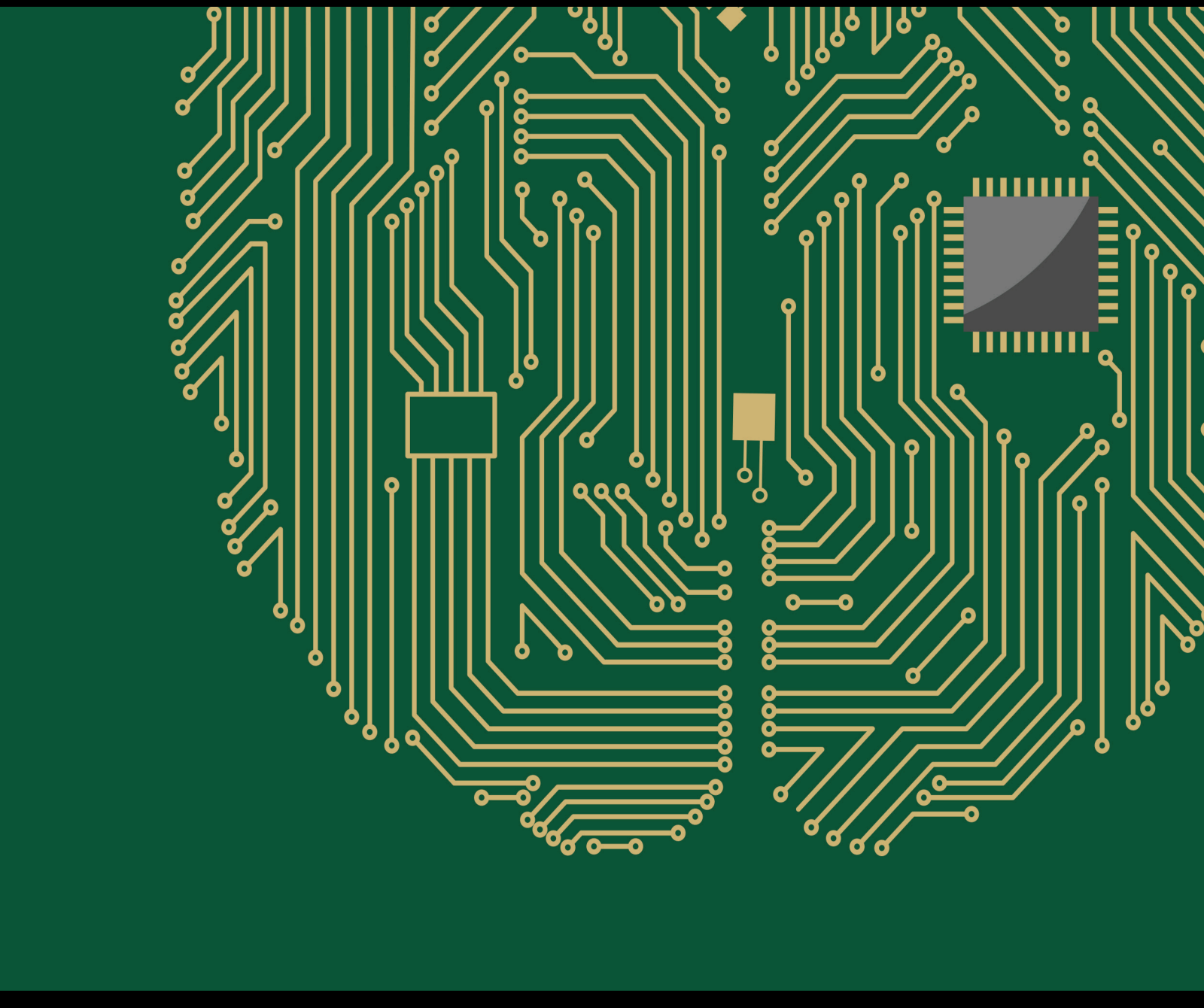
Computational Intelligence and Neuroscience

Advances in Recent Nature-Inspired Algorithms for Neural Engineering

Lead Guest Editor: Ricardo Soto
Guest Editors: Juan A. Gómez-Pulido, Eduardo Rodriguez-Tello, and Pedro Isasi



Advances in Recent Nature-Inspired Algorithms for Neural Engineering

Computational Intelligence and Neuroscience

Advances in Recent Nature-Inspired Algorithms for Neural Engineering

Lead Guest Editor: Ricardo Soto
Guest Editors: Juan A. Gómez-Pulido, Eduardo Rodriguez-Tello, and Pedro Isasi

Contents

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 7836239, 2 pages
https://doi.org/10.1155/2020/7836239

Editorial

Advances in Recent Nature-Inspired Algorithms for Neural Engineering

Ricardo Soto,[1] **Juan A. Gómez-Pulido**,[2] **Eduardo Rodriguez-Tello**,[3] **and Pedro Isasi**[4]

[1]*Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile*
[2]*Universidad de Extremadura, Cáceres, Spain*
[3]*Cinvestav Tamaulipas, Soto la Marina, Tamaulipas, Mexico*
[4]*Universidad Carlos III de Madrid, Getafe, Spain*

Correspondence should be addressed to Ricardo Soto; ricardo.soto@pucv.cl

Received 30 April 2020; Accepted 16 September 2020; Published 28 October 2020

Nature-inspired algorithms are general-purpose problem solvers that operate as a collection of intelligent agents, mimicking interesting phenomena from nature in order to efficiently solve a specific problem. Many optimization techniques belonging to artificial intelligence were born under this paradigm, which are able to combine data, knowledge, learning, and search strategies for building advanced algorithms. This is a particularly interesting area for neural engineering and other AI-related applications.

During the last three years, many new nature-inspired algorithms have been proposed, such as human behavior-based optimization, spotted hyena optimization, dragonfly optimization, Andean Condor Algorithm, water evaporation optimization, collective decision optimization, interactive search algorithm, vapour-liquid equilibrium metaheuristic, selfish herds algorithm, scattering and repulsive swarm intelligence, social engineering optimization, virus colony search, thermal exchange optimization, and kidney-inspired algorithm. Most of them involve interesting novel aspects that have enabled the efficient solving of complex problems, particularly from the NP-hard and NP-complete class of problems. In pursuing these aspects and possibilities, there are many trends and open issues that deserve to be investigated.

This special issue presents six original, high-quality articles, clearly focused on theoretical and practical aspects, including cutting-edge topics about neural networks, neural models, brain-computer interface, machine learning, and optimization algorithms.

The first article in this special issue is entitled "Double-Criteria Active Learning for Multiclass Brain-Computer Interfaces" and focuses on improving the data collection process for developing brain-computer interface (BCI) systems. Brain-computer interfaces (BCIs) allow users to control external devices via observed brain activity. Commonly, these systems are constructed by employing electroencephalography (EEG) signal datasets. However, EEG data acquisition is often lengthy and exhaustive because signals often vary during experiments due to both biological and technical causes. In this scenario, a main concern is to develop a robust BCI system but using as few data as possible. In this paper, the authors propose to combine a two-query active learning algorithm with an extreme learning machine (ELM) to solve this problem. The proposed approach is tested on different benchmark datasets, demonstrating that the performance of the proposed hybrid outperforms several state-of-the-art approaches.

The second paper presents a survey about a recent metaheuristic called cat swarm optimization (CSO). The survey focuses on exploring the different variations of CSO as well as on the different application domains where CSO has been successfully used. During the last 10 years, more than 20 interesting variations of CSO have been reported such as binary, multiobjective, parallel, and enhanced

parallel CSO. Hybrids involving genetic algorithms, particle swarm optimization, and simulated annealing have also been proposed. Weights, opposition-based learning, and quantum behavior have also been used to complement the basic CSO. In the application context, CSO has mainly been employed in electrical engineering, computer vision, signal processing, and system management. Applications of CSO for wireless devices, petroleum engineering, and civil engineering can also be found in the literature. Finally, CSO is tested on 23 classical benchmark functions and 10 recent benchmark functions confirming the performance of this interesting metaheuristic algorithm.

The third manuscript is called "A Dendritic Neuron Model with Adaptive Synapses Trained by Differential Evolution Algorithm" that proposes to improve the computational efficiency of the dendritic neuron model (DNM) by implementing adaptive synapses (DMAS) which are trained through a differential evolution (DE) algorithm. Extensive experimentation is presented, using five classification datasets from the UCI Machine Learning Repository, for assessing the performance of the proposed DE-DMAS algorithm with respect to those reached by a DNM and a neural network, both trained with the classic backpropagation algorithm. The experimental results in terms of correct rate, convergence rate, ROC curve, and cross-validation confirmed the superiority of DE-DMAS over the other compared algorithms, highlighting certain advantages of applying the self-adaptive synapses proposed in the paper: a stronger robustness, an improved performance of DNM, and a reduction of the input parameters needed by a DNM.

