An Ensemble Co-Evolutionary based Algorithm for Classification Problems

Vu Van Truong[1], Bui Thu Lam[1], Nguyen Thanh Trung[2]
[1] Le Quy Don Technical University, Hanoi, Vietnam
[2] Liverpool Logistics Offshore and Marine Research Institute, Liverpool, United Kingdom
Correspondence: Vu Van Truong, truongvv@mta.edu.vn
Communication: received 14 April 2019, revised 22 August 2019, accepted 28 August 2019
Online early access: 28 August 2019, Digital Object Identifier: 10.32913/mic-ict-research.v2019.n1.852
The Editor coordinating the review of this article and deciding to accept it was Prof. Le Hoang Son

Abstract: In this paper, the authors propose a co-evolutionary algorithm using an ensemble learning approach (E-SOCA) to simultaneously solve both feature subset selection and optimal classifier design. Unlike previous studies where each population retains only one best individual (Elite) after co-evolution, in this study, an elite community will be stored and calculated together through an ensemble learning algorithm to produce the finally classified result. Experimental results on the University of California, Irvine (UCI) problems with a variety of input features ranging from small to large sizes show that the proposed algorithm results in more accuracy and stability than traditional algorithms.

Keywords: *Ensemble learning, co-evolution algorithm, feature selection, genetic algorithm.*

I. Introduction

For the classification domain there are basically two main problems to be solved. To be specific, one problem is to select the classification algorithm and the other is to select the input feature sets. Normally for each object to be classified, there will be many features. However, it is necessary to identify important features having the greatest impact on the classification results, and at the same time, with this selected feature set, what is the best configuration for the classification algorithm. Typically, the more features there are, the longer the training and classification process will take. In addition, it also makes the program take up more memory and hard disk space. Therefore, for machine learning methods, it is necessary to select a smaller subset that still guarantees the acceptable accuracy of the classification process. That is called a feature selection (or other names like variable selection, feature reduction, attribute selection or variable subset selection). In contrast to other dimensionality reduction techniques like PCA (Principal Component Analysis) or compression, feature selection techniques only select a subset of them rather than alter the original representation of the variables.

So far there have been many methods used to solve the feature selection problem (more details are presented in Section II.4). In general, these methods can be solved by using only a single algorithm or combining multiple algorithms. The co-evolutionary algorithm is one of the second methods. These algorithms use more than two populations to solve the problem. Depending on how these populations interact with each other, they can be divided into two main categories: co-operative and competitive. In co-operative co-evolution, a problem is broken down into many sub-problems and each population is designed to solve a sub-problem. The final solution will be the combination of those populations' outputs. Meanwhile, in competitive co-evolution, during evolution, individuals in this population will have to fight against a group of individuals in the remaining population and vice versa, this leads to an arms race between these two populations.

The number of competitive co-evolution studies is mainly focused on games and robots [1, 2]. Studies related to classification problems are still relatively small and have started to be more interested in recent years [3–5]. In [3], the authors have proposed a competitive co-evolutionary model using a double-elimination tournament (DET) mechanism. This research employs two populations which are the population of the multilayer perceptron artificial neural network (MLPANN) and the population of basis radial artificial neural network (RBFANN). During evolutionary process, individuals between these two populations interact with each other through the DET mechanism. The proposed method achieved high accuracy rates in the comparison with other versions of co-evolutionary ANNs, co-evolutionary RBFNNs and other classification algorithms on 15 datasets. In [4], a Predator and Prey model was

used to train multi-layer perceptron (MLP) classifiers. The Predators are MLPs and the Preys are training samples. In the evolutionary process, the Predator tries to classify as correct as possible, while the Prey tries to offer the most difficult training data sets to make the Predator misclassify as much as possible. Thanks to this process, the algorithm will find a strong classifier that has the ability to classify difficult data sets. It is noting that the idea of the Predator-Prey model was also used in the Generative Adversarial Networks (GANs) model [5], which is considered a revolution in deep learning. In this model, the Prey acts as a neural network Generator, it always tries to generate fake data that is as similar as possible with real data. Meanwhile, the Predator acts as a Discriminator network, always trying to identify the fake data with the real data.

In this study, we focus primarily on the co-operative co-evolution. Up to now, there have been many publications related to the classification problem. In [6], the authors presented a hybrid approach based on a co-operative co-evolutionary algorithm with dual populations for designing the RBFNN (Radial Basis Function Neural Network) models with feature selection problem (named DC-RBFNN). In this research, each feature is encoded as a binary string and an RBFNN structure is encoded in a real-encoded matrix-form. The authors used five objectives to evaluate the fitness of individuals. The proposed method was tested on 26 datasets from the UCI Repository. Experiment results showed that the proposed algorithm was able to obtain better results on complicated classification tasks than other training algorithms did. In [7], these authors proposed an ensemble coevolutionary algorithm (named Co-NNE). Unlike the DC-RBFNN, in Co-NNE, the authors used multiple subpopulations instead of dual populations, and utilized an Elite Pool to store a community of Elites. Each Elite is the best individual of a subpopulation. Experimental results showed that the Co-NNE got better results on complicated classification tasks in comparison with other ensemble algorithms. Following the success of this research direction, in [8] the authors continued to improve the DC-RBFNN with a new algorithm called subspace-based RBFNN model (SBRBFNN). This algorithm has two versions which are hard CEA-SBRBFNN (CEA-SBRBFNNH - using binary encoding) and soft CEA-SBRBFNN (CEA-SBRBFNNS – using real encoding). Experimental results on 22 UCI datasets show that the proposed method outperforms the DC-RBFNN and other feature selection algorithms. In [9], the authors proposed a multi-population co-evolutionary algorithm using genetic programming (MOGP) for software defect prediction (SDP) problem. In this study, the authors utilized three ensemble selection strategies to select a subset of solutions at the end of the coevolution process. Their results show that the MOGP was a promising solution for solving the SDP with unbalanced data. In [10] and [11], the authors used a co-operative multi-objective evolutionary approach with the support vector machines (SVMs) for solving multiclass classification problems. The idea of the algorithm is to break down the multiclass problem into multiple binary classification problems and utilize a subpopulation to solve each binary problem. From results obtained from 25 datasets, the proposed method outperformed some SVMs Multiclass Extensions.

