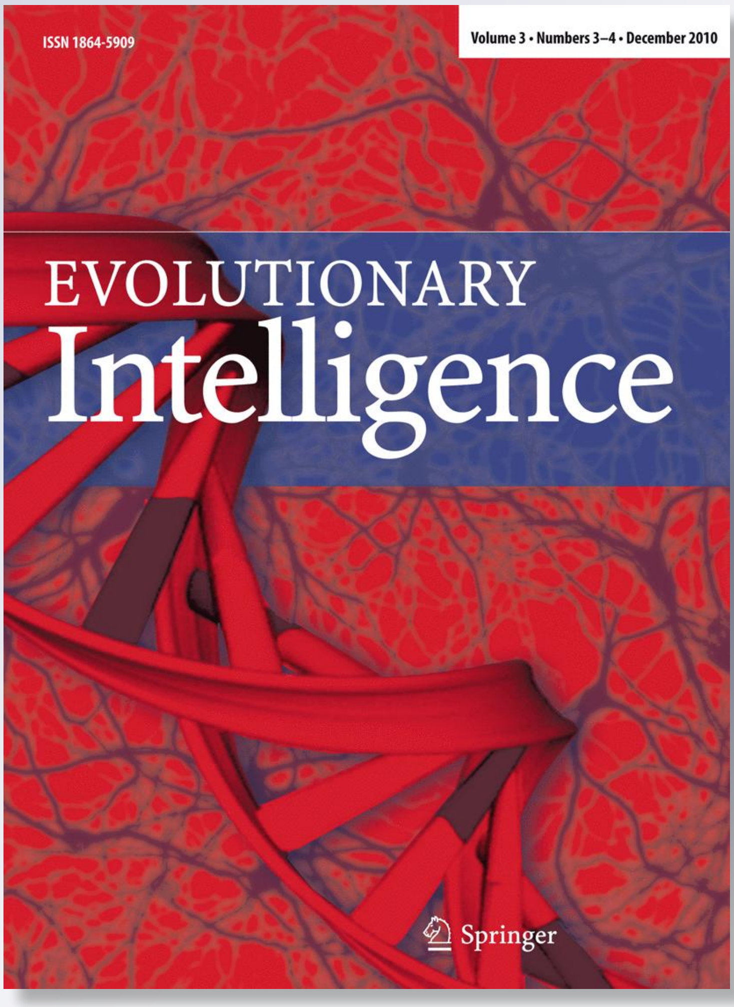
Memetic pareto differential evolutionary artificial neural networks to determine growth multi-classes in predictive microbiology

Evolutionary Intelligence

ISSN 1864-5909
Volume 3
Combined 3-4

Evol. Intel. (2010) 3:187-199
DOI 10.1007/
s12065-010-0045-9



Evol. Intel. (2010) 3:187–199
DOI 10.1007/s12065-010-0045-9

RESEARCH PAPER

Memetic pareto differential evolutionary artificial neural networks to determine growth multi-classes in predictive microbiology

M. Cruz-Ramírez · J. Sánchez-Monedero ·
F. Fernández-Navarro · J. C. Fernández ·
C. Hervás-Martínez

Received: 6 October 2010 / Accepted: 6 October 2010 / Published online: 26 October 2010

Abstract The main objective of this research is to automatically design Artificial Neural Network models with sigmoid basis units for multiclassification tasks in predictive microbiology. The classifiers obtained achieve a double objective: a high classification level in the dataset and high classification levels for each class. The Memetic Pareto Differential Evolution Neural Network chosen to learn the structure and weights of the Neural Networks is a Differential Evolutionary approach based on the Pareto Differential Evolution multiobjective evolutionary algorithm. The Pareto Differential Evolution algorithm is augmented with a local search using the improved Resilient Backpropagation with backtracking–*iRprop*$^+$ algorithm. To analyze the robustness of this methodology, it has been applied to two complex classification problems in predictive microbiology (*Staphylococcus aureus* and *Shigella flexneri*). The results obtained show that the generalization ability and the classification rate in each class can be more efficiently improved within this multiobjective algorithm.

Keywords Differential evolution · Memetic algorithms · Multiclassification · Multiobjective · Neural networks · Predictive microbiology

This paper is a very significant extension of a contribution appearing in the 23rd International Conference on Industrial and Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE2010).

M. Cruz-Ramírez (✉) · J. Sánchez-Monedero ·
F. Fernández-Navarro · J. C. Fernández · C. Hervás-Martínez
Department of Computer Science and Numerical Analysis, University of Córdoba, Rabanales Campus, Albert Einstein building 3 floor, 14071 Córdoba, Spain
e-mail: mcruz@uco.es

1 Introduction

Growth/No-Growth models have appeared in the predictive microbiology field as an approach to determine the growth ability of microorganisms. In this respect, many studies have been published in recent years for both spoilage and pathogenic microorganisms. This fact is mainly due to the need to gain more knowledge about microbial behaviour in limiting conditions that prevent growth, by using mathematical models. Consequently, these mathematical models may lead to more realistic estimations of food safety risks and can provide useful quantitative data for the development of processes which lead to the production of safer food products [14].

The main problems in modeling microbial interface are related to its abrupt transition, i.e., the great change in the value of growth probability (p) within the very narrow range of environmental factors encountered between growth and no-growth conditions. Thus, to more properly define the interface, growth and no-growth should only be considered if all replicas (reproductions of the same conditions in environmental parameters), or at least a very high percentage of them, grow or do not grow, respectively. In this paper four observed microbial responses are obtained based on the growth probability of a microorganism, ($p = 1$ (growth), G; $0.5 \leq p < 1$ (high growth probability), GHP; $0 < p < 0.5$ (low growth probability), GLP, and $p = 0$ (no-growth), NG).

A classifier design method is usually an algorithm that develops a classifier to approximate an unknown input–output mapping function from finitely available data, i.e., training samples. Once this classifier has been designed, it can be used to predict class labels (G, GHP, GLP or NG) that correspond to unseen samples. Hence, the objective in developing a good classifier is to ensure high prediction accuracy for unseen future data, i.e., testing capability. Many techniques have been proposed to improve the overall testing capability of classifier designed methods (assuming, for example, the maximization of the correct classification rate or Accuracy, *C*), but very few methods maintain this capacity in all classes [assuming, for example, maximization of the correct classification of each class (Minimum sensitivity, *MS*)].

Thus this paper proposes the simultaneous optimization of the two conflicting objectives for the determination of the growth limits of two pathogens, *Staphylococcus aureus* and *Shigella flexneri*. The algorithms that optimize two or more objectives are known as Multi-Objective Evolutionary learning Algorithms (MOEA) [8]. When these algorithms train Artificial Neural Networks (ANNs) [49], this is known as Multiobjective Evolutionary Artificial Neural Networks (MOEANNs), some of its main exponents being Abbass [2] and Jin [29].

Our algorithm is based on Differential Evolution (DE), proposed as a new heuristic by Storn and Price [42] for the minimization of functions in totally ordered spaces. DE selects three parents and applies the crossover operator to generate new individuals that will be included in the population. The main author who uses DE with MOEANNs is H. Abbass in his PDE algorithm (Pareto Differential Evolution, [2]) and the MPANN ones (Memetic Pareto Artificial Neural Networks, [1]).

The basic structure of our MOEA has been modified by introducing an additional step, where some individuals in the population have been enhanced by a local search method. The local search algorithm used is the improved Resilient Backpropagation (iRprop$^+$) algorithm [28]. For this purpose, a Memetic Pareto Differential Evolution Neural Network (MPDENN) algorithm has been developed.

The rest of the paper is organized as follows: Sect. 2 shows related work with predictive microbiology while Sect. 3 describes the MPDENN algorithm, followed by the experimental design in Sect. 4. Section 5 shows the results obtained and finally, our conclusions are explained in Sect. 6.

2 Related studies

2.1 Predictive microbiology

In food microbiology, during the past few years, much effort has been directed to develop models describing the combined effects of the factors in microbe growth [3, 9, 48]. Response Surface Models (RSM) are the most frequent techniques used to describe the relationships between the combinations of factors and growth curve parameters [7, 26], but new methods based on Artificial Intelligence (AI), are being introduced, such as the application of ANN [13, 15–18, 21, 24, 25, 44].

Knowledge of how different food product properties, environment and history are able to influence the microflora which develop when food is stored is an important first step towards forecasting its commercial shelf-life, alterations and safety.