The fourth paper is entitled "A Db-Scan Binarization Algorithm Applied to Matrix Covering Problems" that presents a novel binarization algorithm, based on the db-scan unsupervised machine learning technique, which enables the use of continuous swarm optimization metaheuristics for the efficient solution of discrete combinatorial optimization problems. Using the well-known set covering problem as a case study, the behavior of the proposed db-scan binarization technique is analyzed when it is hybridized with two distinct algorithms: a particle swarm optimization and a cuckoo search. The individual performance of each one of these two hybrid metaheuristics was compared with respect to the one produced by other reference binarization methods employing k-means clusters and transfer functions (TFs). The experimental results allow the authors to conclude that the db-scan binarization consistently produced better results, both in terms of solution quality and expended computational time when compared with TFs. With regard to the cluster-based binarization, no significant differences in solution quality could be detected; however, the db-scan binarization significantly improved the convergence times.

The fifth paper is entitled "A Neural Network-Inspired Approach for Improved and True Movie Recommendations" that demonstrates how the application of a recent implementation of neural network improves the intelligent semantic analysis of user reviews and certain external resources in recommender systems. These systems, based on collaborative filtering, allow recommendations to be provided on certain items based on the experience of users when rating them, which finds many interesting applications nowadays, such as movie rating. Moreover, the authors of this article include, in addition to the ratings themselves, multiple external data resources related to user interaction in the context to make an intelligent analysis of the recommendation. The neural network considered is a recurrent implementation that processes the sequence of words semantically with user movie attention, which is a semantic emotion. This way, the recommender system evaluates multivariate movies (ratings, votes, Twitter likes, and reviews) to give a high-accurate recommendation. The system has been satisfactorily tested on a mobile app.

Finally, the article "Image Processing-Based Detection of Pipe Corrosion Using Texture Analysis and Metaheuristic-Optimized Machine Learning Approach" describes an interesting research case where a supervised learning technique adjusted by means of a nature-inspired optimization algorithm is applied to solve an image processing problem. The image processing is used for recognizing corroded and intact pipe surfaces, which is essential to determine current condition of the water supply and waste disposal systems. Nevertheless, the accurate detection is a crucial task where conventional methodologies based on human inspection can be quite inefficient. The intelligent approach constructs a decision boundary for this purpose, applying a support vector machine. In turn, this approach is optimized by using a nature-inspired metaheuristic, the differential flower pollination algorithm, which obtains the best hyperparameters of the support vector machine. As a result, this proposal gives an accuracy rate of 92.8% on a dataset consisting of 2,000 image samples.

Conflicts of Interest

The guest editors declare no conflicts of interest with regard to the articles published in this special issue.

Acknowledgments

The guest editors thank all authors who have submitted their manuscripts to this special issue and the reviewers for their hard work with the reviewing process.

Ricardo Soto
Juan A. Gómez-Pulido
Eduardo Rodriguez-Tello
Pedro Isasi

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 3287589, 13 pages
https://doi.org/10.1155/2020/3287589

Research Article

Double-Criteria Active Learning for Multiclass Brain-Computer Interfaces

Qingshan She,[1] **Kang Chen**,[1] **Zhizeng Luo**,[1] **Thinh Nguyen**,[2] **Thomas Potter**,[2] **and Yingchun Zhang**[2]

[1]*Institute of Intelligent Control and Robotics, Hangzhou Dianzi University, Hangzhou, Zhejiang 310018, China*
[2]*Department of Biomedical Engineering, University of Houston, Houston, TX 77204, USA*

Correspondence should be addressed to Qingshan She; qsshe@hdu.edu.cn and Yingchun Zhang; yzhang94@uh.edu

Received 25 July 2019; Accepted 11 February 2020; Published 10 March 2020

Guest Editor: Eduardo Rodriguez-Tello

Recent technological advances have enabled researchers to collect large amounts of electroencephalography (EEG) signals in labeled and unlabeled datasets. It is expensive and time consuming to collect labeled EEG data for use in brain-computer interface (BCI) systems, however. In this paper, a novel active learning method is proposed to minimize the amount of labeled, subject-specific EEG data required for effective classifier training, by combining measures of uncertainty and representativeness within an extreme learning machine (ELM). Following this approach, an ELM classifier was first used to select a relatively large batch of unlabeled examples, whose uncertainty was measured through the best-versus-second-best (BvSB) strategy. The diversity of each sample was then measured between the limited labeled training data and previously selected unlabeled samples, and similarity is measured among the previously selected samples. Finally, a tradeoff parameter is introduced to control the balance between informative and representative samples, and these samples are then used to construct a powerful ELM classifier. Extensive experiments were conducted using benchmark and multiclass motor imagery EEG datasets to evaluate the efficacy of the proposed method. Experimental results show that the performance of the new algorithm exceeds or matches those of several state-of-the-art active learning algorithms. It is thereby shown that the proposed method improves classifier performance and reduces the need for training samples in BCI applications.

1. Introduction

Brain-computer interfaces (BCIs) are systems that allow users to control external devices via observed brain activity, without relying on peripheral nerve or muscle activity [1]. The most common and useful BCIs are constructed using noninvasive brain activity recording techniques, such as electroencephalography (EEG) [2]. While EEG has become widely used for medical monitoring, rehabilitation, neuroprosthesis, and other healthcare applications [3–5], the data acquisition process can be lengthy and exhaustive for users [6]. In addition, EEG signals often vary over the course of an experiment due to both biological and technical causes, including subject-specific anatomical differences, intersession variability, and the attentional drift of subjects [7]. Consequently, users must often undergo a long data collection process to train a suitable BCI system. This poses a prohibitive burden for individuals with paralysis or a severely injured central nervous system, making it a major hurdle for therapeutic applications. It is therefore of the utmost importance that developed BCI systems achieve efficient and robust performance with as few samples as possible.

One approach that has been effectively applied to cases with limited training sets is the introduction of active learning (AL) to the BCI calibration procedure. AL queries the class labels of informative samples within the unlabeled sample space to maximize the efficiency of the learning model, and its application greatly reduces the complexity of training samples without any obvious loss of classification accuracy [8]. In essence, AL is an iterative sampling and labeling procedure. On each iteration, AL extracts the

sample or batch of samples that are most valuable for improving the current classification model from the unlabeled data pool, and these samples are then manually labeled. The greatest challenge for AL methods is identifying the most informative samples so that the maximum prediction accuracy can be achieved. A number of sample-selection criteria have then been applied to this task, including (1) query-by-committee (QBC), in which several distinct classifiers are used and the selected samples are those with the largest difference between the labels predicted by different classifiers [9–11]; (2) margin uncertainty sampling, wherein the samples are selected according to the maximum uncertainty based on their respective distances from the classification boundaries [12, 13]; (3) max-entropy sampling, which uses entropy as the uncertainty measure via probabilistic modeling [14, 15]; and (4) diversity sampling, which prefers selecting representative samples [16].

Over the past few decades, many supervised learning models have been adopted as baseline classifiers for AL, including linear discriminate analysis (LDA) [12, 17], support vector machine (SVM) [18, 19], artificial neural network (ANN) [20], and extreme learning machine (ELM) [21, 22]. Among these, the ELM has shown a high learning speed and good generalizability in preliminary testing. Additionally, it can be directly applied to both two-class and multiclass classification. To date, few studies have attempted to introduce AL algorithms into the ELM framework, although these have shown the method to be competitive with active SVMs [13, 14, 23]. Specifically, Yu et al. [13] proposed an active learning method called AL-ELM with the goal of saving training time, and results showed a classification performance comparable to that of AL-SVM [18]. Zhang and Er [23] then introduced the SEAL-ELM method by combining the online sequential ELM (OS-ELM) with AL, yielding a higher classification accuracy than offline combinations of AL and SVM on most test datasets. Regrettably, these existing active ELMs only consider a single-querying strategy, leaving space for improvement. The intuitive next step was to then introduce multiple querying strategies to select desirable samples. In fact, researchers have tried to combine two strategies in AL with base classifier SVM, with each performing better than their single-query counterparts [24–26]. At present, however, few implementations of active learning with ELM have been explored and applied for motor imagery- (MI-) based BCI systems [8, 13].

The present investigation intends to fill this gap by combining a two-query AL algorithm with an ELM and testing the method in a BCI application. A well-defined, general framework for active learning is thereby developed in a manner that accounts for both informativeness and representativeness in a multiclass situation. First, an uncertainty sampling strategy is adopted to select a relative large number of samples using the base ELM classifier. The degree of diversity between labeled training data and previously selected, unlabeled samples is then assessed, along with the degree of similarity between the unlabeled samples. Finally, highly informative and representative samples are used to update the ELM classifier through the introduction of a tradeoff parameter. The method is then tested on several benchmark datasets, along with a multiclass MI EEG dataset from BCI Competition IV Dataset 2a. Results demonstrate that the performance of the new method compares favorably with that of existing AL approaches.

Compared to existing ELM-based active learning algorithms, the new method has several noteworthy aspects:

(1) Considering that the use of a single uncertainty strategy may not take full advantage of the abundant information with unlabeled data, the AL-ELM algorithm is extended to combine two querying strategies (uncertainty and diversity) in order to select the most valuable samples from the unlabeled EEG data pool.

