Towards the use of metaheuristics for optimizing the combination of classifier and cluster ensembles

Luiz F. S. Coletta*, Eduardo R. Hruschka*, Ayan Acharya† and Joydeep Ghosh†
*Department of Computer Science
University of Sao Paulo (USP) at Sao Carlos, Brazil
{luizfsc,erh}@icmc.usp.br
†Department of Electrical & Computer Engineering
University of Texas (UT) at Austin, USA
aacharya@utexas.edu, ghosh@ece.utexas.edu

***Abstract*—Unsupervised models can provide supplementary soft constraints to help classify new data since similar instances are more likely to share the same class label. In this context, we investigate how to make an existing algorithm, named C^{3}E (from Combining Classifier and Cluster Ensembles), more user-friendly by automatically tunning its main parameters with the use of metaheuristics. In particular, the C^{3}E algorithm is based on a general optimization framework that takes as input class membership estimates from existing classifiers, as well as a similarity matrix from a cluster ensemble operating solely on the new (target) data to be classified, and yields a consensus labeling of the new data. To do so, two parameters have to be defined *a priori*, namely: the relative importance of classifier and cluster ensembles and the number of iterations of the algorithm. In some practical applications, these parameters can be optimized via (time consuming) grid search approaches based on cross-validation procedures. This paper shows that metaheuristics can be more computationally efficient alternatives to optimize such parameters. More precisely, analyses of statistical significance made from experiments performed on fourteen datasets show that five metaheuristics can provide classifiers as accurate as those obtained from grid search, but taking half the running time.**

I. Introduction

The combination of multiple classifiers to generate a single classifier has been an active area of research for the last two decades [1], [2]. For instance, an analytical framework to quantify the improvements in classification results due to combining multiple models has been addressed in [3]. More recently, a survey of ensemble techniques — including applications of them to many difficult real-world problems such as remote sensing, person recognition, one vs. all recognition, and medicine — has been presented in [4]. In brief, the literature on the subject has shown that from independent, diversified classifiers, the ensemble created is usually more accurate than its individual components. Analogously, just as ensemble learning has been proved to be more useful compared to single-model solutions for classification problems, several research efforts have shown that cluster ensembles can improve the quality of results as compared to a single clustering solution — *e.g.*, see [5].

In this paper, we investigate the use of metaheuristics to estimate values for user-defined parameters of an algorithm that combines ensembles of classifiers and clusterers. Most of the motivations for combining ensembles of classifiers and clusterers are similar to those that hold for the standalone use of either classifier ensembles or cluster ensembles. However, some additional nice properties can emerge from such a combination — *e.g.*, unsupervised models can provide a variety of supplementary constraints for classifying new (target) data [7]. From this viewpoint, the underlying assumption is that similar new instances in the target set are more likely to share the same class label. Thus, the supplementary constraints provided by the cluster ensemble can be useful for improving the generalization capability of the resulting classifier.

In particular, Acharya et al. [6] introduced a framework that combines ensembles of classifiers and clusterers to generate a more consolidated classification. In this framework, an ensemble of classifiers is first learned on an initial labeled training dataset. These classifiers are then used to obtain initial estimates of class probability distributions for new, unlabeled (target) data. In addition, a cluster ensemble is applied to the target data to yield a similarity matrix that, by its turn, is used to refine the initial class probability distributions obtained from the classifier ensemble. This framework is materialized through an optimization algorithm that exploits properties of Bregman divergences[1] in conjunction with Legendre duality to yield a principled and scalable approach. As we discuss later, by using a squared loss function, the algorithm gets simpler, and there are only two user-defined parameters — as opposed to the more general version of the algorithm, which has three user-defined parameters. The study of the optimization of such parameters, by means of metaheuristics, is the main contribution of this work.

The remainder of the paper is organized as follows. Section II reviews the combination of classifier and cluster ensembles as done by the C^{3}E Algorithm, with particular emphasis on its simpler version based on a squared loss function, which was not explicitly studied in [6]. Section III addresses the five metaheuristics used in our study to optimize the user-defined parameters of C^{3}E, namely: three Evolutionary Algorithms — (1+1)-Evolutionary Strategy (ES) [8], [9], Differential Evolution (DE) [10], and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11] — as well as two other nature-inspired algorithms that also perform some kind of stochastic guided search — Simulated Annealing (SA) [12] and Particle

[1]Bregman Divergences include a large number of useful loss functions such as the well-known Squared Loss, KL-divergence, Mahalanobis Distance, and I-Divergence.

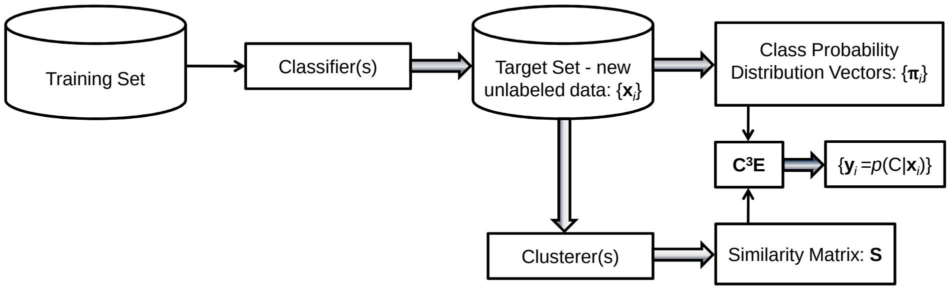
Training Set
Classifier(s)
Target Set - new unlabeled data: {$\mathbf{x}_i$}
Class Probability Distribution Vectors: {$\boldsymbol{\pi}_i$}
C³E
{$\mathbf{y}_i = p(C|\mathbf{x}_i)$}
Clusterer(s)
Similarity Matrix: **S**

Fig. 1. Framework to combine ensembles of classifiers and clusterers using C^{3}E algorithm [6].

