Appl Intell (2018) 48:416–431
DOI 10.1007/s10489-017-0982-4

An exploratory study of mono and multi-objective metaheuristics to ensemble of classifiers

Antonino A. Feitosa Neto[1] · Anne M. P. Canuto[1]

Published online: 14 July 2017

Abstract This paper performs an exploratory study of the use of metaheuristic optimization techniques to select important parameters (features and members) in the design of ensemble of classifiers. In order to do this, an empirical investigation, using 10 different optimization techniques applied to 23 classification problems, will be performed. Furthermore, we will analyze the performance of both mono and multi-objective versions of these techniques, using all different combinations of three objectives, classification error as well as two important diversity measures to ensembles, which are good and bad diversity measures. Additionally, the optimization techniques will also have to select members for heterogeneous ensembles, using *k*-NN, Decision Tree and Naive Bayes as individual classifiers and they are all combined using the majority vote technique. The main aim of this study is to define which optimization techniques obtained the best results in the context of mono and multi-objective as well as to provide a comparison with classical ensemble techniques, such as bagging, boosting and random forest. Our findings indicated that three optimization techniques, Memetic, SA and PSO, provided better performance than the other optimization techniques as well as traditional ensemble generator (bagging, boosting and random forest).

✉ Anne M. P. Canuto
anne@dimap.ufrn.br

Antonino A. Feitosa Neto
antonino_feitosa@yahoo.com.br

1 Department of Informatics and Applied Mathematics (DIMAp), Federal University of Rio Grande do Norte (UFRN), Caixa Postal 1.598 59.072-970, Natal, RN, Brazil

Keywords Ensemble of classifiers · Metaheuristic optimization techniques · Feature selection · Member selection

1 Introduction

In the Pattern Recognition, there has been an intense investigation of many efficient approaches and methodologies in recent decades. This is the result of the increasing complexity and widening applicability of such systems. In the supervised approach of pattern recognition, for instance, the search for efficient and robust classification methods (classifier) has been intensified drastically in the last two decades. Several classifiers from different classes have been efficiently applied to several classification domains [1]. However, according to the No-Free-Lunch theorem [2], the best classifier will not be the same for all problem domains. A prominent solution to this scenario is the use of ensembles of classifiers. These systems are pattern classification structures that combine the outputs of several classification algorithms in order to achieve efficient performance in pattern classification [3, 4].

One important aspect in the design of efficient ensembles is the diversity promoted by the individual components of this system, called ensemble diversity. Ensemble diversity can be reached when the individual classifiers are built under different circumstances, such as: parameter setting, classification type and datasets. In the case of different datasets, the use of feature selection methods in ensemble systems usually increases the diversity of their components. The problem of feature selection can be considered a search problem, in which it intends to find the optimal subsets of attributes for all components of an ensemble system. In this sense, optimization techniques can be easily used to solve this problem.

The choice of the individual classifiers (member selection) of an ensemble is another important aspect in the design of ensemble systems, since it plays an important role in the design of efficient ensembles. In some applications, the ideal size of an ensemble system could vary enormously in order to obtain an ensemble with a reasonable accuracy. Therefore, the search for the optimal size of an ensemble is not an easy task. The problem of member selection can also be considered as a search problem, in which it is intended to find the optimal size (and composition) of an ensemble system.

The majority of papers in the literature usually address only one important aspect of the design of ensemble systems, either feature selection or member selection in ensembles. As a contribution to this important subject, in this paper, we will investigate extensively the influence of optimization techniques in the automatic design of ensemble systems. In order to do this, we selected ten well-known optimization techniques (genetic algorithm, tabu search, GRASP, iterated local search, variable neighborhood search, memetic algorithms, ant colony optimization, particle swarm optimization, simulated annealing and multi start) to define two important parameters in the design of ensemble systems (the individual classifiers and features). These techniques will be used in their mono and multi-objective versions and they use accuracy (error rate) as well as two diversity measures (good and bad diversity, described in (1) and (3) as objective functions in the search process. In this analysis, we will design heterogeneous ensembles in which k-NN (k nearest neighbor), Decision Tree and Naive Bayes classifiers will be used as individual classifiers of these systems and they will be combined by the Majority Voting technique.

In [5, 6], the authors presented the first attempt to analyze the use of these two diversity measures for the automatic design of homogeneous ensemble systems, using only two optimization techniques, tabu search and genetic algorithms. It could be observed, for instance, that there is a high correlation between bad diversity and error rate, which shows that this measure could be an interesting guide in the design of ensembles. In this paper, we extend the work done in [5, 6], performing an exploratory study with more optimization objectives, datasets and using heterogeneous structures of ensembles, aiming at performing a more extensive investigation. The main goal is to explore the full potential of the optimization algorithms in the automatic design of ensemble systems. Therefore, the main contributions of this paper can be described in the following topics.

1. To use 10 optimization techniques, in both mono and multi-objective versions, as a feature and member selector in heterogeneous ensemble systems;
2. To investigate the importance of two diversity measures to guide the design of ensemble systems using them as objectives in all ten optimization techniques;
3. To compare the results of optimization algorithms, using accuracy, good and bad diversity to build their solutions for the feature and member selection problem.

This paper is divided into six sections and it is organized as follows. Section 2 describes the research related to the subject of this paper. In Section 3, the optimization techniques are described and the methodology used in the experimental work of this paper is presented in Section 4. An analysis of the results provided by the empirical analysis is shown in Section 4. Finally, Section 5 presents the final remarks of this paper.

2 Recent studies on optimization techniques for ensemble systems

Finding an optimal subset of features that maximizes classification accuracy is still an open problem. In this paper, we work with a modification of this problem in which we aim at maximizing accuracy and diversity in ensemble systems through the selection of features and members of an ensemble. A recent review on the optimization techniques can be found at [30].

In the literature, we can find studies with similar topics like the application of optimization techniques (metaheuristics) applied to the selection of features in ensembles, such as in [23, 24, 28], among others. However, most of them usually apply one or two techniques, such as Bee Colony optimization [28], Genetic Algorithms [29], Particle Swarm Optimization [32, 33] and Bee Colony optimization and Particle Swarm Optimization for feature selection and Genetic Algorithms for configuration of ensemble parameters [23]. To the best of the authors knowledge, the only study that uses more than two optimization techniques can be found in [24], in which the authors performed a comparative analysis with six different metaheuristics (genetic algorithm, particle swarm optimization, ant colony optimization, harmony search, differential evolution, and quantum-inspired evolutionary algorithm) applied to the problem of feature selection, but they applied these techniques for individual classifiers and not for ensemble systems.

We can also find studies related to the use of optimization techniques for selecting ensemble members, such as in [27] that selected ensemble members using GRASP and in [26] that applied Genetic Algorithms. Once again, these studies applied only one optimization technique.

Additionally, all the aforementioned studies applied mono-objective optimization techniques. There are also some studies using the multi-objective version of these techniques for classification algorithms, such as in [25], in which the authors applied the multi-objective version of two techniques, Memetic Algorithms and Particle Swarm Optimization, to build a radial basis function network for classification problems. However, there are a few studies that deal with multi-objective optimization techniques in the context of ensembles, such as [22, 37, 38]. In [37], for instance, the authors proposed to build ensemble systems taking into account accuracy and diversity, but differently from this work, it does not address diversity as an optimization objective. In [38], the authors applied simulated annealing to calculate competence and diversity of ensemble components. Once again, diversity was not used as an optimization objective.

One recent study, [39], showed that that diversity can be used to design ensemble systems, but not on its own (along with error rate). We can conclude that it is necessary to use multi-objective techniques optimizing accuracy and diversity in order to achieve more accurate ensembles. Therefore, this paper aims at providing an exploratory study of the role of two important diversity measures, along with accuracy, in the automatic design of ensemble systems and using them in the context of several optimization techniques (mono and multi-objective versions).

3 Optimization techniques

As mentioned previously, we will analyze the performance of ten optimization techniques and they will be briefly described in the next subsections. The choice of these ten methods is that, first, we would like to investigate the use of neighborhood-based algorithms in the automatic design of ensemble systems, not yet explored in the literature. Then, we selected well-known neighborhood-based methods, that have been successfully applied in traditional optimization problems. Additionally, we would like to investigate the performance of population-based algorithms (the most common option for this problem) and we selected the most used algorithms in the Machine Learning area. Finally, we selected a hybrid algorithm, Memetic, since it has been successfully applied in traditional optimization problems and we believe it can provide good performance in the automatic design of ensemble systems.