The food industry is constantly creating new microbial habitats, either deliberately when developing new products or new formulae for traditional ones, or by chance, as a consequence of the deviations in the composition of raw materials or in the production process. In order to be able to predict microbial behavior in each new situation and estimate its consequences with respect to the safety and quality of food, there has to be an exact definition of the food environment and how this will influence microbial growth and survival.

Boundary models have been used in the predictive microbiology field as an approach to determine the growth ability of microorganisms. In this respect, several studies have been published in recent years about both spoilage and pathogenic microorganisms. This fact is mainly due to the need to use mathematical models to gain knowledge about microbial behaviour in limiting conditions that just prevent growth. Consequently, these mathematical models may lead to more realistic estimations of food safety risks, and can provide useful quantitative data for the development of processes that permit the production of safer food products [30]. Several mathematical approaches have been developed based on: deterministic estimates of the minimal values of environmental parameters where growth can occur [36]; polynomial and non-linear equations [37, 43] that can be built using a logistic regression procedure proposed by [39]; or ANNs which can be applied to define growth/no-growth interfaces of microorganisms [22].

Hajmeer and Basheer [20] used a Probabilistic Neural Network (PNN) approach for the classification of bacterial growth/no-growth data and for modeling the growth probability of a pathogenic *Escherichia coli* R31 in response to temperature and water activity. They found that PNNs were shown to outperform linear and non-linear regression models in both classification accuracy and ease. Later on, ANNs were used to predict the growth/no-growth interface [44] and survival/death of *E. coli* O157:H7 in mayonnaise model systems [47], or *Listeria monocytogenes* in chorizos [23].

Many pattern classification systems were developed for two-class classification problems and theoretical studies of learning have focused almost entirely on learning binary functions [35] including the well-known support vector machines (SVM) [46] or ANN algorithms such as the perceptron or the error back-propagation algorithm [6]. For most of these algorithms, the extension from two-class to multi-class pattern classification problems is not trivial, and often leads to unexpected complexity or weaker performances [19, 38].

To the best of our knowledge, very few attempts have been made in the predictive microbiology field, including a multi-classification structure to model microbial growth/no-growth. One approach was the study by Le Marc et al. [31] that combined the concept of the Minimum Convex Polyhedron (MCP) previously introduced in [4] with a logistic regression method to model microbial growth/no-growth boundaries. They obtained predicted probabilities corresponding to zones of the model domain belonging to growth, no-growth and uncertainty regions. However, the uncertainty region was built in zones where no data were available. Besides, the MCP was linked to microbial observations in ComBase (where no available data are found in some cases) and has also been used in combination with a logistic regression model.

3 Learning methodology

The beginning of this section describes the neural networks, an explanation of Accuracy and Minimum sensitivity and the fitness functions employed. The proposed algorithms are shown and the section concludes with a description of the local search algorithm used.

3.1 Base classifier

The standard feedforward Multilayer Perceptron (MLP) neural networks considered for multiclassification problems had one input layer with independent variables or features, one hidden layer with sigmoidal hidden nodes and one output layer with J linear nodes.

Let a coded "1-of-J" outcome variable be $\mathbf{y}$, (that is, the outcomes have the form $\mathbf{y} = (y^{(1)}, y^{(2)},\ldots, y^{(J)})$, where $y^{(j)} = 1$ if the pattern belongs to class j, and $y^{(j)} = 0$, in other cases); and a vector $\mathbf{x} = (1, x_1, x_2,\ldots, x_K)$ of input variables, where K is the number of inputs (assuming that the vector of inputs includes the constant term to accommodate the intercept or bias). The model of an MLP can be described by the following equation:

$$f_l(\mathbf{x}, \boldsymbol{\theta}_l) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l \sigma_j\left(w_0^j + \sum_{i=1}^{K} w_i^j x_i\right), \quad \text{for} \quad l = 1,\ldots,J,$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_J\}^T$ is the transpose matrix containing all the neural net weights, $\boldsymbol{\theta}_l = \{\beta_0^l,\ldots,\beta_M^l, \mathbf{w}_1,\ldots,\mathbf{w}_M\}$ is the vector of weights of the *l-th* output node, $\boldsymbol{\beta}^l = \{\beta_0^l,\ldots,\beta_M^l\}$ is the vector of the connection weight between the hidden layer and the output layer, M is the number of hidden nodes, $\mathbf{w}_j = \{w_0^j,\ldots,w_K^j\}$, for $j = 1,\ldots,M$, is the vector of input weights of the hidden node j and $\sigma(\cdot)$ is the sigmoidal activation function.

In order to tackle this classification problem, the outputs of the model have been interpreted from the point of view of probability through the use of the softmax activation function [40], which is given by:

$$p_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \boldsymbol{\theta}_j)}, \quad \text{for} \quad l = 1,\ldots,J, \tag{1}$$

where $f_j(\mathbf{x}, \boldsymbol{\theta}_l)$ is the output of the j output neuron for pattern $\mathbf{x}$ and $p_l(\mathbf{x}, \boldsymbol{\theta}_l)$ is the probability that pattern $\mathbf{x}$ has of belonging to class j.

Using the softmax activation function presented in expression 1, the class predicted by the MLP corresponds to the node in the output layer whose output value is the greatest. In this way, the optimum classification rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x}) = \hat{l}, \quad \text{where} \quad \hat{l} = \operatorname{argmax}_l p_l(\mathbf{x}, \boldsymbol{\theta}_l), \quad \text{for } l = 1,\ldots,J.$$

The best MLP is determined by means of a MOEA (detailed in Sect. 3.4) that optimizes the error function given by the negative log-likelihood for N observations associated with the MLP model:

$$L^*(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \log \sum_{l=1}^{J-1} \exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l)\right], \tag{2}$$

where $y_n^{(l)}$ is equal to 1 if pattern $\mathbf{x}_n$ belongs to the *l-th* class and is equal to 0 otherwise. From a statistical point of view, this approach can be seen as nonlinear multinominal logistic regression, where log-likelihood is optimized using a MOEA.

3.2 Accuracy and minimum sensitivity

This section presents two measures to evaluate a classifier: the Correct Classification Rate or Accuracy, C, and Minimum sensitivity, MS. To evaluate a classifier, the machine learning community has traditionally used C to measure its default performance. Actually, it is only necessary to realize that C cannot capture all the different behavioral aspects found in two different classifiers in multiclassification problems. For these problems, two performance measures are considered: traditionally-used C and MS in all classes, that is, the lowest percentage of examples correctly predicted as belonging to each class, S_i, with respect to the total number of examples in the corresponding class, $MS = \min\{S_i\}$. The pair made up of MS versus

C (MS, C) expresses two features associated with a classifier: global performance (C) and the rate of the worst classified class (MS). The selection of MS as a measure that is complementary to C can be justified by considering that C is the weighted average of the Sensitivities of each class. For a more detailed description of these measures, see [12].

The *MS–C* point of view allows us to represent the classifiers in a two dimensional space to visualize their performance, regardless of the number of classes in the problem. Concretely, MS is represented on the horizontal axis and C on the vertical axis. One point in (MS, C) space *dominates* another if it is above it and to the right, i.e. it has greater C and the best MS. Let C and MS be associated with the classifier g, then $MS \leq C \leq 1 - (1 - MS)p^*$, where p^* is the minimum for estimated prior probabilities. Therefore, each classifier will be represented as a point in the white region in Fig. 1; hence the area outside of the triangle is marked as unfeasible.

The area inside the triangle in Fig. 1 may be feasible or not (attainable), depending upon the classifier and the difficulty of the problem. *A priori*, it could seem that MS and C objectives could be positively correlated, but while this may be true for small values of MS and C, it is not so for values close to 1 in either MS or C. Thus competitive objectives are at the top right corner of the white region. This fact justifies the use of a MOEA.