Swarm Optimization (PSO) [13]. In Section IV, we present our experimental results. Finally, Section V presents concluding remarks.

II. Combination of Classifier and Cluster Ensembles

In [6], the authors designed a framework that combines classifiers and clusterers to generate a more consolidated classification. This framework, whose core is the C^{3}E algorithm, is depicted in Figure 1. It is assumed that a set of classifiers (consisting of one or more classifiers) have been previously induced from a training set. Such an ensemble of classifiers is employed to estimate initial class probability distributions for every instance $\mathbf{x}_i$ of a target set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$. These probability distributions are stored as c-dimensional vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$ (c is the number of classes) and will be refined with the help of a cluster ensemble. From this viewpoint, the cluster ensemble provides supplementary constraints for classifying the instances of $\mathcal{X}$, with the rationale that similar instances are more likely to share the same class label. In order to capture similarities between the instances of $\mathcal{X}$, C^{3}E takes as input a similarity (co-association) matrix $\mathbf{S}$, where each entry corresponds to the relative co-occurrence of two instances in the same cluster [5], [14] — considering all the data partitions that form the cluster ensemble induced from $\mathcal{X}$.

To sum up, C^{3}E receives as inputs a set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$ and the similarity matrix $\mathbf{S}$, and outputs a consolidated classification for every instance in $\mathcal{X}$ — represented by a set of vectors $\{\mathbf{y}_i\}_{i=1}^n$, where $\mathbf{y}_i = p(C \mid \mathbf{x}_i)$ — i.e., $\mathbf{y}_i$ is the estimated posterior class probability assignment for every instance in $\mathcal{X}$. To do so, C^{3}E solves an optimization problem whose objective is to minimize J in (1) with respect to the set of probability vectors $\{\mathbf{y}_i\}_{i=1}^n$:

$$J = \sum_{i \in \mathcal{X}} \mathcal{L}(\mathbf{y}_i, \boldsymbol{\pi}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} \mathcal{L}(\mathbf{y}_i, \mathbf{y}_j) \tag{1}$$

The quantity $\mathcal{L}(\cdot,\cdot)$ denotes a loss function. Informally, the first term in Equation (1) captures dissimilarities between the class probabilities provided by the ensemble of classifiers and the output vectors $\{\mathbf{y}_i\}_{i=1}^n$. The second term encodes the cumulative weighted dissimilarity between all possible pairs $(\mathbf{y}_i, \mathbf{y}_j)$. The weights to these pairs are assigned in proportion to the similarity values $s_{ij} \in [0,1]$ of matrix $\mathbf{S}$, and the coefficient $\alpha \in \mathbb{R}_+$ controls the relative importance of classifier and cluster ensembles.

The C^{3}E algorithm, as proposed in [6], exploits general properties of a large class of loss functions, described by Bregman divergences [15], in conjunction with Legendre duality and a notion of variable splitting that is also used in alternating direction method of multipliers [16] to yield a principled and scalable solution. If one chooses a squared loss function, the C^{3}E algorithm gets simpler. In particular, the variable splitting approach, in which two copies of $\mathbf{y}_i$ are updated iteratively, is not needed anymore. As such, we can get rid of one of the user-defined parameters of the algorithm, λ, which is an optimization constraint to ensure that the two copies of the variables remain close during the optimization process. Also, the update equations are still available in closed form solution for each $\mathbf{y}_i$, as explained next.

A. *C^{3}E with Squared Loss Function*

Choosing the Squared Loss (SL) function as the Bregman divergence in the optimization problem formulated in Equation (1) we get:

$$J_{SL} = \frac{1}{2} \sum_{i \in \mathcal{X}} \|\mathbf{y}_i - \boldsymbol{\pi}_i\|^2 + \alpha \frac{1}{2} \sum_{(i,j) \in \mathcal{X}} s_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \tag{2}$$

The second term of J_{SL} shows that the variables are coupled. In order to circumvent this difficulty, and following an approach analogous to the one adopted for the more general case [6] — where any Bregman Divergence can be used — we can design an iterative update procedure of the variables to be optimized. In particular, keeping $\{\mathbf{y}_j\}_{j=1}^n \setminus \{\mathbf{y}_i\}$ fixed, we can minimize J_{SL} in Equation (2) for every $\mathbf{y}_i$ by setting:

$$\frac{\partial J_{SL}}{\partial \mathbf{y}_i} = \mathbf{0}. \tag{3}$$

Considering that the similarity matrix $\mathbf{S}$ is symmetric, and noting that $\frac{\partial \|\mathbf{x}\|^2}{\partial \mathbf{x}} = 2\mathbf{x}$, we get:

$$\mathbf{y}_i = \frac{\boldsymbol{\pi}_i + \alpha' \sum_{j \neq i} s_{ij} \mathbf{y}_j}{1 + \alpha' \sum_{j \neq i} s_{ij}}, \tag{4}$$

where we set $\alpha' = 2\alpha$ for mathematical convenience. Equation (4) can be computed iteratively by using the Algorithm 1, which summarizes the main steps of C^{3}E with Squared Loss function. Since each update of $\mathbf{y}_i$ contributes to minimize J_{SL}, the algorithm converges. A stopping criterion can be defined as either the maximum number of iterations, I, or a predefined threshold on the difference between values of objective function in (2) computed from two consecutive iterations of the algorithm.

