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 on all criteria is on the results of the 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., Naïve 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 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 $\{g_1, g_2, \ldots, g_m\}$ and let $\{c^1, c^2, \ldots, c^k\}$ be the set of $k$ classes. The different steps of the classification procedure follow.

1) Initialization: For each class $c^h$, $h = 1, 2, \ldots, k$, a set of $L_a$ prototypes $B^h = \{b^h_1, b^h_2, \ldots, b^h_{L_a}\}$ are determined. For each prototype $b^h_i$ and each attribute $g_j$, an interval $[S^1_j(b^h_i), S^2_j(b^h_i)]$ and the preference thresholds $d^1_j(b^h_i) \geq 0$ and $d^2_j(b^h_i) \geq 0$ are defined where $S^1_j(b^h_i) = S^1_j(b^h_i)$, with $j = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, L_a$.

Figure 1 depicts the representation of PROAFTN’s intervals. To apply PROAFTN, the pessimistic interval $[S^1_j(b^h_i), S^2_j(b^h_i)]$ and the optimistic interval $[q^1_j(b^h_i), q^2_j(b^h_i)]$ for each attribute in each class need to be determined [21], where:

$$q^1_j = S^1_j - d^1_j$$
$$q^2_j = S^2_j + d^2_j$$

applied to:

$$q^1_j \leq S^1_j$$
$$q^2_j \geq S^2_j$$

Hence, $S^1_{ja} = S^1_j(b^h_i)$, $S^2_{ja} = S^2_j(b^h_i)$, $q^1_{ja} = q^1_j(b^h_i)$, $q^2_{ja} = q^2_j(b^h_i)$, $d^1_{ja} = d^1_j(b^h_i)$, and $d^2_{ja} = d^2_j(b^h_i)$. 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 $I(a, b^h_i)$: The initial stage of the classification procedure is performed by calculating the fuzzy indifference relation $I(a, b^h_i)$. The fuzzy indifference relation is based on the concordance and non-discordance principle [28], [21], which is identified by:

$$I(a, b^h_i) = \left( \frac{m}{\sum_{j=1}^{m} w_{j}^h C_j(a, b^h_i)} \right) \prod_{j=1}^{m} (1 - D_j(a, b^h_i)^{w_{j}^h})$$

Fig. 1. Graphical representation of the partial indifference concordance index between the object $a$ and the prototype $b^h_i$ represented by intervals.
where $w_{jk}$ is the weight that measures the relative importance of a relevant attribute $g_j$ of a specific class $C^h$:

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

$C_j(a, b_i^h)$, $j = 1, 2, ..., m$, 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:

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

where

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

and

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

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, $v^1_j(b_i^h)$ and $v^2_j(b_i^h)$, 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:

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

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):**
  $$C_j(a, b_i^h) = 1 \Leftrightarrow g_j(a) \in [S_{jk}^1, S_{jk}^2]$$
  (i.e., $S_{jk}^1 \leq g_j(a) \leq S_{jk}^2$)

- **case 2 (no indifference):**
  $$C_j(a, b_i^h) = 0 \Leftrightarrow g_j(a) \leq q_{1jk}^1 \text{ or } g_j(a) \geq q_{2jk}^1$$

- **case 3 (weak indifference):**
  The value of $C_j(a, b_i^h) \in (0, 1)$ is calculated based on Eq. (4). (i.e., $g_j(a) \in [q_{1jk}^1, q_{1jk}^2]$ or $g_j(a) \in [S_{jk}^2, q_{2jk}^2]$)

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:

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

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:

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

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**

```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| Initialization | \[ 
| Mutation | \[ 
| Recombination | \[ 
| Evaluation | \[ 
| Selection | \[ 
| repeat | until (termination criteria are met) |
```

DE’s working mechanism can be described as follows. At the beginning, a population of $N_{pop}$ solution vectors $x$, 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 $x^*$ is sought. Here, $x^*$ is represented by a vector of $D$ parameters where its components are denoted by $x^*_i$, where $i = 1, 2, ..., D$. When starting the optimization, each element $i$ of individual $i$ in the population (i.e., $i = 1, ..., N_{pop}$) is randomly initialized within the boundary constraints as follows:

$$x_i = l_i + (u_i - l_i)rand_i$$

where $l_i$ and $u_i$ are lower and upper bounds for each element. $rand_i$ 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 $x$, 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 $x_i$, and the other two individuals are $x_{r_2}$ and $x_{r_3}$.
2) Creation of the trial vector $v_i$: The trial individual vector is generated based on the mutation and crossover operations:

