Evolutionary Algorithms Applied to Classifier Ensemble Selection

Eulanda M. dos Santos

[1]Institute of Computing – Federal University of Amazonas (UFAM)
Manaus – AM

{emsantos}@icomp.ufam.edu.br

Abstract. *Classifier ensemble selection is focused on finding the most relevant subset of classifiers among a large pool of initial classifiers. Different non-exhaustive search algorithms may be applied in this problem. Taking into account that the definition of the best search algorithm for finding the best subset of classifiers is still an open question, in this paper, we present an experimental study on comparing five search algorithms applied to the classifier ensemble selection problem. Three multi-objective genetic algorithms and two single-objective evolutionary algorithms, namely Genetic Algorithm and Particle Swarm Optimization, are investigated.*

1. Introduction

Learning algorithms are used to solve tasks for which the design of software using traditional programming techniques is difficult. Machine failures prediction, filter for electronic mail messages and handwritten digits recognition are examples of these tasks. Several different learning algorithms have been proposed in the literature such as Decision Trees, Neural Networks (NN), k Nearest Neighbors (kNN) and Support Vector Machines (SVM). Given a sample x and its class label ω_k with an unknown function $\omega_k = f(x)$, all these learning algorithms focus on finding the best approximation function h, which is a classifier, to the function $f(x)$. Hence, the goal is the design of a robust well-suited single classifier to the problem concerned.

Classifier ensembles attempt to overcome the complex task of designing a robust, well-suited individual classifier by combining the decisions of relatively simpler classifiers. It has been shown that significant performance improvements can be obtained by creating classifier ensembles and combining their members' outputs instead of using single classifiers. Altinçay [Altinçay 2007] showed that ensemble of kNN is superior to single kNN, Zhang [Zhang 2008] concluded that ensemble of SVM outperforms single SVM and Ruta and Gabrys [Ruta and Gabrys 2007] demonstrated performance improvements by combining ensemble of NN, instead of using a single NN.

The construction of classifier ensembles may be performed by adopting different strategies. Two of the most popular methods are Bagging [Breiman 1996] and Random Subspace [Ho 1998]. In addition, two different strategies may be applied when dealing with classifier ensembles: fusion and selection. In the first, all classifier members are combined since it is assumed that they are all important and independent. By contrast, classifier selection chooses one individual classifier to assign the label of the samples. A combination of both approaches is the *overproduce-and-choose strategy* (OCS). The objective of OCS is to find the most relevant subset of classifiers, based on the assumption that there is redundancy among classifiers [Zhou et al. 2002]. Once the best subset of classifiers has been selected, the output of its members must be combined.

OCS is clearly an optimization problem, which may be accomplished by using search algorithms. Given an initial pool of classifiers $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, when performing OCS using search algorithms, it is important to choose the best search criterion and the best search algorithm. Recognition rate and diversity are two classical search criteria employed in the literature. In terms of search algorithm, Sharkey and Sharkey [Sharkey and Sharkey 2000] proposed an exhaustive search algorithm to find the best subset of classifiers from an initial pool composed of NNs. The candidate ensembles' recognition rates were used to guide the search process. Based on the same idea, Zhou et al. [Zhu et al. 2004] developed equations, which were used to identify the classifiers that should be eliminated from $\mathcal{C}$ in order to keep the combination with optimal accuracy. Their initial pool of classifiers was also composed of NNs. Nonetheless, non-exhaustive search algorithms might not be used when a large $\mathcal{C}$ is available due to the high computing complexity of an exhaustive search, since the size of $\mathcal{P}(\mathcal{C})$ is 2^n, where $\mathcal{P}(\mathcal{C})$ is the powerset of $\mathcal{C}$ defining the population of all possible candidate ensembles.

