Chapter 10
Evolving Connection Weights of Artificial Neural Network Using a Multi-Objective Approach with Application to Class Prediction

Andrei Strickler and Aurora Pozo

In Artificial Neural Network (ANN), the selection of connection weights is a key issue and Genetic and Evolution Strategies have been found to be promising algorithms to solve this important task. Motivated by that, this study investigates the applicability of using two novel Multi-Objective Evolutionary Algorithms (MOEA): Speed constrained Multi-Objective Particle Swarm Optimization (SMPSO) and Multi-Objective Differential Evolution Algorithm based on Decomposition with Dynamical Resource Allocation (MOEA/D-DE-DRA). ANNs are training to learn data classification using sensibility and specificity for different UCI databases. The results are compared using the Hypervolume as quality indicator and statistical test.

10.1 Introduction

Most training algorithms, such as Backpropagation (BP) and conjugate gradient algorithms, are based on gradient descent [15]. There have been many successful applications of BP in various areas, but BP has drawbacks due to the use of gradient descent. It often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or non-differentiable.

In the other side, Evolutionary Algorithms (EAs) can help to avoid the problem of convergence to local minima and explore global search for training MLP. EAs

A. Strickler (✉) · A. Pozo
Computer Science Department, Federal University of Paraná, Curitiba, Brazil
e-mail: astrickler@inf.ufpr.br

A. Pozo
e-mail: aurora@inf.ufpr.br

© Springer International Publishing Switzerland 2017

N. Nedjah et al. (eds.), *Designing with Computational Intelligence*,
Studies in Computational Intelligence 664,
DOI 10.1007/978-3-319-44735-3_10

can be used effectively to find a near-optimal set of connection weights without computing gradient information. The fitness of an ANN can be defined according to different needs. Moreover, the task of learning the connection weights can be stated as a Multi-Objective task and Multi-Objective Evolutionary Algorithms (MOEAs) can be used to solve this task.

In this study, two different MOEAs are investigated: Speed constrained Multi-Objective Particle Swarm Optimization (SMPSO) [8] and Multi-Objective Differential Evolution Algorithm Based on Decomposition (MOEA/D-DE) [19] with Dynamical Resource Allocation (DRA - MOEA/D-DE-DRA) [20].

The algorithm of Speed constrained Multi-Objective Particle Swarm Optimization (SMPSO) is a technique of optimization based on Particle Swarm Optimization (PSO). PSO developed by Kennedy and Eberhart [8], is a population-based heuristic inspired by the social behavior of bird flocking aiming to find food. PSO have some similarities with evolutionary algorithms: both systems are initialized with a set of solutions, possibly random, and search for optima by updating generations. Despite these similarities, there are two main differences between them. First, there is no notion of offspring in PSO, the search is guided by the use of leaders. Secondly, PSO has no evolution operators such as crossover or mutation. In Particle Swarm Optimization, the set of possible solutions is a set of particles, called swarms moving in the search space, in a cooperative search procedure. These moves are performed by an operator that is guided by a local and a social component [9]. SMPSO algorithm is an extension of PSO for solving Multi-Objective problem. Researchers like SMPSO algorithm because this algorithm is easy to program when compared to other MOEAs.

Multi-Objective Differential Evolution Algorithm based on Decomposition (MOEA/D) is an evolutionary algorithm that optimize multi-objectives problems, using the idea of decomposition [19]. MOEA/D decompose the multi-objective problem into different sub-problems using scalar weight functions. Thus, the algorithm solves these sub problems simultaneously evolving a population of solutions using differential evolution operators. In each generation, the population is composed by the best solution found so far for each sub-problem. The relation among sub-problems are set based on the distances between their weighting vectors [19]. The MOEA/D-DE-DRA algorithm [20] uses the same concepts of MOEA/D [19], but the amount of computational resources (memory) reserved to solve each sub-problem is based on a utility function. Nowadays, MOEA/D-DRA is a state of art on MOEAs.

These two MOEAs are used to train ANN to classify data. With this purpose, two fitness functions are used: the sensitivity and specificity criteria that are directly related to the quality of the classification. An empirical evaluation is made using different UCI databases and the comparison show the effectiveness of these algorithms.

This work is structured as follow: Sect. 10.2 present the basic concepts of ANN (Sect. 10.2.1), Evolutionary Algorithms (Sect. 10.2.2), SMPSO (Sect. 10.2.2.1), MOEA/D-DE-DRA (Sect. 10.2.2.2), Hypervolume (Sect. 10.2.3) and the classification problem (Sect. 10.2.4); Sect. 10.3 describes the configuration of experiments and the obtained results. Finally, Sect. 10.4 has the conclusion and future works.

10.2 Elementary Concepts

In this section, we describe concepts of MLP, multi-objective optimization and the algorithms used in the study. Moreover, elementary concepts of classification are presented.

10.2.1 Artificial Neural Networks - ANN

Researches on neural networks look to the organization of the brain as a model for building intelligent machines. Moreover, the human brain processes information in an entirely different way than conventional digital computer [5]. The brain is a highly complex computer, non-linear and parallel. It has the ability to organize their structural components, known as neurons, in order to perform certain tasks, such as pattern recognition, sense and motor control, much faster than the fastest existing digital computer.