3.3 Fitness functions

When there is an available training dataset $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2,\ldots, N\}$, where $\mathbf{x}_n = (x1_n,\ldots, xk_n)$ is the random vector of measurements taking values in $\Omega \subset R^K$, and $\mathbf{y}_n$ is the class level of the *n-th* individual, C is defined by:

$$C = (1/N)\sum_{n=1}^{N}(I(C(\mathbf{x}_n) = \mathbf{y}_n)),$$

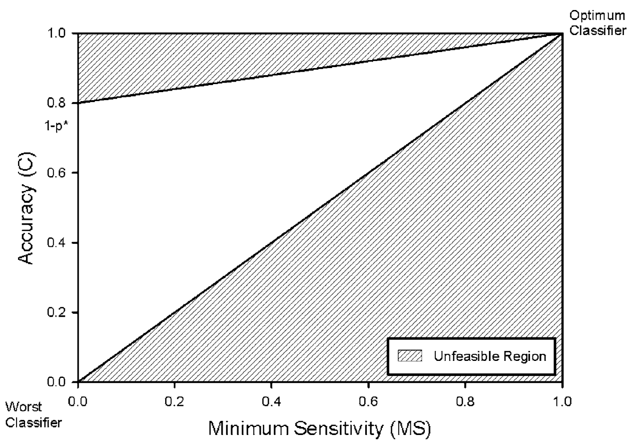


Fig. 1 Unfeasible region in two dimensional (MS, C) space

where $I(\cdot)$ is the zero-one loss function, $\mathbf{y}_n$ is the desired output for pattern n and N is the total number of patterns in the dataset. A good classifier tries to achieve the highest possible C in a given problem. However, the C measure is a discontinuous function, which makes convergence more difficult in neural network optimization.

Thus, instead of C, it is the continuous function given in expression 2 that is considered, which is also called Entropy (E). The advantage of using the error function $E(g, \boldsymbol{\theta})$ instead of C is that this is a continuous function, which makes the convergence more robust.

As a first objective, a strictly decreasing transformation of the $E(g, \boldsymbol{\theta})$ is proposed as the fitness measure to maximize:

$$A(g) = \frac{1}{1 + E(g, \boldsymbol{\theta})}, 0 < A(g) \leq 1,$$

where g is the multivaluated function:

$$g(\mathbf{x}, \boldsymbol{\theta}) = (g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_J(\mathbf{x}, \boldsymbol{\theta}_J)).$$

The second objective to maximize is the MS of the classifier, that is, maximizing the lowest percentage of examples correctly predicted as belonging to each class with respect to the total number of examples in the corresponding class.

3.4 Memetic pareto algorithm

A MOEA is constructed with a local search algorithm, called Memetic Pareto Differential Evolutionary Neural Network (MPDENN), that tries to move the classifier population towards the optimum classifier located at the (1,1) point in the (MS, C) space (see Fig. 1). The MOEA proposed is based on the PDE [2] and the local search algorithm is the Improved Resilient Backpropagation–*iRprop*$^+$ [27].

The Memetic Multiobjective Evolutionary Neural Network algorithm used in this work considers a fully specified ANN as an individual and it evolves architectures and connection weights simultaneously. The ANNs are represented using an object-oriented approach and the algorithm deals directly with the ANN phenotype. Each connection is specified by a binary value, which indicates whether the connection exists, and a real value representing its weight.

The MPDENN is based on the algorithm described in [11]. In MPDENN, local search does not apply to all offspring to be added to the population. Instead, the most representative offspring of the population are optimized throughout several generations. The pseudocode of MPDENN is shown in Fig. 2.

The algorithm starts generating a random population P_0 of size M. The population is sorted according to the non-domination concept (one individual dominates another if it

Fig. 2 MPDENN algorithm pseudocode

```
1:  Create a random initial population
2:  while Stop condition is not met do
3:    Evaluate population
4:    Adjust the size of the population
5:    while The population is not complete do
6:      Select parents
7:      Cross parents
8:      Mutate the child
9:      Evaluate the child
10:     Add the child in the population according to dominance relationships with the main
        parent
11:   end while
12:   if Local search in this generation then
13:     if Number of individuals of the first Pareto front of $P_k < num$ then
14:       Apply iRprop$^+$ to the individuals of the first Pareto front
15:     else
16:       Generate $num$ cluster in the first Pareto front using K-means
17:       Apply iRprop$^+$ to the $num$ centers
18:     end if
19:   end if
20: end while
```

is better or equal in all objectives and better in at least one of them). Dominated individuals are removed from the population. Then the population is adjusted until its size is between 3 and a half the maximum size by adding dominated individuals or deleting individuals according to their distance from the nearest neighbour respectively. After that, the population is completed with new offspring generated from three randomly selected individuals in the population. The child is generated applying the crossover operator to the three parents (α_1, α_2 and α_3). The resultant child is a perturbation of the main parent (α_1). This perturbation occurs with a probability p_c for each neuron. It may be: structural, according to expression (1), so that neurons are removed or added to the hidden layer; or parametric, according to expression (2) (for the hidden layer); or (3) (for the output layer), where the weight of the main parent (α_1) is modified by its difference from the weights of the secondary parents (α_2 and α_3), and multiplied by a random variable with normal distribution, $N(0,1)$.

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if\left(\rho_h^{\alpha_1}+\right)N(0,1)\left(\rho_h^{\alpha_2}-\rho_h^{\alpha_3}\right) \geq 0.5 \\ 0 & otherwise \end{cases}, \tag{3}$$

$$w_{ih}^{child} \leftarrow w_{ih}^{\alpha_1} + N(0,1)\left(w_{ih}^{\alpha_2} - w_{ih}^{\alpha_3}\right), \tag{4}$$

$$w_{ho}^{child} \leftarrow w_{ho}^{\alpha_1} + N(0,1)\left(w_{ho}^{\alpha_2} - w_{ho}^{\alpha_3}\right), \tag{5}$$

where $\rho_h^{\alpha_1}$, $\rho_h^{\alpha_2}$ and $\rho_h^{\alpha_3}$ represent whether or not the hidden neuron h is in the parents α_1, α_2 and α_3, respectively; $w_{ih}^{\alpha_1}$ is the weight between the input neuron i and hidden neuron h in the parent α_1 and $w_{ho}^{\alpha_1}$ is the weight between the hidden neuron h and output neuron o in the parent α_1.

Afterwards, the mutation operator is applied to the child. The mutation operator consists in adding or deleting neurons in the hidden layer depending on a p_m probability for each of them. Taking into account the maximum number of hidden neurons that may exist in an individual in a specific problem, the probability will be used the same number of times as the number of neurons that are found in the classifier. If the neuron exists, it is deleted, but if it does not exist, then it is created and the weights are established randomly, according to expression (6).

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if\, \rho_h^{child} = 0 \\ 0 & otherwise \end{cases}. \tag{6}$$

Finally, the child is added to the population according to dominance relationships with the main parent, that is, the child is added if it dominates the main parent α_1, if there is not dominance relationship with him or if it is the best child of the M rejected children (where M is the population size). In some generations, depending on the size of the first Pareto front, local search is applied to all individuals in the first Pareto front or the most representative individuals in this front (obtained by the K-means algorithm [32]). Local search is explained in Sect. 3.5.

Figure 3 shows the framework of the algorithm proposed in this paper.

3.5 Local search algorithm

Evolutionary Algorithms (EAs) are improved by the incorporation of local search procedures throughout their evolution. Some studies that were carried out on the convergence process of a genetic algorithm in a concrete optimization problem, show that although the genetic algorithm quickly finds good solutions to the problem, it needs many generations to reach the optimum solution and it has great difficulties in finding the best solution when it is in a region near a global optimum. It is well-known that certain local procedures are able to find the local optimum when the search is carried out in a small region of the