In [12], the authors proposed a single objective coevolutionary algorithm (named COCEA) for the classification problems. In COCEA, the authors use two populations with different individual coding, one population using binary encoding (we call it as binary or feature population) and the other using real number encoding (real number population). The authors used the Differential evolution (DE) algorithm to evolve the real population and the GA algorithm to evolve the binary one. The results were tested on both oil spill dataset and three other UCI datasets. A notable point in the COCEA algorithm is that both populations share a fitness function which is the MSE error. This means both populations try to evolve to achieve the best possible classification value. The results show that this idea is appropriate because the classification result of the COCEA algorithm obtains the best one on all test data sets. However, the authors did not focus on further analysis of the aspect of the final selected features.

Through the literature review, we found that studies of co-evolution still have great potential and are still being studied extensively to solve problems related to the classification problem. In particular, after studying carefully the dual-population co-evolutionary models in previous studies, we have found that these studies can still be further expanded, and this is the motivation for us to continue studying in this direction. In this paper, we propose a wrapper method using an ensemble co-operative co-evolution approach with a dual population to optimize the ANN classifier and deal with the feature selection problem.

The novelty and main contributions of the article are as follows. First, in terms of elite retention mechanism, we utilize an Elite Pool to store the best-known individuals from two populations during the coevolutionary process (each the best individual is called an elite). However, unlike previous studies when only two elites were stored from beginning to end of evolution. In this study, all intermediate elites found during co-evolution are stored for further processing. This may increase the chances of finding a good classifier.

Second, in terms of ensemble learning step, in previous studies, the combination of two individuals in the Elite pool

was often considered as the final solution. Some studies using a multi-objective co-evolution solution, after the end of co-evolution, a community of individuals on the first Pareto front was chosen as an ensemble of classifiers. This selection method may encounter problems with the diversity of classifiers. The voting technique was commonly used in this case to make the final decision. Our approach is completely different, the mechanism of storing elites found through each generation may help us maintain the diversity. A boosting algorithm is then used to combine these classifiers to create a more powerful classifier.

This paper extends preliminary results in the conference paper [12]. In general, compared to the algorithm in the conference paper (COCEA), the proposed algorithm has three main differences. First, the COCEA algorithm works only with binary classification problems. In this study, the proposed method works with multiclass classification problems.

Second, in the COCEA, the real population was evolved by using the DE algorithm, meanwhile, the binary population used the GA algorithm to evolve. It is noted that the DE algorithm has proven to be able to converge faster than the conventional GA algorithm. This may lead to an imbalance between the two populations. In this proposed algorithm we use the conventional GA instead of the DE algorithms with some changes in evolutionary steps corresponding to two different populations.

Third, in the COCEA algorithm, after the end of the co-evolution process, only the best individual (elite) of each population is retained for calculation. In the proposed algorithm, a community of elite individuals in the evolutionary process is archived. These elites play as weak classifiers then an ensemble learning algorithm (the AdaBoost) is used to create a strong classifier from these weak learners.

The rest of the paper is organized as follows: Section II describes the fundamentals related to the proposed method. In Section III, the proposed algorithm is presented in detail. Section IV reports the experiments of the proposed method on six UCI datasets together with other algorithms. Finally, Section V concludes with remarks for future research.

II. BACKGROUND

1. The Artificial Neural Network (ANN)

For many years, the ANN has been considered a powerful tool to solve machine learning problems, especially classification problems. When working with the ANN, there are two main issues need to be addressed: one is to select the network architecture and the other is to select the initial initialization value for the Weight set. The ANN randomly generates initially values for the weight set, then a learning algorithm is used to train the ANN; the most common training algorithm is Backpropagation (BP). Because the BP algorithm is based on the gradient descending mechanism, it can quickly converge to the global extreme. However, it also can be easily stuck at local extremes in the weight-space. This depends greatly on initiating the ANN's weight. There have been many studies addressing this problem [13, 14]. In this study, we use one of two populations of co-evolution approach to solve this problem.

2. Genetic Algorithm (GA)

Genetic algorithm [15] is an algorithm based on Darwin's theory to simulate the process of natural evolution. The GA applies the genetic principles: natural selection, mutation, and cross-exchange. In the GA, a population is a collection of chromosomes, each chromosome represents a solution of the problem (in this study each chromosome will be the weight matrix of the ANN or a binary string present for a feature set). The GA is often used widely for feature selection problems. There are many types of encoding strategies for an individual in the GA. The most common strategy is to treat each individual as a binary string in which value "1" means the corresponding feature is selected and vice versa with the value "0". Another encoding type is to use a real number sequence in the range $[0, 1]$ to represent the probability of being selected for a feature. By comparing a given threshold it is possible to convert this real number encoding to conventional binary encoding. Another strategy is a bio-encoding, each chromosome is encoded as a pair of strings. The first one is a binary string to represent the selection of features and the second one is encoded as real numbers to show the weights of features. In this study, we use the binary encoding to present the first population and the real number to encode the second one.

3. Ensemble Learning

In machine learning, ensemble learning is a method using multiple learning algorithms to obtain a better performance than any single learning algorithm [16]. In ensemble learning, Bagging and Boosting are two common techniques. By far, the most common algorithm of boosting is AdaBoost. The AdaBoost (standing for "Adaptive Boosting") was first introduced by Freund and Schapire [17]. After that, there were several versions of this algorithm such as AdaBoost.M1 and AdaBoost.M2 [18]. The AdaBoost combines multiple "weak classifiers" into a single "strong classifier". A weak classifier is simply a classifier like an ANN or a KMeans or any classification algorithm. The AdaBoost determines weights of each classifier. These weights are then used to combine weak learners to form

a stronger classifier. The larger weight value, the higher the level of importance of that weak learner. In this study, we will use the AdaBoost algorithm to combine ANN classifiers to create a more powerful classifier.