Different non-exhaustive search algorithms have been applied in the literature for the selection of classifier ensembles. Roli et al. [Roli et al. 2001] compared forward (FS), backward (BS) and tabu (TS) search algorithms, guided by recognition rate and three diversity measures. Their initial pool $\mathcal{C}$ was composed of heterogeneous classifiers such as NN and kNN. They concluded that the search criteria and the search algorithms investigated presented equivalent results. Following the idea of identifying the best search criteria and search algorithm, Ruta and Gabrys [Ruta and Gabrys 2005] used the candidate ensembles' error rate and 12 diversity measures to guide the following 5 single-objective search algorithms: FS, BS, Genetic Algorithm (GA), stochastic hill-climbing (HC) search and population-based incremental learning (PBIL). They concluded that the Greedy algorithms FS and BS outperformed the remaining methods. However, Ruta and Gabrys [Ruta and Gabrys 2005] observed that unlike Greedy algorithms, evolutionary algorithms allow the possibility of dealing with a population of classifier ensembles rather than one individual best candidate ensemble at each iteration. This property can be important in performing a post-processing phase.

Therefore, since the definition of the best search algorithm for finding the best subset of classifiers is still an open question, several search methods may be used. In terms of evolutionary algorithms, different methods are available in the literature, for instance single- and *multi-objective* GA (MOGA) and Particle Swarm Optimization (PSO). In this paper, we present an experimental study on comparing GA, MOGA and PSO applied to the classifier ensemble selection problem. First, three different MOGAs are investigated, namely (1) non-dominated sorting GA (NSGA) [Deb 2001]; (2) elitist non-dominated sorting GA (NSGA-II) [Deb et al. 2000]; and (3) controlled elitist NSGA [Deb and Goel 2001]. Second, two single-objective evolutionary algorithms are compared: GA and PSO. Finally, GA and NSGA-II are compared since, in our initial experiments, these algorithms are identified as the best single- and multi-objective search algorithms respectively. In this paper, the Random Subspace method is used to generate ensemble members, while kNN classifiers are used for the creation of homogeneous ensembles. The objective functions employed is the ensemble's classification error rate, ensemble size and a diversity measure.

The paper is organized as follows. The definition of classifier ensemble selection is presented in section 2. In section 3, the search algorithms investigated are described. Then, the experiments and the results are presented in section 4. Conclusions and suggestions for future work are discussed in section 5.

2. Classifier Ensemble Selection

Methods based on OCS are divided into two phases: (1) *overproduction*; and (2) *selection*. The overproduction phase must construct an initial *large* pool of candidate classifiers $\mathcal{C}$, using a training dataset $\mathcal{T}$. The second phase is devoted to identify the best performing subset of classifiers in $\mathcal{P}(\mathcal{C})$. The selected ensemble C_j^* is then combined to estimate the class labels of the samples contained in the test dataset $\mathcal{G}$. Figure 1 illustrates the OCS phases.