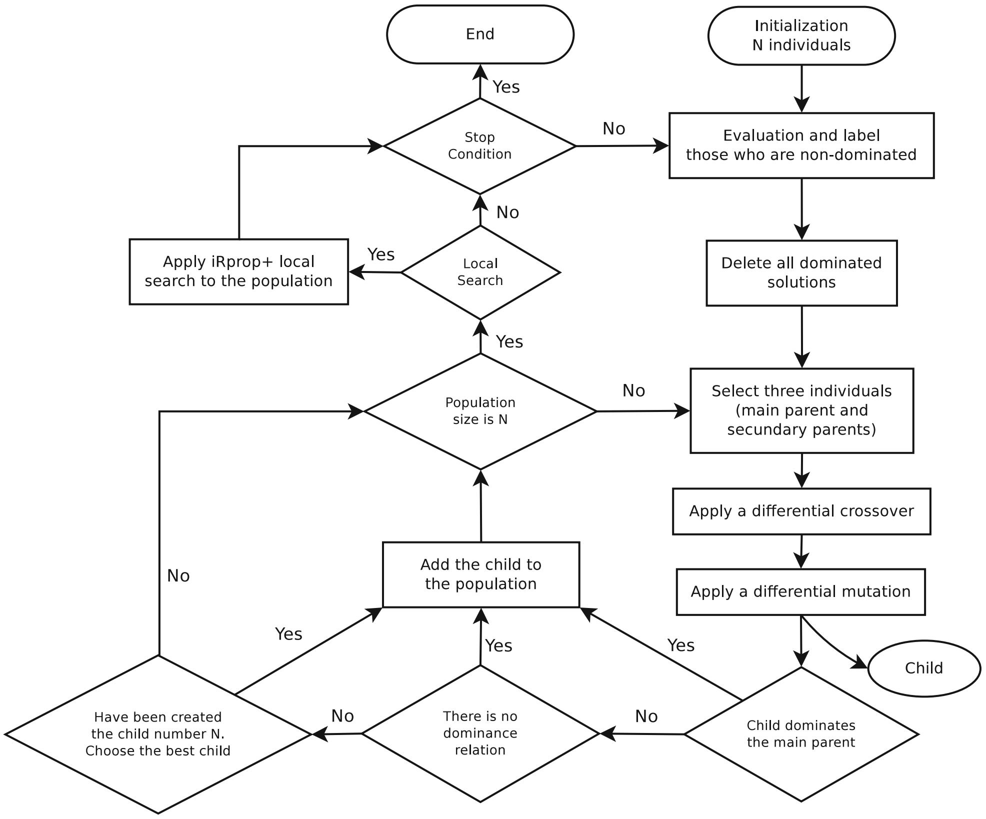


Fig. 3 Framework for MPDENN

space. Therefore, in the combination of EA and local procedures, EA is going to carry out a global search inside the solution space, locating ANNs near the global optimum, and the local procedure will quickly and efficiently find the best solution. This type of algorithm is called the Memetic or Hybrid Algorithm [34].

Many MOEAs use local optimizers to fine-tune ANN weights. This is called "lifetime learning" and it consists in updating each individual regarding the approximation error. In addition, the weights modified during lifetime learning are encoded back to the chromosome, which is known as the Lamarckian type of inheritance. This procedure has a high computational cost which is something that should be avoided; this is the reason behind the following proposal:

The local search algorithm is only applied in three generations of evolution once the population is completed (the first initially, the second in the middle and the third at the end). Thus, local search is not applied to those offspring who are rejected. Local search does not apply to all individuals, only to the most representative. The process for selecting these individuals is as follows: if the number of individuals in the first Pareto front is lower than or equal to the desired number of clusters (*num*), a local search is carried out on all individuals in the first front without needing to apply K-means [32]. But, if the number of individuals in the first front is greater than *num*, the K-means is applied to the first front to get the most representative *num* individuals, who will then be the object of a local search.

This local search will improve the obtained Pareto front in only one objective, specifically in the direction of the objective that tries to minimize the classification error.

In our opinion, one of the best techniques in terms of convergence speed, accuracy and robustness with respect to its parameters is the *Rprop* (resilient Backpropagation) algorithm [28], although classic algorithms like Backpropagation are also frequently used. *Rprop* is a learning heuristic for supervised learning in artificial neural networks. Like the Manhattan update rule, *Rprop* takes into account only the sign of the partial derivative in all patterns (not the magnitude), and acts independently on each weight. For each weight, if there is a sign of change in the partial derivative of the total error function compared to the previous iteration, the update value for that weight is multiplied by a factor η^-. If the last iteration produces the same sign, the update value is multiplied by a factor η^+. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own updated value, in the opposite direction of that weight's partial derivative, so as to minimize the total error function.

The *improved Rprop–iRprop*$^+$ algorithm has recently been proposed. It applies a backtracking strategy (i.e. it decides whether to take a step back along a weight direction or not by means of a heuristic, "+" being the incorporation of backtracking). The improvement is based on the consideration that a change in the partial derivative sign implies that the algorithm has jumped over a local minimum, but does not indicate whether the weight update has caused an increase or a decrease. The idea behind modifying *Rprop*$^+$ is to make the step reversal dependent on the evolution of the error. These considerations lead to the rule that those weight updates that have caused changes to the signs of their corresponding partial derivatives are reverted, but only in the case of an increase in error. It has been shown in several benchmark problems [27] that the *improved Rprop* with backtracking exhibits consistently better performance than the original *Rprop* algorithm, and that is why it is used. The *iRprop*$^+$ local optimizer has been adapted to (1) the softmax activation function, and (2) the cross-entropy error function, modifying the gradient function for weights in the hidden and output layers.

4 Experiments

To analyze the robustness of the proposed methodology, the experimental design considers two complex problems on predictive microbiology for describing the behavior of pathogen and spoilage micro-organisms under a given set of environmental conditions. The objective is to determine the conditions under which these microorganisms can be classified as G/GHP/GLP/NG, and to create a neural classifier for this purpose. The problems that have been specifically taken into consideration are the pathogen growth limits of *Staphylococcus aureus* and *Shigella flexneri*.

In all the experiments, the population size for MPDENN is established at $M = 25$. The crossover probability is 0.8 and the mutation probability is 0.1. For *iRprop*$^+$, the adopted parameters are $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_0 = 0.0125$ (the initial value of the Δ_{ij}), $\Delta_{\min} = 0$, $\Delta_{\max} = 50$ and $Epochs = 25$, see [28] for the *iRprop*$^+$ parameter description. The optimization process is applied 3 times during the execution (every 33.33% of generations) and uses $num = 5$ cluster in the clustering algorithm. To start processing data, each one of the input variables was scaled in the interval [−1.0, 1.0] to avoid saturation of the signal.

Table 1 shows the features for each dataset. The total number of instances or patterns in each dataset appears, as well as the number of instances in training and testing sets, the number of input variables, the total number of instances per class and the p^* value (the minimum of prior estimated probabilities). A fractional factorial design matrix form was used. This design is normally used in predictive microbiology (in [45] the fractional factorial design for *Staphylococcus aureus* is presented and in [48], for *Shighella flexneri*). For example, to determine the data belonging to the training set and the generalization set of the *S. flexneri* dataset, the conditions of sodium chloride and sodium nitrite were selected alternately at the same level of temperature and pH, as shown in Table 2. The objective of this selection was to define the training set data that actually represents the border areas in order to obtain a better fit.

During the experiment, models were trained using the fitness function $A(g)$ (based on E, see Sect. 3.3) and MS as objective functions, but when validated, C was used. $A(g)$ was used instead of C in training because C is a discontinuous function, which makes convergence more difficult in optimization.

The results obtained with the MPDENN algorithm have been compared to those obtained by PDE, MPANN and MPENSGA2 (Memetic Pareto Evolutionary approach based on the NSGA2 evolutionary algorithm) algorithms. The PDE [2] and MPANN [1] algorithms have been developed by H. Abbass and are a benchmark in the evolution differential with neural networks. The MPENSGA2 algorithm has been developed by Fernández et al. [12] and

Table 2 Fractional factorial design for *S. flexneri*

SN (ppm) SC(%)	0	50	100	200	1,000
0.5	○	♦	○	♦	○
2.5	♦	○	♦	○	♦
4.0	○	♦	○	♦	○

SN sodium nitrite, *SC* sodium chloride

○ = training patterns, ♦ = generalization patterns

Table 1 Datasets characteristics

Dataset	#Patterns	#Training patterns	#Test patterns	#Input variables	#Patterns per class	p^*
S. aureus	287	146	141	3	(117, 45, 12, 113)	0.0418
S. flexneri	123	76	47	5	(39, 8, 7, 69)	0.0569

is a memetic version of the NSGA2 algorithm (Non-dominated Sort Genetic Algorithm 2, [10]).

Once the Pareto front is built, two methodologies are considered in order to build a neural network model which then includes the information about the models within it. These are called MethodName-E and MethodName-MS (where MethodName takes the values MPDENN, PDE, MPANN and MPENSGA2). These methodologies provide single models that can be compared to other classification methods found in the literature. The process followed in these methodologies is the following: once the first Pareto front is calculated using training set patterns, the best individual belonging to the Pareto front on *E* (*EI*) is chosen for MethodName-E, and the best individual in terms of *MS* (*MSI*) is selected for MethodName-MS. Once this is done, the values of *C* and *MS* are obtained by testing the *EI* and *MSI* individual models. Therefore an individual $EI_G = (MS_G^{EI}, C_G^{EI})$ is obtained along with an individual $MSI_G = (MS_G^{MSI}, C_G^{MSI})$. This is repeated 30 times and then estimations are made of the average and standard deviation obtained from the individuals $\overline{EI}_G = (\overline{MS}_G^{EI}, \overline{C}_G^{EI})$ and $\overline{MSI}_G = (\overline{MS}_G^{MSI}, \overline{C}_G^{MSI})$. The first expression is the average obtained taking *E* into account as the primary objective, and the second one is obtained by taking *MS* into account as the primary objective. So, the opposite extremes of the Pareto front are taken in each of the executions. Hence, the first procedure is called MethodName-E and the second MethodName-MS.

