International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:4, 2010

A Hybrid Metaheuristic Framework for Evolving the PROAFTN Classifier

Feras Al-Obeidat, *Dept. of Computer Science, University of New Brunswick, Canada*
Nabil Belacel, *National Research Council, Canada*
Juan A. Carretero, *Dept. of Mechanical Engineering, University of New Brunswick, Canada*
Prabhat Mahanti, *Dept. of Computer Science, University of New Brunswick, Canada*

Abstract—In this paper, a new learning algorithm based on a hybrid metaheuristic integrating Differential Evolution (DE) and Reduced Variable Neighborhood Search (RVNS) is introduced to train the classification method PROAFTN. To apply PROAFTN, values of several parameters need to be determined prior to classification. These parameters include boundaries of intervals and relative weights for each attribute. Based on these requirements, the hybrid approach, named DEPRO-RVNS, is presented in this study. In some cases, the major problem when applying DE to some classification problems was the premature convergence of some individuals to local optima. To eliminate this shortcoming and to improve the exploration and exploitation capabilities of DE, such individuals were set to iteratively re-explored using RVNS. Based on the generated results on both training and testing data, it is shown that the performance of PROAFTN is significantly improved. Furthermore, the experimental study shows that DEPRO-RVNS outperforms well-known machine learning classifiers in a variety of problems.

Keywords—Knowledge Discovery, Differential Evolution, Reduced Variable Neighborhood Search, Multiple criteria classification, PROAFTN, Supervised Learning.

I. Introduction

DATA mining, a well-known paradigm in the area of knowledge discovery process, encompasses several techniques from different areas such as databases, machine learning and artificial intelligence. Two main techniques are broadly used in data mining to discover knowledge in data: supervised and unsupervised [1]. In supervised learning, the classification model is first established based on input of labeled (categorized) examples, where each example consists of vectors of features (attributes) and a class label. During the learning process, the classification model is built based on the available observations, and the induced model is evaluated using unknown instances. Unsupervised learning techniques are used to determine how the data are organized based on their characterizations. These techniques are distinguished from supervised learning in that the input is only made of unlabeled examples [2].

Data classification is a widely used supervised learning approach. It requires the development of a classification model that identifies behaviors and characteristics of the available objects to recommend the assignment of unknown objects to predefined classes [3]. The goal of the classification is to accurately assign the target class for each instance in the dataset. For instance, in medical diagnoses, patients are assigned to disease classes (positive or negative) according to a set of symptoms. In this context, the classification model is built on historical data and then the generated model is used to classify unseen instances.

Multi-criteria decision analysis (MCDA) [4] is another field that addresses the study of decision making [5] and classification problems. MCDA techniques were originally developed mainly in the fields of operations research, social psychology, and business management. In recent years, the field of MCDA has been attracting many researchers and decision-makers from many areas, including health, data mining and business [6]. The classification problem in MCDA consists of the formulation of the decision problem in the form of class prototypes that are used for assigning objects to classes. Each prototype is described by a set of attributes and is considered to be a good representative of its class [7].

This paper will focus on the MCDA classification method PROAFTN, which belongs to the class of supervised learning algorithms. PROAFTN is a relatively new classification method introduced in [8]. It has been applied to the resolution of many real-world practical problems such as medical diagnosis, asthma treatment, and e-Health [9], [10], [11]. PROAFTN has several advantages; for example, it uses the MCDA paradigm and therefore can be used to gain more understanding about the problem domain. Furthermore, PROAFTN has direct techniques that enable a Decision Maker (DM) to adjust its parameters.

PROAFTN is based on the preference relational system described by Roy [4]. Because of this, it employs a comparison between the alternatives through the scores of different attributes. So, it avoids resorting to distance measurements such as the Euclidean distance. Moreover, it helps overcome some difficulties encountered when data are expressed in different units or have different orders of magnitude. The aggregation on all criteria is on the results of the comparison between objects to be assigned and the prototypes. However, to apply PROAFTN, the values of several parameters need to be determined prior to classification. These parameters include the boundaries of intervals and the relative weights for each attribute. This consists of the formulation of the decision problem in the form of prototypes – representing each class – to be used for assigning each object to the target class.

Recently, some related work introduced in [12] was done to improve PROAFTN's performance. There, the unsupervised discretization algorithm *k*-means and a Genetic Algorithm are used to obtain PROAFTN intervals from data. In this paper, a new work is proposed based on Differential Evolution (DE)

and Reduced Variable Neighborhood Search (RVNS) to obtain PROAFTN parameters.

DE is a simple and efficient evolutionary algorithm proposed by Storn and Price [13] that has proven very effective over the last decade. The major properties of DE are its ability to handle non-differentiable, nonlinear, real-valued parameters which is the case in this work. DE has been applied in classification problems and partitional clustering [14]. Moreover, DE has been used for training Neural Networks [15], [16]. As demonstrated by most of the aforementioned applications, DE gets better results in a faster and more efficient way compared with other evolutionary population-based methods.

RVNS is a variation of the metaheuristic Variable Neighborhood search (VNS) [17] [18]. VNS and RVNS have been applied successfully in the domain of classification problems [19], [20], [21]. The main difference between DE and RVNS is that the former is a population-based metaheuristic, whereas the latter is a single-point based solution. Hence, integrating the two approaches improves the exploration and exploitation strategies.

Based on the aforementioned motivation and the structure of PROAFTN, a new methodology based on a hybrid metaheuristic is proposed in this study for training the PROAFTN method. In this context, a formulation of the optimization problem is introduced first; then, the integrated DE and RVNs are proposed for solving this optimization problem. The proposed training technique utilizes DE and RVNS to train and improve the performance of PROAFTN. During the learning phase, DE and RVNS are utilized as an inductive approach to infer the best PROAFTN parameters from the training samples. The generated parameters are then used to compose the prototypes, which represent the classification model that will be used for assigning unknown samples. The target is to find the prototypes that maximize the classification accuracy on each dataset.