An ANN consists of a set of processing elements, also known as neurons or nodes, which are interconnected. It can be described as a directed graph in which each node i performs a transfer function f_i as described by Eq. 10.1

$$y_i = f_i\left(\sum_{j=1}^{n}(w_{ij} \cdot x_j) + bias\right) \tag{10.1}$$

where y_i is the output of the node i, x_i is the j_{th} input to the node, and w_{ij} is the connection weight between nodes i and j. The threshold is the *bias* of the node. Usually, f_i is nonlinear, such as a heaviside, sigmoid, or Gaussian function. Equation 10.2 shows the sigmoid function.

$$out = \frac{1}{1 + e^{-net}} \tag{10.2}$$

A neural network topology represents the way in which neurons are connected to form a network. In other words, the neural network topology can be seen as the relationship between the neurons by means of their connections. The topology of ANNs can be divided into feedforward (FFNN) and recurrent classes according to their connectivity. An ANN is a feedforward if the information flow is unidirectional. A unit sends information to another unit from which it does not receive any information. There are no feedback loops. They are used in pattern generation, recognition and classification. In recurrent ANNs, feedback loops are allowed. They are used in content addressable memories.

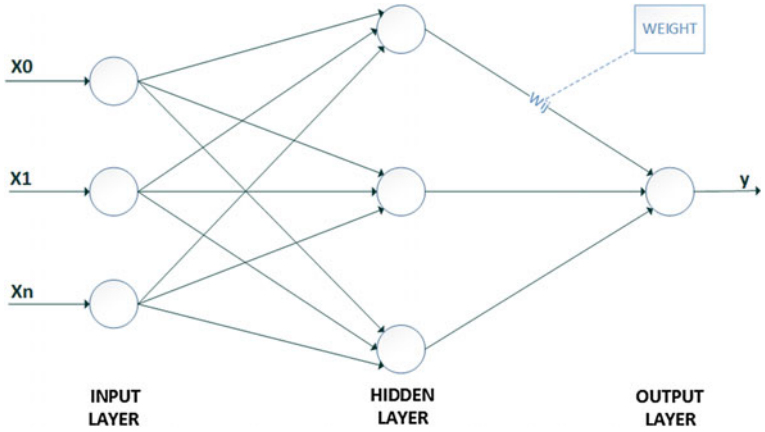

Fig. 10.1 Representation of an ANN - MLP

Basically, there are two kinds of FFNN: single-layer perceptron (SLP), and multi-layer perceptron (MLP). The SLP networks consist of a single layer of output nodes, which are fed directly by input layer via a set of weights. MLP networks consist of multiple layers: an input layer, one or more hidden layers and an output layer. Each layer has nodes and each node is fully weighted interconnected to all nodes in the subsequent layer. Figure 10.1 shows an illustration of an ANN of the type MLP.

The most important feature of an ANN is how its learning process occurs. According to Haykin [5], the learning is defined as a process where the free parameters of a neural network are adjusted by a stimulation process by the environment where it is inserted.

In supervised learning, training is performed by presenting a large set of examples, called the training set, to the network. Each example consists of a set of inputs presented to the input layer and the respective set of desired outputs. Although training an ANN can be time-consuming, once this stage is successful completed, the input–output mapping is evaluated almost instantaneously. However, care must be taken to use an adequate training set, representative of the sampling space. In many cases this is not feasible, and the sampling space must be restricted to a specific sub-domain. This means that ANNs are best applied to specific well and defined problems [3].

When using a MLP to solve a problem, the first activity is to train the MLP. Training depends on chosen initial weights and usually applies gradient learning algorithms to adapt weight values. Among these algorithms, error Backpropagation (BP) method [15] is one of the most used. In BP, the weight adjustment starts in the output nodes, where the measure of the error is available, and proceeds back-propagating this error through the previous layers. BP is a method based in gradient descendent, what means BP does not assure to find a global minimum and can get stuck on local minima, where it will stay indefinitely. However, BP is popular and widely used on ANN training [17].

As alternative, evolutionary algorithms can be applied to global searches within the weight space of a typical feedforward neural network (FFNN) and outline local minima and enable adaptive selection of control parameters [7, 16].

10.2.2 Multi-Objective Evolutionary Algorithms-MOEAs

According to Yao [18], the EAs can be used in the global evolution, to find a set of optimal (or near-optimal) weights of connections, and without gradient calculation. The error value can be defined based on the specific needs of the task to run. A commonly used factor in the formulation of the error function is the difference, called the error between the expected output and the actual output.

Two MOEAs are chosen for this study: MOEA/D-DE-DRA a state of art on MOEAs and SMPSO algorithm because this algorithm is easy to program when compared to other MOEAs.

10.2.2.1 SMPSO

Particle Swarm Optimization (PSO) is a stochastic meta-heuristic based on the movement of bird flocks looking for food, created to optimize nonlinear functions. In this method a swarm (population) of particles (solutions) moves across the search space (evolves) guided by personal and social leaders. A particle as two components: position and velocity. These components are updated at each generation.

Equations 10.3 and 10.4 present the rules for updating the speed (v_i) and position (p_i) of a particle i. The first member of the Eq. 10.3 is the inertia term, the second term is a movement to the personal best position $pBest_i^t$ and the third term is a movement towards the global best position $gBest_i^t$ (social term).

To expand the PSO to solve multi-objective problems, and create a Multi-Objective Particle Swarm Optimization (MOPSO) [14] algorithm, some modifications are needed. The first of them is the creation of an external archive (repository) to store the better (non-dominated) solutions found so far, another modification is in the leader selection scheme, which has to choose from a set of equally good leaders according to some criterion. As the number of non- dominated solutions may become very large, an archiving method is needed to prune the repository and keep only a predefined number of solutions, discarding some non-dominated solutions according to its criterion.