The runtime of each algorithm will also be measured during the evolutionary process. This measure will allow us to estimate the computational cost and efficiency of each of the algorithms.

Therefore, to evaluate the goodness and performance of the algorithms, the results of *C*, *MS*, *RMSE* (Root Mean Squared Error) and Cohen's kappa [5] are analyzed with respect to generalization. Moreover, the runtime of the training process will also be considered.

The following subsections describe the two real problems selected for predictive microbiology.

4.1 Staphylococcus aureus

Staphylococcus aureus has been recognized as an indicator of deficient food and processing hygiene and is a major cause of food gastroenteritis worldwide [41]. A fractional factorial design was followed in order to ascertain the growth limits of *Staphylococcus aureus* [45] by carefully choosing a subset (fraction) of the experimental runs of a full factorial design in order to reduce experimental time and resources. The selection was based on delimiting the levels of the environmental factors studied for the growth/no-growth domain of *S. aureus* [43]. Since no growth was detected at 7.5°C or below, data were collected at 8°, 10°, 13°, 16° and 19°C, at pH levels from 4.5 to 7.5 (0.5 intervals) and at 19a_w levels (from 0.856 to 0.999 at regular intervals). The initial dataset (287 conditions) was divided into two parts: model data (training set, 146 conditions covering the extreme domain of the model) and validation data (generalization set, 141 conditions within the interpolation region of the model). Among the different conditions, there were 117 cases of G, 45 cases of GHP, 12 cases of GLP and 113 cases of NG. The purpose of this selection was to define a dataset for model data focused on the extreme regions of the growth/no-growth domain that the boundary zones actually represent. In this study, the number of replicates per condition ($n = 30$) increased compared to other studies obtaining the growth/no-growth transition.

4.2 Shigella flexneri

Shighella flexneri is an important causative agent of gastrointestinal illness [48]. An incomplete factorial design was used to assess the effects of temperature (12°, 15°, 19°, 28°, 37°C), initial pH (5.5, 6.0, 6.5, 7.0, 7.5), sodium chloride (0.5, 2.5, 4.0%) and sodium nitrite (0, 50, 100, 200, 1,000 ppm). Data are obtained from 375 cultures, representing 123 variable combinations. The number of replicate cultures tested for each variable combination is given in Table 2 in the cited paper [48]. This data was used to derive the models to predict the anaerobic growth of *Shighella flexneri* as a function of temperature, sodium chloride and sodium nitrite concentrations and initial pH. The growth kinetics data for each variable combination are summarized in the cited Table 2. Growth of *Shighella flexneri* was not observed under the conditions corresponding to 40 of the variable combinations studied. An additional 15 variable combinations resulted in environments where some of the replicate cultures grew, while others did not produce any; these are listed in Table 3 in the cited paper. Among the different conditions, there were 39 cases of G, 8 cases of GHP, 7 cases of GLP and 69 cases of NG.

5 Results

Table 3 presents the values of average and Standard Deviation (SD) for testing *C*, testing *MS*, testing *RMSE*, testing Cohen's kappa and training runtime in 30 runs of all the experiments performed. It can be seen that the MPDENN algorithm produces good results in *Staphylococcus aureus* and similar results to a better algorithm (MPENSGA2) in *Shighella flexneri*, but with less runtime.

In fact, from a purely descriptive point of view, the MPDENN algorithm is the fastest algorithm and it obtains

Table 3 Statistical results for MPDENN, PDE, MPANN and MPENSGA2

Methodology	*C* (%) Mean ± SD	*MS* (%) Mean ± SD	*RMSE* Mean ± SD	Cohen's kappa Mean ± SD	Runtime (s) Mean ± SD
Staphylococcus aureus					
MPDENN-E	*73.03 ± 1.53*	0.00 ± 0.00	*0.30 ± 0.01*	*0.56 ± 0.03*	**48.72 ± 3.70**
MPDENN-MS	55.01 ± 7.97	**21.81 ± 14.02**	0.39 ± 0.03	0.37 ± 0.09	**48.72 ± 3.70**
PDE-E	71.27 ± 2.50	0.00 ± 0.00	0.32 ± 0.01	0.53 ± 0.03	*49.34 ± 5.24*
PDE-MS	53.52 ± 6.57	*20.59 ± 10.98*	0.38 ± 0.02	0.39 ± 0.05	*49.34 ± 5.24*
MPANN-E	**74.44 ± 1.41**	0.00 ± 0.00	**0.29 ± 0.01**	**0.59 ± 0.02**	309.14 ± 11.51
MPANN-MS	66.71 ± 8.87	9.36 ± 10.82	0.33 ± 0.04	0.49 ± 0.11	309.14 ± 11.51
MPENSGA2-E	69.65 ± 2.92	0.00 ± 0.00	0.35 ± 0.01	0.49 ± 0.05	158.92 ± 36.13
MPENSGA2-MS	56.71 ± 5.09	20.00 ± 10.50	0.39 ± 0.01	0.39 ± 0.06	158.92 ± 36.13
Shighella flexneri					
MPDENN-E	*87.02 ± 1.16*	0.00 ± 0.00	*0.22 ± 0.01*	*0.76 ± 0.02*	*249.12 ± 13.99*
MPDENN-MS	86.59 ± 2.80	2.22 ± 8.45	**0.21 ± 0.01**	*0.76 ± 0.02*	*249.12 ± 13.99*
PDE-E	83.75 ± 1.04	0.00 ± 0.00	*0.22 ± 0.01*	0.73 ± 0.04	**234.90 ± 17.93**
PDE-MS	84.04 ± 3.77	**9.77 ± 14.82**	*0.22 ± 0.02*	0.73 ± 0.04	**234.90 ± 17.93**
MPANN-E	*87.02 ± 0.85*	0.00 ± 0.00	0.23 ± 0.01	*0.76 ± 0.02*	1956.93 ± 45.28
MPANN-MS	*87.02 ± 0.85*	0.00 ± 0.00	0.23 ± 0.01	*0.76 ± 0.02*	1956.93 ± 45.28
MPENSGA2-E	**88.63 ± 1.63**	*5.55 ± 12.63*	**0.21 ± 0.01**	**0.79 ± 0.03**	1681.41 ± 203.78
MPENSGA2-MS	**88.63 ± 1.63**	*5.55 ± 12.63*	**0.21 ± 0.01**	**0.79 ± 0.03**	1681.41 ± 203.78

The best result is in bold face and the second best result in *italic*

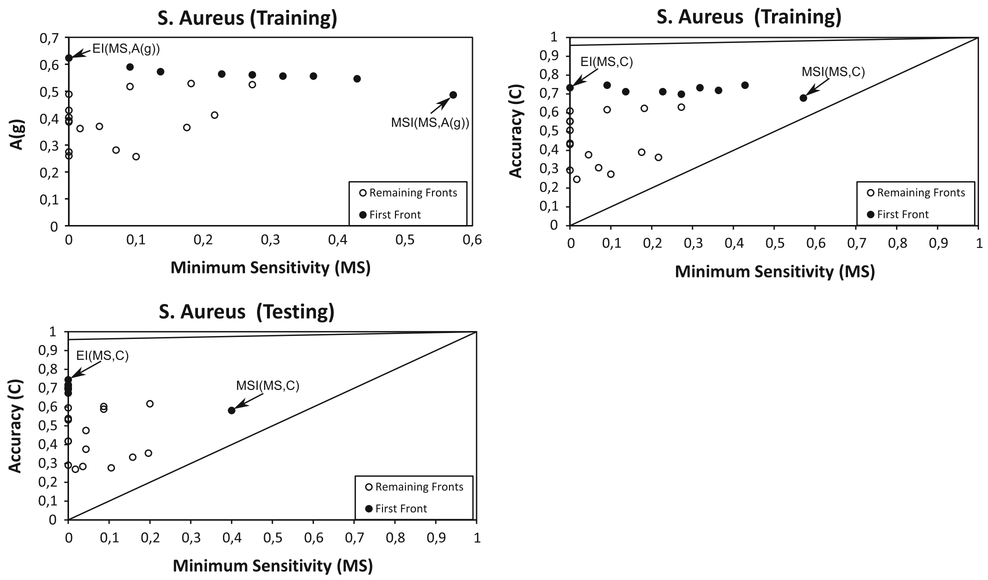


Fig. 4 Pareto front in training (*MS*, *A*(*g*)) and (*MS*, *C*) associated values in training and testing for *S. aureus* dataset in one specific run

best value of MS in *S. aureus* (getting the second best result in the other metrics). In *S. flexneri*, the MPDENN algorithm is the second best algorithm, because it obtains the second best result in four of the five metrics. For all these reasons, we can say that the algorithm MPDENN is competitive with other algorithms in accuracy and much better at runtime.