Algorithm 1: C^{3}E with Squared Loss function – C^{3}E-SL

```
   Input: $\{\boldsymbol{\pi}_i\}, \mathbf{S}, \alpha$.
   Output: $\{\mathbf{y}_i\}$.
 1 Initialize $\{\mathbf{y}_i\}$ such that $\mathbf{y}_{i\ell} = \frac{1}{c}\ \forall i \in \{1, 2, ..., n\}$,
   $\forall \ell \in \{1, 2, ..., c\}$;
 2 Repeat
 3 |   Update $\mathbf{y}_i$ using Equation (4) $\forall i \in \{1, 2, ..., n\}$;
 4 until convergence;
```

The asymptotic time complexity of the algorithm is $O(c \cdot n^2)$, where c is the number of class labels and n is the number of instances in the target set. Note that n, the number of new instances to be classified, is usually much less that the number of instances in the training set — used to build the classifier ensemble. Furthermore, as in the more general case addressed in [6], the resulting minimization procedure can be performed in parallel — by updating one or more variables per processor.

In practice, the choice of the Bregman divergence is dependent on the application domain. At this point, however, we can anticipate that our empirical results are very similar to those found in [6] across a variety of datasets, thereby suggesting that, as usual, simpler approaches should be tried first. In this sense, we investigate how to automatically optimize the user-defined parameters of the algorithm based on the Squared Loss function (C^{3}E-SL), namely: the coefficient α, which controls the relative importance of classifier and cluster ensembles, and the number of iterations (I) of the algorithm.

As expected, this is a difficult (multi-modal) optimization problem — as Figure 2 illustrates for the *Wine Red Quality* dataset [17]. It is widely known that metaheuristics have been successfully used in this kind of problem. Therefore, as a starting point, we decided to investigate the use of some popular metaheuristics to optimize the parameters of C^{3}E-SL.

III. Optimizing the C^{3}E-SL Parameters with Metaheuristics

A variety of metaheuristics have been designed to address problems that cannot be suitably tackled through traditional optimization algorithms. In multi-modal optimization problems, such as those tackled when optimizing the C^{3}E-SL parameters, it is of particular interest to use algorithms capable of escaping from local optima, hopefully being able to reach the global optimum — or at least a very good local optimum solution. Metaheuristics have shown to be promising approaches to solve this kind of problem.

Essentially, metaheuristics consist of stochastic algorithms with randomization and local search that are used for global optimization, including evolutionary-based algorithms [18], [19], and also those based on thermodynamic principles [12] and swarm intelligence [13]. In general, metaheuristics have

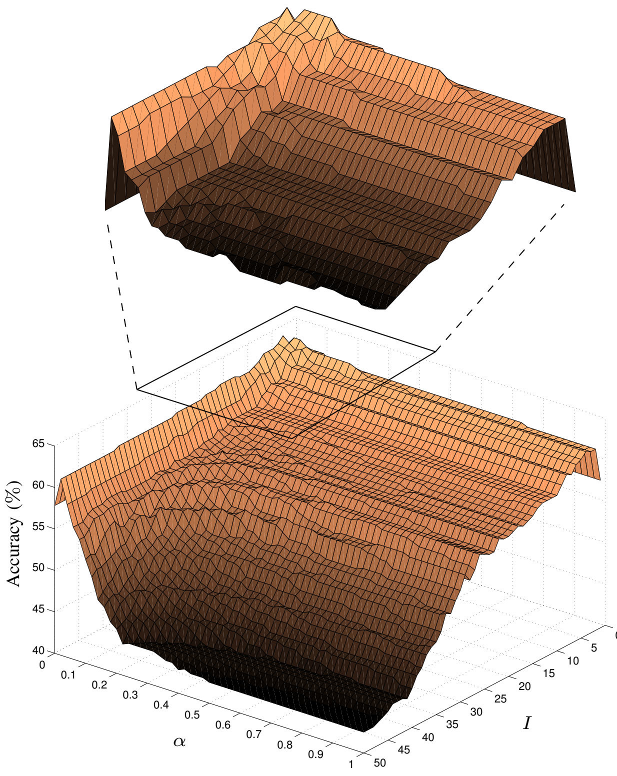

Fig. 2. C^{3}E-SL classification accuracy (%) with respect to different combinations of values for the parameters α and I.

two main characteristics: diversification and intensification [20]. Diversification involves exploring the search space on the global scale by generating diverse random solutions, while intensification focuses on the search in a specific region of the search space by exploiting good solutions that are found in it. Several studies have shown that metaheuristics are indeed capable to find acceptable solutions for complex (optimization) problems in a reasonably time, preventing entrapment in local minima [21], [22], [23], [24]. From this perspective, metaheuristics are promising alternatives to optimize the C^{3}E-SL parameters.

Initially, we decided to investigate the performance of a variety of well-known metaheuristics to optimize the parameters of C^{3}E-SL, namely: α and I. In particular five metaheuristics have been used:

(1+1)-ES: this metaheuristic is an Evolution Strategy (ES) [8], [9], in which evolution is a process based on two individuals, which are candidate solutions to a given problem. In brief, from one parent (a real-valued vector) a descendant is created by means of adding normally distributed random numbers [19], [18]. Then, the best of both individuals serves as the ancestor for the following generation, thus contributing to evolve better solutions. Typically, two parameters (c_d and c_i), which are based on the $1/5$-success rule of Rechenberg [8], must be provided to adjust the convergence rate. Schwefel [9] suggests using $c_d = 0.82$ and $c_i = 1.22$[2].

[2]At this point of our research, we are not particularly interested in fine-tunning parameter values of the metaheuristics themselves. Instead, we are taking advantage of recommendations made in previous studies already available in the literature. This approach has also been adopted to choose parameter values for the other metaheuristics here employed.

CMA-ES: this metaheuristic is an extension of Evolution Strategies. CMA-ES [11] — that stands for Covariance Matrix Adaptation Evolution Strategy — adapts the covariance matrix of a multi-variate normal search distribution. This adaptation consists of leaning dependencies among the parameters. The only user-defined parameter of CMA-ES is the population size. As for (1+1)-ES, by following previous studies [11], [25] we used 20 individuals in our work.