(2) The proposed algorithm provides a straightforward and meaningful way to measure representativeness by assaying two kinds of similarity: the similarity between a query sample and the labeled dataset, and the similarity between any two possible query samples. Employing this modified diversity strategy can help isolate highly representative samples during the active learning process.

2. Background Knowledge

2.1. Active Learning. Active learning methods typically comprise five basic components: **L**, **U**, **T**, **Q**, and **S**. **L** is the limited labeled dataset, **U** is the pool of samples/instances that contains abundant unlabeled instances, **T** is the classification model trained by **L**, **Q** is a query strategy to select the most valuable instances from **U**, and **S** is a human annotator that labels the selected instances correctly. AL is an iterative procedure that gradually adds the most important samples, queried by **Q** and labeled by **S**, from **U** to **L** to update the classification model **T**. The iterative AL process will continue in this manner until a predefined criterion is met. The ability to identify both an excellent classification model **T** and an effective query strategy **Q** is highly important for active learning algorithms.

Depending on the number of querying samples at each iteration, AL can be divided into two groups: stream-based AL and pool-based AL. In stream-based AL, the learner can only access one sample per iteration, while pool-based AL allows the learner to select a batch of samples during each iteration. Adjusting the selection method and number of queried samples then creates different AL algorithms, such as the QBC strategy, the uncertainty strategy, and the diversity strategy.

2.2. Basic ELM. Single-hidden-layer feedforward neural networks (SLFNs) are capable of universal approximation [21]. Consider a dataset containing N training samples, $\{X, Y\} = \{x_j, y_j\}_{j=1}^{N}$, with the input $x_j = [x_{j1}, x_{j2}, \ldots, x_{jp}]^T \in \mathbb{R}^p$ and a corresponding desired output of $y_j = [y_{j1}, y_{j2}, \ldots, y_{jq}]^T \in \mathbb{R}^q$, where p and q represent the respective dimensions and T denotes a transpose operation. Assuming that M is the number of hidden neurons, the output function of the SLFNs is mathematically modeled as

$$\mathbf{y}_j = \sum_{i=1}^{M} \boldsymbol{\beta}_i g\left(\mathbf{a}_i^{\mathrm{T}} \mathbf{x}_j + b_i\right), \quad j = 1, \ldots, N, \tag{1}$$

where $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{iq}]^{\mathrm{T}} \in \mathbb{R}^q$ is the weight vector that connects the i-th hidden neuron to the output neurons, $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{ip}]^{\mathrm{T}} \in \mathbb{R}^p$ is a randomly chosen input weight vector connecting the i-th hidden neuron to the input neurons, $b_i \in \mathbb{R}\,(i = 1, \ldots, M)$ is a randomly chosen bias of the i-th hidden node, and $g(\bullet)$ is the activation function, which can be any nonlinear piecewise continuous function (such as a sigmoid function or Gaussian function).

For convenience, equation (1) can be rewritten in matrix notation as

$$\mathbf{Y} = \mathbf{H}\boldsymbol{\beta}, \tag{2}$$

where $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times q}$ is the expected network output, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_M]^{\mathrm{T}} \in \mathbb{R}^{M \times q}$ denotes the weight of output layer, and $\mathbf{H}$ is the hidden layer output matrix which is defined as

$$\mathbf{H} = \begin{bmatrix} g\left(\mathbf{a}_1^{\mathrm{T}} \mathbf{x}_1 + b_1\right) & \ldots & g\left(\mathbf{a}_M^{\mathrm{T}} \mathbf{x}_1 + b_M\right) \\ \ldots & \ldots & \ldots \\ g\left(\mathbf{a}_1^{\mathrm{T}} \mathbf{x}_N + b_1\right) & \ldots & g\left(\mathbf{a}_M^{\mathrm{T}} \mathbf{x}_N + b_M\right) \end{bmatrix}_{N \times M}. \tag{3}$$

Unlike SLFNs, which require that the parameters of hidden neurons are adjusted during training, ELM adopts randomly generated hidden layer parameters and a tuning-free training strategy [22]. Even with these random hidden node parameters, ELM maintains the universal approximation capability of SLFNs [21]. The ELM training then aims to find suitable network parameters to minimize the approximation error $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2$. To achieve better generalization performance, a regularization parameter c is introduced in [27], with its corresponding objective function given as

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 + \frac{c}{2}\|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2, \tag{4}$$

where $\|\cdot\|_2$ denotes the l_2-norm of a matrix or a vector. We can obtain the output weight vector $\boldsymbol{\beta}$ using the Moore-Penrose principle. The solution of equation (4) is then $\boldsymbol{\beta} = ((\mathbf{I}/c) + \mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y}$ if $N > M$, and $\boldsymbol{\beta} = \mathbf{H}^{\mathrm{T}}((\mathbf{I}/c) + \mathbf{H}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{Y}$ if $N < M$.

3. The D-AL-ELM Method

In this section, we present a novel active learning algorithm, D-AL-ELM, that incorporates both the uncertainty and diversity strategies into consecutive steps. This identifies the most valuable, informative instances, which can then be selected to update the baseline classifier ELM during each learning round.

3.1. Discriminative Information by the Uncertainty Criterion. The uncertainty criterion is used to measure the informativeness of each sample. Uncertain samples which lie along the boundaries of different classes carry more information and play a more significant role in the construction of a classifier. In this implementation, the best-versus-second-best (BvSB) strategy is adopted to estimate the uncertainty of each sample. The BvSB strategy is based on a calculation of posterior probability, which considers the difference in probability values between the two classes with the highest estimated probabilities [28]. The outputs of the ELM then approximate the posterior probabilities of the different classes [13]. To do this, a sigmoid function is used to construct a mapping relationship between the real outputs of the ELM and the posterior probabilities, which is described as

$$p\left(y = 1 | f_i(x)\right) = \frac{1}{1 + \exp\left(-f_i(x)\right)}, \tag{5}$$

where $f_i(x)$ denotes the actual output of the i–th output node corresponding to the time instance x. In practice, equation (5) is only applied to two-class problems, such that the sum of the converted posterior probabilities for the instance x is always 1. However, application in multiclass problem may create a summed posterior proximity that exceeds 1, so calculated probabilities were normalized using the following formula:

$$\overline{p}\left(y = 1 | f_i(x)\right) = \frac{p\left(y = 1 | f_i(x)\right)}{\sum_{j=1}^{q} p\left(y = 1 | f_j(x)\right)}, \tag{6}$$

where $p(y = 1 | f_i(x))$ is the original probability of the i – th class.

Based on the above parameters, the BvSB strategy for each sample x can be expressed as

$$f(\mathbf{x})^{BvSB} = p\left(y_{\text{best}} | x\right) - p\left(y_{\text{second-best}} | x\right), \tag{7}$$

where $p(y_{\text{best}} | x)$ and $p(y_{\text{second-best}} | x)$ are the largest and second largest posterior probabilities of x, respectively. It should be noted that $f(\mathbf{x})^{BvSB}$ values are inversely related to the amount of uncertainty in a sample, with smaller values indicating greater uncertainty.

3.2. Representative Information by the Diversity Criterion. The selection of redundant or overly similar samples is of little use when attempting to construct a robust classifier. It is therefore necessary to use a diversity criterion to select a batch of samples which are diverse in nature. A feasible way of measuring the diversity of uncertain samples is the cosine angle distance. Following this approach, the similarity between two samples x_i and x_j is given by

$$S\left(x_i, x_j\right) = \left|\cos\left(x_i, x_j\right)\right| = \frac{\left|x_i \cdot x_j\right|}{\left\|x_i\right\|\left\|x_j\right\|}. \tag{8}$$

As can be seen from equation (8), the similarity $S(x_i, x_j)$ between the two samples x_i and x_j is small if these two samples are far from each other, and vice versa.

Suppose a batch of samples $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$. If the value of $max_{i=1,\ldots,n} S(x, w_i)$ is small, then the new sample x is diverse from the samples in $\mathbf{W}$. The similarity between a new sample x and $\mathbf{W}$ is defined as

$$div(x, \mathbf{W}) = \max_{w_j \in W} S\left(\mathrm{x}, \mathrm{w}_j\right). \tag{9}$$

Note that a smaller $div(x, \mathbf{W})$ value implies more diversity between x and $\mathbf{W}$.

In order to avoid selecting highly redundant samples, a novel diversity criterion is defined by combining the similarity between a query sample and the labeled set, and the similarity between any two candidate query samples at the same time. This calculation is given by

$$div(w_i) = div(w_i, \mathbf{W}) + div(w_i, \mathbf{L}), \tag{10}$$

where $div(w_i, \mathbf{W})$ represents the diversity between the sample w_i and the candidate set $\mathbf{W}$ (apart from w_i), and $div(w_i, \mathbf{L})$ represents the diversity between the sample w_i and the labeled training set $\mathbf{L}$.

3.3. Proposed D-AL-ELM Algorithm. The BvSB sampling method is a highly effective strategy for sample selection in active learning. Unfortunately, the BvSB may also select some uncertain samples which contain highly redundant information, which reduces the information available for classification. To address this problem, optimal samples were selected for classification. An ideal sample would not only furnish significant information for the classifier but also show diversity from the candidate unlabeled set and a minimal amount of redundancy within the labeled set.