In all algorithms, the objective function is the error rate (accuracy) and/or good and bad diversity measures. For mono-objective optimization problems, only one objective function $f(X)$ is usually applied. In order to use accuracy, good and bad diversity as optimization objectives, we can use the equations that describe these metrics as objectives. In this sense, we should try to maximize good diversity and accuracy. For instance, $f_{gd}(X) = max\{D^+\}$ represents the objective function for good diversity (D^+ is given by (1))

$$D^+ = \frac{1}{N}\sum_{i=1}^{P^+} v_i^- \tag{1}$$

Where P^+ represents the set of instances correctly classified by the ensemble system; and v_i^- represents the number of incorrect votes for i-th instance in P^+ [10]. For accuracy, we will use the following objective function.

$$f_{ac}(X) = max\left\{Acc = \frac{P_+}{P_+ + P_-}\right\} \tag{2}$$

where P_+ and P_- are the number of instances correctly and incorrectly classified by an ensemble. In contrast, we should minimize bad diversity, $f_{bd}(X) = min\ \{D^-\}$ (D^- is given by (3)).

$$D^- = \frac{1}{N}\sum_{i=1}^{P^-} v_i^+ \tag{3}$$

Where P^- represents the set of instances incorrectly classified by the ensemble system; v_i^+ denotes the number of correct votes for the i-th instance in P^-.

For the multi-objective versions, each optimization technique will have up to 3 objectives. In this context, we will use the inverse function of bad diversity in order to allow the use of all three objectives in a maximization problem.

We divided the optimization techniques into three classes, neighborhood-based, population-based and hybrid ones. In the next subsection, these algorithms will be described in more details.

3.1 Neighborhood-based algorithms

Neighborhood-based Algorithms (NbA) usually starts with one solution s and employs a local search to iteratively move from this solution s to an improved solution s^* in the neighborhood of s. Therefore, it will be necessary to define a local search method, for both mono and multi-objective techniques. The mono-objective local search starts with an initial solution s and it is composed of information about the individual classifiers N_C and the attributes N_A of these classifiers. In the individual classifiers part, each part represents one ensemble member and can be one possible classifier type (assign an integer to represent a classifier type). Then, the local search can replace each part by other type of classifiers (classification algorithm). Each replacement generates a new solution and it belongs to the solution neighborhood (neighbors). In the attribute part, the local search performs a number of random replacements and each replacement changes the boolean value of the attribute, that defines the presence or absence of the attribute in the initial solution.

The multi-objective local search is similar to its mono-objective version, differing in the objective function, that is a set of functions instead of just one. This change implies in a performance evaluation of the solutions that it is now done by a dominance relationship procedure. The algorithm 1 describes the step of the multi-objective local search.

Algorithm 1 Multi objective local search algorithm

```
**Require:** $F(set\ of\ functions), N_c(.), N_a(.), s\ (solution)$
  $T \leftarrow \{s' \in N_c(s)$ or $s' \in N_a(s) | s'$ dominates $s$ in $F\}$
  **while** $|T| > 0$
    $s' \leftarrow$ choice $f(x)$ where $\{x | x \in T\}$
    $s \leftarrow s'$
    $T \leftarrow \{s' \in N_c(s)$ or $s' \in N_a(s) | s'$dominates $s$ in $F\}$
  **end while**
  **return** ND
```

The neighborhood-based Algorithms used in this paper are the following ones.

Tabu Search (TS) it is a metaheuristic technique originally proposed in [11]. As a Neighborhood-based Algorithm, it employs a local search to iteratively move from one potential solution s to an improved solution $s*$ in the neighborhood of s, until a stopping criterion has been satisfied. It also has an aspiration function, that determines if a solution should be evaluated, in case its cost is lower than the cost of the best solution s. In this paper, the tabu list size is set to the maximum between 1 and 50% of number of attributes and may vary between 1 and 25% of the number of attributes. The way that the TS algorithm updates the tabu list is also modified in this paper. In this paper, a movement remains in the tabu list for a variable number of iterations that is represented by the following equation.

$$I = (0.5 * N_A) * N_C * R \quad (4)$$

where N_A is the number of attributes of the used dataset; N_C represents the number of classifiers of the solution and R is a random number that lies in the interval [1,50% of the size of the tabu list]. Finally, we also added a diversification strategy to restart the search every 20 iterations, using the best solution to trigger this restart.

In the multi-objective context, we selected the Multinomial Tabu Search algorithm (MTS) that, in each iteration, selects one goal to be considered in the MTS processing. The main difference from the mono-objective version is that it works with a set of possible solutions, called non-dominated solutions ND, at the end of the MTS processing. In this paper, this set is unlimited and all goals have equal chances of being chosen. In terms of parameters, MTS includes a parameter D, in comparison to TS, that indicates the number of iterations in which the diversification zprocess must occur. As each iteration represents one goal to be optimized, D represents the number of goals that must be involved in the diversification process. In addition, the cost function is modified to represent a set of functions F, according to the used set of objectives and, for each function f_i in F, there is a probability p_i of selecting f_i to be the objective to be optimized.

GRASP (Greedy Randomized Adaptive Search Procedure) The GRASP (Greedy Randomized Adaptive Search Procedure) algorithm has two distinct phases: In the first one, called constructive phase, a set of solutions is built and the best solution is selected. In the second phase, called refinement phase, a local search is applied to this selected solution [12, 20].

In the first phase, a criterion is used to define the quality of the solutions. In this paper, we use the objective functions to select the best solution of the initial set of solutions. Once the best solution is selected, the second phase starts with the individual classifiers of the selected solution, but they contain all attributes. In each iteration, an attribute is removed while better solutions are achieved. This procedure is applied to all individual classifiers of the selected solution. In addition, a path-relinking operator is used as a refinement procedure in order to obtain better solutions [19].

The multi-objective version is similar to the mono-objective version, changing the construction mode of solutions to deal with multi-objective solutions. It also changes the local search in a way that the path-relinking operator is applied to all ND solutions.

Iterated Local Search (ILS) The idea of this algorithm is to iteratively apply a "perturbed" local search algorithm [12]. In other words, ILS builds a sequence of locally optimal solutions by perturbing the current local minimum and applying local search after starting from the modified solution.

In this paper, we use five types of perturbations, which are the the following amendments:

1. Change the type of one individual classifier;
2. Change 1/3 of the individual classifiers;
3. Change 1 attribute of 2/3 of the individual classifiers;
4. Change 1/3 of the attributes of each individual classifier;
5. Change 2/3 of the attributes of each individual classifier.

All perturbations are performed on a random and uniform way. Finally, the acceptance criterion means that a new solution is accepted if it presents a lower cost in f than the original solution s.

In the multi-objective version of ILS, the perturbations in the solutions are implemented in the same way as in the mono-objective version, but using the set of objectives.

Multi-Start Algorithm This algorithm builds a random solution and refines through a local search process [12]. This algorithm selects a random initial solution and applies some improvements in this solution in order to obtain better solutions. The multi-objective version uses the multi-objective local search described in 1.

Simulated annealing it is a metaheuristic to approximate global optimization in a large search space. It works by emulating the physical process in which a solid is slowly cooled so that when eventually its structure is "frozen", this happens at a minimum energy configuration [12, 15].

In this paper, we employ a standard mono-objective simulated annealing. The multi-objective version of Simulated Annealing is similar to the MTS algorithm, in which each iteration of the algorithm selects one objective and uses it as the current objective evaluation of solutions. In addition, each objective has an equal probability of being selected. For both versions, the initial temperature T was set to 0.9, the rate of decay of temperature $alpha$ is set to 0.5 and the number of iterations or neighbors M in each temperature is 20.

Variable Neighborhood Search It consists of a systematic exchange of the neighborhood structures to gradually explore different solutions from the current one [12]. In this paper, we use three different neighborhood structures, which are:

1. Consider all individual classifiers, in which replacing them generates a neighbor;
2. Consider each possible combination of two individual classifiers in a solution, in which exchanging them generates a neighbor;
3. Consider a proportion, ATT, of attributes to be replaced, generating a neighbor.

During the search process, type 1 neighborhood is the first to be expanded, followed by type 2 and finally type 3. The multi-objective version is similar to the mono-objective one, using a set of objectives and the idea of non-dominant solution set.

3.2 Population-based algorithms

In this class of algorithms, we start with a population of solutions, instead of just one. In this paper, we analyze three population-based methods, which are:

Genetic algorithm it is a well-known bio-inspired metaheuristic algorithm, originally proposed by John Holland [13]. It can be seen as optimization technique whose functioning is based on biological mechanisms as hereditary and evolution. In genetic algorithm, a chromosome is used to describe each solution of a problem. Additionally, in each step of a genetic algorithm, a population of chromosomes (individuals) is considered.

In this paper, we apply a standard genetic algorithm, with selection, genetic operator (crossover and mutation) and evaluation. For the multi-objective context, we employ the NSGA-II algorithm [14]. For both versions, we use the following parameters: a standard two-point crossover (crossover probability = 80%) and bit flipping mutation; a random and uniform choice of the solution parents; and population of 30 chromosomes. The mutation operator is applied with a probability of 10%, in order to avoid premature convergence.

Ant Colony Optimization As genetic algorithm, Ant colony optimization (ACO) can be seen as a population-based (ants) method which is applied in computational problems that can be reduced to the idea of defining good paths in graph structures. In this paper, a path will be represented by a set of attributes and classifiers. Each ant builds a solution by choosing the set of attributes and/or classifiers to be part of a solution. It also updates pheromone, based on the quality of the corresponding solution [12, 16, 17].