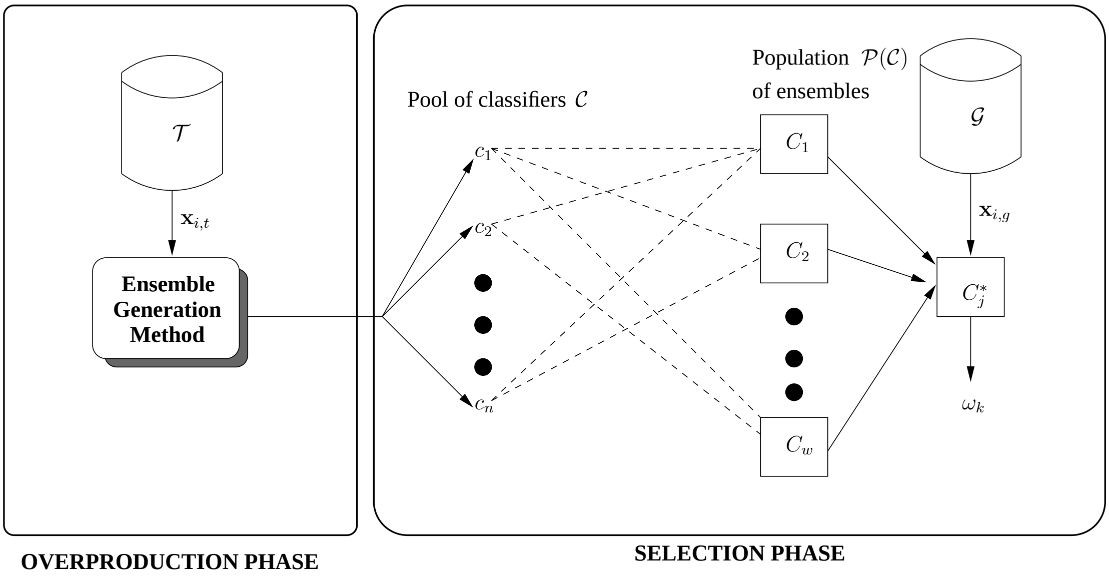
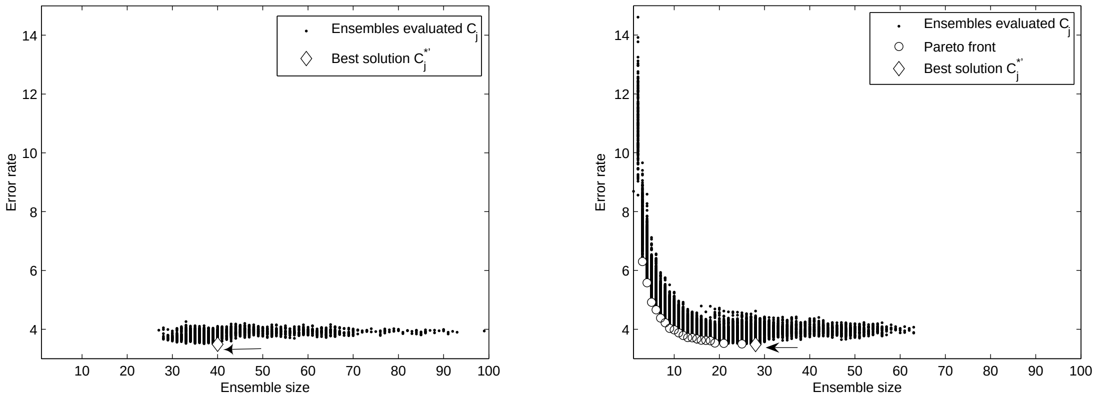


Figure 1. Overview of the OCS process. OCS is divided into the overproduction and the selection phases. The overproduction phase creates a large pool of classifiers, while the selection phase focus on finding the most performing subset of classifiers.

Single- and *multi-objective* search algorithms are the two strategies available when dealing with the selection phase of OCS. Figure 2(a) depicts an example of the evolution of the optimization process for $1,000$ generations using single-objective GA as the search algorithm and the minimization of the error rate as the objective function. Even though we employed the error rate as objective function, we show in Figure 2(a) plots of the error rate versus number of classifiers to better illustrate the problem. Each point on the plot corresponds to a candidate ensemble C_j taken from $\mathcal{P}(\mathcal{C})$ and evaluated during the optimization process. Indeed, these points represent the complete search space explored for $1,000$ generations. The number of individuals at any generation is 128. When using single-objective GA, the solution with lowest error rate is selected as the best solution C_j^*, which is further used to classify the test samples. Diamond represents C_j^* in Figure 2(a).



(a) (b)

Figure 2. Optimization using single-objective GA with the error rate as the objective function in Figure 2(a). Optimization using NSGA-II and the pair of objective functions: the error rate and ensemble size in Figure 2(b). The best performing solutions are highlighted by arrows and circles represent the Pareto front.

When using multi-objective search algorithms, it is possible to conduct optimization using simultaneous multi-objective functions. An optimization process performed by NSGA-II for $1,000$ generations, with 128 individuals at each generation, is illustrated in Figure 2(b). NSGA-II was employed, using the pair of objective functions: jointly minimize the error rate and the ensemble size. Circles on the plot represent the Pareto front. Due to the fact that all solutions over the Pareto front are equally important, the selection of the best candidate ensemble C_j^* is more complex. Several works reported in the literature take into account only one objective function to perform the selection. In [Radtke et al. 2009], the candidate ensemble with lowest error rate was chosen as the best solution C_j^*, even though the optimization process was guided regarding multi-objective functions. In this paper, we also select the solution with lowest error rate as C_j^* to classify the test samples since it allows the main objective in pattern recognition, i.e. finding predictors with a high recognition rate, and it avoids additional experiments in our work. Diamond indicates the solution C_j^* in Figure 2(b). The evolutionary algorithms investigated in our experiments are described in the next section.