The specific steps for each iteration of the D-AL-ELM algorithm are as follows:

Step 1: the BvSB strategy is adopted to select the h most uncertain samples from the unlabeled samples pool $\mathbf{U}$.

Step 2: let h represent the most uncertain samples, denoted by $\mathbf{W} = \{w_1, w_2, \ldots, w_h\} \subseteq \mathbf{U}$, and $\mathbf{S}_m$ be an arbitrary subset containing $m\,(m \leq h)$ samples selected from $\mathbf{W}$. Two evaluations are then performed, including the diversity from the labeled set $\mathbf{L}$ and the candidate set $\mathbf{S}_m$, and the similarity to the samples in $\mathbf{S}_m$.

Step 3: combining the discriminative and representative parts, the following formulation is obtained to select the m samples which are uncertain and diverse from each other:

$$\hat{\mathrm{x}}^{BvSB-div} = \arg\min_{s_i \in \mathbf{S}_m} \left\{ \lambda f(s_i)^{BvSB} + (1-\lambda)\left(div(s_i, \mathbf{S}_m) + div(s_i, \mathbf{L})\right)\right\}, \tag{11}$$

where λ is a tradeoff parameter that can balance the informativeness and representativeness criteria, and $\mathbf{L}$ is the labeled training set. $\hat{\mathbf{x}}^{BvSB-div}$ denotes the unlabeled sample that will be annotated and then included into the labeled training dataset for updating the ELM classifier.

The implementation of the proposed method is summarized in Algorithm 1.

In order to quantitatively evaluate the quality of each learning algorithm, area under the learning curve (ALC) [13] was calculated as a performance metric, which is described as

$$\mathrm{ALC} = \sum_{i=0}^{N_{iter}-1} \frac{y_i + y_{i+1}}{2N_{iter}}, \tag{12}$$

where N_{iter} denotes the number of learning iterations and y_i denotes the classification accuracy at the i-th learning round, such that ALC $\in [0, 1]$. It is noted that the larger the ALC value, the better the performance of the learning algorithm.

4. Experimental Results and Discussions

In this section, several experiments were performed on benchmark datasets and multiclass MI EEG datasets to evaluate the performance of the proposed D-AL-ELM method, in comparison with the other state-of-the-art approaches, including passive learning-based ELM, AL-ELM [13], and entropy-based ELM [14]. All methods were implemented using the MATLAB 2014b environment on a computer with a 2.5 GHz processor and 4.0 GB RAM.

4.1. Experiments on the Benchmark Datasets

4.1.1. Description of the Benchmark Datasets. A series of experiments were performed to evaluate the D-AL-ELM algorithm on 9 benchmark datasets from the KEEL dataset [29] and UCI dataset repositories [30]. Datasets included both binary and multiclass classification problems. As in [13], each raw dataset was divided into three parts: a small initial labeled set, a large unlabeled set, and a testing set. Testing instances comprised 50% of the total number of samples, while the percentage of initially labeled instances was assigned based on the size of the raw dataset and the number of categories. Detailed information regarding these datasets is presented in Table 1.

4.1.2. The Compared Algorithms, Parameter Settings, and the Performance Metric. In our experiments, we compare the proposed method with other state-of-the-art learning algorithms, including the following:

(1) PL-ELM: a passive learning algorithm that randomly selects some instances from the unlabeled set to train the initial classifier

(2) AL-ELM: a batch-mode active learning method based on ELM that uses the margin sampling strategy to select most uncertain examples for labeling [13]

(3) ELM-Entropy: querying discriminative samples through entropy measures [14]

In this study, the ELM adopted a sigmoid function as the activation function on the hidden level. A grid search based on tenfold cross-validation was then used to find the optimal number of hidden nodes M in the initial labeled set. For the regularization parameter c, a leave-one-out (LOO) cross-validation strategy was adopted based on the minimum MSE^{PRES} to find the optimal parameter value [31]. The optimal parameters M and c were determined from $M \in \{10, 20, \ldots, 200\}$ and $c \in \{e^{-5}, e^{-4.9}, \ldots, e^{5}\}$ on all the datasets except for the Letter dataset, where the parameter M was searched among $\{100, 200, \ldots, 1000\}$. Additionally, the tradeoff parameter $\lambda \in \{0.1, 0.2, \ldots, 0.9\}$ for equation (10) was chosen by grid search when M and c were fixed through the aforementioned methods. It should be noted that the

Inputs: $\mathbf{L} = \{(x_i, y_i)\}$ with n_l labeled samples, $\mathbf{U} = \{x_i\}$ with n_u unlabeled samples ($n_u \gg n_l$), the tradeoff parameter (λ), the number of samples selected on basis of their uncertainty (h), the batch size (m), and the terminating condition.
Output: The final learned ELM classifier.
(1) Train the ELM classifier using labeled set **L**.
(2) **Repeat**
(3) Calculate the estimated probability for the samples in **U** with the pretrained ELM classifier according to equation (5) or (6).
(4) Calculate the uncertainty level of each sample in **U** using equation (7).
(5) Include the h most uncertain samples into the set **W**.
(6) Select m samples from **W** using equation (11).
(7) Label the selected m samples.
(8) Update the labeled set **L** and unlabeled set **U**.
(9) Use the extended set **L** to train a new ELM classifier.
(10) **Until** the terminating condition is satisfied.
(11) Return the output the final learned ELM classifier.

Algorithm 1: The double-criteria active learning with the ELM algorithm.

Table 1: Details of the datasets including the numbers of the corresponding features and samples.

Dataset	Number of			Percentage of initial labeled instances (%)	Percentage of initial unlabeled instances (%)	Percentage of test instances (%)
	Features	Instances	Classes			
Liver	7	345	2	10	40	50
Diabetes	8	768	2	10	40	50
Wdbc	30	569	2	10	40	50
Twonorm	20	7400	2	1	49	50
Hayes-Roth	4	160	3	10	40	50
Iris	4	150	3	10	40	50
Wine	13	178	3	10	40	50
Segment	19	2310	7	10	40	50
Letter	16	20000	26	1	49	50

ELM parameter selection process was implemented in the same manner for all four methods.

Parameter details are shown in Table 2. It should be noted that the regularization parameter c was automatically identified using the LOO cross-validation and was not fixed during the learning process (thus, not shown in Table 2).

The batch mode was adopted to add new labeled instances. For the proposed D-AL-ELM method, h samples were first selected from the unlabeled set using equation (7), and then m samples were selected from the h samples using equation (11) and added to the labeled set for each iteration. In this experiment, h was empirically set to $h = 5m$ while m was 5% of the total instances in the original unlabeled set for 8 of the 9 datasets (except Letter). For the Letter dataset, m was 1% of the total instances in the original unlabeled set and h was set to $h = 2m$. These parameters were chosen to decrease the labeling cost, considering the size of the raw dataset and the number of categories.

To provide a fair comparison, all four methods queried m instances on each iteration. For each dataset, the procedure was stopped when the prediction accuracy stabilized or the number of selected samples was greater than 80% of the original unlabeled set. Additionally, to ensure the validity of experimental results, ten runs were performed for each learning method in each experiment, and average results were calculated.

4.1.3. Comparisons with Relevant State-of-the-Art Algorithms. Figure 1shows the trends of classification accuracy for the classifiers when trained by increasing numbers of data points across the various datasets. The results show that the proposed D-AL-ELM algorithm yielded the highest accuracy of all four methods on most of datasets (excepting the Wine and Iris datasets) at the last learning round. Specifically, the proposed method performed better than the remaining three methods over the majority of the active learning period for the Twonorm, Hayes-Roth, and Letter datasets. Moreover, the D-AL-ELM yielded the fastest learning rate over the first few iterations of the learning process for most datasets. This phenomenon indicates that the new method begins by effectively identifying the most informative and representative samples, unlike the other algorithms. Additionally, the ELM-Entropy approach generally yielded lower accuracy in multiclass classification, failing to surpass the PL-ELM on the Wine, Hayes-Roth, Iris, and Letter datasets. Another interesting observation was that the performance tended to degrade at a certain interval on the Segment dataset. It was considered that the Segment dataset may have a more irregular data structure, confounding the BvSB strategy and deteriorating the result. In cases such as this, a more adaptive stop criterion should be designed to stop the learning program at a more appropriate right time, before output degrades.

Table 2: Details of the optimal parameter settings for the different datasets using four methods.

Dataset	D-AL-ELM		AL-ELM	ELM-Entropy	PL-ELM
	M	λ	M	M	M
Liver	110	0.1	110	110	110
Diabetes	110	0.3	110	110	110
Wdbc	200	0.1	200	200	200
Twonorm	120	0.1	120	120	120
Hayes-Roth	100	0.3	100	100	100
Iris	170	0.9	170	170	170
Wine	60	0.7	60	60	60
Segment	200	0.6	200	200	200
Letter	700	0.4	700	700	700

Table 3 presents the mean classification accuracies of the four methods across the 9 datasets during the learning process. The ALC values for the four methods are further compared in Table 4. The results shown in Tables 3 and 4 indicate that the D-AL-ELM method yielded the best performance among all datasets for the tested methods. As in [13], the ALC metric not only was related to the learning velocity but also had close relationship to the quality of the learning model. The proposed D-AL-ELM outperformed the other methods in terms of ALC, with the AL-ELM performing second best, with an accuracy close to that of the D-AL-ELM on the Wdbc and Segment datasets. For the Wdbc dataset, although the proposed method had a slightly higher ALC value than the AL-ELM, both algorithms yielded the same mean accuracy for the overall learning process.