A MOPSO that has shown very good results in the literature is the Speed-constrained Multi-objective PSO (SMPSO) [11]. It was noted that in some conditions the velocity of the particles in a MOPSO can become too high, generating erratic movements towards the limits of the decision space. To avoid such situations, SMPSO presents a velocity constriction mechanism based on a factor χ that varies based on the values of the influence coefficients of personal and global leaders (C1 and C2 respectively). In SMPSO, the (global) leader selection method uses a binary

tournament based in the Crowding Distance metric from [2], and the archiving strategy also uses the Crowing Distance.

$$v_i^{t+1} = \overbrace{\omega \cdot v_i^t}^{inertia} + \overbrace{c_1 \cdot r_1^t (pBest_i^t - p_i^t)}^{personal} + \overbrace{c_2 \cdot r_2^t (gBest_i^t - p_i^t)}^{social} \tag{10.3}$$

$$p_i^{t+1} = p_i^t + v_i^{t+1} \tag{10.4}$$

Algorithm 9 Pseudocode of SMPSO algorithm

```
Require: swarm size;
Ensure: repository;
initialize(particles)
repository = initializeRepository(particles)
gen = 0;
while gen < max_generations do
  for each particle in the repository do
    selectGlobalLeader(particle, repository)
    ComputeSpeed(particle)
    updatePosition(particle)
    mutation(particle)
    evaluation(particle)
    updatePersonalLeader(particle)
  end for
  repository = updateRepository(particles)
  gen++;
end while
return repository;
```

At Algorithm 9 the pseudo-code of the SMPSO algorithm is presented. First the swarm and leaders archive (repository) are initialized and the evolutionary process begin. At each generation, for each particle in the population, the leaders are calculated and then the speed and position are updated. After, it is performed the Polynomial mutation for each particle, and the particles are evaluated. Finally, the particles update the leaders archive. The output of SMPSO is the leaders archive or repository.

10.2.2.2 MOEA/D-DE-DRA

The decomposition is another way to solve a problem with multi-objectives. The MOEA/D-DE-DRA decompose one multi-objective optimization problem (MOP), in many single-objective sub-problems.

There are two main components in MOEA/D. First, the mechanism to decompose MOP into sub-problems. Normally weight vectors are generated randomly and

each one defines a sub-problem. The objective of each sub-problem is a (linear or nonlinear) weighted aggregation of all the individual objectives in the MOP.

The second main component is the neighborhood relations among these sub-problems. The neighborhood relations are defined based on the distances between their weight vectors. Each sub-problem (i.e., scalar aggregation function) is optimized in MOEA/D by using information from its neighboring sub-problems.

The MOEA/D-DE with Dynamical Resource Allocation (DRA) is a version where different amounts of computational effort are allocated to different problems. In MOEA/D with Dynamical Resource Allocation (MOEA/D-DE-DRA), the version of MOEA/D used in this paper, the utility π_i for each subproblem is used.

MOEA/D-DE and its variants can use any decomposition approach for defining their sub-problems. This work uses the Tchebycheff [20] approach. Using this decomposition method, each sub-problem can be formulated as in Eq. 10.5:

$$\begin{aligned} &Min\ g^{te}(x \mid \lambda, z^*) = max_{1 \leq j \leq M}\ \{\lambda_j \mid f_j(x) - z_j^* \mid\} \\ &\qquad\qquad subject\ to\ x\ \in\ \Omega \end{aligned} \tag{10.5}$$

wherein g^{te} is the Tchebycheff function, $f(x) = (f_1(x), \ldots, f_M(x))$ is the set of functions that has to be minimized, and $\lambda = (\lambda_1, \ldots, \lambda_M)$ is the weight vectors.

The sub-problems are evolved using Differential Evolution(DE) operators. DE uses a simple mutation operator based on differences between pairs of solutions (called vectors) with the aim of finding a search direction based on the distribution of solutions in the current population. DE also utilizes a steady-state-like replacement mechanism, where the newly generated offspring (called trial vector) competes only against its corresponding parent (old object vector) and replaces it if the offspring has a higher fitness value.

The MOEA/D-DE-DRA is presented at Algorithm 10. The first steps of MOEA/D-DE-DRA is to initialize various data structures, analogous to most MOEA/D variants. The weight vectors λ_i, $i = 1, \ldots, N$, representing coefficients associated with each objective, are generated using a uniform distribution. The neighborhood $(B^i = i_1, \ldots, i_C)$ of weight vector λ_i stores the indexes of the C weight vectors closest to λ_i. The initial population is randomly generated and evaluated. Each individual (x_i) is associated with the i_{th} weight vector. The empirical ideal point (z^*) is initialized as the minimum value of each objective found in the initial population and the generation (g) is set to 1.

After initialization steps, the algorithm enters its main loop. The first step of the main loop is to determine which individuals from the population will be processed. A 10-tournament selection based on the utility value of each sub-problem (π_i, calculated accordingly to Eq. 10.6) is used to determine the individuals to evolve. Next, the scope used during the generation of the individual and the population update is randomly chosen. DE heuristics (mutation strategies and crossover) are applied considering individuals randomly selected from scope. In this work, scope can swap from the neighborhood to the entire population (and vice-versa) It is composed by the indexes