3. Evolutionary Algorithms

As mentioned in the introduction, we compare two single-objective evolutionary algorithms and three multi-objective GA. In this section, we summarize these algorithms. First, we describe PSO and GA.

3.1. Single-Objective Particle Swarm Optimization

PSO simulates the behaviors of bird flocking or fish schooling. Each individual of the population is called particle. All particles have fitness values which are evaluated during the optimization process of the selection phase of OCS. Algorithm 1 summarizes the variant of the PSO algorithm used in this paper. This variant is called *global neighborhood*

[Parsopoulos and Vrahatis 2002] in the literature. The solution $gBest$ denotes the best particle in the whole population. In PSO, the population is called swarm.

Algorithm 1 PSO

```
for each particle do
    Initialize particle
end for
while maximum iterations or stop criteria are not attained do
    for each particle do
        Compute fitness value
        if fitness value is better than the best fitness ($pBest$) in history then
            Set current value as the new $pBest$
        end if
    end for
    Choose the particle with the best fitness as the $gBest$
    for each particle do
        Compute its velocity
        Update its position
    end for
end while
```

3.2. Single-Objective GA

Single- and *multi-objective* GA are the two strategies available when dealing with GAs. Traditionally, when the optimization process is conducted as a single-objective problem, GA is guided by an objective function during a fixed maximum number of generations (user defined $max(g)$). The selection of classifier ensembles is applied in the context of GA based on binary vectors. Each individual, called chromosome, is represented by a binary vector with a size n, since the initial pool of classifiers is composed of n members. Initially, a population with a fixed number of chromosomes is randomly created, i.e. a random population of candidate classifier ensembles. Thus, at each generation step g, the algorithm calculates fitness of each candidate ensemble in the population $\mathbf{C}(g)$, which is the population of ensembles found at each generation g. The population is evolved through the operators of crossover and mutation. Algorithm 2 summarizes single GA.

Algorithm 2 Single-Objective GA

```
Creates initial population $\mathbf{C}(1)$ of $w$ chromosomes
Find $C_j^*(1)$
for each generation $g \in \{1, \ldots, max(g)\}$ do
    perform all genetic operators
    generate new population $\mathbf{C}(g+1)$ and find $C_j^*(g+1)$
end for
return $C_j^*$
```

3.3. Multi-Objective GA

MOGAs often constitute solutions to optimization processes guided by multi-objective functions. These algorithms use Pareto dominance to reproduce the individuals. A Pareto front is a set of nondominated solutions representing different tradeoffs between the multi-objective functions. In our classifier ensemble selection application, a candidate ensemble solution C_i is said to dominate solution C_j, denoted $C_i \preceq C_j$, if C_i is no worse than C_j on all the objective functions and C_i is better than C_j in at least one objective function. Based on this non-domination criterion, solutions over the Pareto front are considered to be equally important.

Among several Pareto-based evolutionary algorithms proposed in the literature, NSGA-II has two important characteristics: a full elite-preservation strategy and a diversity-preserving mechanism using the crowding distance as the distance measure. The crowding distance does not need any parameter to be set [Deb 2001]. Elitism is used to provide the means to keep good solutions among generations, and the diversity-preserving mechanism is used to allow a better spread among the solutions over the Pareto front. In addition, in this paper, we investigated two variants of NSGA-II: (1) NSGA [Deb 2001]; and (3) controlled elitist NSGA [Deb and Goel 2001]. In the first case, a niche distance parameter must be defined. This parameter, which indicates the maximum distance allowed between any two solutions to participate into a niche, is used to control the number of solutions allowed to be concentrated over small regions (niches) over the Pareto front. In the case of controlled elitist NSGA, the portion of the population that is allowed to keep the best non-dominated solutions must be set, which controls the extent of elitism. In order to reduce redundancy, only NSGA-II is described bellow.