Finally, Table 5 reports the average time for the learning stage of each algorithm across all datasets. As expected, the PL-ELM was the fastest method because it lacked any criteria for the evaluation of samples. The proposed D-AL-ELM required slightly more learning time than AL-ELM and ELM-Entropy, since it computed both informativeness and representativeness of each instance. Considering the improvement of classification performance, this extra time may be deemed an acceptable tradeoff.

4.1.4. Analysis of Effect of Different Batch Size Values. In this experiment, the performance of the proposed active learning method was further evaluated using different batch sizes (i.e., h and m values).

The new method was tested with different querying sizes by varying the values of h and m, respectively. The remaining experimental settings were the same as in earlier experiments and testing was conducted on two benchmark datasets: Iris and Wine. The M and λ parameters were set as $M = 100$, $\lambda = 0.5$ to observe the performance with different batch sizes. Results are reported in Figures 2 and 3. In Figure 2, m was fixed at 5% of the total number of instances in the original unlabeled set and h was chosen from a candidate set $\{h = 1.1m, 1.2m, 1.5m, 2m, 4m, 5m\}$. In Figure 3, h was fixed at the value of $2m$ and m was chosen from $\{1\%, 2\%, 4\%, 6\%, 8\%\}$. It can be seen from Figure 2 that learning rates at the start of the curve increased with higher h values. Performance on Iris was less sensitive to the h value when enough instances were queried, and learning curves tended to be similar when query numbers and h values were large. In contrast, performance on the Wine dataset was more sensitive to h. This may be a result of the Wine dataset having a more complex distribution, which is difficult to capture. Although the D-AL-ELM performed differently on the two datasets, relatively larger h values were consistently able to obtain favorable performance. On the other hand, this increase in h value leads to a greater computational burden. Figure 3 shows the effects of different values of m on the Iris and Wine datasets. From this, it was observed that convergence can be more easily achieved with small m values. Alternatively, when m is large, more instances can be learned at each iteration and the number of total iterations greatly reduced, although this boost in performance does not provide substantially increased accuracy. In conclusion, optimizing the h and m values is not crucial for the D-AL-ELM, as most values yield similar results. It should be noted, however, that larger h and m are generally recommended.

4.2. Experiment on Multiclass MI EEG Data

4.2.1. Description of EEG Datasets. This section further evaluates the performance of the proposed D-AL-ELM method on multiclass MI EEG data from the BCI Competition IV Dataset 2a [32]. This dataset consists of the EEG signals from 9 subjects who performed 4 tasks, including left hand, right hand, foot, and tongue MI. EEG signals were recorded using 22 electrodes. Each subject underwent a training and testing session, each consisting of 288 trials (a total of 576 trials across the two sessions).

4.2.2. Experimental Setup and Parameter Settings. Data preprocessing was first performed on the raw EEG data. For each trial, features were extracted from the time segment lasting from 0.5 s to 2.5 s after the cue instructing the subject to perform MI. Each trial was first band-pass filtered from 8–30 Hz using a fifth-order Butterworth filter. Next, the dimension of the EEG signal was reduced to a 24-dimension feature set using the one-versus-rest common spatial pattern (OVR-CSP) algorithm [33], which is an effective and popular feature extraction method for EEG multiclassification that computes the features that discriminate each class from the remaining classes. Finally, the features

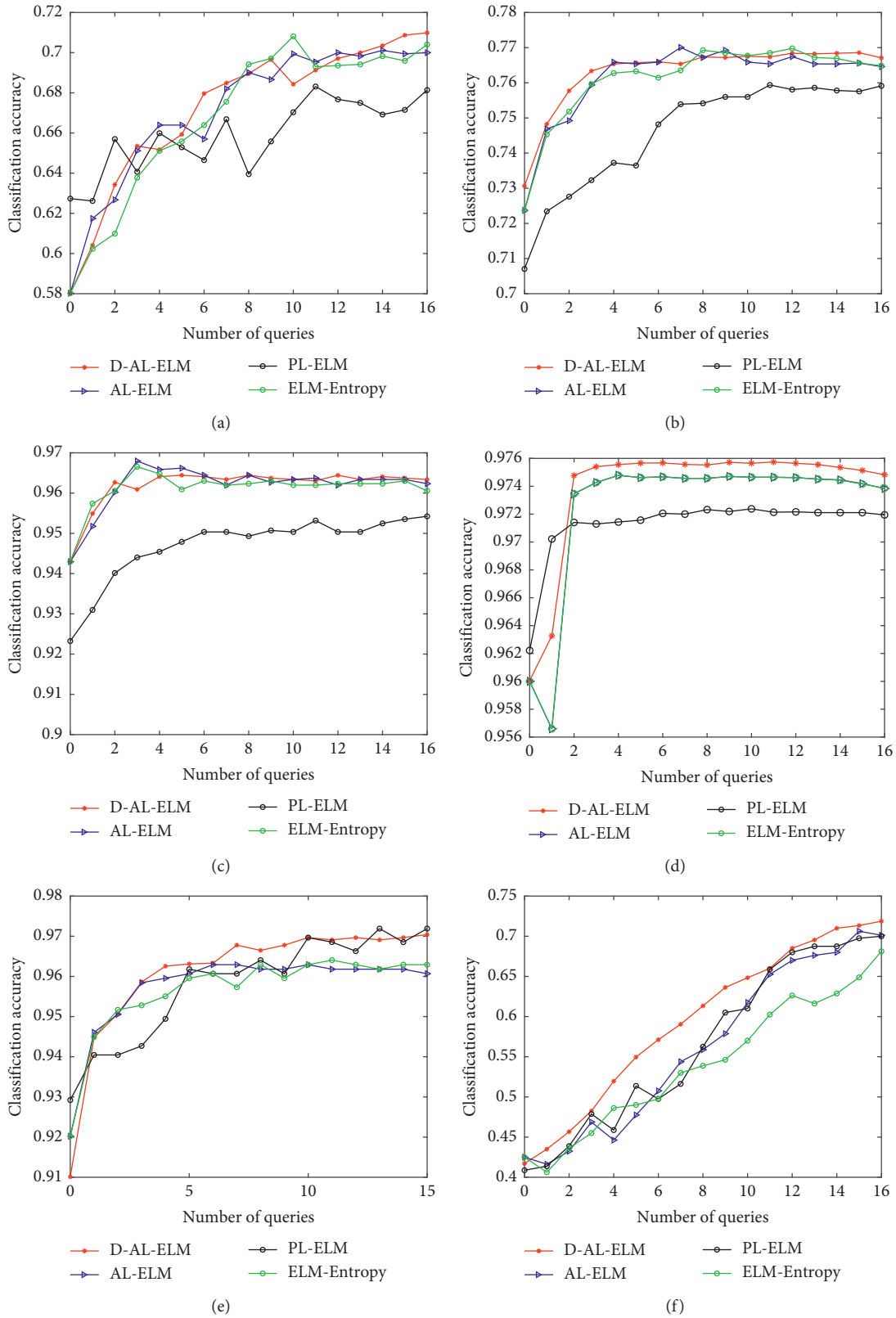


Figure 1: Continued.