The performance of the proposed approach (henceforth referred to as DEPRO-RVNS) is compared with well-known classification algorithms selected from different learning perspectives, including: Logical/Symbolic techniques such as Decision Tree (C4.5 [22]), Statistical learning algorithms, (*e.g.,* Naive Bayes (NB) [23]), Support Vector Machine (SVM [24]), Perceptron-based techniques (*e.g.,* Neural Networks (NN) [25]), Instance-based learning (*e.g., k*-nearest neighbor (*k*-nn) [26]), and the rule-based classifiers such as PART [1], [27]. The comparisons and evaluations are made on 12 well-known datasets by using a stratified 10-fold cross-validation [27].

The rest of the paper is organized as follows: in Section II, the PROAFTN method, the DE algorithm and RVNS are briefly presented. In Section III, the proposed approach DEPRO-RVNS to learn PROAFTN is introduced. The description of the datasets, experimental results, and comparative numerical studies are presented in Section IV. Finally, conclusions are summarized in Section V.

II. Overview of PROAFTN, DE and RVNS

In this section the PROAFTN methodology and the DE and RVNS algorithms are reviewed; the new learning algorithm DEPRO-RVNS is introduced in Section III.

A. PROAFTN Method

The PROAFTN procedure belongs to the class of supervised learning algorithms. Its functioning can be described as follows. From a set of n objects known as a training set, consider a is an object which requires classification; assume this object a is described by a set of m attributes $\{g_1, g_2, ..., g_m\}$ and let $\{C^1, C^2, ..., C^k\}$ be the set of k classes. The different steps of the classification procedure follow.

1) Initialization: For each class C^h, $h = 1, 2, ..., k$, a set of L_h prototypes $B^h = \{b_1^h, b_2^h, ..., b_{L_h}^h\}$ are determined. For each prototype b_i^h and each attribute g_j, an interval $[S_j^1(b_i^h), S_j^2(b_i^h)]$ and the preference thresholds $d_j^1(b_i^h) \geq 0$ and $d_j^2(b_i^h) \geq 0$ are defined where $S_j^2(b_i^h) \geq S_j^1(b_i^h)$, with $j = 1, 2, ..., m$ and $i = 1, 2, ..., L_h$.

Figure 1 depicts the representation of PROAFTN's intervals. To apply PROAFTN, the pessimistic interval $[S_{jh}^1, S_{jh}^2]$ and the optimistic interval $[q_{jh}^1, q_{jh}^2]$ for each attribute in each class need to be determined [21], where:

$$q_{jh}^1 = S_{jh}^1 - d_{jh}^1 \tag{1a}$$

$$q_{jh}^2 = S_{jh}^2 + d_{jh}^2 \tag{1b}$$

applied to:

$$q_{jh}^1 \leq S_{jh}^1 \tag{2a}$$

$$q_{jh}^2 \geq S_{jh}^2 \tag{2b}$$

Hence, $S_{jh}^1 = S_j^1(b_i^h)$, $S_{jh}^2 = S_j^2(b_i^h)$, $q_{jh}^1 = q_j^1(b_i^h)$, $q_{jh}^2 = q_j^2(b_i^h)$, $d_{jh}^1 = d_j^1(b_i^h)$, and $d_{jh}^2 = d_j^2(b_i^h)$. The following subsections explain the stages required to classify the object a to the class C^h using PROAFTN.

2) Computing the fuzzy indifference relation $I(a, b_i^h)$: The initial stage of the classification procedure is performed by calculating the fuzzy indifference relation $I(a, b_i^h)$. The fuzzy indifference relation is based on the concordance and non-discordance principle [28], [21], which is identified by:

$$I(a, b_i^h) = \left(\sum_{j=1}^{m} w_{jh} C_j(a, b_i^h) \right) \prod_{j=1}^{m} \left(1 - D_j(a, b_i^h)^{w_{jh}} \right) \tag{3}$$

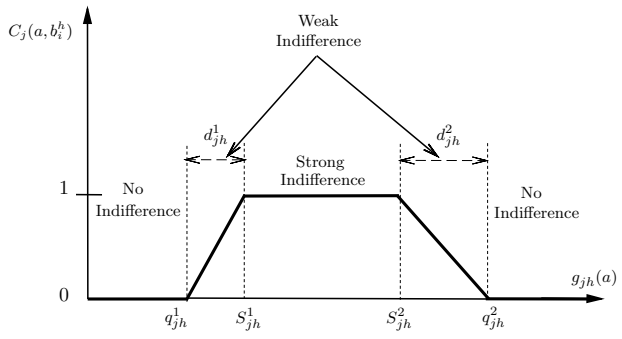

Fig. 1. Graphical representation of the partial indifference concordance index between the object a and the prototype b_i^h represented by intervals.

where w_{jh} is the weight that measures the relative importance of a relevant attribute g_j of a specific class C^h:

$$w_{jh} \in [0,1] \text{ and } \sum_{j=1}^{m} w_{jh} = 1$$

$C_j(a,b_i^h)$, $j = 1,2,...,m$, is the degree that measures the closeness of the object a to the prototype b_i^h according to the attribute g_j. The calculation of $C_j(a,b_i^h)$ is given by:

$$C_j(a,b_i^h) = \min\{C_j^1(a,b_i^h), C_j^2(a,b_i^h)\}, \quad (4)$$

where

$$C_j^1(a,b_i^h) = \frac{d_j^1(b_i^h) - \min\{S_j^1(b_i^h) - g_j(a), d_j^1(b_i^h)\}}{d_j^1(b_i^h) - \min\{S_j^1(b_i^h) - g_j(a), 0\}}$$

and

$$C_j^2(a,b_i^h) = \frac{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_i^h), d_j^2(b_i^h)\}}{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_i^h), 0\}}$$

while $D_j(a,b_i^h)$ is the degree that measures how far the object a is from the prototype b_i^h according to the attribute g_j. Two veto thresholds, $v_j^1(b_i^h)$ and $v_j^2(b_i^h)$, are used to define these values, where object a is considered perfectly different from the prototype b_i^h based on the value of attribute g_j. Generally, the determination of veto thresholds through inductive learning is risky. These values need to be obtained by an expert familiar with the problem. For more details about veto thresholds see [8]. However, in cases where these values cannot be attained from such expert, the effect of the veto thresholds is eliminated by setting them to infinity. Therefore, only the concordance principle is used, and Eq. (3) is summarized by:

$$I(a,b_i^h) = \sum_{j=1}^{m} w_{jh} C_j(a,b_i^h) \quad (5)$$

To sum up, three comparative procedures between object a and prototype b_i^h according to attribute value g_j are performed (Fig. 1):