NSGA-II works as follows. At each generation step g, a parent population $\mathbf{C}(g)$ of size w evolves and an offspring population $\mathbf{C}^q(g)$, also of size w, is created. These two populations are combined to create a third population $\mathbf{C}^r(g)$ of size $2w$. The population $\mathbf{C}^r(g)$ is sorted according to the nondominance criteria, and different nondominated fronts are obtained. Then, the new population $\mathbf{C}(g+1)$ is filled by the fronts according to the Pareto ranking. In this way, the worst fronts are discarded, since the size of $\mathbf{C}(g+1)$ is w. When the last front allowed to be included in $\mathbf{C}(g+1)$ has more solutions than the $\mathbf{C}(g+1)$ available free space, the crowding distance is measured in order to select the most isolated solutions in the objective space in order to increase diversity. Algorithm 3 summarizes NSGA-II.

4. Experiments

The experiments were conducted using an initial pool of 100 kNN (k=1) classifiers generated using the Random Subspace method (RSS) during the overproduction phase. Parameters such as: k value, the number of subspace dimensions and the number of classifier members were defined experimentally. Our experiments were broken down into three main series. In the first, NSGA-II, NSGA and controlled elitist NSGA are compared in order to identify the best MOGA for classifier ensemble selection. In the second series, PSO and GA are also compared. The objective is to point out the best single-objective method. Finally, in the third series, the best search algorithm identified in both previous series are compared to each other.

It is important to mention that all the experiments were replicated 30 times and the results were tested on multiple comparisons using the Kruskal-Wallis nonparametric statistical test by testing the equality between mean values. The confidence level was 95% ($\alpha = 0.05$), and the Dunn-Sidak correction was applied to the critical values. Classical

Algorithm 3 NSGA-II

```
Creates initial population $\mathbf{C}(1)$ of $w$ chromosomes
while $g < max(g)$ do
    creates $\mathbf{C}^q(g)$
    set $\mathbf{C}^r(g) = \mathbf{C}(g) \cup \mathbf{C}^q(g)$
    perform a nondominated sorting to $\mathbf{C}^r(g)$ and identify different fronts $\mathbf{C}_k$, $k = 1, 2, \ldots, etc$
    while $|\mathbf{C}(g+1)| + |\mathbf{C}_k| \leq w$ do
        set $\mathbf{C}(g+1)$:=$\mathbf{C}(g+1) \cup \mathbf{C}_k$
        set $k$:=$k+1$
    end while
    perform crowding distance sort to $\mathbf{C}_k$
    set $\mathbf{C}(g+1)$:=$\mathbf{C}(g+1) \cup \mathbf{C}_k\lfloor 1 : (w - |\mathbf{C}(g+1)|)\rfloor$
    creates $\mathbf{C}^q(g+1)$ from $\mathbf{C}(g+1)$
    set $g$:=$g+1$
end while
```

Table 1. Specifications of the datasets used in the experiments

Dataset	# of features	Training Set ($\mathcal{T}$)	Optimization Set ($\mathcal{O}$)	Validation Set ($\mathcal{V}$)	Test Set ($\mathcal{G}$)	Features RSS	Pool $\mathcal{C}$ size
NIST-digits	132	5,000	10,000	10,000	test1 60,089 test2 58,646	32	100
NIST-letters	132	43,160	3,980	7,960	12,092	32	100

holdout validation strategy is employed for the evaluation of performance. Thus, the original datasets are partitioned into four independent datasets: train, optimization, validation and test. The first partition is used to train ensemble members; the optimization partition is used during the search process; validation partition is employed to set parameters; and test set is used to performance evaluation.

4.1. Multi-Objetive Genetic Algorithms Comparison

The experiments were carried out using the NIST Special Database 19 (NIST SD19) which is a popular database used to investigate digit recognition algorithms. It was originally divided into two test sets: data-test1 (60,089 samples) and data-test2 (58,646 samples). In this first series of experiments, only data-test1 investigated. On the basis of the results available in the literature, the representation proposed by Oliveira et al. [Oliveira et al. 2002] appears to be well defined and well suited to the NIST SD19 database. The features are a combination of the concavity, contour and surface of characters. The final feature vector is composed of 132 components: 78 for concavity, 48 for contour and 6 for surface. Table 1 lists important information about the database and the partitions used to compose the four separate datasets.