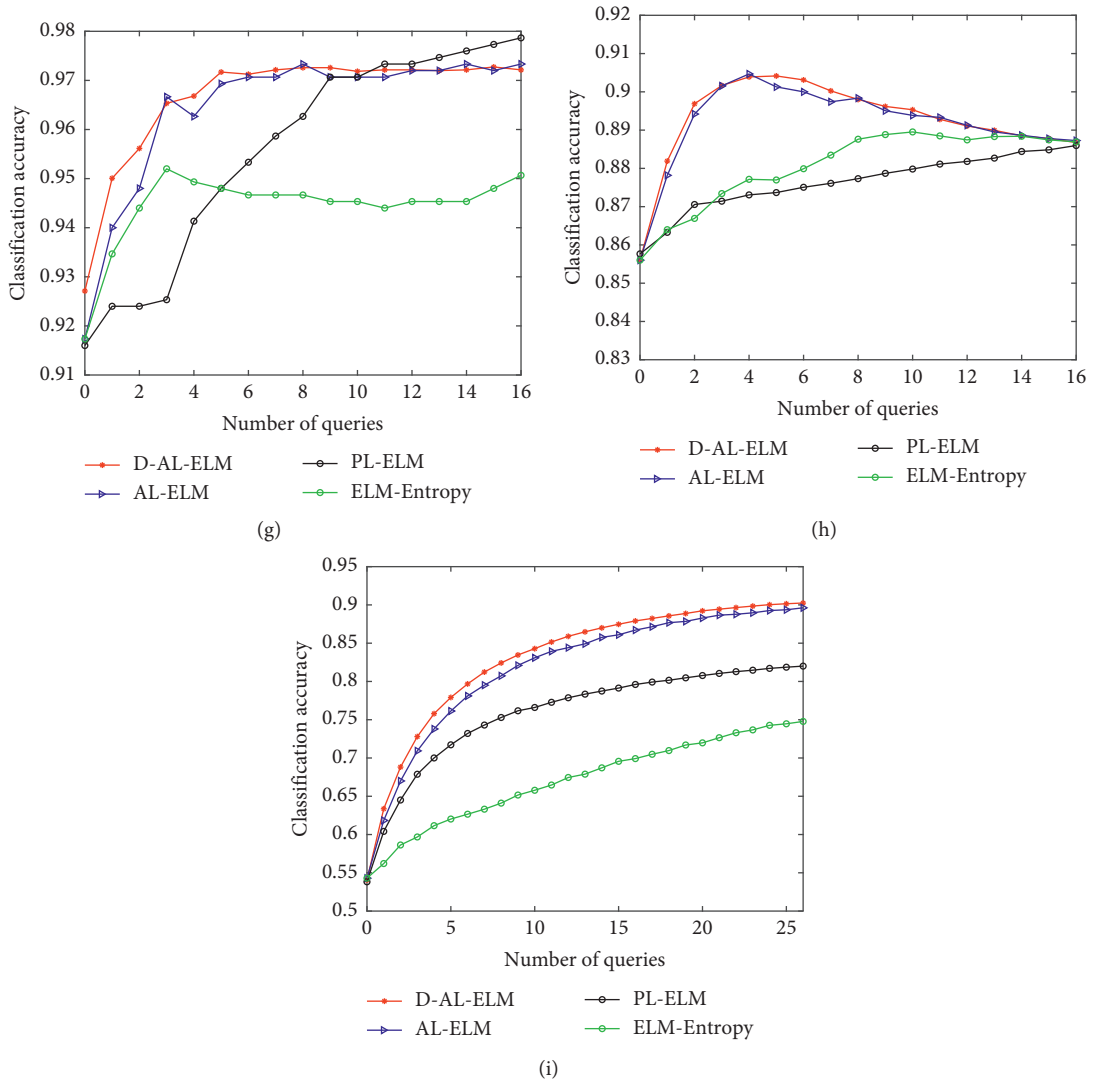


Figure 1: The learning curves of the four different learning algorithms on 9 benchmark datasets. (a) Liver. (b) Diabetes. (c) Wdbc. (d) Twonorm. (e) Wine. (f) Hayes-Roth. (g) Iris. (h) Segment. (i) Letter.

Table 3: Mean accuracy results of the learning processes on 9 datasets (%).

Dataset	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
Liver	**67.59**	67.46	67.14	66.14
Diabetes	**76.31**	76.13	76.12	74.60
Wdbc	**96.18**	**96.18**	96.11	94.69
Twonorm	**97.38**	97.25	97.25	97.13
Wine	**96.08**	95.72	95.64	95.79
Hayes-Roth	**59.43**	56.23	54.03	56.56
Iris	**96.65**	96.43	94.44	95.58
Segment	**89.26**	89.17	88.06	87.63
Letter	**82.89**	81.67	67.09	75.76

Table 4: ALC comparisons of four methods on 9 datasets.

Dataset	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
Liver	**0.7624**	0.7610	0.7572	0.7447
Diabetes	**0.7640**	0.7624	0.7622	0.7469
Wdbc	**0.9624**	0.9623	0.9616	0.9474
Twonorm	**0.9742**	0.9729	0.9729	0.9715
Wine	**0.9622**	0.9584	0.9573	0.9584
Hayes-Roth	**0.5959**	0.5622	0.5395	0.5663
Iris	**0.9676**	0.9655	0.9450	0.9563
Segment	**0.8939**	0.8929	0.8812	0.8766
Letter	**0.8330**	0.8204	0.6718	0.7606

extracted by OVR-CSP were discriminated using the different classification methods.

Optimal selection of the M, λ, and c parameters was performed in the same manner described in Section 4.1.2. The number of hidden nodes M was searched within $\{10, 20, \ldots, 150\}$. For each subject, the first 400 trials were considered as the training set, while the remaining 176 trials were used as the independent testing set [11]. The values for m and h were set at $m = 10$ and $h = 5m$. Finally, experiments included ten runs for each learning method from which average results were calculated.

Table 5: Average running time (s) for each learning algorithm.

Dataset	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
Liver	0.9141	0.7531	0.7719	0.7453
Diabetes	1.2031	1.0438	1.0060	0.9719
Wdbc	1.3484	1.2500	1.2828	1.2234
Twonorm	7.9813	5.3047	5.5875	4.8844
Wine	0.5391	0.4625	0.4625	0.4391
Hayes-Roth	0.5109	0.4516	0.4594	0.4203
Iris	0.5250	0.4469	0.4856	0.4313
Segment	3.6578	3.3906	3312	3.3250
Letter	121.8047	115.5203	123.0641	111.4641

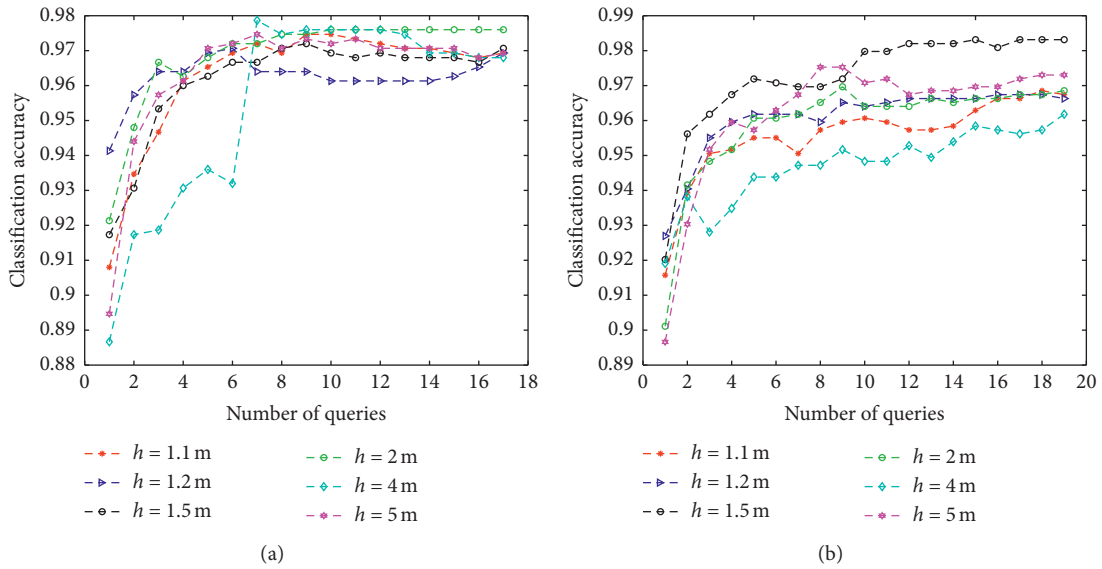


Figure 2: The learning curves of the proposed algorithm with different h values on Iris and Wine. (a) Iris. (b) Wine.

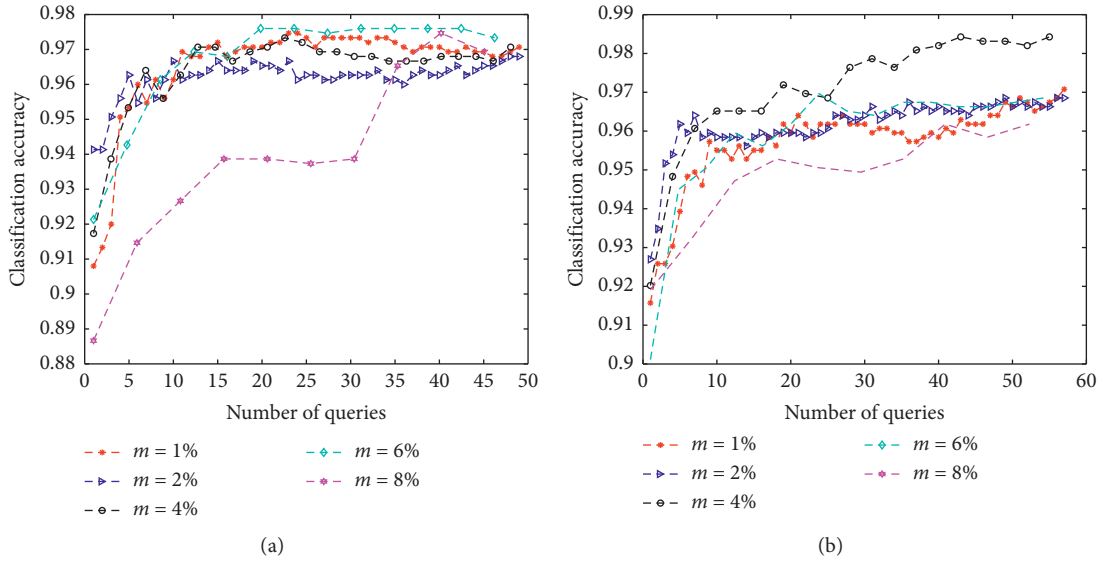


Figure 3: The learning curves of the proposed algorithm with different m values on Iris and Wine. (a) Iris. (b) Wine.