a) Mutation: the mutant parameter is generated by

$$v_i = x_{i} + F (x_{i} - x_{j})$$

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, $x_i$, are combined with those from the trial vector $v_i$ to produce the offspring, $x^*_{iτ}$, where $n$ stands for the new generated individual $x$:

$$x^*_{iτ} = \begin{cases} v_{iτ}, & \text{if } (rand_{τ} < κ) \text{ or } (ρ = τ) \\ x_{iτ}, & \text{otherwise.} \end{cases}$$

and $rand_{τ}$ is a random number generator and $rand_{τ} \in [0, 1]$. $κ$ is the crossover parameter set by the user, and $κ \in [0, 1]$. $ρ$ is a randomly selected index, where $ρ \in [1, 2, ..., D]$. The condition $ρ = τ$ 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 $x^*$ and the current individual $x_i$ 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:

if $f(x^*_i) \geq f(x_i)$ then $x_i ← x^*_i$, $f_i ← f(x_i)$
and if $f(x^*_i) < f(x_i)$ then $x^* ← x_i$

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) $x$, the algorithm selects a random solutions $x'$ from first neighborhood. If the generated $x'$ is better than $x$, it replaces $x$ 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 $x$
- Choose stopping condition

**repeat**

$k ← 1$

**while** $k < k_{max}$ **do**

**Shaking:**

Generate a point $x'$ at random from the $k$th neighborhood of $x$ ($x' \in N_k(x)$)

**Move or not:**

if $x'$ is better than the incumbent $x$ then

$x ← x'$

$k ← 1$

else

set $k ← k + 1$

**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 $C^i$. 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:

$$Entropy = \sum_{i=1}^{C} P_i \log_2 (P_i)$$

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,

$$P: \text{Maximize} \quad f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh}, n)$$

**Subject to:**

$$S^1_{jh} \leq S^2_{jh}; d^1_{jh}, d^2_{jh} \geq 0$$

$$\sum_{j=1}^{m} w_{jh} = 1$$

$$0 \leq w_{jh} \leq 1$$

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.
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:

$$
\mathbf{v}_{\tau,jh} = \begin{cases} 
\mathcal{X}_1 \mathcal{J}_{\mathbf{v}_i} + F(\mathcal{X}_2 \mathcal{J}_{\mathbf{v}_i} - \mathcal{X}_3 \mathcal{J}_{\mathbf{v}_i}), & \text{if } (\text{rand}_1 < \kappa \text{ or } \rho = \tau) \\
\mathcal{X}_1 \mathcal{J}_{\mathbf{v}_i}, & \text{otherwise.}
\end{cases}
$$

(11)

where, as described in Section II-B, $F$ is the mutation factor in $[0,2]$ and $\kappa$ is the crossover factor. This modified operation (i.e., Eq. (11)) forces the mutation and crossover process to be applied on each element $\tau$ selected randomly for each set of 4 parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}$ and $d^2_{jh}$ in $\mathbf{v}_i$ for all $j = 1, 2, ..., m$.

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: $F, \kappa$

Boundary constraints for $D$ parameters: $1, u$

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 $\mathbf{v}_{\tau,jh}$ elements (Eq. (11))

Repair parameters violating constraints:

if $[\mathbf{v}_{\tau,jh} \not\in [l_{\tau,jh}, u_{\tau,jh}]]$ then

$\mathbf{v}_{\tau,jh} \leftarrow \text{rand}[l_{\tau,jh}, u_{\tau,jh}]$

end if

end for

end for

end for

return $\mathbf{v}_i$

$b = 1, 2, ..., L_h$ and $h = 1, 2, ..., k$. Algorithm 3 illustrates the required steps to create the trial individual $\mathbf{v}_i$.

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 $(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh})$ are updated. The lower boundary is identified as:

$$
l_{\tau,jh} = x_{\tau,jh} - \frac{(k/k_{\text{max}})}{x_{\tau,jh}}
$$

(12)

where $l_{\tau,jh}$ represents the lower boundary for each element $\tau \in D$. $k/k_{\text{max}}$ 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 $(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh})$ before update.

$$
u_{\tau,jh} = x_{\tau,jh} + \frac{(k/k_{\text{max}})}{x_{\tau,jh}}
$$

(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:

$$x_{\tau,jh} = l_{\tau,jh} + (u_{\tau,jh} - l_{\tau,jh}) \cdot \text{rand}(0,1)
$$