Three search criteria are investigated: error rate, ensemble size and diversity measure. Ambiguity (as defined in equations 1 and 2) is used as the diversity measure. Taking into account that it has been shown in the literature that ensemble size and diversity

Table 2. Genetic Algorithms parameters

Population size	128
Number of generations	1000
Probability of crossover	0.8
Probability of mutation	0.01
One-point crossover and bit-flip mutation	

measures are not conflicting objective functions [DosSantos et al. 2008], the three search criteria are not combined together as objective functions in this paper. Instead, two pairs of objective functions are used to guide the three MOGAs: the maximization of ambiguity combined with the minimization of the error rate; and the minimization of ensemble size combined with the minimization of the error rate.

Ambiguity - The classification ambiguity is defined as:

$$a_i(x) = \begin{cases} 0 & \text{if } y_i = \omega_k \\ 1 & \text{otherwise} \end{cases} \qquad (1)$$

where a_i is the ambiguity and y_i is the output of the i^{th} classifier on the observation x, and ω_k is the candidate ensemble output. Given n the number of samples and l the number of classifiers, the ambiguity of the ensemble C_j is:

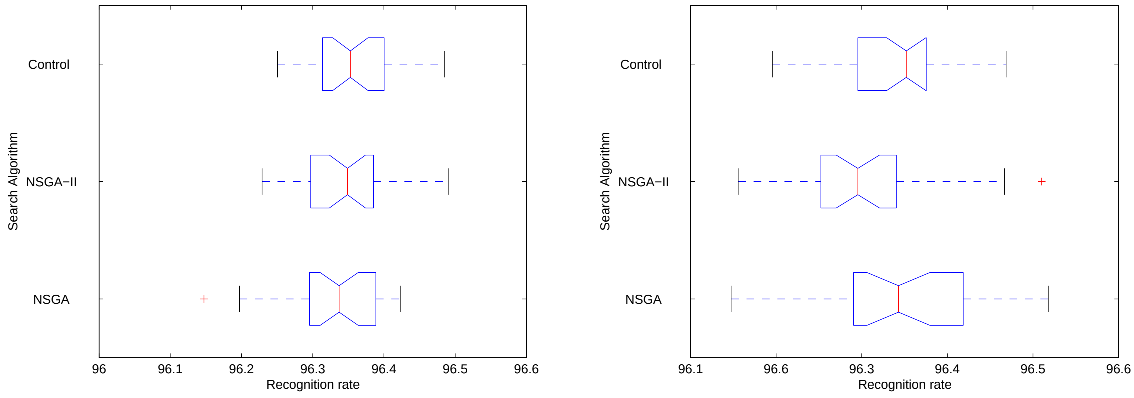
$$\gamma = \frac{1}{n.l} \sum_{i \in C_j} \sum_{x \in X} a_i(x) \qquad (2)$$

The general GA parameters presented in Table 2, are used for all three MOGAs. However, NSGA and controlled elitist NSGA need some parameters to be set. In the first case, a niche distance parameter must be defined. In the second case, the portion of the population that is allowed to keep the best non-dominated solutions must be set. The level of elitism defined is equal to 50% for controlled elitist NSGA. In terms of NSGA's parameters, we set the best niche distance values also experimentally, as defined bellow. The level of elitism for controlled elitist NSGA is also mentioned.

- NSGA guided by ambiguity and the error rate: niche distance=0.025;
- NSGA guided by ensemble size and the error rate: niche distance=0.05;
- Controlled elitist NSGA elitism level: 50%.

Figure 3(a) shows the comparison results of 30 replications on data-test1 obtained using ambiguity and the error rate as objective functions. These results show that controlled elitism NSGA and NSGA-II presented equivalent performances, while NSGA was slightly worse.

The results obtained by combining ensemble size and the error rate in a pair of objective functions to guide the three MOGAs are shown in Figure 3(b). Using this second pair of objective functions, we observe that controlled elitism NSGA and NSGA presented equivalent performances. Thus, unlike our previous results, NSGA-II was the worst search algorithm. However, it is important to note that the variance of the recognition rates achieved by each algorithm is small (from 96.1% to 96,6%).