4. Feature Selection Algorithms

In general, depending on the combination of the feature selection search and the classification model, the Feature subset selection techniques can be organized into filter approach and wrapper approach [19]. In the Filter approach [20], a feature relevance score is first calculated and features having low-scoring are removed. After that, the selected features becoming the input of the classification algorithm, they are done independently of the classification algorithm. Two steps are done independently. Although they are computationally simple, they are assessed by another criterion rather than classifiers, which leads to worse classification performance. Conversely, in the wrapper approach, the feature subset selection algorithm is wrapped with the classification function. Various subsets of features are generated, and the predictor plays as a black box to evaluate these subsets. In general, the wrapper approach outperforms the filter approach. There are several search algorithms used to find the optimal subsets. In [21–23], the authors used exhausting strategies to solve a feature selection problem. Normally, these algorithms only work effectively in relatively small dimensional data problems, and vice versa, because the computational space is too large these algorithms are not effective. In recent years, evolutionary techniques (EC) have been used extensively for solving this problem. Readers can be found more details of studies using the EC techniques for feature selection problems in a review paper [24].

III. Proposed Method

The general diagram of the E-SOCA is given in Algorithm 1 and Figure 1. In this study, the authors use two populations. The first population (called the feature population or binary population) and the second one (called the populations of ANNs or real population). At each generation, these two populations will interact with each other. This population will use the best individual (i.e. Elite) of the other population. $Elite1$ will determine the number of inputs of ANNs in the population 2 and vice versa, $Elite2$ will be used to calculate the adaptive value of each individual in population 1. Undergoing evolution, the overall solution will be a combination of the Elites from two populations. As can be seen that the biggest difference between the proposed solution and the conventional feature selection methods is that the classifiers (i.e. ANNs) change constantly during the evolution process instead of fixed configuration.

In traditional wrapper methods, a classifier is fixed to evaluate the performance of each selected feature set. This can reduce the performance of the classifier, since each ANN is usually appropriate with a set of inputs. It cannot fit all types of inputs. Therefore, the use of the co-evolutionary solution here will hopefully solve this problem. Another interesting point is that unlike previous co-evolutionary studies, only a pair of Elites in each population is selected at the last evolutionary step. In this study, a list of Elite pairs is stored so that we can continue to use an ensemble learning algorithm to combine them together.

In Figure 1, the E-SOCA consists of four main steps: population initialization, co-evolution, ensemble learning, and fine-tune. A more detailed explanation of the E-SOCA will be presented below.

1. Encoding

Two populations use two different coding strategies. In the feature population, suppose that we have m individuals $(F_1, F_2, \ldots, F_m)$, each individual F_i is encoded as a binary string. In particular, the value 1 corresponds to the feature being selected and vice versa (Figure 2).

In the second population, suppose that this population has n individuals $(\mathrm{ANN}_1, \mathrm{ANN}_2, \ldots, \mathrm{ANN}_n)$, each individual ANN_i is a weight and bias of an ANN. Therefore, we use a real-encoding to represent an ANN (Figure 3).

2. Initialization

In general, at the initialization step, individuals will be generated randomly. However, if individuals in the real population receive any random value, random in the binary population initialization must ensure the following two constraints: (1) this binary string must contain at least one bit of "1" to ensure there must be at least one selected feature. (2) There is only one string containing all bits of "1" (line 3) to ensure that all features must be considered.

At first, an $Elite1$ with all the bits assigned to "1" will be used as the input of the individuals in the population 2. Thus, it is possible to assess the adaptability of each individual in the second population. After that we make a ranking based on the fitness value and choose the best individual as an elite (i.e. $Elite2$) and put it into the Elite Pool. The $Elite2$ is then used to evaluate fitness values of the individuals in population 1, after ranking, the best one (named $Elite1$) is selected and put into the Elite Pool. At this time, $\langle Elite1, Elite2 \rangle$ will be stored in the Elite Pool, at the same time $Elite2$ is also included in a list (named $BestList$) to serve the ensemble learning step later.

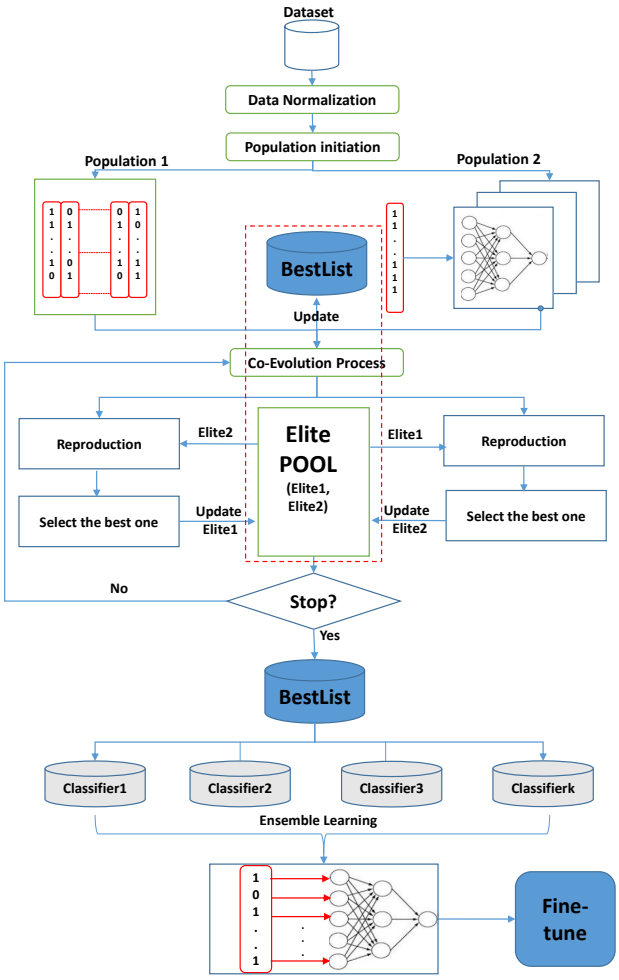

Figure 1. Diagram of the proposed method.