(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 \( x \) which contains \( d_{jh}^1, S_{jh}^1, d_{jh}^2, S_{jh}^2 \)
- Calculate the objective function \( f(x) \) of the optimization problem in Eq. (10)
- Stopping condition \( k \) is set to 4

repeat
  \( k \leftarrow 1 \)
  Shaking:
  while \( k < k_{\text{max}} \) do
    for each parameter \((S_{jh}^1, S_{jh}^2, d_{jh}^1, d_{jh}^2) \in x \) do
      - Update the boundary for each parameter \( \tau \in D \) according to Eq. (12) and Eq. (13)
      - Generate randomly a new position \( x \) from \( k \)th neighborhood for \( r \)th parameter \( \in N_k(\tau) \) Eq. (14)
    end for
    Submit \( x \) to calculate the new fitness value \( f(x) \) according to Eq. (10)
  end while
  Move or not:
  if \( f(\bar{x}) \) is better than the incumbent \( f(x) \) then
    \( x \leftarrow \bar{x} \)
    \( k \leftarrow 1 \)
  else
    set \( k \leftarrow k + 1 \)
  end if
until stopping condition is met
return the best generated point \( x' \) to DE to continue the search

optimization stage, the optimal parameters \( x' \) 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 (STAus), 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 \( D = \dim(x) \) describes the number of elements of each individual required by DEPRO-RVNS. Considering two prototypes for each class, and four parameters for each attribute \( S_{jh}^1, S_{jh}^2, d_{jh}^1, d_{jh}^2 \) 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.

### Algorithm 5 DEPRO-RVNS Algorithm

Require:
- Training data \( NT \), testing data \( NS \)
- Number of classes \( k \), number of attributes \( m \)
- Problem dimension: \( D = \dim(x) \)
- Control parameters: \( N_{\text{pop}}, F, \kappa \)
- Stopping condition \( N_{\text{gen}} \); number of generation or iteration
- Boundary constraints for \( D \) parameters: \( l, u \)

Initialisation:
- Initialize population \( N_{\text{pop}} \) \( \leftarrow \text{rand} \in [l, u] \)
- Evaluate the objective function \( f(x) \) (Eq. (10))

Start the optimization:
for \( gen = 1 \) to \( N_{\text{gen}} \) do
  for \( i = 1 \) to \( N_{\text{pop}} \) do
    - Choose randomly \( r_1, r_2, r_3 \in [1, ..., N_{\text{pop}}] \) where: \( r_1 \neq r_2 \neq r_3 \neq i \)
    - Create the trial individual \( v_i \) (Algorithm 3)
    - Evaluate the objective function \( f(v_i) \) (Eq. (10))
    - Select the best solution \( x \)
    - Apply local search (RVNS):\( \text{LocalSearch}(x) \) (Algorithm 4)
    if \( f'(x') > f(x) \) then
      \( x \leftarrow x' \)
    end if
  end for
end for

Classification of unknown data:
Submit the best solution \( x' \) and testing data \( (NS) \) to PROAFTN for evaluation

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

A. Parameters Settings

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

- The bounds for \( S_{jh}^1 \) and \( S_{jh}^2 \) vary between \( \mu_{jh} - 6\sigma_{jh} \) and \( \mu_{jh} + 6\sigma_{jh} \), where \( \mu_{jh} \) and \( \sigma_{jh} \) represent mean and standard deviation for each attribute in each class, respectively;
- The bounds for \( d_{jh}^1 \) and \( d_{jh}^2 \) vary in the range \([0, 6\sigma_{jh}]\).
These regions are defined before starting the training stage. During the optimization phase, the parameters $S_{j_k}^1, S_{j_k}^2, d_j, d_{j_k}$ 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 $\kappa$ 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.