Differencial Evolution (DE): as (1+1)-ES and CMA-ES, DE is also an Evolutionary Algorithm (EA) for parameter optimization [10]. DE aims at solving an optimization problem by maintaining a population of candidate solutions. From this population, new candidate solutions are created by combining existing ones. Then, the best candidate solutions are kept for the next iteration. New candidate solutions are obtained from perturbations (the so-called mutations) in existing solutions. In particular, consider a given parameter vector. After being mutated, the parameters of this vector are mixed with the parameters of another predetermined (target) vector to provide the so-called trial vector. If the trial vector encodes a better solution than the target vector, this gets replaced by the trial vector in the next generation. DE can be materialized by means of a variety of algorithms — we used $DE/best/2/bin$ with population size $P = 20$, the scaling constant $F = 0.5$, and the crossover constant $CR = 0.9$ [10], [26].

Simulated Annealing (SA): inspired by thermodynamic processes, SA [12] starts its search from an initial random solution (s) and, in each iteration, a "neighbor" s' of the current solution is randomly generated. Differences in the fitness function are captured by a parameter, Δ. In minimization problems, if $\Delta < 0$, then s' becomes the new current solution, otherwise s' is accepted with a probability $e^{-\Delta/T}$, where T is the so-called temperature, a parameter that is decreased over the iterations. This process is repeated until T is so small that no solution can be accepted anymore. In each iteration, the temperature is updated by doing $T' = r \cdot T$, where r is the cooling rate. In our study, as suggested by [12], we used $r = 0.95$.

Particle Swarm Optimization (PSO): based on swarm intelligence [13], PSO starts the search from a population of randomly generated candidate solutions, which are named particles. These particles are moved around in the search-space by taking into account their positions and velocities. Each particle's movement is influenced by its local best known position. Also, particles are guided towards the best positions known so far over the iterations. In particular, the velocity of the particle is updated via information of the best position found by itself and the best position found so far — considering all particles. The velocities for each dimension are limited by a maximum value that affects the precision with which the regions near the current solutions will be explored — following the general guidelines from [27], we adopted $V_\alpha = 0.3$, and $V_I = 15$. Also, we used a population formed by 20 particles and we set both the *cognition* and *social* quantities to 2. The inertia weight, w, was set to linearly decrease from 0.9 to 0.4 during the first 1,500 iterations.

The metaheuristics just addressed usually find good solutions in reasonable running time [22], [23], [24]. However, the suitability of any heuristic for a particular application scenario is an empirical matter. In the next section, we present experimental results that show that the chosen metaheuristics can efficiently optimize the C^{3}E-SL parameters, thereby providing good classifiers for a variety of datasets.

IV. Empirical Evaluation

The optimization of the parameters of the C^{3}E-SL algorithm (described in Section II-A) consists of searching for values of α and I that yield to best classification accuracy. In other words, ideally we are looking for an optimal pair of values $<\alpha^*, I^*>$ that results in the most accurate classifier for a given problem. In order to estimate these values, we employ the metaheuristics addressed in Section III. The details of our experimental setup, followed by the achieved empirical results, are presented next.

A. Experimental Setup

Fourteen datasets from the UCI Machine Learning Repository [17] were used in the experiments. The main characteristics of these datasets are reported in Table I.

TABLE I. Main characteristics of the used datasets.

ID	Dataset	Instances	Attributes	Classes
Wisc	Breast Cancer Wisconsin (Original)	683	9	2
Pima	Pima Indians Diabetes	768	8	2
Yeast	Yeast	205	20	2
Spam	Spambase	4601	57	2
Iono	Ionosphere	351	33	2
Iris	Iris	150	4	3
Wred	Wine Red Quality	1599	12	6
Blood	Blood Transfusion Service Center	748	4	2
Glass	Glass Identification	214	9	7
Sonar	Con. Bench (Sonar, Mines vs. Rocks)	208	60	2
Vert	Vertebral Column	310	6	3
Seeds	Seeds	210	7	3
Ecoli	Ecoli	336	7	8
Ilpd	Indian Liver Patient Dataset	579	10	2

Essentially, the optimal values for the parameters of C^{3}E-SL — $<\alpha^*, I^*>$ — can be estimated via cross-validation procedures, in which we usually have three datasets: training, validation, and test. In a controlled experimental setting, the new data (target set) is frequently referred to either test or validation set. These two terms have been used interchangeably in the literature, sometimes causing confusion. In this paper, it is assumed that the target/test set has not been used at all in the process of building the ensemble of classifiers — *i.e.*, it is an independent set of data not used at all to optimize any parameter of the algorithm and, as a consequence, of the resulting classifier. Thus, as usual [28], only the training and validation sets are used to optimize the parameters of the algorithm.

Figure 3 portrays the two main steps of the adopted optimization procedure. First, we split the instances into training, validation, and target/test sets. With the training set, the ensemble of classifiers is built. Then, we use metaheuristics to estimate α^* and I^* by using the validation set, where a cluster ensemble is induced. The resulting values for the parameters are finally employed to assess the classification accuracy in the target/test set where, again and as requested by C^{3}E, a cluster ensemble similar to the one built for the validation set must be induced. Provided that, in a controlled experimental setting, we do know the true class labels of all instances, we can repeat this process multiple times and compute statistics of interest from the target/test set. Thus, we used the cross-validation procedure for empirically estimating the generalization capability of the C^{3}E-SL algorithm (as further discussed next).