Algorithm 10 Pseudocode of MOEA/D-DE-DRA algorithm

Require: Population size (N); number of objectives (M)
1: λ^i = genWeightVectors(N);
2: $\lambda^i = (\lambda^i_1, ..., \lambda^i_M)$; $i = 1, ..., N$
3: **for** $i = 1, ..., N$ **do**
4: define the set of neighbor indexes $B^i = \{i_1, ..., i_C\}$, where $\{\lambda^{i_1}, ..., \lambda^{i_C}\}$ are C weight vectors closest to λ^i (Euclidian Distance)
5: **end for**
6: pop ← initializeRandomly();
7: Evaluate each individual $i \in$ pop and associate to its weight vector λ^i;
8: Initialize $z^* = (z^*_1, ..., z^*_M)$;
9: $z^*_j = min_{1 \leq i \leq N} f_j(x^i)$
10: g = 1;
11: **while** g > max evaluations **do**
12: I = Select using 10-tournament with (π^i);
13: **for** each Individual $i \in$ I **do**
14: **if** $rand < \delta$ **then**
15: scope = B^i;
16: **else**
17: scope = $\{1, ..., N\}$;
18: **end if**
19: y = Crossover(DE/Rand/1/bin, i);
20: $y^{'}$ = PolynomialMutation(y);
21: evaluate($y^{'}$);
22: update z^*; $z^*_j = min(z^*_j, f_j(y^{'}))$
23: **for** each subproblem k (k randomly selected from scope) **do**
24: **if** $g^{te}(y^{'}|\lambda^k, z^*) < g^{te}(x^k|\lambda^k, z^*)$ **then**
25: **if** a new replacement may occur **then**
26: Replace x^k by $y^{'}$ and increment n_r;
27: **end if**
28: **end if**
29: **end for**
30: g++;
31: **end for**
32: computeUtility();
33: **end while**

of chromosomes from either the neighborhood B^i (with probability δ) or from the entire population (with probability $1 - \delta$). Based on the chosen strategy, a modified chromosome y is generated in step 19 and modified by the polynomial mutation in step 20, generating $y^{'} = (y^{'}_1, \ldots, y^{'}_n)$ from y.

In step 22, if the new chromosome $y^{'}$ has an objective value better than the value stored in the empirical ideal point, z^* is updated with this value. The next steps involve the population update process (steps 23–26) which is based on the comparison of the fitness of individuals. In the MOEA/D-DE framework, the fitness of an individual is measured accordingly to a decomposition function. In this work the Tchebycheff function is used (Eq. 10.5) Accordingly to what is selected for the scope (steps 15 or 17), the neighborhood or the entire population is updated.

To avoid the proliferation of y' to a great part of the population, a maximum number of updates (NR) is used. The population update is as follows: if a new replacement may occur, (i.e., while $nr < NR$ and there are unselected indexes in scope), a random index (k) from scope is chosen. If y' has a better Tchebycheff value than x_k (both using the k_{th} weight vector - λ_k) then y' replaces x_k and the number of updated chromosomes (nr) is incremented. If the current generation is a multiple of 50, then the utility value of each sub-problem is updated using Eq. 10.6. The evolutionary process stops when the maximum number of evaluations is reached.

$$\pi^2 = \begin{cases} 1, & if\, \Delta^i > 0.001 \\ (0.95 + 0.05 * \Delta^i / 0.001) * \pi^i, & otherwise \end{cases} \tag{10.6}$$

10.2.3 Hypervolume

The performance comparison of one or more multi-objective optimization methods is a complex task. Two goals of multi-objective optimization are: convergence and diversity of solutions.

A widely used metric in the evaluation of multi-objectives algorithms is the indicator of Hypervolume (HV). In HV, the volume of the covered area between the points of the solutions on the Pareto front P (non-dominated solutions) and a reference point W is calculated. Each solution $i \in P$, constitutes a hypercube, v_i with reference to a point W [21]. This reference point can be found by building a vector with the worst values of the objective function. The union of all hypercubes found is the result of the metric and, as higher is the value of HV better are the results. Higher values of HV indicate that there is a higher spreading between the solutions in P and indicate that there is a better convergence to the Pareto front.

Hypervolume corresponds to the area formed by the union of all rectangles, as shown in Fig. 10.2.

10.2.4 Classification Problem

Classification is one of the main tasks of Data Mining. According to Han and Kamber [4] classification is the process of finding a model or function that describes and distinguishes data elements or concepts in order to be able to use the model to predict the class of an object whose class is unknown. The derived model is based on analysis of a set of training data.

The training data consist of pairs of inputs (vectors) and desired outputs. For example, in a classification problem, a hospital may want to classify medical patients into those who have high, medium or low risk to acquiring a certain illness.

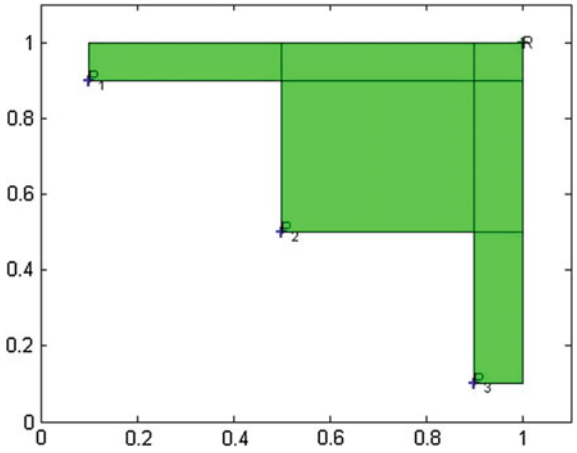

Fig. 10.2 Hypervolume area

The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before. Therefore, a key objective of the learning algorithm is to build models with good generalization capability; i.e., models that accurately predict the class labels of previously unknown records.

A general approach for solving classification problems consist of two steps. First, a training set consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the test set, which consists of records with unknown class labels.

Evaluation of the performance of a classification model is based on the counts of test records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a confusion matrix (Table 10.1).