(a) MOGAs guided by ambiguity and the error rate

(b) MOGAs guided by ensemble size and the error rate

Figure 3. Results of 30 replications using NSGA, NSGA-II and controlled elitist NSGA.

Table 3. Comparing GA and PSO in terms of error rate. Values are shown in bold when the error rates are significantly different. The results were calculated using NIST-letters and NIST-digits.

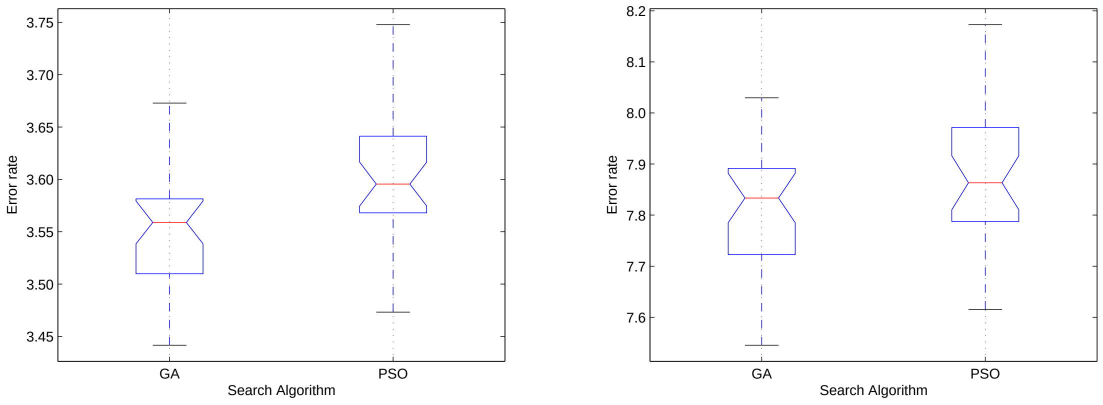
Search Algorithm	data-test1	data-test2	NIST-letters
GA	**3.55**	7.80	6.44
PSO	3.62	7.88	6.49

Therefore, our results globally showed that the three MOGAs investigated in this series of experiments are equally competent to guide the optimization process involved at the selection phase of OCS. Since no search algorithm can be claimed to be the best, NSGA-II is the MOGA applied in the experiments presented further. This choice is specially due to the fact that NSGA-II does not need any parameter to be set.

4.2. Single-Objective Evolutionary Algorithms Comparison

In order to develop our experiments on comparing GA and PSO, we use the two datasets described in Table 1: NIST-digits (both test sets) and NIST-letters. The same initial pool of 100 kNN classifiers generated by applying the Random Subspace method is investigated here. In addition, only the error rate was used as objective function. Finally, in both search algorithms, individuals (particles) are represented by binary vectors, the number of generations (iterations) was 1000 and the number of individuals (particles) was 128.

Table 3 shows the average results from 30 replications obtained using both databases. Values are shown in bold when the performances are significantly different, according to the Kruskal-Wallis statistical test. It can be observed that GA generates solutions that are better in generalization than PSO taking into account data-test1 of NIST-digits. This result is better illustrated in Figure 4(a). In data-test2 (Figure 4(b)), GA was slightly better than PSO. The average results from 30 replications obtained using NIST-letters dataset are illustrated in Figure 5. This result confirms that the performance of the



(a) data-test1

(b) data-test2

Figure 4. NIST-digits: error rates of the solutions found on 30 replications using GA and PSO. The performances were calculated on the data-test1 and data-test2.

solutions found by GA were slightly better than the solutions found by PSO. Due to this result, GA is used to be compared to NSGA-II in our third series of experiments.

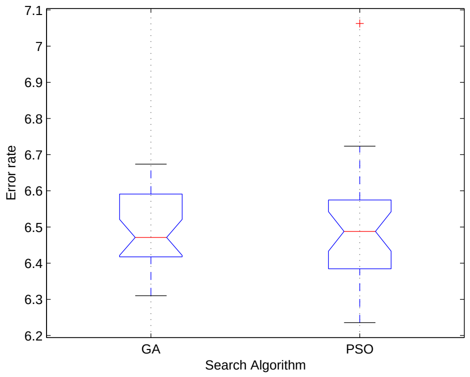


Figure 5. NIST-letters: error rates of the solutions found on 30 replications using GA and PSO.