- case 1 (strong indifference):
  $C_j(a,b_i^h) = 1 \Leftrightarrow g_j(a) \in [S_{jh}^1, S_{jh}^2]$;
  (*i.e.,* $S_{jh}^1 \leq g_j(a) \leq S_{jh}^2$)
- case 2 (no indifference):
  $C_j(a,b_i^h) = 0 \Leftrightarrow g_j(a) \leq q_{jh}^1$, or $g_j(a) \geq q_{jh}^2$
- case 3 (weak indifference):
  The value of $C_j(a,b_i^h) \in (0,1)$ is calculated based on Eq. (4). (*i.e.,* $g_j(a) \in [q_{jh}^1, S_{jh}^1]$ or $g_j(a) \in [S_{jh}^2, q_{jh}^2]$)

3) Evaluation of the membership degree $\delta(a,C^h)$: The membership degree between object a and class C^h is calculated based on the indifference degree between a and its nearest neighbor in B^h. The following formula identifies the nearest neighbor:

$$\delta(a,C^h) = \max\{I(a,b_1^h), I(a,b_2^h), ..., I(a,b_{L_h}^h)\} \quad (6)$$

4) Assignment of an object to the class: The last step is to assign object a to the right class C^h. The evaluation required to find the right class is performed by applying the following decision rule:

$$a \in C^h \Leftrightarrow \delta(a,C^h) = \max\{\delta(a,C^i)/i \in \{1,...,k\}\} \quad (7)$$

B. Differential Evolution (DE) Algorithm

DE is a population-based evolutionary algorithm. The evolution strategy of DE is similar to some extent to other widely used evolutionary algorithms. For instance, DE uses operators such as crossover, mutation and selection that are used by "Genetic algorithms" (GAs). The major difference between the GAs and DE is the way they seek to improve the solutions. GAs, for instance, use crossover to exploit the search space, while DE uses mutation operation as a search mechanism and selection operation to direct the search to the most promising regions of the search space. The DE algorithm also uses a crossover operation that takes child vector parameters from one parent more often than it does from others based on fitness. This recombination operator efficiently updates information, enabling the exploration of more promising regions in the solution space [29], [13]. The general procedure of the DE algorithm is presented in Algorithm 1.

Algorithm 1 Differential Evolution Steps

Initialization
Evolution
repeat
  Mutation
  Recombination
  Evaluation
  Selection
until (termination criteria are met)

DE's working mechanism can be described as follows. At the beginning, a population of N_{pop} solution vectors $\mathbf{x}$, each consisting of D parameters, is randomly initialized. Then, this population is iteratively improved by applying mutation, crossover and selection operators. The DE algorithm terminates when stopping criteria are met.

Assume the optimal solution $\mathbf{x}^*$ is sought. Here, $\mathbf{x}^*$ is represented by a vector of D parameters where its components are denoted by x_τ^*, where $\tau = 1,2,...,D$. When starting the optimization, each element τ of individual i in the population (*i.e.,* $i = 1,...,N_{pop}$) is randomly initialized within the boundary constraints as follows:

$$x_{i\tau} = l_\tau + (u_\tau - l_\tau) rand_\tau,$$

where l_τ and u_τ are lower and upper bounds for each element. $rand_\tau$ is a random number generator between 0 and 1.

Following the initialization phase, the objective function value (a.k.a. fitness) of each individual is calculated and the evolutionary process is started. Typically, the DE algorithm makes N_{gen} iterations, where N_{gen} is used as one of potentially many stopping conditions. At each iteration and for each individual $\mathbf{x}_i$, the following three steps are performed:

1) Random selection of three individuals: The selected individuals are identified by indices r_1, r_2, and r_3, (where $r_1, r_2, r_3 \in [1,...,N_{pop}]$) must be different from each other, and also different from the current individual i (*i.e.,* $r_1 \neq r_2 \neq r_3 \neq i$). The first of the selected individuals is defined as the target vector $\mathbf{x}_{r_1}$ and the other two individuals are $\mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$.

2) Creation of the trial vector $\mathbf{v}_i$: The trial individual vector is generated based on the mutation and crossover operations:
   a) Mutation: the mutant parameter is generated by
   $$\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \quad (8)$$
   where F is the scaling factor used to control the differential variation (a.k.a. mutation factor). Generally, $F \in [0,2]$ is a real constant factor proportional to the diversity of a population [13].
   b) Crossover (recombination): The elements from the parent vector, $\mathbf{x}_i$, are combined with those from the trial vector $\mathbf{v}_i$ to produce the offspring, $x^n_{i\tau}$, where n stands for the new generated individual x:
   $$x^n_{i\tau} = \begin{cases} v_{i\tau}, & \text{if } (rand_\tau < \kappa) \text{ or } (\rho = \tau) \\ x_{i\tau}, & \text{otherwise.} \end{cases}$$
   $$\tau = 1,...,D \; ; \quad i = 1,...,N_{pop}$$
   and $rand_\tau$ is a random number generator and $rand_\tau \in [0,1]$. κ is the crossover parameter set by the user, and $\kappa \in [0,1]$. ρ is a randomly selected index, where $\rho \in [1,2,...,D]$. The condition $\rho = \tau$ enables at least one of the parameters of the offspring to be different from the parent.
3) Selection of the best individual: The last stage is to select the best individual, *i.e.*, the one that provides the best fitness value. In this case, the best between the new generated individual $\mathbf{x}^n_i$ and the current individual $\mathbf{x}_i$ is selected. Also, the new member of the population is compared to the best individual found so far from N_{pop}. If it is better, the best individual's index is updated. The selection procedure is described by:
   $$\text{if } (f(\mathbf{x}^n_i) \geq f(\mathbf{x}_i)) \text{ then}$$
   $$\mathbf{x}_i \leftarrow \mathbf{x}^n_i, \; f_i \leftarrow f(\mathbf{x}_i)$$
   $$\text{and if } (f(\mathbf{x}_i) \geq f(\mathbf{x}^*)) \; \mathbf{x}^* \leftarrow \mathbf{x}_i$$

C. Reduced Variable Neighborhood Search (RVNS)

RVNS is a variation of the metaheuristic Variable Neighborhood Search (VNS) [17], [18]. The basic idea of the VNS algorithm is to search solution space with a systematic change of neighborhood. In RVNS, two procedures are used: shake and move. Starting from the initial solution (the position of prematurely converged individuals) $\mathbf{x}$, the algorithm selects a random solutions $\mathbf{x}'$ from first neighborhood. If the generated $\mathbf{x}'$ is better than $\mathbf{x}$, it replaces $\mathbf{x}$ and the algorithm starts all over again with the same neighborhood. Otherwise, the algorithm continues with the next neighborhood structure. The pseudocode of RVNS is given in Algorithm 2.