Figures 4 and 5 show the graphical results obtained for the MPDENN algorithm for the *Staphylococcus aureus* and *Shigella flexneri* datasets, respectively, in the training (*MS*, *A*(*g*)) and the test (*MS*, *C*) spaces. For the (*MS*, *A*(*g*)) space, the Pareto front is selected for one specific run output of the 30 done for each dataset, concretely the execution that presents the best *E* individual in training, where *A*(*g*) and *MS* are the objectives that guide MPDENN. The (*MS*, *C*) testing graphs show *MS* and *C* values throughout the testing set for the individuals who are reflected in the (*MS*, *A*(*g*)) training graphs. Observe that the

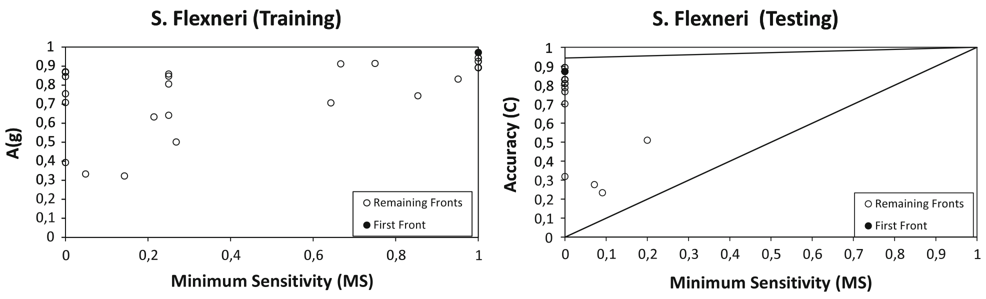


Fig. 5 Pareto front in training (MS, $A(g)$) and (MS, C) associated values in testing for for *S. flexneri* dataset in one specific run

Table 4 Probability expression of the best MLP model for *S. aureus*

Best model for *S. aureus* in (MS, C) space considering MS (MPDENN-MS)

$$p_G(\mathbf{x},\theta) = \frac{e^{f_1(\mathbf{x},\theta)}}{1+\sum_{i=1}^{3} e^{f_i(\mathbf{x},\theta)}} \qquad p_{GHP}(\mathbf{x},\theta) = \frac{e^{f_2(\mathbf{x},\theta)}}{1+\sum_{i=1}^{3} e^{f_i(\mathbf{x},\theta)}}$$

$$p_{GLP}(\mathbf{x},\theta) = \frac{e^{f_3(\mathbf{x},\theta)}}{1+\sum_{i=1}^{3} e^{f_i(\mathbf{x},\theta)}} \qquad p_{NG}(\mathbf{x},\theta) = \frac{1}{1+\sum_{i=1}^{3} e^{f_i(\mathbf{x},\theta)}}$$

$f_1(\mathbf{x},\theta) = 2.08 - 5.00\ MLP_1 - 3.98\ MLP_2 + 5.00\ MLP_3 - 3.45\ MLP_4 - 0.78\ MLP_5$
$+ 3.86\ MLP_6 - 1.41\ MLP_7 - 4.75\ MLP_8 - 0.88\ MLP_9 + 1.68\ MLP_{10}$

$f_2(\mathbf{x},\theta) = 2.82 - 2.12\ MLP_1 - 2.70\ MLP_2 + 5.00\ MLP_3 - 4.88\ MLP_4 + 3.63\ MLP_5$
$- 5.00\ MLP_6 - 4.04\ MLP_7 + 3.17\ MLP_8 - 1.68\ MLP_9 + 4.99\ MLP_{10}$

$f_3(\mathbf{x},\theta) = - 5.54 + 0.15\ MLP_1 - 5.00\ MLP_2 - 5.00\ MLP_3 + 3.38\ MLP_4 + 4.99\ MLP_5$
$- 2.18\ MLP_6 + 5.00\ MLP_7 + 4.34\ MLP_8 + 3.73\ MLP_9 + 0.48\ MLP_{10}$

$f_4(\mathbf{x},\theta) = 0$

$MLP_1 = \sigma(-0.34 - 0.72T^\star - 0.51pH^\star - 0.72a_w^\star)$

$MLP_2 = \sigma(-1.00 - 0.65T^\star - 0.98pH^\star - 1.00a_w^\star)$

$MLP_3 = \sigma(0.21 + 0.35T^\star - 1.00pH^\star + 1.00a_w^\star)$

$MLP_4 = \sigma(0.99 + 0.25T^\star - 0.96pH^\star - 0.42a_w^\star)$

$MLP_5 = \sigma(-0.41 - 0.59T^\star + 1.00pH^\star - 0.99a_w^\star)$

$MLP_6 = \sigma(0.97 + 1.00T^\star - 0.76pH^\star + 1.00a_w^\star)$

$MLP_7 = \sigma(-1.00 - 1.00T^\star + 0.80pH^\star + 0.72a_w^\star)$

$MLP_8 = \sigma(-0.48 - 0.85T^\star - 1.00pH^\star - 1.00a_w^\star)$

$MLP_9 = \sigma(-0.64 - 1.00T^\star - 0.34pH^\star - 0.10a_w^\star)$

$MLP_{10} = \sigma(0.27 + 0.08T^\star + 0.82pH^\star + 1.00a_w^\star)$

$T^\star, pH^\star, a_w^\star \in [-1, 1]$

$C_T = 67.81\%;\ \ MS_T = 57.14\%\ \ RMSE_T = 0.35\ Kappa_T = 0.55$

$C_G = 58.16\%;\ MS_G = 0.40\%\ RMSE_G = 0.38\ Kappa_G = 0.41$

#neurons = 10;#effective connections = 73

$$CM_T = \begin{pmatrix} 47 & 11 & 2 & 0 \\ 3 & 13 & 4 & 2 \\ 0 & 3 & 4 & 0 \\ 0 & 6 & 16 & 35 \end{pmatrix};\quad CM_G = \begin{pmatrix} 41 & 13 & 3 & 0 \\ 5 & 10 & 6 & 2 \\ 0 & 1 & 2 & 2 \\ 2 & 8 & 17 & 29 \end{pmatrix}$$

Performance of this model: Accuracy in the training set (C_T), Accuracy in the generalization set (C_G), Minimum sensitivity in the training set (MS_T), Minimum sensitivity in the generalization set (MS_G), $RMSE$ in the training set ($RMSE_T$), $RMSE$ in the generalization set ($RMSE_G$), Cohen's kappa in the training set ($Kappa_T$) and Cohen's kappa in the generalization set ($Kappa_G$). Confusion Matrix for the training set (CM_T) and for the generalization set (CM_G)

(MS, C) values do not form Pareto fronts in testing, and the individuals that had been in the first Pareto front in the training graphics may now find themselves located in a worse region in the space. In general the structure of a Pareto front in training is not maintained in testing. Sometimes it is very difficult to obtain classifiers with a high percentage of classification and a high percentage of sensitivity, and for this reason some fronts have very few individuals. Finally, so that the reader can observe the training classification error versus MS for the MPDENN algorithm, a graph example has been included in Fig. 4 for the *S. aureus* dataset. This figure shows the Pareto front in training in the (MS, $A(g)$) space and the associated values in the (MS, C) space in training and testing during one specific run.