From the confusion matrix (10.1) is possible to calculate measures such as: True Positive rate (*TP rate*), True Negative rate (*TN rate* or specificity), False Positive rate (*FP rate*) and False Negative rate (*FN rate*). *TP rate*, also called sensitivity, is the precision between the positive examples (Eq. 10.7). Its complement is the *FN rate* (i.e., $FNrate = 1 - FPrate$). Specificity is the precision between the negative examples (Eq. 10.8). Its complement is the *FP rate*.

Table 10.1 Confusion matrix

	$Class = 1$	$Class = 0$	**Predicted class**
$Class = 1$	TP	FP	$TP + FP$
$Class = 0$	FN	TN	$FN + TN$
$ActualClass$	$TP + FN$	$FP + TN$	N

$$sensitivity = \frac{TP}{TP + FN} \tag{10.7}$$

$$specificity = \frac{TN}{TN + FP} \tag{10.8}$$

For several years, the most used performance measure for classifiers was the accuracy [1]. The accuracy is the fraction of examples correctly classified, showed on Eq. 10.9. Despite of its use, the accuracy maximization is not an appropriate goal for many of the real-world tasks [13]. A tacit assumption in the use of classification accuracy as an evaluation metric is that the class distribution among examples is constant and relatively balanced. In real world this case is rare, moreover, the cost associated with the incorrect classification of each class can be different because some classifications can lead to actions which could have serious consequences [12].

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10.9}$$

Classification is one of the most dynamic exploratory and application areas of ANNs. However, as mentioned before the selection of connection weights is a key issue and here this issue is tackle with two MOEAs.

10.3 Experimental Evaluation and Results

The experimental evaluation aims at answering the following research questions:

RQ1: Is there difference of performance among the configurations of each algorithm?
RQ2: Is there difference of performance between SMPSO and MOEA/D-DE-DRA?
RQ3: What are the advantages of the multi-objective versus mono-objective approach for evolving connection weights of ANN for classification task?

To answer RQ1, first different configurations of the algorithms are used to learn ANNs for each training database using sensitivity and specificity as fitness functions. Second, the learned ANNs are applied into the test databases obtaining a new set of values of sensitivity and specificity. Finally, the different configurations are compared using the Hypervolume indicator and the Friedmann rank test [6].

The goal of RQ2 is to verify whether exists one algorithm with better results than the other. The results obtained in RQ1 are now compared using the best configuration obtained for each algorithm. Again the Friedmann rank test is used.

To answer RQ3, the results generated by applying the ANNs to each test databases are analyzed using the accuracy, sensitivity and specificity.

In order to verify statistical difference among the results found by all algorithms and settings, all of them were run 30 times and Friedmann [10] and Mann–Whitney tests were executed with 0.05 significance level.

This section explains the methodology adopted to evolve connection weights of artificial neural network using a multi-objective approach and its application in class Prediction. The Java language was used to implement the ANN and to compute the two fitness functions: sensitivity and specificity. The implementation of SMPSO and MOEA/D-DE-DRA available at the JMetal Framework were used.

The following databases were used:

1. Breast Cancer Wisconsin (Original) Data Set (called as Cancer);
2. Pima Indians Diabetes Data Set (called as Diabetes);
3. Glass Identification Data Set (called as Glass);
4. Statlog (Heart) Data Set (called as Heart).

Each database was divided into 2 groups of instances, each one corresponding to training set and testing set. These groups were set up with different sizes depending on the database as shown in Table 10.2.

The topologies of the ANNs were defined according to the databases. The input layers are defined according to the numbers of attributes and the output layer according to the number of classes. The complete definition of the used topologies is presented at Table 10.3.

The topology defines the size of the individuals that were evolved by the algorithms, one dimension for each connection plus the bias for each neuron, i.e., each individual defines one ANN. The neurons used a sigmoid function.

The algorithms were executed with two different population sizes: 50 and 100 and two different number of generations: 500 and 1000, given four different configurations for each algorithm. C1 with a population size of 50 and number of generations set to 500; C2 with a population size of 50 and number of generations set to 1000; C3 with a population size of 100 and number of generations set to 500 and, C4

Table 10.2 Separation of databases

Data base	Training	Testing	Total
Cancer	500	183	683
Diabetes	650	118	768
Glass	170	44	214
Heart	220	50	270

Table 10.3 Number of neurons of each layer

Base	Attributes (Input)	Classes (Output)	Hidden
Cancer	9	2	5
Diabetes	8	2	10
Glass	9	7	10
Heart	13	2	5

Table 10.4 Parameters values used

Parameter	Value
F	0.3
CR	0.7
NR	2
T	20
Δ (delta)	0.9
c_1	[1.5:2.5]
c_2	[1.5:2.5]
r_1	[0.0:1.0]
r_2	[0.0:1.0]
ω	0.1

with a population size of 100 and number of generations set to 1000. The remaining parameters were set as presented at Table 10.4 using the default values of the JMetal.

Next we present and discuss the results of the experiments in order to answer the research questions.

10.3.1 RQ1 - Comparing Different Configuration of Each Algorithm

As mentioned before, different configurations of each algorithm were compared to set the values of the parameters: population size and number of iterations.

Table 10.5 shows the mean values and standard deviation of Hypervolume indicator. At the top of the Table, the results of the SMPSO are reported and at the bottom the results of MOEA/D-DE-DRA. For SMPSO, the best configuration for Cancer is C1, Diabetes is C4, Glass is C3 and for Heart is C4. In the case of MOEA/D-DE-DRA, the best configuration for Cancer, Diabetes and Heart is C2 and for Glass is C4. However, the difference between the values of Hypervolume is not high. For a deep analysis on these values the Kruskal–Wallis at 0.5 significance level was applied. These results are reported at Table 10.6, for SMPSO and MOEA/D-DE-DRA. Analyzing the Kruskal–Wallis results, is possible to observe that for SMPSO the configuration C4 always get best or equivalent results for all databases. For MOEA/D-DE-DRA, the configuration C2 almost always get best or equivalent results for all databases, with exception in Glass where C4 is the best configuration.