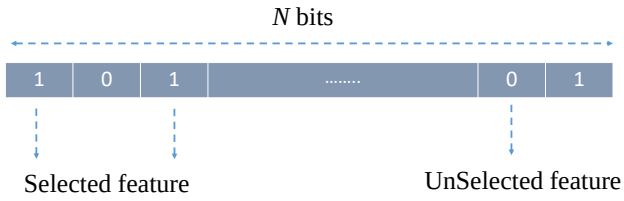

Figure 2. Encoding a feature individual.

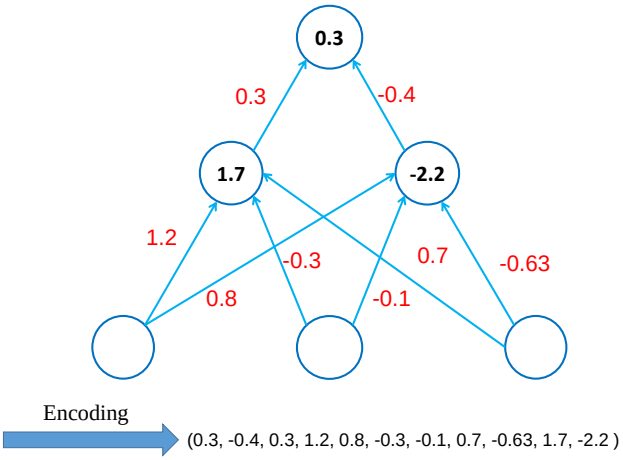

Figure 3. Encoding an ANN individual.

3. Co-Evolution Process

After the population initialization is the process of co-evolution. This is an iterative process; the step basically is quite similar to the initialization step in which each sub-population will try to evolve and find the elites to update to the pool. The elite in this population will be used for the other population in the next iteration.

However, there are two differences in this step compared to the initialization step. Suppose the Elite Pool consists of two elements $\langle Elite1, Elite2\rangle$. The first difference is the $Elite1$ update mechanism. If there is anyone (denoted by $Elite1_t$ at the loop t^{th}) dominating the corresponding elite in the pool (i.e. $Elite1$), the $Elite1$ will be substituted by $Elite1_t$ and the fitness of the new $Elite1$ is updated as well. However, we do not compare $Elite1_t$ with $Elite1$, instead we replace $Elite1$ by $Elite1_t$ immediately. This is also a different point from the COCEA algorithm. The reason for this change is that, through experimenting with the COCEA algorithm, we realize that this update mechanism is the main reason that the number of updated Elite Pool times is quite small. This leads to the result of co-evolution being not significant. Therefore, to increase the probability of updating the Elite Pool, we make a change in this update mechanism for $Elite1$. Meanwhile, with the $Elite2$, the update mechanism still happens normally, which means we choose the best individual from the population 2 (denoted by $Elite2_t$ at the loop t^{th}) compared to the $Elite2$ in the Elite Pool, if the MSE value of the $Elite2_t$ is smaller than the $Elite2$ then we replace the $Elite2$ by the $Elite2_t$. The second difference here is that, whenever the $Elite2$ is replaced by the $Elite2_t$, instead of taking new better values (i.e. $Elite2_t$), we store the old values ($Elite2$) and put them into $BestList$. These elements in $BestList$ continue to be combined through an ensemble learning algorithm.

The main difference between the two populations is the evolutionary operation (the Reproduction step in the Figure 1). Because each population uses different encoding types (binary and real encoding), therefore evolutionary operation applied to them are varied. Specifically, in population 1 we use *Two points crossover* and *Bit flip mutation* operations. Meanwhile, the *SBX Crossover* and *Polynomial Mutation* are applied to the population 2.

Algorithm 1: Ensemble Single Objective Coevolutionary Algorithm (E-SOCA).

1 **Input**: DataSet.
2 **Output**: BestClassifier.
3 $P1 \leftarrow$ InitializePopulation1(N);
4 $P2 \leftarrow$ InitializePopulation2(N);
5 $Elite1 \leftarrow [11 \ldots 1]$;
6 $Pool \leftarrow \Theta$; $BestList \leftarrow \Theta$;
7 Pool.Add(Elite1);
8 **CalculateFitness**(P2, Elite1, DataSet);
9 Eltite2 $\leftarrow Sort(P2)$;
10 Pool.Add(Elite2);
11 BestList.Add(Elite2);
12 **while** *stop condition false* **do**
13 **Reproduction**(P1, Eltite2);
14 Eltite1 $\leftarrow Sort(P1)$;
15 **Pool.Update**(Elite1);
16 **Reproduction**(P2, Eltite1);
17 Eltite2 $\leftarrow Sort(P2)$;
18 **Pool.Update**(Elite2);
19 **if** *update = true* **then**
20 BestList.Add(Elite2);
21 **end**
22 **end**
23 WeakLearners = BestList;
24 BestClassifier = **EnsembleLearning**(WeakLearners);
25 **FineTune**(BestClassifier);
26 **Return** BestClassifier;

4. Ensemble Learning

After the coevolutionary process, a list of Elites is stored in *BestList*. These Elites play as *weakLearners* in the AdaBoost algorithm. Ideas and ways to implement this algorithm are shown in [13]. The biggest difference here is that the AdaBoost version is used for the classification problem instead of the forecasting problem. In the AdaBoost algorithm, weak learners may be different models or the same models with different configurations. In this paper, we use an ensemble of ANNs (i.e. *BestList*) with the same topology but different weights. Besides, each weak learner may be trained on a random subset of the training set and the subsets can overlap. However, in this study, instead of dividing into subsets we use the entire training data set for all weak learners. By using the AdaBoost, each Elite will have different influence coefficients, the better the result of the classification, the greater the effect factor and vice versa. These coefficients will be used to combine Elite sets to create a stronger classifier (named *BestClassifier*).