In this work, we use the pheromone equation considering only the attributes of the solution. For the selection of the individual classifiers, a random selection is made. The choice of each attribute is define by (5).

$$p_{ij}^{k} = \frac{[\Gamma_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in N_k^i} [\Gamma_{il}]^{\alpha} * [\eta_{il}]^{\beta}} \qquad \forall j \in N_k^i \tag{5}$$

Where N_k^i represents the set of attributes not yet selected by ant k, Γ_{ij} represents the pheromone for the $i-th$ attribute of classifier j, η_{il} is a prior heuristic information that defines the importance of the $i-th$ attribute for the ensemble system and α and β are two parameters that control the influence of pheromone and heuristic information respectively. In this paper, η_{il} represents the cost of the solution in case only the i-th attribute is present in the individual classifier.

Given the probability of selection for each attribute, a solution is built by randomly selecting the individual classifiers and selecting the attribute in an iterative process. This iterative process starts with a solution where all attributes are active and each iteration selects an attribute to be removed, according to (5). In order to select an attribute, the current cost of the solution is assessed and compared to the cost of the solution by removing one attribute, if the cost is lower, this process continues with a new iteration. Otherwise this process is interrupted, returning the solution with the attributes deleted until then.

The multi-objective version is similar to the mono-objective one. It uses only one colony and the pheromone matrix is shared among all different objectives. At each iteration, one objective is selected and the pheromone matrix is updated.

Particle Swarm Optimization This population-based algorithm is based on a population of particles that moves in the search space of a problem in order to resolve an optimization problem [21]. In each iteration, all particles carry out a movement operation that consists of three parts: 1) Follow its own path; 2) Return to the best position found (local best); and 3) Go to the best position found by all other particles (global best).

In this paper, we use the implementation similar to the one used in [18] to deal with mono-objective discrete problems. The main difference of this algorithm to a standard PSO is the speed operator, that is divided into three components. The first one represents the particle following its own path (local search), while the second part is when the particle wants to follow its local best (path-relinking) and the last part represents when a particle selects the global best, the best position found among all other particles (path-relinking). For all three parts, a probability of selection is defined and one is selected, based on this probability.

The multi-objective version is similar to the mono-objective version. However there are two main differences:

1. The local search is replaced by local search multi-objective described by the algorithm 1;
2. The path-relinking operator is modified to act in the context of multi-objective.

The multi-objective algorithm assumes that all objectives are normalized in the (0, 1) interval and all goals must be minimized.

3.3 Hybrid algorithm

This class of algorithms uses the idea of neighborhood structure (local search), but it uses a population of solutions, instead of just one. In this paper, we analyze only one hybrid algorithm, which is a memetic algorithm.

Memetic Algorithms are metaheuristic algorithms based on a combination of ideas originated by different metaheuristics. It is also known as a hybrid Genetic Algorithm. Basically, it combines the ideas of neighborhood structure and population algorithms [12].

The global search for the memetic algorithm is a standard genetic algorithm using the same parameter set of the other GA-based methods (crossover probability = 80% and mutation probability = 10%). In addition, we applied an elitism selection operator and, for helping to refine the search process during its functioning, the mono-objective local search described in the beginning of Section 3 is applied.

In this paper, we employ a standard mono-objective memetic algorithm. For the multi-objective version, we update a version of the algorithm NSGA-II to include the combination needed by a Memetic Algorithm. In the algorithm implemented in this paper, we use the following parameters: a standard two-point crossover (crossover probability = 80%) and bit flipping mutation; a random and uniform choice of the solution parents; and population of 30 individuals. The mutation operator is applied with a probability of 10%.

4 Experimental methodology

In order to analyze the effects of using optimization techniques in the automatic design of ensemble systems, an empirical analysis will be conducted. In this analysis, we evaluate the effect of using ten optimization techniques in

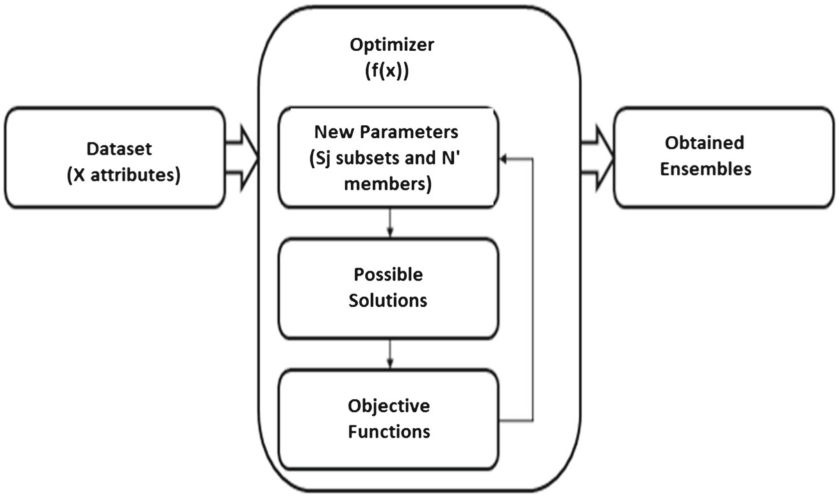

Fig. 1 An illustration of the general framework of the experimental methodology

the selection of important components of ensemble systems. In this analysis, mono and multi-objectives versions are analyzed in which the optimization objectives are all possible combinations of accuracy, good diversity and bad diversity.

The feature selection process is defined in the following way: Suppose that X is the original set of attributes, with s attributes. Then, we select N subsets, $S_j | j = 1, ..., N$ where N represents the number of individual classifiers that compounds an ensemble system. Therefore, each subset of S_j has a cardinality $s_i < s$. In addition, the member selection procedure selects whether or not a classifier will compose an ensemble system. Once it is selected, the classifier type is also defined. In this sense, based on an initial size of an ensemble, *N*, *N'* members will be selected to compose the ensemble system, where $N' < N$.

In this paper, we apply optimization techniques to select these two parameters (attributes and members). The general framework of the experimental methodology is illustrated in Fig. 1 and the main steps of the used methodology is presented in Algorithm 1.

Algorithm 2 General methodology

```
Require: Database, Optimizer, Objectives, Maximum Number of Classifiers Base N, Attributes, Classifiers
  ND is a new set of non-dominated solutions
  while stopping criterion is not satisfied do
    Attributes, Classifiers = Optimizer(Database, Attributes, Classifiers, Objectives, N)   ▷ Attributes and classifiers for each base classifier
    Ensemble = Make Ensemble from (Attributes, Classifiers)
    Eval = Evaluate(Ensemble, Objectives)
    if is non-dominated solution (ND, Eval) then
      Insert Ensemble in ND
    end if
  end while
  return ND
```

For member selection, *k*-NN (*k* nearest neighbor), Decision Tree and Naive Bayes will be used as possible types of individual classifiers of the ensemble systems and they will be combined by the Majority Voting technique. These algorithms are chosen because they are simple and efficient algorithms that apply different classification criteria in their hypothesis. Decision tree, for instance, builds a classification tree during its training and use it during the test procedure, while *k*-NN is an instance-based algorithm. Additionally, all classifiers of the same type used the same parameter setting, once they use different training data. For k-NN, for simplicity reasons, we decided to use its simplest version, using only one neighbor. Then, the ensemble systems uses a standard stacking procedure to define the learning process for the individual classifiers and for the combination methods.

In relation to the feature selection, there is no restriction about the number of classifiers in which an attribute may be assigned to. In other words, an attribute can be assigned to all classifiers or to none of them.

In order to compare the accuracy of the obtained ensemble systems, two statistical tests will be applied, depending on the used scenarios. They are Mann-Whitney (U test) and Friedman tests with a post-hoc test [7] and all applied tests are bi-caudal, in which the null hypothesis is based on the idea that the samples come from the same population. In all three tests, the confidence level is 95% ($\alpha = 0.05$).

4.1 Optimization algorithms

In all optimization algorithms, the solutions are represented by a vector of values indicating the individual classifiers and attributes used by the ensemble. The parameter N is the maximum number of individual classifiers that a solution can have. For example, for $N = 15$, it means that all possible solutions represent ensembles composed from 2 to 15 classifiers. The size of the vector that represents a solution is determined by the number of attributes of the selected dataset multiplied by the maximum number of individual classifiers (plus 1). For example, let us take $N = 15$ and the number of attributes of problem $s = 5$. Then, we will have a vector of size $(s + 1) * N$ positions, 90 positions. The first 6 positions define the configuration of the first classifier. The first position indicates the type of classification algorithm, in which 0 means that this classifier is not selected, 1 indicates a k-NN classifier, and 2 and 3 indicate Decision Tree and Naive Bayes, respectively. The remaining five positions represent the attributes that will be used by this classifier. In the same way, the 6 following positions correspond to the second individual classifier and so on until the last classifier.