Algorithm 2 RVNS-PROCEDURE

Require:
- Define neighborhood structures N_k for $k = 1,2,...,k_{max}$, that will be used in the search
- Get the initial solution $\mathbf{x}$
- Choose stopping condition

repeat
  $k \leftarrow 1$
  while $k < k_{kmax}$ **do**
    Shaking:
      Generate a point $\mathbf{x}'$ at random from the kth neighborhood of $\mathbf{x}$ ($\mathbf{x}' \in N_k(\mathbf{x})$)
    Move or not:
    if $\mathbf{x}'$ is better than the incumbent $\mathbf{x}$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}'$
      $k \leftarrow 1$
    else
      set $k \leftarrow k+1$
    end if
  end while
until stopping condition is met

III. Problem Formulation

In most cases DE may not be able to explore the search space thoroughly. To improve the exploration process, RVNS is used to force individuals to jump to another solution and continue the search using the past experience. In this work, DE is employed in exploring the search space to find good neighborhoods, whereas the exploitation is achieved by RVNS, which is regarded as a local search method for intensive search of the neighborhoods of the best solution generated by DE in each iteration.

As discussed earlier, to apply PROAFTN, the intervals $[S^1_{jh}, S^2_{jh}]$ and $[q^1_{jh}, q^2_{jh}]$ satisfy the constraints in Eq. (2) and the weights w_{jh} are required to be obtained for each attribute g_j belonging to each class C^h. In this study, the induction of weights is based on the calculation of entropy and information gain [27]. An entropy measure of a set of objects is calculated as follows:

$$Entropy = -\sum_{i=1}^{C} (P_i) \; log_2 \; (P_i) \quad (9)$$

where P_i is the proportion of instances in the dataset that take the i_{th} value of the target attribute, and C represents the number of classes.

To simplify the constraints in Eq. (2), the variable substitution based on Eq. (1) is used. As a result, parameters d^1_{jh} and d^2_{jh} are used instead of q^1_{jh} and q^2_{jh}, respectively. Therefore, the optimization problem, which is based on maximizing classification accuracy to provide the optimal parameters, is defined here,

$$\begin{aligned} P: \text{Maximize} \quad & f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh}, n) \\ \text{Subject to:} \quad & S^1_{jh} \leq S^2_{jh}; d^1_{jh}, d^2_{jh} \geq 0 \\ & \sum_{j=1}^{m} w_{jh} = 1 \\ & 0 \leq w_{jh} \leq 1 \end{aligned} \quad (10)$$

where f depends on the classification accuracy and n represents the set of training objects/samples to be assigned to different classes. The procedure for calculating the fitness function $f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh}, n)$ is described in Table I.

TABLE I
PROCEDURE TO CALCULATE OBJECTIVE FUNCTION f

	For all $a \in n$:
Step 1:	- Obtain the weights w_{jh} (Eq. 9)
	- Compute the fuzzy indifference relation $I(a,b_i^h)$ (Eq. (5))
	- Evaluate the membership degree $\delta(a,C^h)$ (Eq. (6))
	- Assign the object to the class (Eq. (7))
Step 2:	- Compare the value of the new class with the true class C^h
	- Calculate the classification accuracy (*i.e.* the fitness value): $f = \frac{\text{number of correctly classified objects}}{n}$

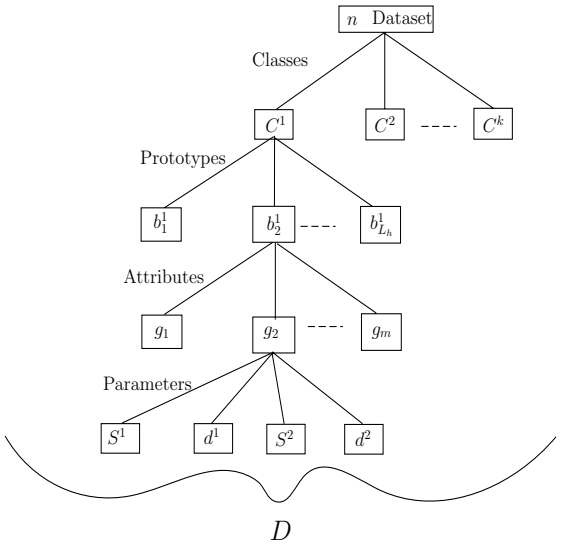

Fig. 2. DEPRO-RVNS Dimension.

In this study, DE and RVNS are utilized to solve the optimization problem described in Eq. (10). The problem dimension D (*i.e.*, the number of search parameters in the optimization problem) is proportional to the number of classes k, prototypes L_h and attributes m in the problem. Figure 2 describes graphically the problem's dimensionality. Because of this hierarchal structure, the creation of the trial vector is different from the typical one used in DE described in Section II-B. The mutation and crossover steps to update the elements of the trial individual $\mathbf{v}_i$ are performed as follows:

$$v_{i\tau jbh} = \begin{cases} x_{r_1\tau jbh} + F(x_{r_2\tau jbh} - x_{r_3\tau jbh}), & \text{if } (rand_\tau < \kappa \text{ or } \rho = \tau) \\ x_{ijlh\tau}, & \text{otherwise.} \end{cases} \quad (11)$$

$$i, r_1, r_2, r_3 \in [1,...,N_{pop}], \quad i \neq r_1 \neq r_2 \neq r_3;$$

$$h = 1,...,k; \quad b = 1,...,L_h; \quad j = 1,...,m; \quad \tau = 1,...,D$$

where, as described in Section II-B, F is the mutation factor $\in [0,2]$ and κ is the crossover factor. This modified operation (*i.e.*, Eq. (11)) forces the mutation and crossover process to be applied on each element τ selected randomly for each set of 4 parameters $S_{jh}^1, S_{jh}^2, d_{jh}^1$ and d_{jh}^2 in $\mathbf{v}_i$ for all $j = 1,2,...,m$, $b = 1,2,...,L_h$ and $h = 1,2,...,k$. Algorithm 3 illustrates the required steps to create the trial individual $\mathbf{v}_i$.