TABLE I
THE PROPERTIES OF THE BENCHMARK DATASETS

Dataset	Train	Val	Test	Class	Features
Sonar	104	-	104	2	60
Satimage	4435	-	2000	7	36
Segment	1162	574	574	7	19
Ionosphere	175	88	88	2	34
Pima	384	192	192	2	8
Heart	134	68	68	2	13

5. Fine-Tune

A population-based evolutionary mechanism helps the E-SOCA approach find out the regions that contain the global maximum points, but it is very hard to find exactly these extreme points. From this characteristic, we use the GA algorithm to optimize the ANNs first (i.e. using the GA to take the ANNs to escape from the local extremes and approach the global ones), then use the Back-Propagation algorithm to put them closer to the global extreme points. This is the reason that we continue using this step for *BestClassifier* (line 24 of Algorithm 1) after going through the ensemble learning step.

6. Fitness function

In this paper, we choose the Mean Squared Error (MSE) which is calculated as the fitness function as follows:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(x_{\text{real}} - x_{\text{output}})^2, \quad (1)$$

where n is the number of data value, x_{real} is a real value, and x_{output} is an output value.

IV. EXPERIMENTS

1. Data Description

In order to evaluate the performance of the proposed method, the authors conduct experiments on 6 different UCI datasets (Sonar, Satimage, Segment, Ionosphere, Pima, and Heart). Most datasets are divided into three subsets: 50% of the dataset are utilized for training, 25% for validating, and the remaining 25% for testing, except the Sonar and the Satimage. For each dataset, the final result is averaged over 20 runs of the algorithms. These datasets are briefly characterized in Table I.

TABLE II
PARAMETER SETTING

Method	Parameters	Value
ANN	Learning rate	0.3
	Number hidden nodes	20
	The iterations of Fine-tune	1000
	Alpha value	2
GA	The population size	20
	The probability of mutation	$1/\lambda$
E-SOCA	The iterations of E-SOCA	150
	The iterations of reproduction	50
	The Population size	100

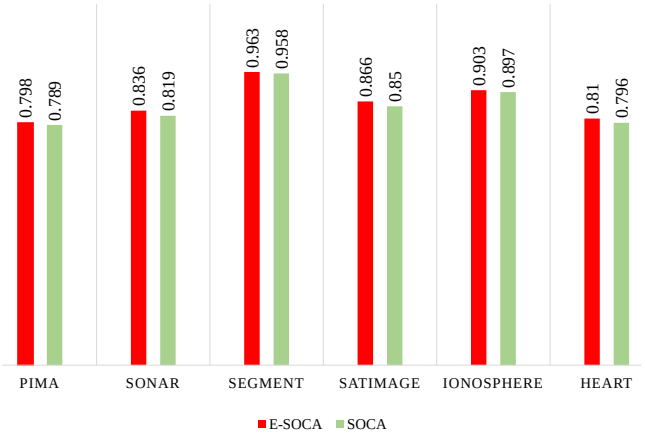

Figure 4. Testing accuracies of the proposed method E-SOCA comparing with the non-ensemble version SOCA.

2. Parameter Setting

The experiment parameters used in the proposed algorithm are in Table II, where λ is the chromosome length.

3. Results and Analysis

To evaluate the potential of the proposed algorithm we conduct some experiments as follows.

Experiment 1. Verify effect of the ensemble learning algorithm:

In order to measure the effect of the ensemble learning method, we compare the proposed method E-SOCA with a version which does not utilize the ensemble learning algorithm (named SOCA - Single Objective Co-Evolutionary Algorithm). As can be seen from Figure 4, the proposed method E-SOCA is better than the SOCA in all test cases. One common point among all 6 experimental data sets is that both methods give quite high results, especially in the Segment dataset, the correct classification results can reach 96.32% and 95.84% by the E-SOCA and the SOCA, respectively. Since both these algorithms differ only in ensemble learning step, only the classification results are different. However, in terms of criteria for the number of features selected, both methods are the same because they use Elite1 at the last step of the co-evolutionary process. Through this experimental result, the effect of community learning can be clearly seen. Especially with the Sonar dataset, thanks to this step, the classification accuracy has increased from 81.9% to 83.6%.

Experiment 2. Verify proposed algorithm with other conventional machine learning algorithms:

There are five machine learning algorithms (including the Naive Bayes - Bayes, the C4.5, the K-means, the K-neighbour algorithm (KNN), the Multilayer Perceptron (MLP). Two other algorithms are also used to compare with the proposed method including the probabilistic neural network (PNN) [25] — a 4-layers MLP widely used in classification problems, and the sequential minimal optimization (SMO) algorithm [26] — an algorithm used for training a support vector classifier. The empirical results with these algorithms are derived from [6].

Table III shows the results of proposed algorithms E-SOCA in comparison with six machine learning algorithms on six data sets. The bold value is the best one in each row. It can be easily seen that the E-SOCA algorithm achieves the best results in four out six test cases except for the Ionosphere and the Heart datasets. This shows the stability and power of the proposed algorithm in terms of classification accuracy criteria. Specifically, the proposed algorithm E-SCOCA outperforms the other ones with Pima dataset. It can reach 79.8%. Meanwhile SMO ranks the second, but its level of accuracy just reaches 77.26%. The remaining algorithms are lower, even the PNN can only reach 69.36%. In the Sonar dataset, the proposed E-SOCA algorithm continues to give the best classification results (83.6%), outperforming other methods. Meanwhile, the figures for the MLP is slightly slower, at 82.01%. In this dataset, PNN results in very low classification, only 53.2%. With the Segment dataset, the proposed algorithm E-SCOCA gives the best classification results compared to other methods. To be specific, it can achieve accuracy of 96.3%, while the C4.5 and the MLP are slightly lower, at 96.26% and 96.01% respectively. the KMeans produces the worst results, only 76%. With the Satimage dataset, the E-SOCA, PNN and SMO algorithms show quite similar results, the figures are 86.6%, 86.42%, 86.42%, and 86.58% respectively. Meanwhile, Bayes gives the worst classification results, reaching only 79.67%. The E- ranks the third compared to other algorithms on the Ionosphere dataset. In this case, the PNN gives the best results, at 94.24% while the KNN produces a very poor result, at only 79.13%. The result with the Heart dataset is similar to the Ionosphere dataset. The E-SOCA does not get the best results, just reaching 81%

TABLE III
TESTING ACCURACIES OF THE PROPOSED METHOD E-SOCA COMPARING WITH SOME OTHER MACHINE LEARNING ALGORITHMS.