During the processing of an optimization technique, one or more initial solutions are instantiated, depending on if the selected algorithm if a population-based method or not. The initial solutions are randomly generated and the optimization process starts, generating and evaluating the obtained solutions. The evaluation of a possible solution is performed based on its performance in a validation set. In order to define the training/validation/test division, a stratified 10-fold cross-validation will be used in which 8 folds will be used for training, one for validation and one for test.

The ending condition of each optimization technique is based on three criteria, which are: maximum run time (from 30 to 60 minutes for neighborhood-based methods and from 60 to 120 minutes for population-based methods, depending on the analysed dataset); 20 iterations without updating the best solution, in the case of the mono-objective algorithms and 20 iterations without upgrading the set of non-dominated solutions, in the case of multi-objective algorithms; and a maximum of 1000 iterations. Finally, as the optimization techniques used are non-deterministic, 10 runs of each technique are performed. Thus, all accuracy results presented in this paper refer to the mean over 10 different test sets.

For each optimization technique, seven possibilities are generated (three mono-objective and four multi-objective ones). The next subsection will describe the multi-objective versions of the used optimization techniques.

4.1.1 The multi-objective algorithms

For all multi-objective neighborhood-based algorithms, we apply the multi-objective local search described in 1. In addition, we apply well known multi-objective versions of the population-based algorithms. As already mentioned, we will use four multi-objective versions, with all combinations of error rate, good and bad diversity. Table 1 presents all multi-objective combinations that are used in this paper.

When using multi-objective optimization techniques, one of the biggest challenges is to compare the outcomes provided by these techniques. In general, the outcome of a multi-objective stochastic optimizer is a set of non-dominated points, called approximation set. Pareto front is the best set of non-dominated (ND) solutions for a problem. In this set, there is no solution that dominates other solution within the ND set. The main aim of the multi-objective optimization techniques is to create a set of ND solutions that is as close as possible to the Pareto front. Therefore, in order to evaluate the quality of the ND set, several evaluation metrics can be used, such as GD (Generational Distance) and IGD (Inverted Generational Distance). However, for these two evaluation metrics, the exact Pareto front must be known.

Nevertheless, in our case, the exact pareto set cannot be known. In this case, we decided to follow the general guidelines that are provided in [34], in which different measures compared to approximation sets are reviewed. The authors also proposed the dominance ranking approach which they claim that it yields general statements about the relative performance of multi-objective optimizers fairly independent of preference information. This approach is recommended as the first step in any comparison. If conclusions cannot be drawn based only on this approach, other measures, named quality indicators, are applied to point out differences among the sets generated by different stochastic optimizers. Moreover, they recommend to use Pareto compliant indicators which are those that whenever an approximation set A is preferable to an other one, B, with respect to weak Pareto dominance, the indicator value for A is at least as good as the indicator value for B. The Pareto-compliant quality indicators are multiplicative binary-ε [35] and hyper-volume [36] and they are used here when significant differences are not devised with the dominance ranking approach. The ε-indicator gives the factor by which an approximation set is worse than another with respect to all objectives [35]. The hyper-volume measures the portion of the objective space that is weakly dominated by an approximation set [34].

At the end of the processing of each optimization technique, one or more solutions are provided. In the case of mono-objective algorithms, a single solution is provided and, in the case of multi-objective algorithms, a set of solutions is provided. In this case, the best solution is selected based on the error rate delivered by all solutions over the validation set (the one that achieved the lowest error rate).

Table 1 Multi-objective Versions for all Optimization Objective

–	Alg.EG	Alg.EB	Alg.GB	Alg.EGB
Error	x			x
Good Diversity	x	x	x	x
Bad Diversity		x	x	x

4.2 Datasets

This experimental analysis is performed using 23 different classification data sets. The majority of databases used in this analysis were obtained from the UCI repository [8]. All datasets are preprocessed removing irrelevant attributes. Table 2 presents a description of the datasets used in this analysis, defining the number of instances, attributes and classes of the datasets.

The datasets that were not taken from UCI were Gaussian, Simulated and Jude. They are synthetic databases that simulate microarray data and were created to test the machine learning algorithms in the gene expression analysis [31].

5 Results

This empirical analysis will be divided into four parts. In the first part, we will analyze the effect of ensemble size, in which we will analyze ensembles with different sizes (number of individual classifiers) in order to define which size should be used in the following parts of this analysis. In the second part, we will investigate which objective set achieves the most accurate ensembles. Then, in the third part, we will analyze the best optimization technique, the one that provides the most accurate ensembles. Finally, we select the best algorithms of the previous part and compare them with some existing techniques for building ensemble systems, in order to compare the performance achieved in this paper

Table 2 Description of the used data sets, defining the number of attributes (C stands for the number of categorical attributes and N for numeric attributes), instances and classes

Index	Base	Attributes(C/N)	Instances	Classes
1	Ecoli	7(7/0)	336	8
2	Wine	13(13/0)	178	3
3	Vehicle	18(18/0)	846	4
4	Flags	28(10/18)	194	6
5	Breast Tissue	9(9/0)	106	6
6	Credit Approval	14(6/8)	690	2
7	Hepatitis	19(6/13)	155	2
8	Annealing	31(6/25)	798	6
9	Pittsburgh Bridges V1	11(3/8)	105	6
10	Congressional Voting Records	16(0/16)	435	2
11	Climate Model Simulation Crashes	20(20/0)	540	2
12	Breast Cancer Wisconsin Prognostic	33(33/0)	198	2
13	Connectionist Bench Vowel	12(10/2)	990	11
14	Labor	16(8/8)	57	2
15	Horse Colic	22(7/15)	368	2
16	Ionosphere	33(33/0)	351	2
17	Planning Relax	12(12/0)	182	2
18	Zoo	17(0/17)	101	7
19	Sick	27(6/21)	3772	2
20	Dermatology	34(1/33)	366	6
21	Protein	120(0/120)	583	5
22	Gaussian	600(0/600)	60	3
23	Simulated	600(0/600)	60	6

with the state-of-the-art in ensemble systems. All four parts of this analysis will be presented in the next subsections.

5.1 Ensemble size

In this first part, we aim at analyzing the effect of ensemble size in the performance of the ensemble systems obtained by optimization algorithms. In order to do this, we selected three ensemble sizes, with $N = 10$, 15 and 30, where N represents the maximum number of individual classifiers for an ensemble. For each ensemble size, all ten optimization algorithms are performed (for all 7 combinations of objective functions) and the obtained ensembles are evaluated. It is important to emphasize that the numbers 10, 15 and 30 are the maximum number of individual classifiers so that an ensemble can have from 2 to N individual classifiers.

In order to do this analysis, we will apply the Friedman test to compare the performance (error rate) of all three ensemble sizes (10, 15 and 30) for each optimization algorithm, for all 23 datasets. The p-values of the Friedman test are presented in Table 3. In this table, we also present the results of the post-hoc Friedman test, comparing each pair of sizes, for the cases in which the Friedman test detected difference in performance ($p - values < 0.05$). The post-hoc Friedman test results are presented in the ”A X B” format, where A and B may have the following values: 10c, 15c and 30c, representing 10, 15 and 30 individual classifiers, respectively.

The Friedman test did not show statistical difference in performance for the majority of algorithms, with exception for GA, ILS, MultiStart and GRASP. In other words, according to the statistical test, there is no statistical difference in performance for ensemble systems using 10, 15 or 30 as the maximum number of individual classifiers. An interesting aspect can be observed in the cases that the Friedman test detected statistical difference in performance. In these cases, the post-hoc test proved that the use of 10 individual classifiers provided more accurate ensemble systems.

The results obtained in Table 3 show that the best option for ensemble size is 10, since it provides ensemble systems with better or similar performance to the other ensemble sizes. Therefore, hereafter, we will be using ensemble systems with 10 as the maximum number of individual classifiers ($N = 10$).

5.2 Objective sets

In this section, we will investigate which objective set achieves the most accurate ensembles. In order to do this, we will analyze the classification error of ensemble systems, when they are generated by all possible combinations of the optimization objectives, which are: error (E), good diversity (G), bad diversity (B), error and good diversity (EG), error and bad diversity (EB), good and bad diversity (GB) as well as error, good and bad diversity (EGB). Due to the large amount of data, we summarize these values in Table 4 that illustrates, for each algorithm, the number of datasets which obtained the best result (lowest error rate) by each

Table 3 Fridman and the Post-hoc Test for Ensemble Size

	MTS	GA	ACO	GRASP	PSO
Friedman	0.4858	0.0099	0.1492	0.0008	0.191
10c x 15c	–	0.0002	–	0.1338	–
10c x 30c	–	0.0001	–	0.0002	–
15c x 30c	–	0.6394	–	0.0003	–
	VNS	SA	Memetic	ILS	MultiStart
	0.1105	0.1854	0.087	0.0246	0.0022
10c x 15c	–	–	–	0.0292	0.4327
10c x 30c	–	–	–	0.0002	0.0002
15c x 30c	–	–	–	0.0108	0.0005

Table 4 Number of Best Results for Each Objective Set

	E	G	B	EG	EB	GB	EGB
TS	7	0	0	7	5	3	3
GA	4	0	3	9	11	4	8
ACO	4	1	1	8	6	2	7
GRASP	17	0	1	3	3	3	2
PSO	15	0	1	5	5	3	2
PSO	11	0	1	8	6	3	3
SA	9	0	0	9	8	3	8
Memetic	20	0	2	3	5	3	3
ILS	15	0	0	3	2	1	5
Multi Start	16	0	1	4	4	2	2
Total	118	1	10	59	55	27	43

E = Error; G = Good Diversity; B = Bad Diversity

objective. For example, for the TS algorithm, error rate (E) objective obtained the best result in 7 datasets, while error and the good diversity objective (EG) obtained the best results in 7 datasets.

