-AIRS, through its global searching mechanism, can achieve high accuracy with fewer Detectors. This leads to higher data reduction.
- Overall, Genetic-AIRS is competitive to AIRS and may, in fact, outperform AIRS.
- Genetic-AIRS forms a different and more integrated distribution of the data set even when only a small percentage of the data set is utilized. The data set is classified more efficiently with the knn classification algorithm than the AIRS algorithm which takes into account the entire data set.

VI. SUMMARY AND FUTURE RESEARCH

In this paper, we presented an analysis of Genetic-Airs, which is a hybrid classification method based on a fusion of Artificial Immune Systems and Genetic Algorithms. Specifically, we describe the mathematical definition of the corresponding optimization problem in order to explain our classification results. Furthermore, we present and describe the parameters of the Genetic-AIRS algorithm. Finally, we experimentally compared Genetic-AIRS to AIRS and identified their relative advantages and disadvantages. In the future, we will evaluate Genetic-AIRS further using on a variety of data sets. We will also present a statistical analysis of the parameters of the algorithm in order to find correlations between the user-defined parameters and the classification accuracy so as to fine-tune the parameters of the algorithm. We will extend Genetic-AIRS by incorporating weights on the classification features. We will also investigate mapping features into higher-dimension data or, on the contrary, reducing the data dimension. This and other research is currently under way and will be presented elsewhere in the near future.

REFERENCES

[1] D. Sotiropoulos, D. Mathioudakis, and G. A. Tsihrintzis, "Genetic-airs: A hybrid classification method based on genetic algorithms and artificial immune systems," pp. 1–5, July 2014.
[2] T. Mitchell, *Machine Learning*, jun 1997, vol. 4, no. 1.
[3] E. Alpaydn, *Introduction to machine learning*, 2014, vol. 1107.
[4] M. Arbib, D. Ballard, J. Bower, and G. Orban, *Neural Networks Algorithms , Applications*, 1994, vol. 7, no. 1.
[5] L. Moseley, "Introduction to machine learning," p. 334, 1988.
[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
[7] D. Fogel, "Evolutionary algorithms in theory and practice," *Complexity*, vol. 2, no. 4, pp. 26–27, mar 1997.
[8] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*, 1993.
[9] A. Watkins, J. Timmis, and L. C. Boggess, "Artificial Immune Recognition System (AIRS) : An Immune Inspired Supervised Machine Learning Algorithm," *Genetic Programming and Evolvable Machines*, vol. 5, no. 3, pp. 291–317, 2004.
[10] L. N. De Castro and F. J. Von Zuben, "aiNet: An Artificial Immune Network for Data Analysis," in *Data Mining A Heuristic Approach*, H. A. Abbass, R. A. Sarker, and C. S. Newton, Eds. Idea Group Publishing, 2001, ch. XII, pp. 231–259.