4.3. Single and Multi–Objective Genetic Algorithms Comparison

In order to compare NSGA-II and GA, we employed NSGA-II guided by ambiguity combined with the error rate, since this pair of objective functions outperformed error rate combined with ensemble size in our first series of experiments. GA was guided by the minimization of the error rate. Table 4 summarizes the mean of the error rates obtained on all three databases investigated on comparing GA and NSGA-II. These results indicate, without exception, that GA was better than NSGA-II. It is important to note that differences between the results achieved by both search algorithms were not significant. Moreover, only one pair of objective functions was used to guide NSGA-II. A different combination of objective functions may help NSGA-II to find solutions with better performances.

Table 4. Mean of the error rates obtained on 30 replications comparing GA and NSGA-II.

Dataset	NSGA-II	GA
NIST-digits Test1	3.63	3.55
NIST-digits Test2	7.87	7.80
NIST-letters	6.54	6.44

5. Conclusion

This work presented the experimental results of a study comparing five search algorithms applied to the selection of classifier ensembles performed as optimization problems using single- and multi-objective search algorithms. The experiments were divided into three series. In the first series, three multi-objective genetic algorithms were compared in one database. A diversity measure and the ensemble size were combined with the error rate to make up pairs of objective functions to guide these algorithms. In the second series of experiments, single-objective GA and PSO were compared in two databases. Finally, in the third series of experiments, the best multi-objective GA was compared to the best single-objective algorithm.

The results show that all multi-objective GAs are equivalent. Then, NSGA-II was picked up to be compared to single GA, since this algorithm outperformed PSO. Based on the results found in the third series of experiments, single GA was slightly better than NSGA-II as search strategy for classifier ensemble selection in our experiments. However, since only one pair of objective functions was employed, these results could be different according to the objective functions used. The next stage of this research will involve to test different search criteria in order to determine strategies to improve the selection results.

References

Altinçay, H. (2007). Ensembling evidential k-nearest neighbor classifiers through multi-modal pertubation. *Applied Soft Computing*, 7(3):1072–1083.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, LTD.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France.

Deb, K. and Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Proceedings of First Evolutionary Multi-Criterion Optimization*, pages 67–81.

DosSantos, E. M., Sabourin, R., and Maupin, P. (2008). Pareto analysis for the selection of classifier ensemble. In *Proceedings of ACM Genetic and Evolutionary Computation Conference*, pages 681–688.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Oliveira, L., Sabourin, R., Bortolozzi, F., and Suen, C. (2002). Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1438–1454.

Parsopoulos, K. and Vrahatis, M. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2):235–306.

Radtke, P., Sabourin, R., and Wong, T. (2009). Solution over-fit control in evolutionary multiobjective optimization of pattern classification systems. *nternational Journal of Pattern Recognition and Artificial Intelligence*, 23(1):1107–1127.

Roli, F., Giacinto, G., and Vernazza, G. (2001). Methods for designing multiple classifier systems. In *Proceedings of the International Workshop on Multiple Classifier System*, pages 78–87.

Ruta, D. and Gabrys, B. (2005). Classifier selection for majority voting. *Information Fusion*, 6(1):163–168.

Ruta, D. and Gabrys, B. (2007). Neural network ensembles for time series prediction. In *Proceedings of the IEEE International Joint Conference on Neural Network*, pages 1204–1209, Orlando, USA.

Sharkey, A. and Sharkey, N. (2000). The ”test and select” approach to ensemble combination. In Kittler, J. and Roli, F., editors, *Multiple Classifier System*, pages 30–44. Spring-Verlag.

Zhang, Z. (2008). Mining relational data from text: from strictly supervised to weakly supervised learning. *Information Systems*, 33(3):300–314.

Zhou, Z.-H., Wu, J., and Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2):239–263.

Zhu, X., Wu, X., and Yang, Y. (2004). Dynamic selection for effective mining from noisy data streams. In *Proceedings of Fourth IEEE International Conference on Data Mining*, pages 305–312.