In observing Table 4 we notice that the best results are concentrated in the optimization of error rate (E), error rate and good diversity (EG), error rate and bad diversity (EB) and error rate, good and bad diversity (EGB). However there are some aspects that need to be considered: ACO has error and good diversity (EG) as the best objective set as well as GA, in which the best objective set is error and bad diversity (EB). It is important to emphasize that genetic algorithms (GA) is the most popular optimization algorithm for applications in ensemble systems. In addition, ACO, along with PSO, can be considered as a second option of optimization algorithm for ensemble systems. In other words, the inclusion of one diversity measure as objective had a positive effect in the performance of the two most popular optimization algorithms for ensemble systems.

Now, in order to compare the performance of the different objective sets, we apply the Friedman test, comparing all objective sets and datasets, for each optimization algorithm. The obtained p-values are presented in Table 5.

When analyzing Table 5, it can be detected a statistical difference in performance among the different objective sets in all optimization algorithms. This means that there is a statistical difference in performance between one or more pairs of objective sets. Then we decided to apply the post-hoc Friedman test in order to evaluate each pair of objective sets. Table 6 presents the p-values of the post-hoc test. For simplicity reasons, we are only presenting the comparison of the objective sets that provided the best results in Table 4, which are: error rate (E), error rate and good diversity (EG), error rate and bad diversity (EB) and error rate, good and bad diversity (EGB). In this table, each column represents the optimization algorithm and each line represents the compared pair of objective sets. In addition, the bold numbers represent the statistically significant results ($p - values < 0.05$).

As we can observe in Table 4, despite the statistical difference observed by the Friedman test, the same behavior was not detected by the post-hoc test, comparing the three best objective sets. For instance, the first column of Table 6 corresponds to the TS algorithms and we can observe that we did not verify statistically significant differences in any pair of objective sets. The same behavior occurs for seven optimization algorithms. However, in two optimization algorithms, GA and ACO, the performance of error rate (E) and error rate and bad diversity (EB) as well as error rate (E) and error rate and good diversity (EG) proved to be statistically different. In both cases, as the performance delivered by EB and EG are better (lower error rate) than the one delivered by E, we can state that the inclusion of both diversity measures had a positive effect in the performance of the ensemble systems, from a statistical point of view. Therefore, the improvement in performance detected in Table 4 proved to be statistically significant.

Now that we analyzed the objective sets only by the error rate produced by the ensemble systems obtained by the optimization techniques, we would like to assess the different optimization algorithms from the multi-objective

Table 5 The Friedman Test Between the Different Sets of Optimization Objectives

	MTS	GA	ACO	GRASP	PSO
Friedman	0.0001	0.0001	0.0001	0.0001	0.0001
	VNS	SA	Memetic	ILS	MultiStart
Friedman	0.0001	0.0001	0.0001	0.0001	0.0001

Table 6 The Post-hoc Friedman Test Between the Different Sets of Optimization Objectives

ObjSet	TS	GA	ACO	GRASP	PSO
E/EB	0.4778	**0.0299**	**0.0424**	0.3962	0.4003
E/EG	0.3584	**0.0037**	**0.0170**	0.2220	0.4174
E/EGB	0.2211	0.1386	0.0817	0.1652	0.1978
EG/EG	0.9300	0.9912	0.9912	0.8510	0.9474
EG/EGB	0.8433	0.9212	0.9387	0.6387	0.7583
EB/EGB	0.9737	0.9825	0.9474	0.8235	0.6445
ObjSet	VNS	SA	Mem	ILS	MS
E/EB	0.3667	0.5212	0.2443	0.0958	0.2056
E/EG	0.5212	0.4950	0.2841	0.1904	0.4176
E/EGB	0.4604	0.5561	0.1382	0.2524	0.3920
EG/EG	0.6683	0.9912	0.8432	0.7417	0.7836
EG/EGB	0.9212	1.0000	0.8518	0.8951	0.8175
EB/EGB	0.6763	0.9825	0.7499	0.7170	0.9737

Table 7 Multi-Objective parameters - GA

	E/EG			E/EB		
Base	DR	H	ϵ	DR	H	ϵ
1	0.4127	0.0952	0.3413	0.0635	0.2381	0.0238
2	0.1667	**0.0159**	**0.0079**	1	0.1667	0.6032
3	0.619	**0.0317**	**0.0317**	**0.0079**	–	–
4	0.2857	0.0079	0.1508	**0.0476**	–	–
5	0.2857	0.8413	0.9683	0.1667	0.0317	0.5714
6	0.746	0.0079	0.0873	0.2063	0.5397	0.6905
7	0.4444	**0.0079**	**0.0238**	0.2381	0.0317	0.3095
8	0.0952	**0.0317**	**0.0397**	0.1746	0.0317	0.8413
9	0.7698	0.0952	0.0952	0.0794	0.0079	0.7381
10	0.9841	0.0079	0.5476	0.5238	0.0952	1
11	1	0.0079	0.8889	**0.0079**	–	–
12	0.5238	0.3095	0.3095	**0.0238**	–	–
13	0.6032	0.0079	0.1349	**0.0159**	–	–
14	1	0.1667	1	1	1	1
15	0.9048	0.5397	0.8333	0.7143	0.0397	0.0635
16	**0.0476**	–	–	0.2778	0.2778	0.7857
17	1	0.5476	0.4206	0.1667	0.0079	0.1905
18	0.4444	0.3095	0.1667	0.4444	0.0952	0.4444
19	0.2619	**0.0079**	**0.0317**	0.5079	0.0079	0.8413
20	0.7143	0.1508	0.5079	1	0.0079	0.8095
21	0.7143	**0.0079**	**0.0317**	**0.0079**	–	–
22	1	0.0079	1	1	1	1
23	1	0.0079	1	1	1	1

DR = Dominance Ranking; H = Hyper-volume ϵ = Binary-ϵ

Table 8 Multi-Objective parameters - ACO

	E / EG			E / EB		
Base	DR	H	ϵ	DR	H	ϵ
1	0.7222	**0.0079**	**0.0159**	0.2857	0.0079	0.5556
2	0.4762	0.7937	1	1	0.9048	0.6825
3	0.4444	0.0556	0.5	0.1429	**0.0079**	**0.0159**
4	0.7143	0.0159	0.0556	0.1667	0.0079	0.9524
5	0.0794	0.0317	0.3889	**0.0079**	–	–
6	0.3333	0.0556	0.6111	0.5238	0.0238	0.3889
7	0.3651	0.0476	0.5635	**0.0079**	–	–
8	0.9206	0.1667	0.381	1	0.0556	0.246
9	**0.0397**	–	–	**0.0079**	–	–
10	0.1429	0.0317	0.6667	1	0.0317	0.1111
11	0.1429	0.0159	0.0556	0.4206	0.0079	0.8413
12	0.1905	**0.0317**	**0.0159**	0.127	0.0079	0.2222
13	1	0.0952	0.8889	0.3968	0.0952	0.8889
14	1	0.1667	1	1	0.4444	1
15	0.7619	0.3016	0.3016	0.7143	0.0873	1
16	0.2857	0.5873	0.5476	0.2063	**0.0079**	**0.0397**
17	0.4444	0.1349	0.127	0.1667	0.1667	1
18	1	0.127	0.4048	1	0.0079	0.4048
19	0.0794	0.4206	0.4444	0.3651	1	1
20	0.7143	0.0159	0.5873	0.5238	0.0079	0.9683
21	0.6429	0.2857	0.6905	0.0714	0.4206	0.3968
22	0.9206	1	1	1	0.4	1
23	0.0476	1	1	0.6429	0.2	1

DR = Dominance Ranking; H = Hyper volume ϵ = Binary-ϵ

context. The main aim of this analysis is to assess the inclusion of one diversity measure, along with error rate, as objective set and if it can provide ensemble systems with better performance. In order to do this, we evaluate three important multi-objective parameters, which are: Dominance Ranking, Hyper-volume and binary-ϵ (described in Section 4.1.1).