Dataset	E-SOCA	Bayes	C4.5	K-means	KNN	MLP	PNN	SMO
Pima	**0.798**	0.7542	0.7345	0.7323	0.7149	0.7476	0.6936	0.7726
Sonar	**0.836**	0.6564	0.7251	0.7087	0.758	0.8201	0.5321	0.7733
Segment	**0.963**	0.8019	0.9626	0.76	0.9383	0.9601	0.9561	0.9229
Satimage	**0.866**	0.7967	0.8608	0.8282	0.8101	0.8614	0.8642	0.8658
Ionosphere	0.903	0.8307	0.8983	0.8848	0.7913	0.896	**0.9424**	0.885
Heart	0.81	**0.8505**	0.7636	0.8064	0.827	0.7922	0.7225	0.8406

while with this data the Bayes gives the highest value with 85.05%. Meanwhile, the PNN is the worst one with the figure is 72.25%.

Experiment 3. Verify proposed algorithm with other ensemble learning algorithms:

To verify the performance of the proposed method, we compare the proposed algorithm with some other ensemble learning algorithms such as the CO-NNE using the majority voting rule (CO-NNE (VOT)), AdaBoost (AB) [18], Bagging (BA) [27], Decision Forests (DF) [28] and Random Subspace method (RS) [29]. The experimental results of these algorithms are shown in [7]. In particular, the CO-NNE (VOT) (published in 2012) is the main algorithm proposed in this study while the other algorithms are renowned ensemble learning machine algorithms. We focus more on comparing the proposed method with the CO-NNE (VOT) algorithm, which is also an ensemble co-evolutionary approach, but the implementation method is completely different. As can be seen in Table IV (the symbol "–" indicates unavailable result), the E-SOCA significantly outperforms the CO-NNE (VOT) on three the datasets Pima, Sonar and Segment. There are improvements of 4.2%, 6.7% and 5.4% on testing accuracy respectively. Meanwhile, with the Ionosphere and Heart datasets, the CO-NNE (VOT) is slightly better than the E-SOCA. The figures for the difference are 1.6% and 1.1% correspondingly. In comparison with other ensemble learnings, the E-SOCA gets the best results on two data sets the Pima and the Sonar. Meanwhile, the DF reaches the best values on the Segment and the Ionosphere datasets with 97.4% and 93.1%, respectively. For the Heart dataset, the BA shows the best testing accuracy at 84.4%, outperforming other algorithms.

Experiment 4. Verify proposed algorithm with other state-of-the-art co-evolutionary algorithms:

To further verify the power of the proposed algorithm, we compare the E-SOCA to the latest recent dual population co-evolutionary algorithms. There are four algorithms used in this research in which two algorithms (the CEA-SBRBFNNH and the CEA-SBRBFNNS algorithms) were proposed in 2016 [8]. Two remaining algorithms (the DC-RBFNN, and the GA-RBFNN algorithm) were proposed in 2010 [6]. As can be seen in Table V, the E-SOCA significantly outperforms the other coevolutionary algorithms on most of the datasets except the Inono and Heart. Especially with the Sonar dataset, the proposed algorithm gives much better results than the remaining algorithms do. The accuracy reaches 83.6%. Meanwhile, the corresponding figures for CEA-SBRBFNNH, CEA-SBRBFNNS, DC-RBFNN and GA-RBFNN algorithms are 77.1%, 75.6%, 77% and 73.4%, respectively. With the Pima and Satimage datasets, the results of E-SOCA are slightly better than other algorithms. In contrast, with the Ionosphere dataset, the algorithm proposes the worst result, however, the difference between the algorithms is not much. The DC-RBFNN algorithm gives the highest outcome at 92.20% while the figure for the E-SOCA is 90.30%. While the two CEA-SBRBFNN versions produced higher results than others with 84.3% and 83.4%, the E-SOCA algorithm produced similar results with DC-RBFNN, reaching 81% and 81.6%, respectively. The GA-RBFNN algorithm gives the lowest result, at 78.2%. In terms of the number of selected features, this is also the biggest difference between co-evolutionary algorithms and other single algorithms (Bayes, C4.5, ...). With the single algorithms, all features are selected to participate in the classification. Meanwhile, with the co-evolutionary algorithms only a certain number of features are selected to classify. If the number of features is large which can be up to hundreds, choosing the most important features will not only help improve classification results but also reduce time and space complexity. To assess the effectiveness of the proposed algorithm in reducing the number of features, we conduct a comparison with another co-evolutionary feature selection algorithm (the DC-RBFNN).