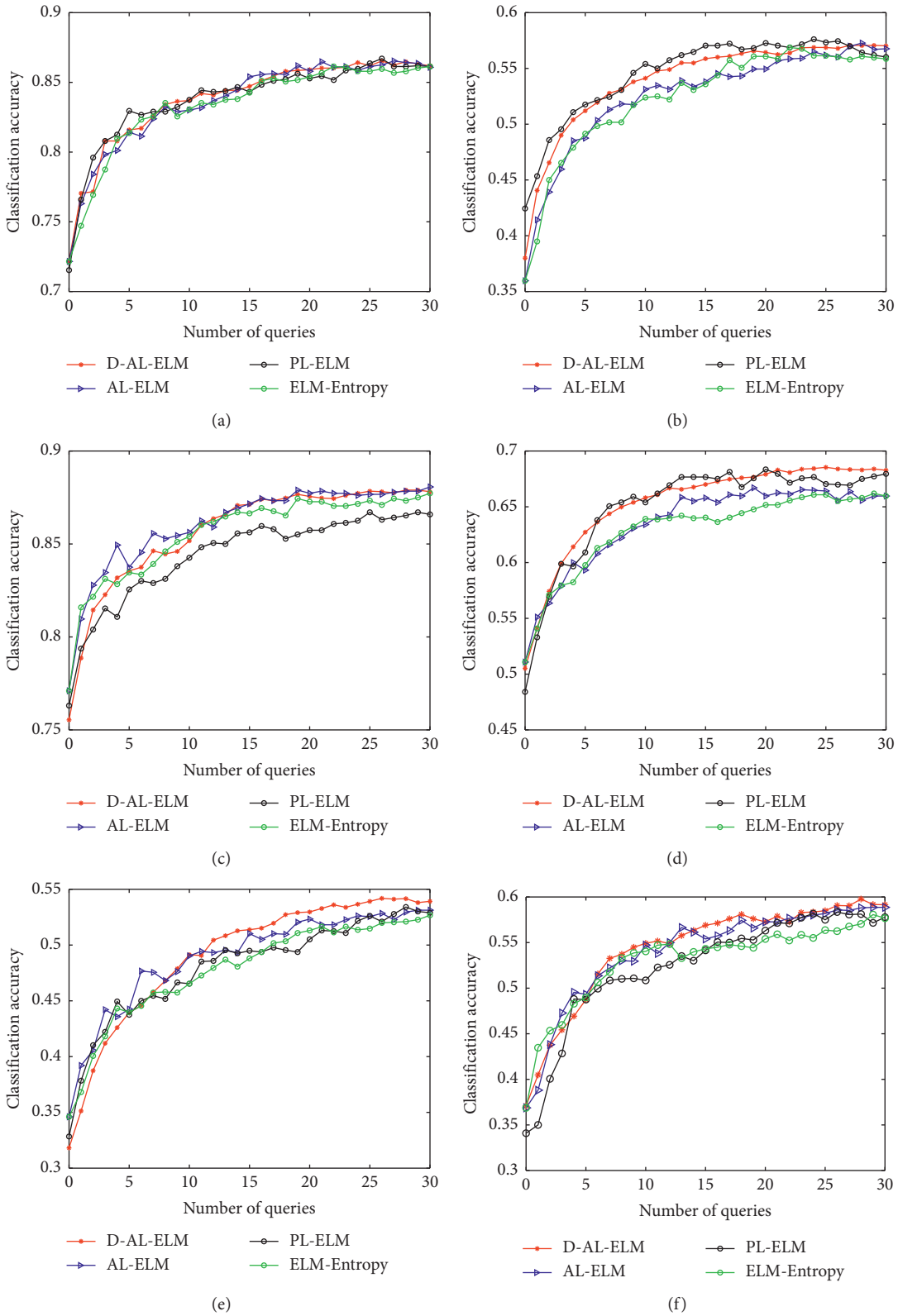


Figure 4: Continued.

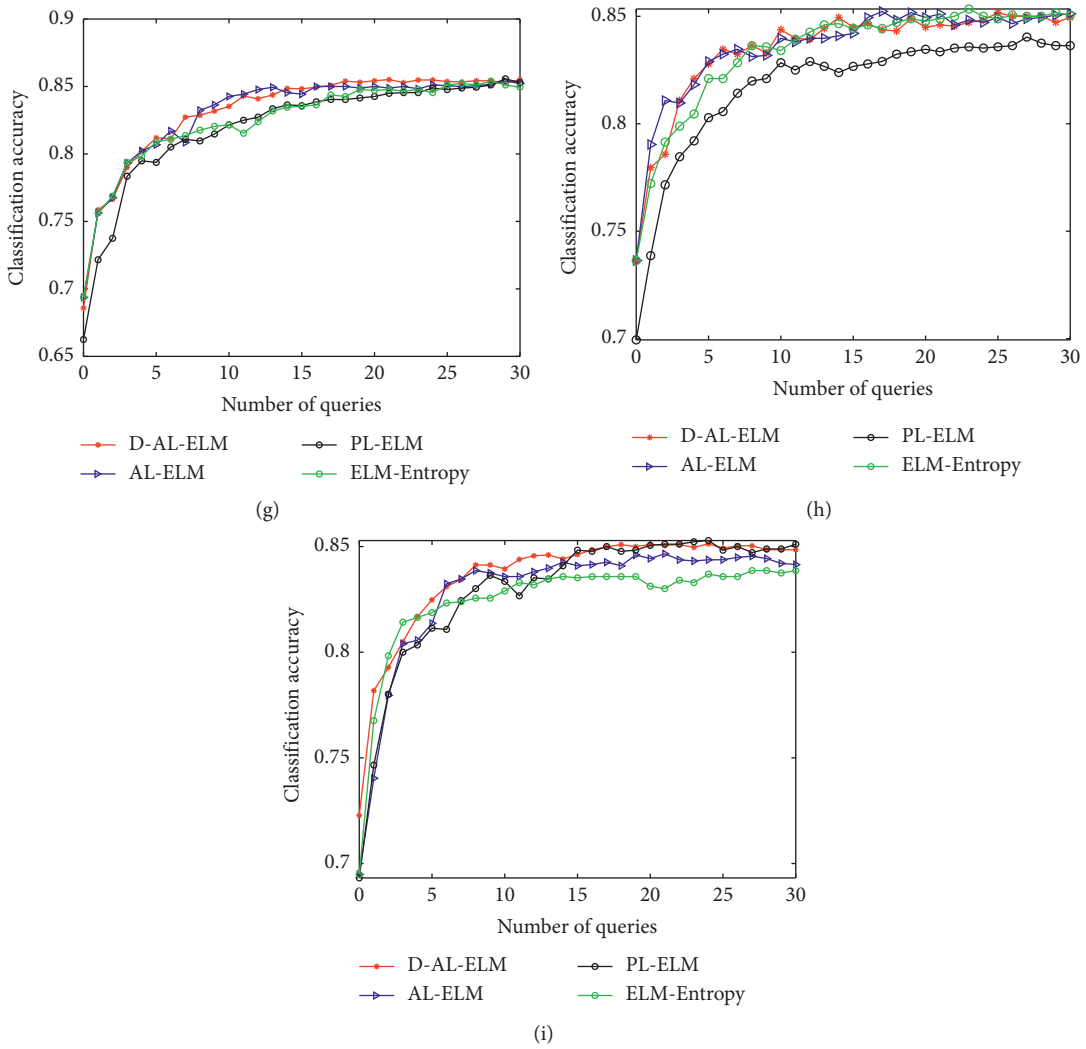


FIGURE 4: Learning curves of the four different learning algorithms on BCI Competition IV Dataset 2a. (a) S1. (b) S2. (c) S3. (d) S4. (e) S5. (f) S6. (g) S7. (h) S8. (i) S9.

4.2.3. Comparisons with Related Algorithms. Figure 4 illustrates the trend lines of classification accuracy when methods were applied to different testing datasets, while Table 6 lists the mean classification accuracies of the four methods during the learning process. Table 7 then provides the ALC results, while Table 8 shows the average running time (s) for the learning stage.

The results show that the performance of D-AL-ELM method is comparable to that of the AL-ELM and better than that of the ELM-Entropy and PL-ELM algorithms for all subjects (except for subject 2 in PL-ELM). Specifically, the proposed method surpassed the AL-ELM approach in 6 of the 9 subjects (1, 2, 4, 5, 6, 9) in terms of the ALC metric. For all 9 subjects, the D-AL-ELM method yielded a mean accuracy of 71.36%, higher than that of AL-ELM (70.92%), ELM-Entropy (70.34%), and PL-ELM (70.51%). These results demonstrate the effectiveness of the D-AL-ELM in selecting both informative and representative instances from unlabeled EEG samples. Additionally, they reveal that the proposed method can calibrate an effective classifier for MI EEG signals without the need for a large number of labeled training samples.