Algorithm 3 Creation of the trial individual $\mathbf{v}_i$

Require:
Number of classes k, number of prototypes L_h and number of attributes m
Problem dimension: $D = dim(\mathbf{x})$
Control parameters: F, κ
Boundary constraints for D parameters: $\mathbf{l}, \mathbf{u}$
The values: r_1, r_2, r_3
Update the contents of $\mathbf{v}_i$:
for $h = 1$ to k **do**
  for $b = 1$ to L_h **do**
    for $j = 1$ to m **do**
      Select randomly one parameter $\tau \in D$
      Update $v_{i\tau jbh}$ elements (Eq. (11))
      Repair parameters violating constraints:
      if $(v_{i\tau jbh} \notin [l_{i\tau jbh}, u_{i\tau jbh}])$ **then**
        $v_{i\tau jbh} \leftarrow rand[l_{i\tau jbh}, u_{i\tau jbh}]$
      end if
    end for
  end for
end for
return $\mathbf{v}_i$

Using RVNS as a local search algorithm, the following equations are considered to update the boundary for the previous solution $\mathbf{x}$ (*i.e.*, the solution provided by DE). In each iteration inside RVNS, the boundaries for each parameter $(S_{jh}^1, S_{jh}^2, d_{jh}^1, d_{jh}^2)$ are updated. The lower boundary is identified as:

$$l_{\tau jh} = x_{\tau jh} - (k/k_{max})x_{\tau jh} \quad (12)$$

where $l_{\tau jh}$ represents the lower boundary for each element $\tau \in D$. k/k_{max} is the dimension or range of the search space. $x_{\tau jh}$ is the initial or previous solution provided by DE, which contains the parameters $(S_{jh}^1, S_{jh}^2, d_{jh}^1, d_{jh}^2)$ before update.

$$u_{\tau jh} = x_{\tau jh} + (k/k_{max})x_{\tau jh} \quad (13)$$

where $u_{\tau jh}$ are the elements of $\mathbf{u}$ which represents the upper boundary. The shaking phase to generate randomly $\mathbf{x}'$ is given by:

$$x'_{\tau jh} = l_{\tau jh} + (u_{\tau jh} - l_{\tau jh}).rand[0,1) \quad (14)$$

The steps which explain the utilization of RVNS to learn PROAFTN are illustrated in Algorithm 4.

The complete procedure, which illustrates the flow of the classification procedure of the proposed DEPRO-RVNS is presented in Algorithm 5. After the initialization of all individuals $\mathbf{x}$, the optimization is then implemented iteratively. At each iteration, a new fitness value (classification accuracy) for each individual according to Eq. (10) is calculated. An individual is replaced by its corresponding trial individual if the latter has better fitness. Furthermore, to enhance the search strategy and to get a better solution, the best solution found so far in each iteration by DE is submitted to RVNS for further exploration and exploitation. After completing the

Algorithm 4 The RVNS based local search for PROAFTN

Require:
Get DE premature-solution as initial solution $\mathbf{x}$ which contains $S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}$
Calculate the objective function $f(\mathbf{x})$ of the optimization problem in Eq. (10)
stopping condition k is set to 4
repeat
  $k \leftarrow 1$
  Shaking:
  while $k < k_{kmax}$ **do**
    for each parameter $(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}) \in \mathbf{x}$ **do**
      - Update the boundary for each parameter $\tau \in D$ according to (Eq. (12)) and Eq. (13)
      - Generate randomly a new position $\mathbf{x}'$ from kth neighborhood for τth parameter $\in N_k(\tau)$ Eq. (14)
    end for
    Submit $\mathbf{x}'$ to calculate the new fitness value (f') according to Eq. (10))
    Move or not:
    if $f'(x')$ is better than the incumbent $f(\mathbf{x})$ **then**
      $\mathbf{x} \leftarrow \mathbf{x}'$
      $k \leftarrow 1$
    else
      set $k \leftarrow k+1$
    end if
  end while
until stopping condition is met
return the best generated point $\mathbf{x}'$ to DE to continue the search

Algorithm 5 DEPRO-RVNS Algorithm

Require:
Training data NT, testing data NS
Number of classes k, number of attributes m
Problem dimension: $D = dim(\mathbf{x})$
Control parameters: N_{pop}, F, κ
Stopping condition N_{gen}; number of generation or iteration
Boundary constraints for D parameters: $\mathbf{l}, \mathbf{u}$
Initialization:
Initialize population $N_{pop} \leftarrow rand \in [\mathbf{l}, \mathbf{u}]$
Evaluate the objective function $f(\mathbf{x})$ (Eq. (10))
Start the optimization:
for $gen = 1$ to N_{gen} **do**
  for $i = 1$ to N_{pop} **do**
    - Choose randomly $r_1, r_2, r_3 \in [1, ..., N_{pop}]$ where: $r_1 \neq r_2 \neq r_3 \neq i$
    - Create the trial individual $\mathbf{v}_i$ (Algorithm 3)
    - Evaluate the objective function $f(\mathbf{v}_i)$ (Eq. (10))
    - Select the best solution $\mathbf{x}$
    Apply local search (RVNS:)
    Find better solution by using RVNS:
    (*i.e.,* $\mathbf{x}'$ = LocalSearch($\mathbf{x}$)) (Algorithm 4)
    if $(f'(\mathbf{x}') > f(\mathbf{x}))$ **then**
      $\mathbf{x} = \mathbf{x}'$
    end if
  end for
end for
Classification of unknown data:
Submit the best solution $\mathbf{x}^*$ and testing data (NS) to PROAFTN for evaluation

optimization stage, the optimal parameters $\mathbf{x}^*$ and the testing data are submitted to PROAFTN to perform classification. The classification procedure based on testing data is carried out using equations (5) to (7).