Figure 5 visually displays the results of the algorithms on these six sample datasets in terms of the feature selected of two algorithms. It can be easily seen that, in the DC-RBFNN algorithm, the number of selected features in the DC-RBFNN algorithm is much lower than the figure in the proposed algorithm, 4.167/8 compared to 6.45/8

TABLE IV
TESTING ACCURACIES OF THE PROPOSED METHOD E-SOCA COMPARING WITH SOME OTHER ENSEMBLE LEARNING METHODS

Dataset	E-SOCA	CO-NNE (VOT)	AB	BA	DF	RS
Pima	**0.798**	0.756	0.743	0.758	0.742	0.753
Sonar	**0.836**	0.769	0.825	0.762	0.799	0.720
Segment	0.963	0.909	0.922	0.961	**0.974**	0.903
Satimage	0.866	–	–	–	–	–
Ionosphere	0.903	0.919	0.933	0.914	**0.931**	0.916
Heart	0.81	0.821	0.806	**0.844**	0.801	0.827

TABLE V
TESTING ACCURACY OF THE PROPOSED METHOD E-SOCA COMPARING WITH STATE-OF-THE-ART COEVOLUTIONARY ALGORITHMS

Dataset	E-SOCA	CEA-SBRBFNNH	CEA-SBRBFNNS	DC-RBFNN	GA-RBFNN
Pima	**0.798**	0.77 0	0.769	0.758	0.750
Sonar	**0.836**	0.771	0.756	0.770	0.734
Segment	**0.963**	–	–	0.951	0.9448
Satimage	**0.866**	–	–	0.8648	0.861
Ionosphere	0.903	0.921	0.914	**0.922**	0.912
Heart	0.810	0.834	**0.843**	0.816	0.782

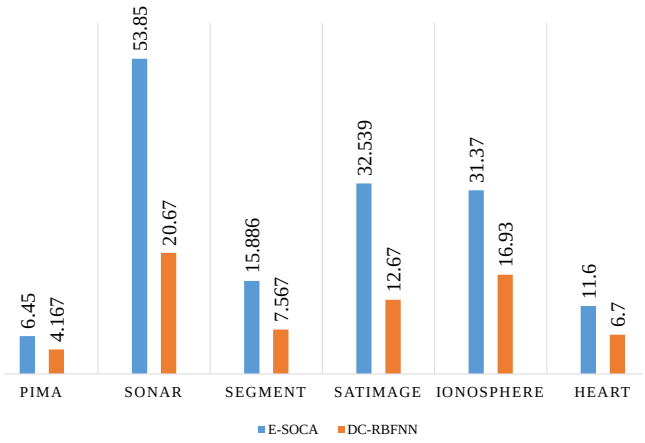

Figure 5. The number of features selected by the proposed method comparing with the DC-RBFNN.

in the Pima dataset; 6.7/13 compared to 11.6/13 in the Heart dataset; 7.567/19 compared to 15.886/19 in the Segment dataset; 16.93/34 compared to 31.37/34 in the Ionosphere dataset; 32.539/36 compared with 12.67/36 in the Satimage dataset and 53.85/60 compared with 20.67/60 in the Sonar dataset. It can be clearly seen that the larger the number of input features, the number of characteristics chosen from the DC-RBFNN algorithm remains stable (approximately 50% of the features are retained), while the figure for the proposed algorithm is variable and unstable. This is the weakness of the E-SOCA. The cause of this phenomenon will be explained more thoroughly in the Discussion section.

Experiment 5. Verify running times of proposed algorithm:

In this experiment, we will check that in the 3 main steps of the proposed algorithm including the co-evolution step, the community learning step (AdaBoost) and the Fine-tune step (BP), which step takes up computational time the most. The running time of each step with the data sets is shown in Figure 6. The unit of measurement in seconds. The computer configuration used here is core I7, 8Gb RAM. One thing that can be seen is that the co-evolution is the most time-consuming step. The remaining two steps take a very small time. The two columns represent the total computational time and the running time of the co-evolutionary step is always approximately the same on all data sets. As can be seen that the calculation time depends mainly on three factors: the number of co-evolution steps, the size of the training set and the size of the input feature set. The Heart data set (with a feature number of 13 and the number of training samples of 134) occupy the smallest running time, reaching 288.27 (s). Meanwhile, the Segment data set, with 19 input features, 1162 training samples, takes more time consuming than the Heart, occupying 1992.58 (s).

4. Discussion

In short, from the proposed algorithm model along with experimental results and comparisons with other algorithms. We can draw some points. First, the strength of the E-SOCA algorithm is that it is possible to give good accuracy classification results with high stability. This is partly because the E-SOCA uses a population of ANNs

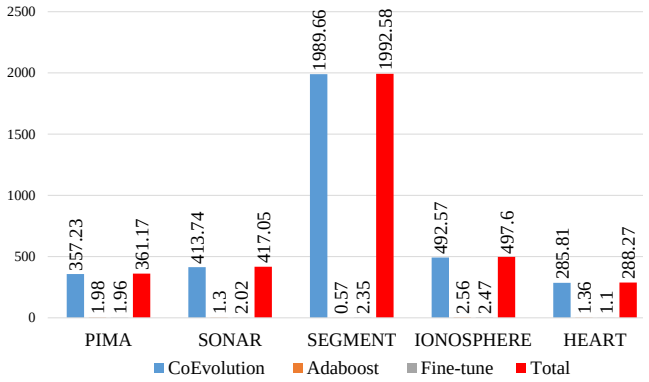

Figure 6. The computational time for each steps in the proposed method.

to evolve and the accuracy of ANNs in next-generations will be better. Another reason is the effect of the ensemble learning algorithm, by combining the weak learners into a strong learner, this algorithm contributes to the stability and also improves the classification accuracy.

Second, the weakness of the E-SOCA lies in the single-objective function. Although the E-SOCA uses two different populations, one for evolving the classifier and the other one for evolving the feature set. However, both populations use the same fitness function. This is the classification accuracy (MSE). This means both populations only focus on optimizing this goal, and the criteria for feature selection are still secondary. This cannot guarantee that the number of features selected will be gradually reduced. This is the main reason why the number of features retained by the E-SOCA is often large and uncontrollable. This problem can be handled by applying a multi-objective optimization algorithm (like the DC-RBFNN) instead of the single-objective algorithm being used in this version.

Third, time-consuming is also a drawback of the proposed method. Because the proposed method is currently still running sequentially, therefore the running time of this method will be greater when the size of the features as well as the training sample number increases. This problem can be solved by using parallel, multi-thread, and distributed computing solutions.

Finally, the E-SOCA algorithm provides a common model that can be applied to many different evolutionary algorithms for each population. In this study we are using the GA algorithms for evolving both populations, other multi-objective evolution algorithms can be applied to this model (such as NSGA-II). Besides, other ensemble learning algorithms can also be applied to the final set of Elites.