Based on Demˇsar’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 $\chi^2 = 25.8304$ 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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Dataset</th>
<th>C4.5</th>
<th>NB</th>
<th>SVM</th>
<th>NN</th>
<th>k-nn</th>
<th>PART</th>
<th>DEPRO</th>
<th>DEPRO-RVNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Bcancer</td>
<td>94.56</td>
<td>95.99</td>
<td>96.70</td>
<td>95.56</td>
<td>97.00</td>
<td>94.28</td>
<td>96.97</td>
<td>97.05</td>
<td></td>
</tr>
<tr>
<td>2 Blood</td>
<td>77.81</td>
<td>75.40</td>
<td>76.20</td>
<td>78.74</td>
<td>74.60</td>
<td>76.07</td>
<td>79.59</td>
<td>79.61</td>
<td></td>
</tr>
<tr>
<td>3 Heart</td>
<td>76.60</td>
<td>83.70</td>
<td>84.10</td>
<td>78.10</td>
<td>78.89</td>
<td>73.33</td>
<td>83.74</td>
<td>83.81</td>
<td></td>
</tr>
<tr>
<td>4 Hepatitis</td>
<td>80.00</td>
<td>85.81</td>
<td>83.87</td>
<td>81.94</td>
<td>84.52</td>
<td>82.58</td>
<td>84.17</td>
<td>85.37</td>
<td></td>
</tr>
<tr>
<td>5 HM</td>
<td>71.90</td>
<td>74.83</td>
<td>73.52</td>
<td>72.87</td>
<td>70.26</td>
<td>72.55</td>
<td>80.36</td>
<td>76.10</td>
<td></td>
</tr>
<tr>
<td>6 Iris</td>
<td>96.00</td>
<td>96.00</td>
<td>96.00</td>
<td>97.33</td>
<td>95.33</td>
<td>94.00</td>
<td>96.47</td>
<td>96.66</td>
<td></td>
</tr>
<tr>
<td>7 Liver</td>
<td>68.7</td>
<td>56.52</td>
<td>58.26</td>
<td>71.59</td>
<td>61.74</td>
<td>63.77</td>
<td>71.01</td>
<td>70.99</td>
<td></td>
</tr>
<tr>
<td>8 MM</td>
<td>82.10</td>
<td>78.35</td>
<td>79.24</td>
<td>82.10</td>
<td>77.21</td>
<td>82.21</td>
<td>84.33</td>
<td>84.77</td>
<td></td>
</tr>
<tr>
<td>9 Pima</td>
<td>71.48</td>
<td>75.78</td>
<td>77.08</td>
<td>75.39</td>
<td>73.44</td>
<td>73.05</td>
<td>75.37</td>
<td>77.23</td>
<td></td>
</tr>
<tr>
<td>10 STAust</td>
<td>85.22</td>
<td>77.25</td>
<td>85.51</td>
<td>84.93</td>
<td>83.62</td>
<td>83.62</td>
<td>85.62</td>
<td>86.04</td>
<td></td>
</tr>
<tr>
<td>11 TA</td>
<td>59.56</td>
<td>52.98</td>
<td>54.34</td>
<td>54.34</td>
<td>50.33</td>
<td>58.28</td>
<td>61.80</td>
<td>62.72</td>
<td></td>
</tr>
<tr>
<td>12 Wine</td>
<td>91.55</td>
<td>97.74</td>
<td>99.35</td>
<td>97.94</td>
<td>95.45</td>
<td>92.86</td>
<td>96.87</td>
<td>97.10</td>
<td></td>
</tr>
</tbody>
</table>

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 the power test. The difference between DEPRO-RVNS and other classifiers is notified by \( p \leq 0.05 \).

TABLE V
AVERAGE RANKINGS OF THE ALGORITHMS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>1.5833</td>
</tr>
<tr>
<td>NN</td>
<td>3.375</td>
</tr>
<tr>
<td>NB</td>
<td>3.5417</td>
</tr>
<tr>
<td>C4.5</td>
<td>4.2917</td>
</tr>
<tr>
<td>PART</td>
<td>5.0417</td>
</tr>
<tr>
<td>3-NN</td>
<td>5.2917</td>
</tr>
</tbody>
</table>

TABLE VI
PSOPRO Versus Other Classifiers for \( \alpha = 0.05 \)

<table>
<thead>
<tr>
<th>i</th>
<th>algorithms</th>
<th>( z = (R_i - R_j)/SE )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SVM</td>
<td>2.0316</td>
<td>0.0423</td>
</tr>
<tr>
<td>2</td>
<td>NN</td>
<td>2.2205</td>
<td>0.0264</td>
</tr>
<tr>
<td>3</td>
<td>NB</td>
<td>3.0709</td>
<td>0.0021</td>
</tr>
<tr>
<td>4</td>
<td>C4.5</td>
<td>3.7324</td>
<td>1.8966E-4</td>
</tr>
<tr>
<td>5</td>
<td>PART</td>
<td>3.9214</td>
<td>8.8043E-5</td>
</tr>
<tr>
<td>6</td>
<td>3-NN</td>
<td>4.2049</td>
<td>2.6125E-5</td>
</tr>
</tbody>
</table>

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