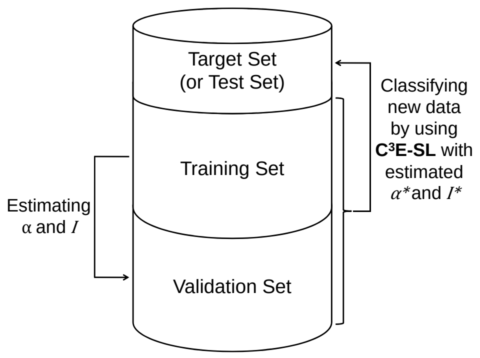

Fig. 3. Optimizing C^{3}E-SL parameters via cross-validation.

A straightforward (but often computationally intensive) way of searching for $<\alpha^*, I^*>$ involves running grid search [29], which has been adopted as a baseline for comparison purposes in our work. Grid search usually refers to exhaustive searching through a subset of the parameter space. In our case, one of the parameters, the number of iterations (I), is naturally discrete, whereas the other parameter (α) is real-valued and thus needs to be discretized before running grid search. For both of them, lower and upper bounds must be set *a priori*. In particular, we used all different combinations of $\alpha = \{0, 0.001, 0.002, ..., 1\}$ and $I = \{1,2,3,...,50\}$. From this setting, C^{3}E-SL was run 50,500 times to estimate the optimal pair of values $<\alpha^*, I^*>$ for each dataset, according to (5):

$$<\alpha^*, I^*> = \underset{\alpha, I}{\mathit{arg\ min}}\ \text{ClassError}\left[\text{C}^3\text{E-SL}(<\alpha, I>)\right] . \quad (5)$$

Metaheuristics guided to minimize the misclassification rate in (5) were run from a time constraint given by grid search. In particular, for each dataset the running times spent by grid search were stored and then *50%* of these running times were given as a time limit for the metaheuristics[3]. Algorithm 2 provides an overview of the framework used for every metaheuristic. Accuracy results obtained from metaheuristics were then compared to those found via grid search. By doing that, we can evaluate the trade-off between running time and classification accuracy provided by metaheuristics.

We adopted a 5-fold cross-validation process [28], in which each fold has 20% of the dataset instances — this is precisely the size of each target/test set. Accordingly, both training and validation sets have 40% of the dataset instances. The classifier ensemble was composed of two well-known classifiers,

[3]All metaheuristics under study were implemented in Matlab, using only the necessary commands. Population initialization was made within of a common region (of α and I). This way, more uniform efficiency comparisons can be performed. The same computer (i7, 3.2 GHz, 12 Gb RAM), running only the operational system, was used for all the controlled experiments.

Algorithm 2: Optimizing the C^{3}E-SL parameters using metaheuristics

```
  Input: MH ← choose a metaheuristic.
  Output: estimated pair of values $<\alpha^*, I^*>$.
1 Run MH:
2     Randomly set initial solution(s) for $<\alpha, I>$ such
      that $\alpha = \{0, 0.001, ..., 1\}$ and $I = \{1, 2, ..., 50\}$ —
      this way a population containing a certain number
      of individuals (solutions) is initialized;
3     Run C$^3$E-SL (Algorithm 1) for each individual of
      the population (each one using its coded solution
      $<\alpha, I>$ as input for the C$^3$E-SL) — the
      misclassification rate, such as in Equation (5), is
      used to determine the goodness of the individuals;
4     Apply the MH operators (for details see Section III);
5     Go to Step 3;
6 until time limit is reached;
7 Select the best individual (that provides the minimal
  misclassification rate) to be the pair of values $<\alpha^*, I^*>$;
```

namely: Naive Bayes and Decision Tree [28]. The similarity matrix $\mathbf{S}$ was constructed from a cluster ensemble based on data partitions found by the K-Means clustering algorithm [30]: one with k clusters and another with $2k$ clusters — where the number of clusters, k, was automatically estimated from data by using the relative validity clustering criterion known as Simplified Silhouette [31], [32].

B. Experimental Results

Before reporting the experimental results achieved by the used metaheuristics, it is instructive to compare the results obtained by C^{3}E based on Squared Loss (SL) with C^{3}E based on I-Divergence [6]. For both variants of C^{3}E, grid-search was performed to optimize their parameters — α and I (as addressed in Section IV-A). Table II shows the average classification accuracies, along with the results obtained for the classifier ensembles.

TABLE II. Accuracies (%) of Classifier Ensembles, C^{3}E based on I-Divergence, and C^{3}E based on Squared Loss (SL) — standard deviations appear within parentheses. For convenience, best results are highlighted in bold face.

Dataset	Classifier Ensemble	C^{3}E - I [a]	C^{3}E - SL
Wisc	95.60 (2.8)	**96.48** (1.2)	**96.48** (1.2)
Pima	76.18 (3.1)	**77.09** (3.6)	75.92 (2.7)
Yeast	95.61 (3.2)	96.59 (2.8)	**97.56** (1.7)
Spam	81.20 (1.2)	92.20 (1.2)	**92.81** (1.1)
Iono	**88.34** (4.3)	85.20 (11.2)	87.75 (7.1)
Iris	95.33 (3.0)	**96.00** (2.8)	**96.00** (2.8)
Wred	55.91 (4.5)	**57.60** (8.7)	57.48 (9.1)
Blood	75.80 (1.9)	76.47 (1.5)	**76.60** (1.7)
Glass	**64.97** (6.7)	62.65 (9.9)	**64.97** (6.7)
Sonar	**73.60** (8.7)	71.20 (7.6)	67.38 (9.4)
Vert	**82.26** (5.9)	59.33 (15.2)	75.16 (16.1)
Seeds	88.57 (7.4)	90.00 (3.1)	**91.90** (3.2)
Ecoli	**84.83** (3.8)	**84.83** (4.5)	84.53 (4.8)
Ilpd	62.69 (3.4)	**66.84** (9.4)	60.98 (8.2)

[a]Following [6] we adopted $\lambda = 1$.