The confirmation of these findings is given by the average rankings of configurations obtained using Friedman test. These results are showed for SMPSO and MOEA/D-DE-DRA at Tables 10.7 and 10.8 respectively. Summarizing, the Friedman test point out configuration C4 for SMPSO and C2 for MOEA/D-DE-

Table 10.5 Results of Hypervolume in each configuration

Algorithm	Data Base	Mean HV C1 (Std)	Mean HV C2 (Std)	Mean HV C3 (Std)	Mean HV C4 (Std)
SMPSO	Cancer	**0.99889** (0.002572)	0.99586 (0.008267)	0.99500 (0.013317)	0.99802 (0.005244)
	Diabetes	0.81054 (0.075324)	0.85761 (0.038138)	0.85706 (0.039694)	**0.85878** (0.050186)
	Glass	0.99573 (0.001395)	0.99035 (0.003285)	**0.99681** (5.8690E-4)	0.99672 (0.001294)
	Heart	0.64645 (0.037708)	0.64999 (0.032952)	0.66789 (0.032417)	**0.67879** (0.027469)
MOEA/D-DE-DRA	Cancer	0.94478 (0.045548)	**0.97901** (0.031272)	0.97803 (0.032959)	0.94298 (0.041491)
	Diabetes	0.50173 (0.027114)	**0.62406** (0.041031)	0.60731 (0.038475)	0.50856 (0.037513)
	Glass	0.83237 (0.002012)	0.98992 (0.035803)	0.99217 (0.026678)	**0.99849** (0.002765)
	Heart	0.65033 (0.074096)	**0.72171** (0.073602)	0.71988 (0.069996)	0.69071 (0.079847)

DRA as the better considering all databases. So, these configurations were chosen for being used in the following experiments.

10.3.2 RQ2 - Comparing Different Algorithms

To answer RQ2, we compared the results from SMPSO algorithm with MOEA/D-DE-DRA, using the configurations chosen according to the results presented previously. Table 10.9 shows the results of the Wilcoxon test at 0.5 significance level and the effect size. It possible to observe that the algorithms present significant different results for each database. However, SMPSO presents better results for Cancer, Diabetes and Glass. For Heart the best results are for MOEA/D-DE-DRA.

Figure 10.3 depicts the obtained fronts using SMPSO and MOEA/D-DE-DRA for Diabetes. Tables 10.10 and 10.11 present the values of sensitivity and specificity of each of the solutions in the fronts, for SMPSO and MOEA/D-DE-DRA respectively. These fronts are the obtained fronts after executing 30 times the algorithms and removing dominated and repeated solutions.

For Diabetes, SMPSO clearly outperforms MOEA/D-DE-DRA. The same happens for Heart but, in this case, is MOEA/D-DE-DRA that outperforms SMPSO. Then, the average rankings was obtained using Friedman test. These results are presented at Table 10.12, there is possible to observe that SMPSO is slightly better than MOEA/D-DE-DRA considering the Hypervolume.

Table 10.6 Kruskal–Wallis at 0.05 significance level for Hypervolume

Dataset	Algorithm	Conf.	C1	C2	C3	C4
Cancer	SMPSO	C1	–	TRUE	TRUE	FALSE
		C2	TRUE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	FALSE	TRUE	TRUE	–
	MOEA/D-DE-DRA	C1	–	TRUE	TRUE	FALSE
		C2	TRUE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	FALSE	TRUE	TRUE	–
Diabetes	SMPSO	C1	–	TRUE	TRUE	TRUE
		C2	TRUE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	TRUE	TRUE	TRUE	–
	MOEA/D-DE-DRA	C1	–	TRUE	TRUE	FALSE
		C2	TRUE	–	FALSE	FALSE
		C3	TRUE	FALSE	–	TRUE
		C4	FALSE	FALSE	TRUE	–
Glass	SMPSO	C1	–	TRUE	TRUE	TRUE
		C2	TRUE	–	TRUE	TRUE
		C3	TRUE	TRUE	–	FALSE
		C4	TRUE	TRUE	FALSE	–
	MOEA/D-DE-DRA	C1	–	TRUE	TRUE	TRUE
		C2	TRUE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	TRUE	TRUE	TRUE	–
Heart	SMPSO	C1	–	FALSE	TRUE	TRUE
		C2	FALSE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	TRUE	TRUE	TRUE	–
	MOEA/D-DE-DRA	C1	–	TRUE	TRUE	TRUE
		C2	TRUE	–	FALSE	TRUE
		C3	TRUE	FALSE	–	TRUE
		C4	TRUE	TRUE	TRUE	–

Table 10.7 SMPSO average rankings of configurations (Friedman)

Configuration	Ranking
C1	3.0
C2	2.75
C3	2.5
C4	1.75

Table 10.8 MOEA/D-DE-DRA average rankings of configurations (Friedman)

Configuration	Ranking
C1	3.75
C2	1.5
C3	2.0
C4	2.75

Table 10.9 Wilcoxon test at 0.05 significance level, SMPSO x MOEA/D-DE-DRA, Hypervolume results

Dataset	p-value	Observation diff.	Critical diff.	Diff.	Effect size
Cancer	0.0008472	16.62222	8.837967	TRUE	0.6822841 (medium)
Diabetes	0.0009271	14.93333	8.837967	TRUE	0.7488889 (large)
Glass	0.0009148	6.81322	8.837967	TRUE	0.573216 (small)
Heart	0.0071166	12.43255	8.837967	TRUE	0.421211 (small)