IV. Application and Analysis of DEPRO-RVNS

The proposed DEPRO-RVNS algorithm (Algorithm 5) is implemented in Java and applied to 12 popular datasets: Breast Cancer Wisconsin Original (Bcancer); 9 attributes out of 10 are used, Transfusion Service Center (Blood), Heart Disease (Heart), Hepatitis, Haberman's Survival (HM), Iris, Liver Disorders (Liver), Mammographic Mass (MM), Pima Indians Diabetes (Pima), Statlog Australian Credit Approval (STAust), Teaching Assistant Evaluation (TA), and Wine. The datasets are in the public domain and are available at the University of California at Irvine (UCI) Machine Learning Repository database [30]. The details of the dataset description and dimensionally are presented in Table II. The dimensionality $D = dim(\mathbf{x})$ describes the number of elements of each individual required by DEPRO-RVNS. Considering two prototypes for each class, and four parameters for each attribute S^1, S^2, d^1, d^2 are needed, the number of components of D for each problem is $2 \times 4 \times k \times m$, where k and m are the number of classes and attributes, respectively.

TABLE II
Dataset Description

	Dataset	Instances	Attributes	Classes	$D = dim(\mathbf{x})$
1	BCancer	699	9	2	144
2	Blood	748	4	2	64
3	Heart	270	13	2	208
4	Hepatitis	155	19	2	304
5	HM	306	3	2	48
6	Iris	150	4	3	96
7	Liver	345	6	2	96
8	MM	961	5	2	80
9	Pima	768	8	2	128
10	STAust	690	14	2	224
11	TA	151	5	3	120
12	Wine	178	13	3	312

A. Parameters Settings

To apply DEPRO-RVNS, the following factors are considered before applying the optimization:

- The bounds for S^1_{jh} and S^2_{jh} vary between $\mu_{jh} - 6\sigma_{jh}$ and $\mu_{jh} + 6\sigma_{jh}$, where μ_{jh} and σ_{jh} represent mean and standard deviation for each attribute in each class, respectively;
- The bounds for d^1_{jh} and d^2_{jh} vary in the range $[0, 6\sigma_{jh}]$.

These regions are defined before starting the training stage. During the optimization phase, the parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}$ evolve within the aforementioned boundary constraints. The parameters are selected so that the classification accuracy is maximized.

Also, the following technical factors are considered for implementing DEPRO-RVNS:

- Setting the best values for κ and F varies from one application to another. However, based on the overall experimental results, it was noticed that setting $\kappa = 0.90$ and $F = 0.50$ was a suitable choice for all datasets.
- The size of the population is fixed at $N_{pop} = 80$;
- The maximum iteration number is fixed at $N_{gen} = 500$.

B. Results and Analysis

The experimentation work is performed in two stages. First, to test the performance and the robustness of DEPRO-RVNS, 10 independent runs are executed over each dataset. Second, to compare the performance of DEPRO-RVNS against other well-known machine learning classifiers, similar experimental work is performed on the same dataset using *Weka* (the open source platform described in [27]).

Since the proposed learning approach is based on meta-heuristics, further investigation of the robustness of the method is examined. In this context, Table III lists the results for the performance of DEPRO-RVNS on both training and testing sets as applied to all datasets in each independent run. By observing the standard deviation values, it is possible to see that DEPRO-RVNS provides good and stable results on all datasets. The classification accuracy on the training and testing datasets is stable since they are situated around their average (small standard deviation for both classifiers and databases). Columns 2, 3 and 4 represent best, mean, and standard deviation of the solutions obtained in 10 independent executions of DEPRO-RVNS on the training data, whereas columns 5 to 7 represent the same quantities but as the algorithm is applied to the testing data. By observing these results one can see that, in all 10 independent runs, the classification accuracy obtained by each dataset did not deviate by more than 0.94 % on the training dataset and 1.79 % on the testing dataset. Furthermore, DEPRO-RVNS was able to provide excellent results in terms of classification accuracy on both training and testing.

To evaluate the performance of DEPRO-RVNS versus other classifiers, further experimental work is conducted with six machine learning techniques. These algorithms are chosen from different machine learning theories; they are: 1) Tree induction C4.5 (J48) [22], 2) statistical modelling, Naive Bayes (NB) [23], 3) Support Vector machines (SVM), SMO [31], 4) Neural Network (NN), multilayer perceptron (MLP) [32], 5) instance-based learning, IBk with k=3 [33], and 6) rule learning, PART [34]. The open source platform *Weka* [27] is used with its default settings to run these algorithms. Furthermore, to evaluate the advantages of using RVNS, similar experimental work to that presented in Table III was performed using DE and PROAFTN (*i.e.*, DEPRO alone without using RVNS).

Table IV documents the results of the classification accuracy obtained by DEPRO, DEPRO-RVNS and other algorithms based on testing dataset. The best results achieved on each application are marked in bold. The average results of 10 runs for DEPRO-RVNS presented in Table III are considered in this comparisons. The same procedure adopted for DEPRO-RVNS is also applied for DEPRO. It is noticed that DEPRO-RVNS gives better results than using DEPRO alone. DEPRO-RVNS gives better results on 10 out of 12 datasets, which means the utilization of RVNS with DE fairly consistently improved the performance of PROAFTN method. Furthermore, it was noticed that when using RVNS inside DE, the best solution could be reached with a smaller number of iterations. For instance, some datasets such as Iris, breast cancer (Bcancer) or Wine, *etc.* the best solution could be obtained in 20 iterations only. Based on these conclusions, DEPRO-RVNS is adopted in the next phase of this study to be compared with other classifiers.