Note that in most of the cases both variants of C^{3}E provided better classification accuracies compared to the classifier ensemble. Also, the overall accuracies of C^{3}E-SL have shown

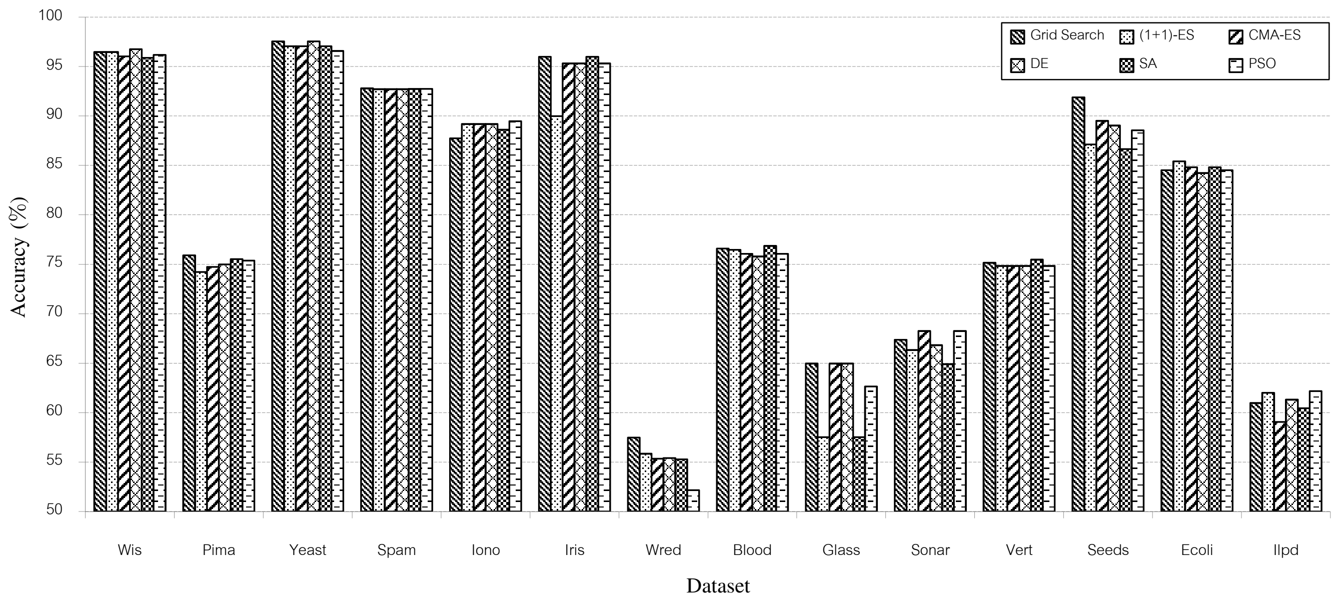

Fig. 4. Accuracies (%) of C^3E-SL with parameters optimized via metaheuristics.

TABLE III. ACCURACIES (%) OF C^3E-SL WITH PARAMETERS OPTIMIZED VIA METAHEURISTICS — STANDARD DEVIATIONS APPEAR WITHIN PARENTHESES. FOR CONVENIENCE, BEST RESULTS ARE HIGHLIGHTED IN BOLD FACE. RECALL THAT METAHEURISTICS WERE GIVEN JUST HALF THE RUNNING TIME OF THE GRID-SEARCH.

Dataset	Grid Search	(1+1)-ES	CMA-ES	DE	SA	PSO
Wisc	96.48 (1.2)	96.48 (1.2)	96.04 (1.1)	**96.77** (1.5)	95.90 (0.8)	96.19 (1.0)
Pima	**75.92** (2.7)	74.23 (2.9)	74.74 (2.8)	75.00 (2.6)	75.53 (3.0)	75.39 (2.6)
Yeast	**97.56** (1.7)	97.07 (2.7)	97.07 (2.0)	**97.56** (1.7)	97.07 (2.7)	96.59 (2.8)
Spam	**92.81** (1.1)	92.72 (1.1)	92.72 (1.1)	92.72 (1.1)	92.74 (1.1)	92.76 (1.1)
Iono	87.75 (7.1)	89.20 (5.3)	89.20 (5.3)	89.20 (5.3)	88.62 (4.4)	**89.48** (5.1)
Iris	**96.00** (2.8)	90.00 (13.3)	95.33 (3.0)	95.33 (3.0)	**96.00** (2.8)	95.33 (3.0)
Wred	**57.48** (9.1)	55.85 (10.1)	55.35 (10.1)	55.41 (9.8)	55.29 (9.7)	52.16 (11.1)
Blood	76.60 (1.7)	76.47 (0.9)	76.07 (2.2)	75.80 (1.9)	**76.87** (1.0)	76.07 (2.2)
Glass	**64.97** (6.7)	57.53 (9.0)	**64.97** (6.7)	**64.97** (6.7)	57.53 (9.0)	62.65 (9.9)
Sonar	67.38 (9.4)	66.34 (11.3)	**68.26** (11.3)	66.83 (13.0)	64.91 (11.6)	**68.26** (11.3)
Vert	75.16 (16.1)	74.84 (15.9)	74.84 (15.9)	74.84 (15.9)	**75.48** (16.0)	74.84 (15.9)
Seeds	**91.90** (3.2)	87.14 (6.4)	89.52 (2.1)	89.05 (2.1)	86.67 (6.2)	88.57 (7.4)
Ecoli	84.53 (4.8)	**85.43** (5.4)	84.83 (4.5)	84.24 (5.3)	84.83 (4.5)	84.53 (4.8)
Ilpd	60.98 (8.2)	62.01 (9.7)	59.07 (9.6)	61.32 (8.6)	60.46 (8.5)	**62.18** (8.4)

to be similar to those got by C^3E based on I-Divergence, which is more complex.