10.3.3 RQ3 - Advantages of a Multi-Objective Approach

In the task of learning classification algorithms as ANNs, the goal is to create algorithms that have good performance for classification. Hence, the great majority of the methods aims to optimize the performance of the classification by improving the accuracy in the set. Despite of its use, the accuracy maximization is not an appropriate goal for many of the real-world tasks [13]. A tacit assumption in the use of classification accuracy as an evaluation metric is that the class distribution among examples is constant and relatively balanced. In real world this is rarely the case, because classification leads to actions which could have serious consequences. Therefore, recent researches point out sensitivity and specificity as better metrics to be used for induction of classification algorithms. Sensitivity is a relative measures of instances of the positive class that are well classified. Hence, the greater the

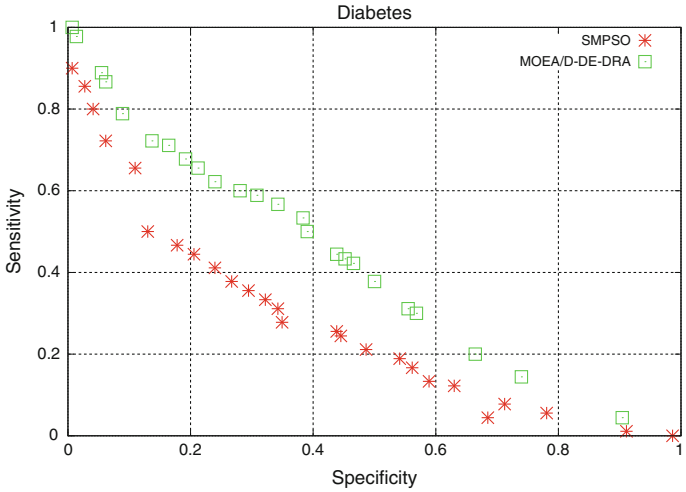

Fig. 10.3 Obtained Fronts, SMPSO and MOEAD/D-DE-DRA for Diabetes

sensitivity, the greater the number of instances in the positive class that are correctly classified. Specificity is the same measure of sensitivity, but for negative instances. The greater its value, the lower the number of instances in the negative class that are misclassified. Sensitivity and specificity are inversely proportional, meaning that as the sensitivity increases, the specificity decreases and vice versa. For understanding the advantages of a multi-objective approach based on these two metrics in the following the ANNs obtained by SMPSO for Diabetes are deeply analyzed. Table 10.13 presents TP, FP, TN, FN and accuracy of the ANNs sorted by increased value of TP. It possible to note that as TP increases, TN decreases. The best value of accuracy achieved is 0.7288135593, with TP = 127, FP = 19, TN = 45 and FN = 45. Or in other words, 127 patients that have diabetes were diagnostic as having diabetes but 19 patients that have not diabetes were included in the diagnostic. In the other hand, 45 patients that have not diabetes were confirmed as not having the diseases but 45 patients that have diabetes were diagnostic as without diabetes. This can be dangerous because a treatment at time can make a good difference on the quality of life for these patients. Having access to all these informations another ANN could be used. That is, the user as more freedom to adequate the ANN that is better for its preference.

10.4 Conclusion

ANNs are specially used to find a general solution in problems where a pattern needs to be extracted, such as data-mining. The main difficulty to apply ANN in some domain problem is to train the ANN to learn and predict. ANN provides different ways to solve many nonlinear problems that are hard to solve by conventional techniques.

Table 10.10 SMPSO obtained Pareto Front for Diabetes

Solution	Sensitivity	Specificity
s1	0.9863013699	0
s2	0.9109589041	0.0111111111
s3	0.7808219178	0.0555555556
s4	0.7123287671	0.0777777778
s5	0.6849315068	0.0444444444
s6	0.6301369863	0.1222222222
s7	0.5890410959	0.1333333333
s8	0.5616438356	0.1666666667
s9	0.5410958904	0.1888888889
s10	0.4863013699	0.2111111111
s11	0.4452054795	0.2444444444
s12	0.4383561644	0.2555555556
s13	0.3493150685	0.2777777778
s14	0.3424657534	0.3111111111
s15	0.3219178082	0.3333333333
s16	0.2945205479	0.3555555556
s17	0.2671232877	0.3777777778
s18	0.2397260274	0.4111111111
s19	0.2054794521	0.4444444444
s20	0.1780821918	0.4666666667
s21	0.1301369863	0.5
s22	0.1095890411	0.6555555556
s23	0.0616438356	0.7222222222
s24	0.0410958904	0.8
s25	0.0273972603	0.8555555556
s26	0.0068493151	0.9

The use of evolutionary algorithms has excelled to problem solving that requires space of global search optimization in several types problems. Theses algorithms have also been used to train ANNs. This paper describes and compares the results obtained in ANN training with two different algorithms: based on particle swarm optimization (SMPSO) and differential evolution(MOEA/D-DE-DRA). ANNs are trained for classification task, moreover, to properly tackle this task, ANNs need to maximize two metrics: sensitivity and specificity.