TABLE III
STATISTICAL EVALUATIONS OF DEPRO-RVNS (IN %) IN TERMS OF THE BEST SOLUTION FOUND, MEAN, AND STANDARD DEVIATION OF 10 INDEPENDENT RUNS

Dataset	Training Data				Testing Data			
	Best	Min	Mean	Std Dev	Best	Min	Mean	Std Dev
Bcancer	98.06	98.01	98.05	0.03	97.71	96.00	97.05	0.30
Blood	82.13	81.71	81.94	0.13	81.52	78.48	79.61	0.95
Heart	89.22	88.31	88.67	0.29	85.56	82.00	83.81	1.29
Hepatitis	92.90	92.54	92.70	0.12	86.67	83.63	85.37	1.14
HM	80.79	79.01	79.58	0.70	77.18	73.61	76.10	1.09
Iris	99.85	99.04	99.36	0.33	97.23	96.00	96.66	0.34
Liver	80.07	77.59	78.70	0.94	72.94	68.22	70.99	1.79
MM	85.14	84.18	84.60	0.43	84.84	83.00	84.77	0.64
Pima	81.60	79.45	80.20	0.87	80.52	75.32	77.23	1.44
STAust	89.55	88.53	89.03	0.36	86.96	85.36	86.04	0.48
TA	68.43	66.00	66.98	0.73	63.09	59.90	62.72	1.67
Wine	100.00	99.93	99.99	0.03	97.59	95.50	97.10	0.72

Based on Demšar's recommendation [35], the Friedman test is used in this work to evaluate whether there is a difference in the performance between DEPRO-RVNS and other classifiers. Provided that the Friedman test indicates statistically significant difference on 12 datasets and the seven classifiers including DEPRO-RVNS, other advanced tests such as Bonferroni-Dunn's, Hochberg's, Hommel's, and Nemenyi's procedures described in [35] and in [36] are used to determine which classifier(s) performs best. More particularly, these tests are used to test whether the difference between DEPRO-RVNS versus other classifiers is meaningful. Based on the classification accuracy results obtained by each classifier presented in Table IV, the algorithms ranking results using Friedman test are shown in Table V.

Friedman's and Iman-Davenport's statistics are respectively:

- Chi-square χ^2 = 25.8304 with 6 degrees of freedom,
- F-distribution F = 6.1541 with $k-1$ and (k-1)(N-1), that is 6 and 66 degrees of freedom.

TABLE IV
EXPERIMENTAL RESULTS BASED ON CLASSIFICATION ACCURACY (IN %) TO MEASURE THE PERFORMANCE OF THE DIFFERENT CLASSIFIER COMPARED WITH DEPRO-RVNS

	Algorithm	*C4.5*	*NB*	*SVM*	*NN*	*k-nn*	*PART*	*DEPRO*	*DEPRO-*
	Dataset	*J48*		*SMO*	*MLP*	*Ibk, k=3*			RVNS
1	BCancer	94.56	95.99	96.70	95.56	97.00	94.28	96.97	**97.05**
2	Blood	77.81	75.40	76.20	78.74	74.60	78.07	79.59	**79.61**
3	Heart	76.60	83.70	**84.10**	78.10	78.89	73.33	83.74	83.81
4	Hepatitis	80.00	**85.81**	83.87	81.94	84.52	82.58	84.17	85.37
5	HM	71.90	74.83	73.52	72.87	70.26	72.55	**80.36**	76.10
6	Iris	96.00	96.00	96.00	**97.33**	95.33	94.00	96.47	96.66
7	Liver	68.7	56.52	58.26	**71.59**	61.74	63.77	71.01	70.99
8	MM	82.10	78.35	79.24	82.10	77.21	82.21	84.33	**84.77**
9	Pima	71.48	75.78	77.08	75.39	73.44	73.05	75.37	**77.23**
10	STAust	85.22	77.25	85.51	84.93	83.62	83.62	85.62	**86.04**
11	TA	59.6	52.98	54.3	54.3	50.33	58.28	61.80	**62.72**
12	Wine	91.55	97.4	**99.35**	97.4	95.45	92.86	96.87	97.10

TABLE V
AVERAGE RANKINGS OF THE ALGORITHMS

Algorithm	Ranking
RVNS-DEPRO	1.5833
SVM	3.375
NN	3.5417
NB	4.2917
C4.5	4.875
PART	5.0417
3-NN	5.2917

k is the number of classifiers and N is the number of datasets. The critical value of $F(6,66)$ for $\alpha = 0.05$ is 2.24; this indicates that the performance of the algorithms is significantly different. Table VI summarizes the hypothesis ordered by their p-value and the adjustment of α's by Bonferroni-Dunn's, Hochberg's, Hommel's, and Nemenyi's statistical procedures [35], [36]. The difference between DEPRO-RVNS notified by R_0 and other classifiers R_i. The standard error between two classifiers is $SE = \sqrt{\frac{k(k+1)}{6N}}$. The p-values are documented in the last column. p-values identify the probability of difference in performance among the classifiers over the datasets. According to this analysis, Nemenyi's procedure rejects those hypotheses that have a p-value ≤ 0.0023. Bonferroni-Dunn's procedure rejects those hypotheses that have a p-value ≤ 0.0083. Hochberg's procedure rejects those hypotheses that have a p-value ≤ 0.05. Hommel's procedure rejects all hypotheses. Based on these outcomes, the pairwise comparisons between DEPRO-RVNS and other classifiers are drawn as follows:

TABLE VI
PSOPRO VERSUS OTHER CLASSIFIERS FOR $\alpha = 0.05$

i	algorithms	$z = (R_0 - R_i)/SE$	p
1	SVM	2.0316	0.0423
2	NN	2.2205	0.0264
3	NB	3.0709	0.0021
4	C4.5	3.7324	1.8966E-4
5	PART	3.9214	8.8043E-5
6	3-NN	4.2049	2.6125E-5

1) DEPRO-RVNS performs strongly better than 3-NN, PART, C4.5, and NB.
2) DEPRO-RVNS performs better than NN and SVM.

Regarding the execution time of DEPRO-RVNS, it was noticed that, as expected, the execution time is dependent mainly on the problem size (*i.e*, the number of training datasets and the number of attributes) and the number of PROAFTN parameters (D) involved in the training process. Even though, the number of iterations is set to 500 in this study, DEPRO-RVNS was able to get the presented results in much less iterations number with a very good speed. It is noticed that the execution time of DEPRO-RVNS was compared favorably with the execution time of NN most of the cases. The remaining algorithms 3-NN, C4.5, NB, PART and SVM, respectively were relatively faster than DEPRO-RVNS.