V. Conclusion

In this paper, an ensemble dual-population based co-operative co-evolutionary approach (E-SOCA) is presented. In the E-SOCA, two sub-populations are simultaneously maintained to achieve both good classification accuracy and prominent input features. These sub-populations interact with each other via an elitist selection mechanism then are combined by an ensemble learning method. The performance of the proposed algorithm is compared with the conventional machine learning algorithms and other state-of-art ensemble learning and coevolutionary algorithms on the six datasets. The empirical results on the test instances demonstrated the effectiveness of the new co-operative co-evolutionary approach for designing the ANN to solve the classification problems. This is our preliminary study, in the follow-up study, we will use parallel computing solutions to speed up the proposed algorithm as well as utilize the multi-objective optimization solution to further optimize both the number of selected features and classification results and in further research, we could apply this model to hyper-parameter searching in Deep Learning models.

References

[1] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 3, pp. 225–240, Sep 1998.

[2] E. Uchibe and M. Asada, "Incremental coevolution with competitive and cooperative tasks in a multirobot environment," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1412–1424, July 2006.

[3] B. Y. Hiew, S. C. Tan, and W. S. Lim, "A double-elimination-tournament-based competitive co-evolutionary artificial neural network classifier," *Neurocomputing*, vol. 249, pp. 345 – 356, 2017.

[4] M. Castellani, "Competitive co-evolution of multi-layer perceptron classifiers," *Soft Computing*, vol. 22, no. 10, pp. 3417–3432, May 2018.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] J. Tian, M. Li, and F. Chen, "Dual-population based coevolutionary algorithm for designing rbfnn with feature selection," *Expert Systems with Applications*, vol. 37, no. 10, pp. 6904 – 6918, 2010.

[7] J. Tian, M. Li, F. Chen, and J. Kou, "Coevolutionary learning of neural network ensemble for complex classification tasks," *Pattern Recognition*, vol. 45, no. 4, pp. 1373 – 1385, 2012.

[8] J. Tian, M. Li, F. Chen, and N. Feng, "Learning subspace-based rbfnn using coevolutionary algorithm for complex classification tasks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 47–61, Jan 2016.

[9] G. Mauša and T. G. Grbac, "Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study," *Applied Soft Computing*, vol. 55, pp. 331 – 351, 2017.

[10] A. Rosales-Pérez, A. E. Gutierrez-Rodríguez, S. García, H. Terashima-Marín, C. A. C. Coello, and F. Herrera, "Cooperative multi-objective evolutionary support vector machines

for multiclass problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 513–520.
[11] A. Rosales-Perez, S. Garcia, H. Terashima-Marin, C. A. Coello Coello, and F. Herrera, "Mc2esvm: Multiclass classification based on cooperative evolution of support vector machines," *IEEE Computational Intelligence Magazine*, vol. 13, no. 2, pp. 18–29, May 2018.
[12] V. Van Truong, B. Lam Thu, and N. Trung Thanh, "A coevolutionary approach for classification problems: Preliminary results," in *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, Nov 2018, pp. 81–86.
[13] L. T. Bui, V. T. Vu, and T. T. H. Dinh, "A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates," *Data & Knowledge Engineering*, vol. 114, pp. 40 – 66, 2018, special Issue on Knowledge and Systems Engineering (KSE 2016).
[14] T. T. H. Dinh, V. T. Vu, and T. L. Bui, "A multi-objective ensemble learning approach based on the non-dominated sorting differential evolution for forecasting currency exchange rates," in *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, Oct 2016, pp. 96–102.
[15] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[16] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, Feb 2010.
[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.
[18] ——, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ser. ICML'96. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, pp. 148–156.
[19] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16 – 28, 2014.
[20] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[21] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 1, pp. 131 – 156, 1997.
[22] H. Liu and Z. Zhao, *Manipulating Data and Dimension Reduction Methods: Feature Selection*. New York, NY: Springer New York, 2012, pp. 1790–1800.
[23] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection: An ever evolving frontier in data mining," in *Proceedings of the Fourth International Workshop on Feature Selection in Data Mining*, ser. Proceedings of Machine Learning Research, H. Liu, H. Motoda, R. Setiono, and Z. Zhao, Eds., vol. 10. Hyderabad, India: PMLR, 21 Jun 2010, pp. 4–13.
[24] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, Aug 2016.
[25] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109 – 118, 1990.
[26] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, March 2001.
[27] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.
[28] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, Aug 1995, pp. 278–282 vol.1.
[29] ——, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, Aug 1998.

Vu Van Truong received the B.S. degree in Geomatics from Le Quy Don Technical University, Ha Noi, Viet Nam in 2010 and the M.S. degree in Information systems from Le Quy Don Technical University, in 2014. He is currently pursuing the Ph.D. degree in Information technology at Le Quy Don Technical university. His research interest includes the Evolutionary Computations specialized with evolutionary multi-objective optimization, Machine learning, GIS and Remote sensing.

Bui Thu Lam received the Ph.D. degree in computer science from the University of New South Wales (UNSW), Australia, in 2007. He did postdoctoral training at UNSW from 2007 until 2009. He has been involved with academics including teaching and research since 1998. Currently, he is an Associate Professor at Le Quy Don Technical University, Hanoi, Vietnam. He is doing research in the field of evolutionary computation, specialized with evolutionary multi-objective optimization.

Nguyen Thanh Trung is a Reader in Operational Research and a co-director of the Liverpool Logistics, Offshore and Marine (LOOM) Research Institute. He is leading a research group on optimization, simulation and operational research, with focuses on maritime, transport (rail and road), logistics and IoT problems. He is working on solving various combinatorial problems such as vehicle scheduling, vehicle fleet sizing, vehicle routing, bin packing, berth planning, seat allocation for trains, minimizing train delays, crane trajectory optimizing, container stacking, vessel stowage planning, container loading, yard layout design problems. He is also leading research on using data analytics to study transportation systems and improve transportation operations.