Table 4 shows the best model for *S. aureus* in the (MS, C) space considering the best model under the methodology MPDENN-MS. Also included in Table 4 are

Table 5 *p*-Values of the Snedecor's F ANOVA I or K–W test and ranking of means of the Tukey Statistical multiple comparison tests or M–W pair test

	Staphylococcus aureus	*Shigella flexneri*
A priori test: Snedecor's F ANOVA I or K–W		
C	0.000(*)	0.000(°)
MS	0.000(°)	0.000(°)
RMSE	0.000(*)	0.008(*)
Cohen's kappa	0.000(*)	0.000(°)
Runtime	0.000(*)	0.000(*)
Post-hoc test: Tukey or M–W		
Means	$\mu_{MPAE} \geq \mu_{ME} \geq \mu_{PE} \geq$	$\mu_{MPEE} \geq \mu_{MPAE}$;
Ranking of the *C*	$\geq \mu_{MPEE} \geq \mu_{MPAMS} >$	$\mu_{MPEE} \geq \mu_{ME}$;
	$> \mu_{MPEMS} \geq \mu_{MMS} \geq \mu_{PMS}$;	$\mu_{MPEE} > \mu_{PE}$;
	$\mu_{MPAE} > \mu_{MPEE}$;	$\mu_{ME} > \mu_{PE}$
	$\mu_{PE} > \mu_{MPAMS}$	
Means	$\mu_{MMS} \geq \mu_{PMS}$;	$\mu_{PMS} \geq \mu_{MPEMS}$;
Ranking of the *MS*	$\mu_{MMS} \geq \mu_{MPEMS}$;	$\mu_{PMS} \geq \mu_{MMS}$;
	$\mu_{MMS} > \mu_{MPAMS}$	$\mu_{MPEMS} \geq \mu_{MMS}$
Means	$\mu_{MPAE} \leq \mu_{ME} < \mu_{PE} \leq$	$\mu_{MPEE} \leq \mu_{MPEMS} \leq \mu_{PE} \leq$
Ranking of the *RMSE*	$\leq \mu_{MPAMS} < \mu_{MPEE} < \mu_{PMS} \leq$	$\leq \mu_{ME} \leq \mu_{MMS} \leq \mu_{PMS} \leq$
	$\leq \mu_{MMS} \leq \mu_{MPEMS}$;	$\leq \mu_{MPAE} \leq \mu_{MPAMS}$;
	$\mu_{PMS} < \mu_{MPEMS}$	$\mu_{MPEMS} < \mu_{PMS}$
Means	$\mu_{MPAE} \geq \mu_{ME} \geq \mu_{PE} \geq$	$\mu_{MPEE} \geq \mu_{MPAE}$;
Ranking of the Cohen's kappa	$\geq \mu_{MPAMS} \geq \mu_{MPEE} > \mu_{PMS} \geq$	$\mu_{MPEE} \geq \mu_{ME}$;
	$\geq \mu_{MPEMS} \geq \mu_{MMS}$;	$\mu_{MPEE} \geq \mu_{PE}$;
	$\mu_{MPAE} > \mu_{PE}$;	$\mu_{MPAE} \geq \mu_{ME}$;
	$\mu_{ME} > \mu_{MPAMS}$	$\mu_{MPAE} \geq \mu_{PE}$;
		$\mu_{ME} \geq \mu_{PE}$
Means	$\mu_{MPDENN} \leq \mu_{PDE} <$	$\mu_{PDE} \leq \mu_{MPDENN} <$
Ranking of the runtime	$< \mu_{MPENSGA2} < \mu_{MPANN}$	$< \mu_{MPENSGA2} < \mu_{MPANN}$

(*) Snedecor's F ANOVA I

(°) Kruskal–Walis Test

$\mu_A \geq \mu_B$: methodology *A* yields better results in mean than methodology *B*, but the difference are not significant; and $\mu_A > \mu_B$: methodology *A* yields better results in mean than methodology *B* with significant differences. The binary relations $>$ and $\geq$ are not transitive

C, *MS*, *RMSE*, Cohen's kappa and confusion matrices, in training and generalization. The model structure is complex with 10 MLP nodes in the hidden layer and 73 coefficients or connections. In addition, the three environmental variables are present in all MLP basic functions, i.e. the three environmental factors are needed to determine the growth/no-growth boundary zones.

In order to determine the best methodology for training MLP neural networks (in the sense of its influence on *C*, *MS*, *RMSE* and Cohen's kappa in the test dataset and, also, the runtime in training), an ANalysis Of the VAriance of one factor (ANOVA I) statistical method or the non parametric Kruskal–Wallis (K–W) tests were carried out, based on a previous Normality Kolmogorov–Smirnov (K–S) test of generalization *C*, *RMSE*, Cohen's kappa and training runtime values. The levels of the factor represent the methodology applied and they are the following: MPDENN-E (ME), MPDENN-MS (MMS), PDE-E (PE), PDE-MS (PMS), MPANN-E (MPAE), MPANN-MS (MPAMS), MPENSGA2-E (MPEE) and MPENSGA2-MS (MPEMS). It is not possible to perform these tests for *MS*, because some methodologies in the two data sets present zero values. The results of the ANOVA or K–W analysis for *C*, *RMSE*, Cohen's kappa and training runtime show that, the effect, in mean, of the methodologies is statistically significant at a level of 5% for both datasets. Table 5 presents the results obtained using the ANOVA or K–W test.

Therefore, a post hoc multiple comparison test is performed of the mean *C*, *MS*, *RMSE*, Cohen's kappa and training runtime obtained in order to establish an order between the different methodologies. First, a Levene test [33] is carried out for evaluating the equality of variances. Then, if the variances are equal (the normality hypothesis is satisfied), a Tukey test [33] is performed. In other cases, a pair-wise Mann–Whitney test in applied. Table 5 presents the results obtained using the post-hoc Tukey test or the Mann–Whitney (M–W) test. The mean difference is significant with p-value $= 0.05$.

If we analyze the test results for *C*, we can observe that the MPDENN-E methodology obtains results that are, in mean, similar to the obtained with the best methodology

(MPANN-E in *S. aureus* and MPENSGA2-E in *S. flexneri*), for a level of signification of 5%. The results for *MS* show that the MPDENN-MS methodology obtains a performance that is better than the rest of methodologies in *S. aureus* and similar to the obtained with the best methodology in *S. flexneri* (PDE-MS) (for a level of signification of 5%). For both *RMSE* and Cohen's kappa, the results show that the MPDENN-E methodology has no significant differences with the best methodology (MPANN-E in *S. aureus* and MPENSGA2-E in *S. flexneri*), with a significance level of 5%. The results for runtime show that the time of MPDENN and PDE are similar and significantly less than the time of MPANN and MPENSGA2, in both datasets.

6 Conclusions

This paper studies the application of a memetic algorithm based on differential evolution in the resolution of multi-classification problems in predictive microbiology. This algorithm is meant to obtain competitive results in the Minimum sensitivity and Accuracy space (*MS*, *C*), and also decrease the computational cost to reduce the runtime. We have proposed applying local search to the most representative individuals in the population, selected through clustering techniques, to optimize the most promising individuals. This memetic algorithm has obtained similar results (no significant differences) in all the metrics used with respect to the best algorithm (MPANN in *S. aureus* and MPENSGA2 in *S. flexneri*), but MPDENN runs faster in both datasets. For this reason we recommend using the MPDENN to address multiclass problems when good results are desired in a reduced runtime.

The existence of classes GHP and GLP are clearly justified since in certain zones of the model domain, microbial responses are more variable and therefore, a classification into G or NG cannot be established. In these conditions which approach microbial growth limits, the probability of belonging to class GHP and GLP is higher.

The proposed methodology can help predictive modelers to better define the growth boundaries of microorganisms and to model the microbial variability associated to these conditions.

Some suggestions for future research are the following: to use the MPDENN algorithm on other predictive microbiology datasets, such as *Listeria monocytogenes* [23] or *Escherichia coli R31* [47], to adapt the algorithm to unbalanced datasets by resampling techniques and to use more metrics to determine the performance of classifiers obtained.

Acknowledgments This work has been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). Manuel Cruz-Ramírez and Francisco Fernndez-Navarro's research have been funded by the "Junta de Andalucía" Predoctoral Program, grant reference P08-TIC-3745. Javier Sánchez-Monedero's research has been funded by the "Junta de Andalucía" Ph. D. Student Program.