Figure 4 reports an overview of the results obtained by the studied metaheuristics — specific results can be found in Table III. In addition, the results obtained from grid search — already reported in the last column of Table II — are again reproduced in the second column of Table III for convenience, thus making easier to compare the whole set of experimental results. As expected, in several datasets the computationally intensive grid search procedure provided the best results. However, note that, with half the running time of grid search, better accuracies were obtained by PSO (for *Iono*, *Sonar*, and *Ilpd*), SA (for *Blood* and *Vert*), DE (for *Wisconsin*), CMA-ES (for *Sonar*), and (1+1)-ES (for *Ecoli*). Note that there are also some ties. Considering the overall results achieved by the metaheuristics, one can note that DE, SA, and PSO obtained the best accuracies in three datasets, CMA-ES in two datasets, and (1+1)-ES only in one dataset. Although the performances of the different metaheuristics are comparable across the assessed datasets, a more careful observation suggest that DE, CMA-ES, and PSO have provided more consistent results compared to grid search. By taking into account this observation, one could prefer to use CMA-ES (because it has less parameters) or DE (due to implementation simplicity).

In practice, choosing an algorithm (among several alternative ones) for a particular application is usually not an easy task. For the problem being tackled, convenience aspects of the metaheuristics could be considered in order to help us to decide which ones are the most suitable ones. In this sense, we would like to answer the following question: Which of the studied metaheuristics is the preferred one for optimizing the C^3E-SL parameters? To do so, we took into account two aspects, namely: how user-friendly a particular metaheuristic is and its accuracy (recall that the running time has been fixed for all of them). For each of these aspects, the metaheuristics were ranked. One can argue that "user-friendly" is a rather subjective criterion. For simplicity, we only consider the number of parameters that each metaheuristic has. Hence, the metaheuristic in the top of the user-friendly criterion ranking is the one that requires the fewest number of user-defined parameters. Similarly, considering the classification accuracy, the metaheuristics were ranked. In this case, ranks have been generated according to Friedman test from the classification accuracies reported in Table III. The Friedman test is a non-parametric counterpart of the well-known ANOVA. Essentially, tests of statistical significance are reported because they provide some reassurance about the validity and non-randomness of the obtained results. More precisely, we carried out statistical tests by following the approach described in [33]. In short, the adopted approach is aimed at comparing multiple algorithms on multiple datasets by using the Friedman test with a corresponding post-hoc test. Analyses of statistical significance showed that there is no statistical significant difference when comparing the results obtained for grid search to the five metaheuristics. From these analyses, one could claim that metaheuristics can provide classifiers as accurate as those obtained from grid search, but taking half the running time. This is particularly relevant for real-world data mining applications where large datasets are available.

Table IV presents a summary of the rankings for the two adopted criteria: user-friendliness and accuracy. According to this table, CMA-ES is the most user-friendly metaheuristic (because it only requires the user to set the population size, which can be set by taking into account the available computational resources[4]), while PSO is the less indicated in this aspect (due to need for setting up several parameters). On the other hand, considering the accuracy criterion, PSO has shown to be the best one. Actually, it is somehow expected that there should be a trade-off between user-friendliness and accuracy. From this observation, a simple approach that takes both criteria into account involves multiplying the respective ranks of each metaheuristic. For instance, for CMA-ES, we computed the combined score as 1 (from user-friendliness) $\times 3$ (from accuracy) $= 3$. From sorting the respective scores for each metaheuristic, we can get a combined ranking of the metaheuristics, which is suggested according to the adopted criteria, namely: CMA-ES, DE, PSO, SA, (1+1)-ES.

From the combined ranking, it is now interesting to focus our analysis on CMA-ES and DE (the top two metaheuristics). From this perspective, Tables V and VI report the minimum, maximum, and average estimated values for α^* and I^* achieved from the cross-validation procedure run for each dataset. One can observe that there is a high variance both for different datasets and for different folds of the same dataset, thereby suggesting that the optimization of the C^{3}E-SL parameters is highly data dependent. We also noticed that C^{3}E-SL, with parameters optimized by CMA-ES, provided equal to or better results than those achieved by classifier ensembles in 64% of the datasets — compared to 57% for DE. Thus, note that C^{3}E-SL with parameter optimization by means of CMA-ES or DE can yield to better results than classifier ensembles in less running time than grid search approaches.

TABLE IV. User-friendliness and accuracy rankings.

Rank	User-friendliness	Accuracy
1	CMA-ES	PSO
2	SA	DE
3	(1+1)-ES	CMA-ES
4	DE	(1+1)-ES
5	PSO	SA

TABLE V. Minimum, maximum, and average values obtained by CMA-ES for α^* and I^* .

Dataset	α^*_{min}	α^*_{max}	α^*_{avg}	I^*_{min}	I^*_{max}	I^*_{avg}
Wisc	0.001	0.793	0.487	2	36	17
Pima	0.000	0.643	0.351	1	21	6
Yeast	0.000	0.944	0.539	1	6	3
Spam	0.245	0.587	0.452	8	11	9
Iono	0.000	0.575	0.250	1	10	4
Iris	0.149	0.539	0.350	1	18	7
Wred	0.268	0.943	0.634	3	12	5
Blood	0.006	0.882	0.394	1	42	21
Glass	0.000	0.344	0.138	1	6	2
Sonar	0.000	0.198	0.106	1	50	20
Vert	0.005	0.754	0.267	1	40	15
Seeds	0.001	0.605	0.255	2	22	11
Ecoli	0.000	0.275	0.124	1	35	9
Ilpd	0.012	0.954	0.588	7	50	37

TABLE VI. Minimum, maximum, and average values obtained by DE for α^* and I^* .