Although these measures are multi-objective parameters, we also apply them in the mono-objective versions. In order to do this, we perform the mono-objective versions in a common way and, we calculate the remaining objectives only in the solution provided by the optimization algorithm. For example, if we apply the error rate as objective, we run the optimization algorithms using only the error rate and, after the optimization process, we calculate the good and the bad diversity presented in the delivered ensemble. Thus, we have all values for the application of above cited parameters.

In order to evaluate the optimization algorithms from the multi-objective context, we apply the Mann-Whitney test in the obtained results in order to check if there is any statistical difference. For simplicity reasons, we selected three optimization algorithms, which are: GA, ACO and SA. In the same way, we selected three objective sets, error (E), error and bad diversity (EB) and error and good diversity (EG) for analysis. These results are presented in Tables 7, 8 and 9 for GA, ACO and SA, respectively.

When analyzing Tables 7, 8 and 9, we notice that only a few statistic differences occur, for all datasets. For GA (Table 7), we have some cases that presented significant difference between the two analyzed objective sets. When comparing error to error and good diversity, we detected statistical significant differences in 7 datasets (1 for DR and 6 for H and ϵ), in which only 5 of them were in favor of error and good diversity (EG) objective. For the comparison of error (E) against error and bad diversity (EB), we have statistical significant cases in 6 datasets (all 6 are in DR), in which 3 of them are in favor of EB objective. For ACO (Table 8), we also have some cases that present a statically significant difference in performance. When comparing E against EG, we have cases in 3 datasets, in which 2 of them are in favor of EG objective. For the comparison to EB, we have statically significant differences in 5 datasets, being 3 of them in favor of EB objective. Finally, for SA (Table 9), when comparing E against EG, we have statically significant differences in 7 datasets (1 for DR and 6 for H and ϵ), in which 6 of them is in favor of EG objective. For the comparison to EB, we have statically significant differences in 7 datasets, being 4 of them in favor of EB objective.

In summary, according to the statistical analysis of three important parameters for multi-objective, the bi-objective versions EG (error and good diversity) and EG (error and bad diversity) can generate similar performance to the mono-objective version using error rate as objective, for the majority of datasets. In the cases in which statistically significant differences in performance were detected, they were in favor of the bi-objective versions, for most of the analyzed cases. It is important to emphasize that we analyzed only three optimization, GA, ACO and SA. However, a similar pattern of behavior was observed in the remaining optimization algorithms.

5.3 Optimization algorithms

As mentioned previously, in this section, we will analyze the performance of all optimization techniques, aiming at assessing which one provides the most accurate ensembles. Table 10 illustrates the average error rate for each algorithm, averaging the results of all objective sets. The main aim is to evaluate which optimization algorithm provides the best overall performance. The bold numbers represent the best performance for a dataset. The AVG line represents the average error rate over all datasets and the values within brackets represent the overall position of the optimization technique, in terms of error rate, ranging from 1 (the one that provides the most accurate ensembles) to 10 (the least accurate ensembles).

As we can observe from Table 10, in terms of the number of best results, PSO delivered the most accurate ensemble systems in 7 datasets, followed by Memetic and SA (5 datasets), GRASP and ILS (2 datasets) and VNS and MS (1 dataset). However, when considering the averaged error rate, Memetic is the optimization technique that provides the most ensemble systems, followed by SA and PSO.

In order to analyze the performance of the optimization from a statistical point of view, we applied the Friedman test in the results of Table 10. As a result of this test, we observed that that there is a significant difference in performance of all optimization algorithms (p-value = 0.001). Then we applied the post-hoc Friedman test for each pair of algorithms. The p-values of the post-hoc test are expressed in Table 11. The results in this table are presented in a (symmetric) matrix form, comparing the line algorithm with the column algorithm. For example, the comparison of TS with GA (cell 2,1) provides p-value = 0.9260. The statistically significant values are described in bold.

When analyzing Table 11 we observe that there are statistically significant results in almost 56% of the analyzed cases (25 out of 45). An important aspect is that TS, GA and ACO showed statistically significant results, when compared to all remaining optimization techniques. As these algorithms provided the least accurate ensembles, this decrease in performance proved to be statistically significant. Among the remaining algorithms, there are only a few statistically significant differences, mainly related to Memetic and SA algorithms. Therefore, instead of selecting one optimization algorithm, we have selected a set of two algorithms, SA, and Memetic. We will use this set of best algorithms in the comparison with the classical approach, that will be done in the next subsection.

Table 9 Multi-Objective parameters - SA

	E/EG			E/EB		
Base	DR	H	ϵ	DR	H	ϵ
1	0.6905	**0.0079**	**0.0317**	0.2063	0.1032	0.7460
2	0.2381	0.0952	1.0000	0.4444	0.1429	1.0000
3	0.2857	0.0079	0.2222	0.4444	0.0079	0.2460
4	0.0952	0.0238	0.6349	1.0000	0.0079	0.8413
5	0.7302	**0.0079**	**0.0079**	**0.0476**	–	–
6	0.7937	0.0079	0.0952	0.5238	0.1032	1.0000
7	1.0000	0.0079	0.1905	0.4444	0.0079	0.6190
8	0.1349	0.0079	0.0635	**0.0159**	–	–
9	0.9206	**0.0079**	**0.0159**	0.4444	0.0079	0.0556
10	**0.0476**	–	–	0.1667	0.0079	0.1667
11	0.6032	0.0397	0.1508	0.4444	**0.0079**	**0.0476**
12	0.7222	0.0317	0.2857	1.0000	0.0079	0.1270
13	0.3333	0.0079	0.1905	**0.0476**	–	–
14	1.0000	0.0476	1.0000	1.0000	0.4444	1.0000
15	0.4286	0.0317	0.0635	0.5238	0.0794	0.2857
16	0.2302	**0.0079**	**0.0079**	0.1667	**0.0079**	**0.0238**
17	0.9206	**0.0079**	**0.0317**	**0.0476**	–	–
18	1.0000	**0.0079**	**0.0079**	1.0000	0.0079	0.1667
19	0.8810	0.0079	0.4206	0.4444	0.0079	0.1270
20	1.0000	0.0397	0.1429	0.1667	**0.0079**	**0.0079**
21	0.1270	0.0079	0.3095	0.1667	0.0079	0.0397
22	1.0000	0.0079	1.0000	1.0000	1.0000	1.0000
23	1.0000	0.0079	1.0000	1.0000	1.0000	1.0000

DR = Dominance Ranking; H = Hyper volume ϵ = Binary-ϵ

5.4 Comparative analysis: classical techniques

Table 12 presents the comparison of the best optimization algorithms with some classical ensemble techniques, which are: random selection, bagging, boosting and random forest (more details about these methods can be found [9]). The random selection generates a heterogeneous committee by randomly choosing individual classifiers, the choices are:

Table 10 Average error rate of all optimization algorithms

Dataset	TS	GA	ACO	GRASP	PSO
1	0.1472	0.1412	0.1202	0.1070	**0.1070**
2	0.0114	0.0122	0.0090	0.0048	**0.0000**
3	0.2788	0.2600	0.2484	0.2528	0.2724
4	0.2164	0.2360	0.2136	0.2073	0.1844
5	0.2492	0.2454	0.2336	0.2226	0.2058
6	0.1304	0.1313	0.1298	0.1260	0.1338
7	0.0852	0.0878	0.0826	0.0772	0.0504
8	0.0238	0.0134	0.0088	0.0060	0.0100
9	0.2648	0.2594	0.2422	0.2250	**0.2000**
10	0.0524	0.0418	0.0226	**0.0210**	0.0232
11	0.0606	0.0712	0.0460	**0.0443**	0.0544
12	0.1860	0.1980	0.1920	0.1782	0.1680
13	0.0488	0.0166	0.0112	0.0387	0.0178
14	0.0036	0.0000	0.0000	0.0000	**0.0000**
15	0.0444	0.0292	0.0160	0.0110	0.0110
16	0.0540	0.0570	0.0618	0.0576	0.0650
17	0.2758	0.2726	0.2800	0.2660	**0.2418**
18	0.0240	0.0340	0.0220	0.0200	**0.0060**
19	0.0513	0.0190	0.0430	0.9000	0.0184
20	0.0214	0.0202	0.0092	0.0080	0.0126
21	0.1744	0.1596	0.1860	0.1630	0.1666
22	0.1102	0.0000	0.2667	0.0068	0.0100
23	0.0332	0.0000	0.0734	0.0170	**0.0000**
AVG	0.1029(10)	0.0906(7)	0.0993(9)	0.0827(5)	0.0771(3)
Dataset	VNS	SA	Memetic	ILS	MS
1	0.1166	0.1138	0.1160	0.1154	0.1167
2	0.0070	0.0036	0.0045	0.0056	0.0024
3	0.2434	**0.2398**	0.2478	0.2460	0.2413
4	0.1960	0.1940	**0.1803**	0.1888	0.1900
5	0.2116	**0.1935**	0.2170	0.2134	0.2170
6	0.1236	0.1238	0.1186	**0.1162**	0.1175
7	0.0648	0.0624	0.0618	**0.0492**	0.0582
8	0.0050	0.0050	**0.0046**	0.0080	0.0130
9	0.2230	0.2212	0.2268	0.2174	0.2210
10	0.0256	0.0225	0.0253	0.0232	0.0215
11	0.0494	0.0468	0.0510	0.0636	0.0460
12	0.1750	0.1690	**0.1630**	0.1631	0.1690
13	**0.0060**	0.0068	0.0100	0.0095	0.0137
14	0.0000	0.0000	0.0000	0.0000	0.0000
15	0.0110	0.0125	**0.0095**	0.0115	0.0104
16	0.0524	0.0528	**0.0483**	0.0490	0.0502
17	0.2526	0.2518	0.2500	0.2504	0.2528
18	0.0100	0.0100	0.0067	0.0160	0.0140
19	0.9000	**0.0108**	0.0210	0.9000	0.9000
20	0.0100	0.0086	0.0086	0.0115	**0.0065**
21	0.1648	**0.1572**	0.1586	0.1616	0.1590
22	0.0000	**0.0000**	0.0000	0.0834	0.1915
23	0.0268	0.0000	0.0000	0.0298	0.0665
AVG	0.0788(4)	0.0754(2)	0.0751(1)	0.0837(6)	0.0913(8)