V. CONCLUSIONS

In this paper, a new methodology based on the metaheuristic algorithms DE and RVNS is proposed for training the MCDA classification method PROAFTN. The proposed technique for solving classification problems is named DEPRO-RVNS. During the learning stage, DE and RVNS are utilized to induce the classification model for PROAFTN by inferring the best parameters from data with high classification accuracy.

The performance of DEPRO-RVNS applied to 12 classification dataset demonstrates that DEPRO-RVNS outperforms the well-known classification methods PART, 3-nn, C4.5, NB, SVM, and NN. PROAFTN requires some parameters and uses fuzzy approach to assign objects to classes. As a result there is richer information, more flexibility, and therefore an improved chance of assigning objects to the preferred classes. In this study, using the metaheuristics DE and RVNS to obtain these parameters proved to be a successful approach for training PROAFTN and thus greatly improving its performance.

In conclusion, it was observed that DEPRO-RVNS significantly improved the performance of the PROAFTN method. Hence, the utilization of hybrid algorithm seems to be a promising and efficient approach for learning other classification methods from different paradigms. Likewise, PROAFTN is a relatively new classification method that merits further investigation in the area of data mining and knowledge discovery.

ACKNOWLEDGMENT

We gratefully acknowledge the support from NSERC's Discovery Award (RGPIN293261-05) granted to Dr. Nabil Belacel.

REFERENCES

[1] D. Dutton and G. Conroy, "A review of machine learning," *The Knowledge Engineering Review*, vol. 12:4, pp. 341–367, 1996.
[2] D. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, 2005.
[3] E. Alpaydin, "Introduction to machine learning (adaptive computation and machine learning)," *MIT Press*, 2004.

[4] B. Roy, "Multicriteria methodology for decision aiding," *Kluwer Academic*, 1996.
[5] N. E. Fenton and W. Wang, "Risk and confidence analysis for fuzzy multicriteria decision making," *Knowledge-Based Systems*, vol. 19, no. 6, pp. 430–437, 2006.
[6] C. Zopounidis and M. Doumpos, "Multicriteria classification and sorting methods: A literature review," *European Journal of Operational Research*, vol. 138, no. 2, pp. 229–246, 2002.
[7] K. Jabeur and A. Guitouni, "A generalized framework for concordance/discordance-based multi-criteria classification methods," in *Information Fusion, 2007 10th International Conference on*, July 2007, pp. 1–8.
[8] N. Belacel, "Multicriteria assignment method PROAFTN: methodology and medical application," *European Journal of Operational Research*, vol. 125, no. 1, pp. 175–183, 2000.
[9] N. Belacel and M. Boulassel, "Multicriteria fuzzy assignment method: A useful tool to assist medical diagnosis," *Artificial Intelligence in Medicine*, vol. 21, no. 1-3, pp. 201–207, 2001.
[10] N. Belacel, P. Vincke, M. Scheiff, and M. Boulassel, "Acute leukemia diagnosis aid using multicriteria fuzzy assignment methodology," *Computer Methods and Programs in Biomedicine*, vol. 64, no. 2, pp. 145–151, 2001.
[11] N. Belacel, Q. Wang, and R. Richard, "Web-integration of PROAFTN methodology for acute leukemia diagnosis," *Telemedicine Journal and e-Health*, vol. 11, no. 6, pp. 652–659, 2005.
[12] F. Al-Obeidat, N. Belacel, P. Mahanti, and J. A. Carretero, "Discretization techniques and genetic algorithm for learning the classification method proaftn," in *Eighth International Conference On Machine Learning and Applications*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 685–688.
[13] R. Storn and K. Price, "Differential evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, December 1997.
[14] S. Paterlini and T. Krink, "Differential evolution and particle swarm optimisation in partitional clustering," *Comput. Stat. Data Anal*, vol. 50, pp. 1220–1247, 2006.
[15] B. Subudhi and D. Jena, "Differential evolution and levenberg marquardt trained neural network scheme for nonlinear system identification," *Neural Process. Lett.*, vol. 27, no. 3, pp. 285–296, 2008.
[16] M. Tayel and A. H. Yassin, "An introduced neural network-differential evolution model for small signal modeling of phemts," *International Conference on Electronic Computer Technology*, vol. 0, pp. 499–506, 2009.
[17] P. Hansen and N. Mladenovic, "Variable neighborhood search for the p-median," *Location Science*, vol. 5, pp. 207–226, 1997.
[18] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, no. 130, pp. 449–467, 2001.
[19] *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008*. IEEE, 2008.
[20] C.-Y. Tsai and C.-C. Chiu, "A vns-based hierarchical clustering method," in *CIMMACS'06: Proceedings of the 5th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2006, pp. 268–275.
[21] N. Belacel, H. Raval, and A. Punnen, "Learning multicriteria fuzzy classification method PROAFTN from data," *Computers and Operations Research*, vol. 34, no. 7, pp. 1885–1898, 2007.
[22] J. R. Quinlan, "Improved use of continuous attributes in c4.5," *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, 1996.
[23] G. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.
[24] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 1–47, 1998.
[25] G. Castellano, A. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 519–531, 1997.
[26] B. Twala, "Multiple classifier application to credit risk assessment," *Expert Systems with Applications*, vol. In Press, Uncorrected Proof, pp. – , 2009.
[27] H. Witten, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems, 2005.
[28] N. Belacel, "Multicriteria classification methods: Methodology and medical applications," Ph.D. dissertation, Free University of Brussels, Belgium, 1999.
[29] J. Kacprzyk, *Advances in Differential Evolution*. Springer, 2008.
[30] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.
[31] S. Pang, D. Kim, and S. Bang, "Face membership authentication using SVM classification tree generated by membership-based lle data partition," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 436–446, 2005.
[32] Y. Shirvany, M. Hayati, and R. Moradian, "Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations," *Appl. Soft Comput.*, vol. 9, no. 1, pp. 20–29, 2009.
[33] D. Aha, "Lazy learning," *Dordrecht: Kluwer Academic Publishers*, 1997.
[34] D. K. Subramanian, V. S. Ananthanarayana, and M. Narasimha Murty, "Knowledge-based association rule mining using and-or taxonomies," *Knowledge-Based Systems*, vol. 16, no. 1, pp. 37–45, 2003.
[35] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research.*, vol. 7, pp. 1–30, 2006.
[36] S. Garcia and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2009.