Dataset	α^*_{min}	α^*_{max}	α^*_{avg}	I^*_{min}	I^*_{max}	I^*_{avg}
Wisc	0.000	0.519	0.240	2	26	7
Pima	0.014	0.519	0.410	1	5	2
Yeast	0.498	0.519	0.515	1	2	3
Spam	0.257	0.508	0.390	7	11	9
Iono	0.004	0.920	0.492	1	2	2
Iris	0.498	0.595	0.530	1	24	6
Wred	0.026	0.882	0.450	3	5	4
Blood	0.385	0.704	0.521	1	48	17
Glass	0.312	0.498	0.461	1	1	1
Sonar	0.171	0.519	0.394	1	25	6
Vert	0.498	0.729	0.553	1	24	6
Seeds	0.060	0.519	0.409	2	32	10
Ecoli	0.002	0.519	0.277	1	8	3
Ilpd	0.060	0.958	0.312	4	50	28

V. Conclusions

We studied how to make an existing algorithm, named C^{3}E (from Combining Classifier and Cluster Ensembles), more user-friendly by automatically tunning its main parameters — the relative importance of classifier and cluster ensembles and the number of iterations of the algorithm — with the use of metaheuristics. Although, as expected, these parameters can be straightforwardly optimized via (time consuming) grid search approaches (based on well-known cross-validation procedures), we empirically showed that metaheuristics can be more computationally efficient alternatives. More precisely, analyses of statistical significance made from experiments performed on fourteen datasets show that five metaheuristics can provide classifiers as accurate as those obtained from grid search, but taking half the running time. Our work also shows that the classification accuracies of the different metaheuristics are comparable across the assessed datasets. From this viewpoint, all of them are in principle eligible to be used to optimize C^{3}E-SL parameters. However, CMA-ES and DE could be preferred over the other metaheuristics because the former has less parameters, whereas the later is possibly the simplest one to implement. More specifically, from the results achieved so far and by assuming that there is a trade-off between user-friendliness and classification accuracy, these metaheuristics are the most recommended ones for optimizing the parameters of C^{3}E-SL.

[4]The population size and the number of generations required for convergence are somewhat related. From a practical viewpoint, one may expect that the more genotypes the less the required generations for convergence.

Finally, promising venues for future work involve combining C^{3}E-SL and metaheuristics to solve difficult real-world problems in semi-supervised, active, and transfer learning scenarios.

Acknowledgment

We acknowledge the Brazilian Research Agencies CNPq and FAPESP for their financial support to this work.

References

[1] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ: Wiley-Interscience, 2004.

[2] J. Kittler and F. Roli, Eds., *Multiple Classifier Systems*. LNCS Vol. 2634, Springer, 2002.

[3] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined neural classifiers," *Pattern Recognition*, vol. 29, no. 2, pp. 341–348, 1996.

[4] N. C. Oza and K. Tumer, "Classifier ensembles: Select real-world applications," *Information Fusion*, vol. 9, no. 1, pp. 4–20, 2008.

[5] J. Ghosh and A. Acharya, "Cluster ensembles," *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, vol. 1, no. 4, pp. 305–315, 2011.

[6] A. Acharya, E. R. Hruschka, J. Ghosh, and S. Acharyya, "C3E: A framework for combining ensembles of classifiers and clusterers," in *Multiple Classifier Systems*. LNCS Vol. 6713, Springer, 2011, pp. 269–278.

[7] S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 1st ed. Chapman & Hall/CRC Press, 2008.

[8] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, ser. Problemata. Stuttgart-Bad Cannstatt: Frommann-Holzboog Verlag, 1973, no. 15.

[9] H. P. Schwefel, *Numerical Optimization of Computer Models*. Chichester: John Wiley & Sons, Inc., 1981.

[10] K. Price, "Differential evolution: a fast and simple numerical optimizer," in *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*. IEEE, 1996, pp. 524–527.

[11] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science, Number 4598*, vol. 220, 4598, pp. 671–680, 1983.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, 1995, pp. 1942 –1948.

[14] A. Strehl and J. Ghosh, "Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[15] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *J. Mach. Learn. Res.*, vol. 6, pp. 1705–1749, 2005.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[17] A. Frank and A. Asuncion, "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]," University of California, Irvine, School of Information and Computer Sciences. 2010.

[18] T. Bäck, F. Hoffmeister, and H. P. Schwefel, "A survey of evolution strategies," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 2–9.

[19] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, pp. 1–23, 1993.

[20] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Luniver Press, 2010.

[21] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, 2008.

[22] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK: IOP Publishing, 1997.

[23] P. Fleming and R. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control Engineering Practice*, vol. 10, no. 11, pp. 1223 – 1241, 2002.

[24] A. A. Freitas, *A survey of evolutionary algorithms for data mining and knowledge discovery*. New York, USA: Springer-Verlag Inc., 2003, pp. 819–845.

[25] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation*, ser. Studies in Fuzziness and Soft Computing. Springer Berlin / Heidelberg, 2006, vol. 192, pp. 75–102.

[26] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Jornal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[27] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, ser. EP '98. London, UK: Springer-Verlag, 1998, pp. 591–600.

[28] I. H. Witten and E. Frank, *Data mining: practical machine learning tools and techniques*, 2nd ed., ser. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2005.

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[30] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010.

[31] R. Campello and E. Hruschka, "A fuzzy extension of the silhouette width criterion for cluster analysis," *Fuzzy Sets and Systems*, vol. 157, no. 21, pp. 2858–2875, 2006.

[32] E. R. Hruschka, R. J. Campello, and L. N. D. Castro, "Evolving clusters in gene-expression data," *Information Sciences*, vol. 176, no. 13, pp. 1898–1927, 2006.

[33] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.