decision tree, k-NN, Naive Bayes and none. The ensemble system is also composed of up to 10 individual classifiers. In addition, it has a feature selection step, in which a random selection of around 70% of the feature set is selected for each individual classifier. The random forest method is composed of decision trees and it was implemented as originally proposed. Finally, bagging and boosting methods are composed of k-NN and combined by majority voting. In this table, the bold numbers represent the ensemble generation methods that delivered the best performance of the corresponding dataset.

As we can observe from Table 12, SA and Memetic algorithms delivered the most accurate ensembles, for almost 80% of the analyzed datasets. This is a promising result since bagging, boosting and random forest are classical ensemble generation techniques that have been widely used in the literature. In order to evaluate the performance of these techniques, from a statistical point of view, we applied the Friedman test in the results of Table 12 and we observed that that there is a significant difference in performance of all optimization algorithms (p-value = 0.0007). Then we applied the post-hoc Test for each pair of techniques. The p-values of the test are expressed in Table 13.

As we can see in Table 13, the improvement in performance detected in both optimization techniques proved to be statistically significant. In other words, both algorithms, SA and Memetic, delivered more accurate ensemble systems, from a statistical point of view, when compared to bagging, boosting, random forest and random methods.

5.5 Result analysis

In the previous sections, a huge variety of results were presented. In this section we will try to analyze and to explain the obtained results. In the first analysis, ensemble size, the maximum number of individual classifiers varied from $N = 10$ to 15 and 30. Unlike what was expected, the obtained results were similar using all three ensemble sizes, for most datasets. In cases where a statistical difference was detected, it is favorable to the smallest ensemble size ($N = 10$). As N represents the maximum number of individual classifiers, we calculated the average size of the obtained ensemble systems and we observed that they are close to 10 (average size 9 for $N = 10$, 11 for $N = 11$ and 16 for $N = 30$). In other words, even when we used a maximum number equal to 30, the final ensemble configurations returned by the optimization techniques were much smaller. We believe that the average ensemble sizes were close to 10 for two reasons:

– The used individual classifiers (k-NN, Naive Bayesian and Decision Tree) have good computational power and this may mean that their combination on a large heterogeneous ensemble does not bring benefits to the

Table 11 p-values of the Post-hoc Friedman test, comparing the optimization algorithms

X	TS	GA	ACO	GRASP	PSO	VNS	SA	Mem	ILS
TS	–	–	–	–	–	–	–	–	–
GA	0.9260	–	–	–	–	–	–	–	–
ACO	**0.0190**	0.5563	–	–	–	–	–	–	–
GRAS	**0.0000**	**0.0000**	**0.0136**	–	–	–	–	–	–
PSO	**0.0000**	**0.0000**	**0.0000**	0.8919	–	–	–	–	–
VNS	**0.0000**	**0.0000**	**0.0000**	0.6626	1.0000	–	–	–	–
SA	**0.0000**	**0.0000**	**0.0000**	**0.0499**	0.8165	0.9632	–	–	–
Mem	**0.0000**	**0.0000**	**0.0000**	**0.0073**	0.4350	0.7203	0.9999	–	–
ILS	**0.0000**	**0.0000**	**0.0037**	1.0000	0.9783	0.8645	0.1292	**0.0244**	–
MS	**0.0000**	**0.0000**	**0.0000**	0.3638	0.9980	1.0000	0.9982	0.9293	0.6024

ensemble's performance. In this case, the use of weaker classifiers or homogeneous structures in this analysis may modify this scenario;

– One of the stopping conditions for the optimization techniques is time (varying from 30 to 120 minutes). We observed that the processing time was the ending condition used by most of the optimization techniques. When $N = 15$ and $N = 30$, the optimization techniques may need more processing time to validate all the solutions and this might be reflecting in the obtained results. In these cases, they exploit less problem space, affecting their overall performance.

In the second analysis, we compared the performance of the different objective sets. In general, we can conclude that the use of the diversity measures in the mono-objective algorithms did cause a decrease in the performance of the ensemble systems, for all optimization techniques. This was an expected result since we presumed that the use of a diversity measure on its own would not surpass the performance of the error rate in the guide to provide accurate ensemble systems. However, an important observation must be highlighted, the inclusion of one diversity measure, along with error rate, can have a positive affect in the search for accurate ensemble systems. Table 14 summarizes the best results (Table 4) obtained by the ensemble systems when using only error rate (E) and when using error rate along with a diversity measure (EB+EG).

In observing Table 14, we can observe that the inclusion of one diversity measure caused an increase in the obtained best results, for six optimization techniques (TS, GA, ACO, PSO, SA and Memetic). It is important to emphasize that among these six techniques, we can find all analysed population-based techniques and one hybrid technique (Memetic). Therefore, as the population-based

Table 12 Comparison with Classical Techniques

Base	SA	Memetic	Bagging	Boosting	Random	Random Forest
1	**0.1142**	0.1160	0.1369	0.1280	0.1521	0.1726
2	**0.0036**	0.0045	0.0281	0.0225	0.0281	0.0169
3	**0.2398**	0.2478	0.5402	0.5307	0.3301	0.2541
4	0.1940	**0.1803**	0.3505	0.2113	0.2876	0.3144
5	**0.1964**	0.2170	0.3302	0.2642	0.3123	0.3113
6	0.1240	**0.1186**	0.2058	0.2261	0.1562	0.1580
7	0.0624	**0.0618**	0.1355	0.1097	0.1226	0.1097
8	0.0052	**0.0046**	0.0614	0.1228	0.0524	0.0063
9	0.2212	0.2268	0.3143	**0.2095**	0.3076	0.2762
10	**0.0230**	0.0253	0.0368	0.0805	0.0637	0.0414
11	0.0468	0.0510	0.0630	**0.0426**	0.0685	0.0685
12	0.1690	**0.1630**	0.3182	0.2727	0.2354	0.2020
13	**0.0070**	0.0100	0.1909	0.2747	0.1713	0.0414
14	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0281	**0.0000**
15	0.0134	**0.0095**	0.0924	0.1223	0.0878	0.0299
16	0.0528	**0.0483**	0.0883	0.1795	0.1054	0.0712
17	0.2518	**0.2500**	0.3956	0.2747	0.3077	0.3352
18	0.0100	0.0067	0.0693	**0.0000**	0.0554	0.0396
19	**0.0108**	0.0210	0.0604	0.0626	0.0219	0.0191
20	**0.0086**	**0.0086**	0.0273	0.0109	0.0235	0.0464
21	**0.1572**	0.1586	0.2178	0.2195	0.2142	0.2161
22	**0.0000**	**0.0000**	0.2333	0.3833	0.1683	0.2167
23	**0.0000**	**0.0000**	0.6166	0.2500	0.1000	0.0833
NumWin	12	12	1	4	0	1

Table 13 Post Hoc - Comparison with Classical Techniques

x	SA	Memetic	Boosting	Bagging	Random
SA	–	–	–	–	–
Memetic	1.0000	–	–	–	–
Boosting	**0.0000**	**0.0000**	–	–	–
Bagging	**0.0002**	**0.0002**	0.5835	–	–
Random	**0.0001**	**0.0001**	0.7045	0.9997	–
Random Forest	**0.0010**	**0.0020**	0.1693	0.9739	0.9039

Table 14 Comparison of the best results: Error rate (E) and Error rate with one diversity measure (EG+EB)

x	TS	GA	ACO	GRASP	PSO
E	7	4	4	17	11
EG+EB	12	20	14	6	14
x	VNS	SA	Memetic	ILS	MS
E	15	9	8	15	16
EG+EB	10	17	16	5	8

algorithms tend to explore more rapidly the search space, it would be important to use more than one criteria and the use of a diversity measure can help the error rate in the search for more accurate ensemble systems.