An experiment was conducted using different benchmark databases. First the goal was to determine the values of two important parameters of the algorithms: the population size and number of generations. After then, the best configurations were

Table 10.11 MOEAD obtained Pareto Front for Diabetes

Solution	Sensitivity	Specificity
S1	0.6643835616	0.2
S2	0.5684931507	0.3
S3	0.5547945205	0.3111111111
S4	0.5	0.3777777778
S5	0.4657534247	0.4222222222
S6	0.4520547945	0.4333333333
S7	0.4383561644	0.4444444444
S8	0.3904109589	0.5
S9	0.3835616438	0.5333333333
S10	0.3424657534	0.5666666667
S11	0.3082191781	0.5888888889
S12	0.2808219178	0.6
S13	0.2397260274	0.6222222222
S14	0.2123287671	0.6555555556
S15	0.1917808219	0.6777777778
S16	0.1643835616	0.7111111111
S17	0.1369863014	0.7222222222
S18	0.0890410959	0.7888888889
S19	0.0616438356	0.8666666667
S20	0.0547945205	0.8888888889
S21	0.0136986301	0.9777777778
S22	0.0068493151	1

Table 10.12 Average rankings of the algorithms (Friedman)

Algorithm	Ranking
SMPSO	2.3125
MOEAD	2.6875

compared to answer which is the best algorithm for the task. Here, it was possible to observe that the best algorithm depends on the database, however, SMPSO presented slightly better results. Finally, using the results found for Diabetes the advantages of using sensibility and specificity were illustrated.

Future works include analyzing the influence of other parameters of the algorithms, for example to use an adaptive version of MOEA/D-DE-DRA. It is known that an appropriate configuration of parameters can produce better results.

Table 10.13 SMPSO obtained solutions for Diabetes

Solution	TP	FP	TN	FN	Accuracy
S1	2	144	90	0	0.3898305085
S2	13	133	89	1	0.4322033898
S3	32	114	85	5	0.4957627119
S4	42	104	83	7	0.5296610169
S5	46	100	86	4	0.5593220339
S6	54	92	79	11	0.563559322
S7	60	86	78	12	0.5847457627
S8	64	82	75	15	0.5889830508
S9	67	79	73	17	0.593220339
S10	75	71	71	19	0.6186440678
S11	81	65	68	22	0.6313559322
S12	82	64	67	23	0.6313559322
S13	95	51	65	25	0.6779661017
S14	96	50	62	28	0.6694915254
S15	99	47	60	30	0.6737288136
S16	103	43	58	32	0.6822033898
S17	107	39	56	34	0.6906779661
S18	111	35	53	37	0.6949152542
S19	116	30	50	40	0.7033898305
S20	120	26	48	42	0.7118644068
S21	127	19	45	45	0.7288135593
S22	130	16	31	59	0.6822033898
S23	137	9	25	65	0.686440678
S24	140	6	18	72	0.6694915254
S25	142	4	13	77	0.656779661
S26	145	1	9	81	0.6525423729

Acknowledgments Authors would like to thank CNPq and CAPES for financial support.

References

1. Baronti F, Starita A (2007) Hypothesis testing with classifier systems for rule-based risk prediction (chap), pp 24–34). doi:10.1007/978-3-540-71783-6_3
2. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Proceedings of the 6th international conference on parallel problem solving from nature, PPSN VISpringer, London, UK, pp 849–858

3. Gaspar-Cunha A, Vieira A (2005) A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. Int J Comput Syst Signal 6(1):18–36
4. Han J, Kamber M (2006) Data mining: concepts and techniques. Morgam Kaufmann Publishers, Amsterdam
5. Haykin S (2001) Redes neurais. Bookman
6. Hodges JL, Lehmann E (2012) Rank methods for combination of independent experiments in analysis of variance. In: Rojo J (ed) Selected works of E.L. Lehmann, selected works in probability and statistics. Springer, US, pp. 403–418. doi:10.1007/978-1-4614-1412-4_35
7. Ilonen J, Kamarainen JK, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17(1):93–105. doi:10.1023/A:1022995128597
8. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ieee international conference on neural networks, 1995, vol 4, pp 1942–1948. doi:10.1109/ICNN.1995.488968
9. Kennedy J, Eberhart RC (2001) Swarm intelligence. Morgan Kaufmann Publishers Inc., San Francisco
10. Kruskal WH (1952) A nonparametric test for the several sample problem. Ann Math Statist 23(4):525–540. doi:10.1214/aoms/1177729332
11. Nebro AJ, Durillo JJ, Garcia-Nieto J, Coello CAC, Luna F, Alba E (2009) SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In: Computational intelligence in multi-criteria decision-making, IEEE, pp. 66–73
12. Provost FJ, Fawcett T (1997) Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: KDD, pp 43–48
13. Provost F, Fawcett T, Kohavi R (1998) The case against accuracy estimation for comparing induction algorithms. In: proceedings 15th international conference on machine learning, Morgan Kaufmann, San Francisco, CA, pp 445–453
14. Reyes-Sierra M, Coello CAC (2006) Multi-objective particle swarm optimizers: a survey of'the state-of-the-art. Int J Comput Intell Res 2(3):287–308
15. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. Parallel distributed processing: explorations in the microstructure of cognition, vol 1. MIT Press, Cambridge, pp 318–362. http://dl.acm.org/citation.cfm?id=104279.104293
16. Slowik A (2011) Application of an adaptive differential evolution algorithm with multiple trial vectors to artificial neural network training. IEEE Trans Ind Electron 58(8):3160–3167. doi:10.1109/TIE.2010.2062474
17. van Ooyen A, Nienhuis B (1992) Improving the convergence of the back-propagation algorithm. Neural Netw 5(3):465–471. doi:10.1016/0893-6080(92)90008-7
18. Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447. doi:10.1109/5.784219
19. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
20. Zhang Q, Liu W, Li H (2009) The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. IEEE Congr Evol Comput 1:203–208
21. Zitzler E, Thiele L, Laumanns M, Fonseca C, da Fonseca V (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(2):117–132. doi:10.1109/TEVC.2003.810758