References

1. Abbass HA (2001) A memetic pareto evolutionary approach to artificial neural networks. In: Brooks M, Corbet D, Stumptner M (eds) AI2001. LNAI 2256, Springer, pp 1–12
2. Abbass HA, Sarker R, Newton C (2001) Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the 2001 congress on evolutionary computation, vol 2. Seoul, South Korea
3. Baranyi J, Roberts T (1995) Mathematics of predictive food microbiology. Int J Food Microbiol 26:199–218
4. Baranyi J, Ross T, McMeekin TA, Roberts TA (1996) Effects of parameterization on the performance of empirical models used in predictive microbiology. Int J Food Microbiol 13:83–91
5. Ben-David A (2007) A lot of randomness is hiding in accuracy. Eng Appl Artif Intell 20(7):875–885
6. Bishop CM (1996) Neural networks for pattern recognition. Oxford University Press, Oxford
7. Buchanan RLB (1993) Expansion of response surface analysis for the growth of *Escherichia coli* O157:H7 to include sodium nitrite as a variable. Int J Food Microbiol 23:317–332
8. Coello Coello CA, Lamont GB, Van Veldhuizen DA (2007) Evolutionary algorithms for solving multi-objective problems, vol XXII, 2nd edn. Springer, Berlin
9. Davey K (1991) Applicability of the Davey, (linear Arrhenius) predictive model to the lag phase of microbial growth. J Appl Bacteriol 70:253–257
10. Deb K, Pratab A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga2. IEEE Trans Evol Comput 6(2):182–197
11. Fernández JC, Hervás C, Martínez FJ, Gutiérrez PA, Cruz M (2009) Memetic pareto differential evolution for designing artificial neural networks in multiclassification problems using cross-entropy versus sensitivity. In: Hybrid artificial intelligence systems, vol 5572. Springer, Berlin, pp 433–441
12. Fernández JC, Martínez FJ, Hervás C, Gutiérrez PA (2010) Sensitivity versus accuracy in multi-class problems using memetic pareto evolutionary neural networks. IEEE Trans Neural Netw 21(5):750–770
13. Fernández-Navarro F, Valero A, Hervás-Martínez C, Gutiérrez PA, García-Gimeno RM, Zurera-Cosano G (2010) Development of a multi-classification neural network model to determine the microbial growth/no growth interface. Int J Food Microbiol 141:203–212
14. García D, Ramos AJ, Sanchis V, Marin S (2009) Predicting mycotoxins in foods: a review. Food Microbiol 26:757–769
15. Garcia-Gimeno RM, Hervas-Martinez C, Barco-Alcala E, Zurera-Cosano G, Sanz-Tapia E (2003) An artificial neural network approach to *Escherichia coli* O157:H7 growth estimation. J Food Sci 68(2):639–645
16. Garcia-Gimeno RM, Hervas-Martinez C, Rodriguez-Perez R, Zurera-Cosano G (2005) Modelling the growth of *Leuconostoc mesenteroides* by artificial neural networks. Int J Food Microbiol 105(3):317–332
17. Garcia-Gimeno RM, Hervas-Martinez C, Sanz-Tapia E, Zurera-Cosano G (2002) Estimation of microbial growth parameters by

means of artificial neural networks. Food Sci Technol Int 8(2):73–80

18. Garcia-Gimeno RM, Hervas-Martinez C, de Siloniz MI (2002) Improving artificial neural networks with a pruning methodology and genetic algorithms for their application in microbial growth prediction in food. Int J Food Microbiol 72(1–2):19–30
19. Gelenbe E, Hussain KF (2002) Learning in the multiple class random neural network. IEEE Trans Neural Netw 13(6):1257–1267
20. Hajmeer M, Basheer I (2002) A probabilistic neural network approach for modeling and classification of bacterial growth/no-growth data. J Microbiol Methods 51:217–226
21. Hajmeer M, Basheer I, Najjar Y (1997) Computational neural networks for predictive microbiology: II. Application to microbe growth. Int J Food Microbiol 34:51–66
22. Hajmeer MN, Basheer IA (2003) A hybrid bayesian-neural network approach for probabilistic modeling of bacterial growth/no-growth interface. Int J Food Microbiol 82:233–243
23. Hajmeer MN, Basheer IA, Cliver DO (2006) Reliability-based estimation of the survival of listeria monocytogenes in chorizos. J Sci Food Agric 86(14):2337–2344
24. Hervas C, Zurera G, Garcia RM, Martinez JA (2001) Optimization of computational neural network for its application in the prediction of microbial growth in foods. Food Sci Technol Int 7(2):159–163
25. Hervas-Martinez C, Garcia-Gimeno RM, Martinez-Estudillo AC, Martinez-Estudillo FJ, Zurera-Cosano G (2006) Improving microbial growth prediction by product unit neural networks. J Food Sci 71(2):M31–M38
26. Hudson J (1992) Construction of and comparison between response surface analysis for *Aeromonas hydrophila* ATCC 7966 and food isolate under aerobic conditions. J Food Prot 55:968–972
27. Igel C, Hüsken M (2000) Improving the rprop learning algorithm. In: Proceedings of the second international ICSC symposium on neural computation (NC 2000). ICSC Academic Press, pp 115–121
28. Igel C, Hüsken M (2003) Empirical evaluation of the improved rprop learning algorithms. Neurocomputing 50(6):105–123
29. Jin Y, Sendhoff B (2008) Pareto-based multiobjective machine learning: an overview and case studies. IEEE Trans Syst Man Cybern C Appl Rev 38(3):397–415
30. Koutsoumanis K, Taoukis P, Nychas G (2005) Development of a safety mon itoring and assurance system for chilled food products. Int J Food Microbiol 100(1–3):253–260
31. Le Marc Y, Pin C, Baranyi J (2002) Methods to determine the growth domain in a multidimensional environmental space. Int J Food Microbiol 100(1–3):312
32. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. U. C. Berkeley Press, pp 281–297
33. Miller R (1996) Beyond ANOVA, basics of app. statistics. Chapman & Hall, London
34. Moscato P, Cotta C (2003) A gentle introduction to memetic algorithms. In: Handbook of Metaheuristics. International series in operations research and management science, vol 57. Springer, New York, pp 105–144
35. Natarajan BK (1991) Machine learning: a theoretical approach. Morgan Kaufmann, San Mateo. ISBN: 1558601481
36. Pitt R (1992) A descriptive model of mold growth and aflatoxin formation as affected by environmental conditions. J Food Prot 56:139–146
37. Presser K, Ross T, Ratkowsky D (1998) Modelling the growth limits (growth/no growth interface) of *Escherichia coli* as a function of temperature, pH, lactic acid concentration and water activity. Appl Environ Microbiol 64(5):1773–1779
38. Price D, Knerr S, Personnaz L, Dreyfus G (1995) Pairwise neural network classifiers with probabilistic outputs. In: Tesauro G, Touretzky DS, Leen TK (eds) Advances in neural information processing systems 7 (NIPS Conference, Denver, Colorado, USA, 1994). MIT Press, Cambridge, pp 1109–1116
39. Ratkowsky DA, Ross T (1995) Modeling the bacterial no growth interface. Lett Appl Microbiol 20:29–33
40. Richard D, David ER (1989) Product units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput 1(1):133–142
41. Soriano JM, Font G, Moltó JC, manes J (2002) Enterotoxigenic staphylococci and their toxins in restaurant foods. Trends Food Sci Technol 13:60–67
42. Storn R (2008) Differential evolution research—trends and open questions. In: Advances in differential evolution. Studies in computational intelligence, vol 143, pp 1–31. doi:10.1007/978-3-540-68830-3
43. Valero A, Hervás C, García-Gimeno RM, Zurera G (2007) Product unit neural network models for predicting the growth limits of *Listeria monocytogenes*. Food Microbiol 24(5):452–464
44. Valero A, Hervás C, García-Gimeno RM, Zurera G (2007) Searching for new mathematical growth model approaches for *Listeria monocytogenes*. J Food Sci 72(1):M16–M25
45. Valero A, Pérez-Rodríguez F, Carrasco E, Fuentes-Alventosa JM, Garcfa-Gimeno RM, Zurera G (2009) Modelling the growth boundaries of *Staphylococcus aureus*: effect of temperature, pH and water activity. Int J Food Microbiol 133:186–194
46. Vapnik V (1998) Statistical learning theory. Wiley, New York
47. Yu C, Davidson VJ, Yang SX (2006) A neural network approach to predict survival/death and growth/no-growth interfaces for *Escherichia coli* O157:H7. Food Microbiol 23(6):552560
48. Zaika LL, Moulden E, Weimer L, Phillips JG, Buchanan RL (1994) Model for the combined effects of temperature, initial pH, sodium chloride and sodium nitrite concentrations on anaerobic growth of *Shigella flexneri*. Int J Food Microbiol 23:345–358
49. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern C Appl Rev 30(4):451–462