In the third analysis of this paper, an analysis of the optimization algorithms, we can conclude that the best optimization technique is Memetic, followed closely by SA and PSO. The interesting aspect of these results is that Memetic is a hybrid algorithm, PSO is a population-based algorithm and SA is a neighborhood-based algorithm. In fact, Memetic uses most of the genetic algorithm functionality along with a local search (neighborhood-based functionality). For instance, the Memetic performance is better than GA performance, showing that the hybridization used in the Memetic algorithm had a positive effect in the automatic design of ensemble systems. Additionally, SA is a neighborhood-based algorithm that emulated a physical process in which a solid is slowly cooled so that when eventually its structure is ”frozen”. This elaborated metaheuristic also led to the design of accurate ensembles, even when compared to some population-based algorithms (ACO and GA). Finally, PSO is a simple population-based algorithm that applies a probabilistic selection of the direction to follow (follow their own path; back to the best position found so far; or follow the path of the best solution in the swarm). This direction process is similar to the processes made by some neighborhood-based algorithms, making PSO a population-based algorithm with some functionalities of a neighborhood-based algorithms. This combination also had a positive effect in the PSO performance to produce accurate ensemble systems.

6 Final remarks

This paper presented different ways to generate ensembles through the use of several optimization algorithms in the design of these systems, that were assessed in both mono and multi-objective context. These optimization algorithms were evaluated when searching the optimal values for the number of individual classifiers and feature subsets to compose the ensemble systems. In this analysis, we aimed at evaluating the effect of using ten optimization techniques in the selection of important components for ensemble systems. Therefore, we wanted to explore the full potential of the optimization algorithms in the automatic design of ensemble systems.

The obtained results showed that the use of the diversity measures in the mono-objective algorithms did cause a decrease in the performance of the ensemble systems, for all optimization techniques. However, an important observation must be highlighted, the inclusion of one diversity measure, along with error rate, can have a positive affect in the search for accurate ensemble systems. The obtained results also showed that the variation in the number individual classifiers did not cause a significant improvement in the performance of ensemble systems and, as it was not expected, the best performance was achieved when using a maximum of 10 individual classifiers ($N = 10$). In addition, when we compare all ten optimization techniques, in a general perspective, we observed that two optimization techniques, Memetic and SA, provided better performance than the other techniques. When comparing these two algorithms with classical ensemble techniques, bagging, boosting, random forest and random selection, we observed that both algorithms delivered more accurate ensemble systems.

The results obtained in this paper are very promising and, as future work, we can think of using other optimization techniques, for example, hybrid metaheuristics (combination of two or more algorithms taking advantage of the synergy between them). Thus, we aim at developing hybrid metaheuristics with the combination of traditional metaheuristics as genetic algorithms and tabu search to provide an algorithm that is more efficient than the individual metaheuristics.

Acknowledgments This work has the financial support of CNPq (Brazilian Research Councils).

References

1. Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res 15:3133–3181
2. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1-1:67–82
3. Kuncheva LI (2004) Combining pattern classifiers: Methods and Algorithms. Wiley
4. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles. Mach Learn 51:181–207
5. Feitosa Neto A, Canuto A, Goldbarg E, Goldbarg M (2011) Optimization Techniques for the Selection of Members and Attributes in Ensemble System. IEEE Proceedings of Congress on Evolutionary Computation (CEC)
6. Feitosa Neto A, Canuto A, Dantas C (2016) Multiobjective optimization techniques for selecting important metrics in the

design of ensemble systems. Computational Intelligence Journal. doi:10.1111/coin.12090
7. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Asunción A, Newman DJ (2007) UCI Machine Learning Repository, University of California at Irvine, http://ics.uci.edu/~mlearn/MLRepository.html
9. Witten IH, Frank E (2005) Data Mining - Pratical Learning Tools and Techniques, 2nd edn. Morgan Kaufmann
10. Brown G, Kuncheva L (2010) "good" and "bad" diversity in majority vote ensembles. In: El Gayar N, Kittler J, Roli F (eds) Multiple Classifier Systems, ser. Lecture Notes in Computer Science, vol 5997. Springer, Berlin Heidelberg, pp 124–133
11. Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 5:553–549
12. Gendreau M, Potvin J (2010) Handbook of Metaheuristics, 2nd edn. Springer
13. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Berkeley
14. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii, vol 6
15. Kirkpatrick S, Gellat DC, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
16. Dorigo M (1992) Optimization, Learning and Natural Algorithms. Phd thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, p 140
17. Dorigo M (1997) Comput Oper Res 24:10971100
18. Goldbarg EFG, Goldbarg MC, de Souza GR (2006) Particle Swarm Optimization Algorithm for the Traveling Salesman Problem. In: Gottlieb J, Raidl GR (eds) EvoCOP 2006, LNCS 3906. Springer, Berlin, pp 99–110
19. Glover F (1996) Tabu Search and adaptive memory programming - advances, applications and challenges. In: Barr R., Helgason R., Kennington J. (eds) Interfaces in Computer Sciences and Operations Research. Kluwer Academic Publishers, pp 1–75
20. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6:109–133
21. Lee KY, El-Sharkawi MA (2008) Modern heuristic optimization techniques - theory and applications to power systems. Wiley-Interscience
22. Santana LE, Canuto AM (2014) Filter-based optimization techniques for selection of feature subsets in ensemble systems. Expert Systems with Applications 41(4 Part 2):1622–1631
23. Khan SA, Nazir M, Riaz N (2013) Optimized features selection for gender classification using optimization algorithms. Turk J Electr Eng Comput Sci 21:1479–1494
24. Wang L, Ni H, Yang R, Pappu V, Fenn MB, Pardalos PM (2014) Feature selection based on meta-heuristics for biomedicine. Optimization Methods and Software 29(4):703–719. doi:10.1080/10556788.2013.834900
25. Sultan NQ, Shamsuddin SM, Hashim SZM, Darus M, Al-Shammari E (2013) Memetic multiobjective particle swarm optimization-based radial basis function network for classificationproblems. Inf Sci 239:165–190
26. Oh D-Y, Gray JB (2013) GA-ensemble: a genetic algorithm for robust ensembles. Comput Stat 28:2333–2347
27. Liu Z, Dai Q, Liu N (2014) Ensemble selection by GRASP. Appl Intell 41:128–144
28. Palanisamy S, Kanmani S (2012) Classifier Ensemble Design using Artificial Bee Colony based Feature Selection. IJCSI Int J Comput Sci Issues 9(3). No 2
29. Zhang T, Dai Q, Ma Z (2015) Extreme learning machines ensemble selection with GRASP. Appl Intell 43:439–459
30. Boussaïd I., Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
31. Monti S, Tamayo P, Mesirov J, Golub T (2003) Consensus clustering – A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning: Functional Genomics Special Issue, 91–118
32. Chen Y, Wong M-L, Li H (2014) Applying Ant Colony Optimization to configuring stacking ensembles for data mining. Expert Systems with Applications 41:2688–2702
33. Chen Y, Zhao Y (2008) A novel ensemble of classifiers for microarray data classification. Appl Soft Comput 8:1664–1669
34. Thiele L, Zitzler E, Knowles J (2006) A tutorial on the performance assessment of stochastic multiobjective optimizers, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Tech. Rep. TIK Report 214
35. Zitzler E, Thiele L, Laumanns M, Fonseca C, da Fonseca V (2003) Performance assessment of multiobjective optimizers: an analysis and review, vol 7
36. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, vol 3
37. Mao S, Jiao LC, Xiong L, Gou S (2011) Greedy optimization classifiers ensemble based on diversity. Patter Recogn 44(6):1245–1261
38. Lysiak R, Kurzynski M, Woloszynski T (2014) Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers. Neurocomputing 126:29–35
39. Slubana B, Lavrača N (2015) Relating ensemble diversity and performance: A study in class noise detection. Neurocomputing 160:120–131

Antonino A. Feitosa Neto received the B.S. degree from the Federal University of Rio Grande do Norte, Brazil, in 2010, the M.Sc. degree from the Federal University of Rio Grande do Norte, Brazil, in 2012 and the Ph.D. degree Federal University of Rio Grande do Norte, in 2016. He is actually a research assistant at the Federal University of Rio Grande do Norte (UFRN). His research interests include ensemble of classifiers and optimization techniques, unsupervised learning, machine learning and computational intelligence.

Anne M. P. Canuto received the B.S. degree from the Federal University of Rio Grande do Norte, Brazil, in 1992, the M.Sc. degree from the Federal University of Pernambuco (UFPE), Brazil, in 1995, and the Ph.D. degree from the University of Kent, in 2001. Currently, she is an associate professor in the informatics and applied mathematics department, Federal University of Rio Grande do Norte. She has published several articles in scientific journals and conferences. Her interests include pattern recognition, clustering algorithms, biometrics, classifier combination and multi-agent systems.