For comparative purposes, Table 8 also provides the average running time of each learning algorithm. Although the D-AL-ELM exhibited slightly longer training time than the other three methods, this may be considered a worthwhile tradeoff for the improved classification performance of the D-AL-ELM.

4.3. Discussion. In these experiments, the proposed D-AL-ELM method exhibited excellent performance in both classification accuracy and computational efficiency, as demonstrated on several benchmark datasets and an experimental MI EEG dataset. When compared to a passive learning-based ELM, D-AL-ELM achieved improved performance by effectively extracting the most valuable unlabeled samples. The D-AL-ELM also outperformed the AL-ELM and ELM-Entropy algorithms, which both employed a single-query strategy. Improvement was seen on all nine

Table 6: Mean accuracy (%) of the learning process on BCI Competition IV Dataset 2a.

Datasets	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
S1	**83.78**	83.64	83.32	83.76
S2	53.91	52.50	52.33	**54.55**
S3	85.66	**86.06**	85.57	84.39
S4	**65.32**	63.48	62.94	64.95
S5	**49.06**	48.90	47.79	48.00
S6	**54.31**	53.96	52.97	52.36
S7	83.15	**83.34**	82.45	81.96
S8	83.46	**83.56**	83.34	81.71
S9	**83.57**	82.80	82.39	82.91
Mean	**71.36**	70.92	70.34	70.51

Table 7: ALC values of the four methods on BCI Competition IV Dataset 2a.

Datasets	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
S1	**81.22**	0.8109	0.8076	81.21
S2	0.5238	0.5100	0.5085	**0.5296**
S3	0.8302	**0.8340**	0.8291	0.8177
S4	**0.6340**	0.6159	0.6105	0.6308
S5	**0.4764**	0.4749	0.4638	0.4662
S6	**0.5276**	0.5241	0.5144	0.5088
S7	0.8066	**0.8085**	0.7996	0.7952
S8	0.8090	**0.8100**	0.8079	0.7923
S9	**0.8103**	0.8032	0.7992	0.8042

Table 8: Average running time (s) of each learning algorithm.

Datasets	D-AL-ELM	AL-ELM	ELM-Entropy	PL-ELM
S1	1.7266	1.4625	1.4594	1.4172
S2	2.5125	2.2813	2.3859	2.2469
S3	1.7219	1.4859	1.4578	1.4234
S4	2.0719	1.7891	1.8328	1.8125
S5	2.0781	1.8281	1.8141	1.7719
S6	1.7609	1.4594	1.4094	1.3609
S7	2.4188	2.1969	2.1734	2.1641
S8	1.5562	1.3375	1.3609	1.3078
S9	1.2906	0.9484	0.9563	0.8906
Mean	1.9042	1.6432	1.6500	1.5995

datasets in Section 4.1, evidencing the ability of the D-AL-ELM to boost overall learning performance by combining the uncertainty and diversity strategies when updating the classifier with the selected samples. In terms of computational efficiency, the slight increase in training time for the D-AL-ELM, as compared to the PL-ELM, AL-ELM, and ELM-Entropy, was negligible in practice, especially when considering the improved classification accuracy. The experimental results then demonstrate that the proposed algorithm can effectively and comprehensively measure the representativeness of samples. Simultaneously, the proposed approach also measures how informative individual examples are, contributing to the improved classifier performance. Combining these factors, suitable instances can be selected for classifier construction.

Finally, the effectiveness of the D-AL-ELM was shown in its application to an experimental multiclass MI task from the BCI Competition IV Dataset 2a. Due to the low signal-to-noise ratio of EEG data, the applied algorithms struggled to generate adequate results. Consequently, hand-designed features were first extracted from the raw EEG data using the OVR-CSP approach, and the different AL algorithms were then used to further extract the unlabeled samples and calibrate a robust classifier. For subjects S1, S3, S7, S8, and S9, the D-AL-ELM yielded an acceptably high mean classification accuracy of over 80% for the whole learning process. Unfortunately, all tested methods performed poorly on subject S5. The proposed algorithm was only able to achieve 49.06% accuracy which, though insufficient, still ranked the highest among the applied methods.

5. Conclusion

In this paper, a novel active learning method with ELM, the D-AL-ELM, was developed for multiclassification. This new algorithm combines the uncertainty and diversity strategies and effectively reduces the expense and time cost of obtaining labeled data manually. For each sample, the proposed algorithm employs a BvSB strategy to measure informativeness and the cosine angle distance to measure diversity. The modified diversity measure not only estimates the diversity between the limited labeled training data and previously selected unlabeled samples, but also calculates the similarity among previously selected samples. Experimental results from several benchmark datasets and the multiclass MI EEG data from BCI Competition IV Dataset 2a were then used to verify the efficacy of the proposed D-AL-ELM algorithm. These results indicate that the performance of the proposed algorithm is consistently better than, or at least comparable to, that of other popular active learning techniques. Future work will then aim to develop online learning for the D-AL-ELM [23, 34]. In addition, an adaptive stopping criterion may be applied to promote the efficiency of the D-AL-ELM and improve its abilities for the classification and evaluation of MI EEG signals.

Data Availability

The BCI Competition IV Dataset 2a was used in our study, which is publicly available via the following link: http://www.bbci.de/competition/iv/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61871427 and 61671197). The authors would like to acknowledge the BCI Competition IV Dataset 2a which was used to test the algorithms proposed in this study.

References

[1] F. Lotte, L. Bougrain, A. Cichocki et al., "A review of classification algorithms for EEG-based brain-computer interfaces: a 10-year update," *Journal of Neural Engineering*, vol. 15, no. 3, p. 031005, 2018.

[2] P. Gonzalez-Navarro, M. Moghadamfalahi, M. Akcakaya, and D. Erdogmus, "Spatio-temporal EEG models for brain interfaces," *Signal Processing*, vol. 131, pp. 333–343, 2017.

[3] K. K. Ang and C. Guan, "EEG-based strategies to detect motor imagery for control and rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 4, pp. 392–401, 2017.

[4] R. Zhang, Y. Li, Y. Yan et al., "Control of a wheelchair in an indoor environment based on a brain-computer interface and automated navigation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 1, pp. 128–139, 2016.

[5] Q. She, H. Gan, Y. Ma et al., "Scale-Dependent signal identification in low-dimensional subspace: motor imagery task classification," *Neural Plasticity*, vol. 2016, p. 15, 2016.

[6] R. Li, T. Potter, W. Huang et al., "Enhancing performance of a hybrid EEG-fNIRS system using channel selection and early temporal features," *Frontiers in Human Neuroscience*, vol. 11, p. 462, 2017.

[7] P. Gaur, R. B. Pachori, H. Wang, and G. Prasad, "A multi-class EEG-based BCI classification using multivariate empirical mode decomposition based filtering and riemannian geometry," *Expert Systems with Applications*, vol. 95, no. 1, pp. 201–211, 2018.

[8] J. Li and L. Zhang, "Active training paradigm for motor imagery BCI," *Experimental Brain Research*, vol. 219, no. 2, pp. 245–254, 2012.

[9] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, no. 2/3, pp. 133–168, 1997.

[10] S. Kee, E. Del Castillo, and G. Runger, "Query-by-committee improvement with diversity and density in batch active learning," *Information Sciences*, vol. 454-455, pp. 401–418, 2018.

[11] V. Lawhern, D. Slayback, D. Wu et al., "Efficient labeling of EEG signal artifacts using active learning," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3217–3222, Kowloon, China, October 2015.

[12] M. Chen, X. Tan, J. Q. Gan et al., "A batch-mode active learning method based on the nearest average-class distance (NACD) for multiclass brain-computer interfaces," *Journal of Fiber Bioengineering & Informatics*, vol. 7, no. 4, pp. 627–636, 2014.

[13] H. Yu, C. Sun, W. Yang, X. Yang, and X. Zuo, "AL-ELM: one uncertainty-based active learning algorithm using extreme learning machine," *Neurocomputing*, vol. 166, pp. 140–150, 2015.

[14] R. Wang, S. Kwong, Q. Jiang et al., "Active learning based on single-hidden layer feed-forward neural network," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2158–2163, Kowloon, China, October 2015.

[15] Z. Qiu, D. J. Miller, and G. Kesidis, "A maximum entropy framework for semisupervised and active learning with unknown and label-scarce classes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 4, pp. 917–933, 2017.

[16] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Batch mode active sampling based on marginal probability distribution matching," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 3, pp. 1–25, 2013.

[17] H. Ibrahim, K. Abbas, H. Imali et al., "Multiclass informative instance transfer learning framework for motor imagery-based brain-computer interface," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 6323414, 12 pages, 2018.

[18] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 45–66, 2001.

[19] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Semisupervised SVM batch mode active learning with applications to image retrieval," *ACM Transactions on Information Systems*, vol. 27, no. 3, pp. 1–29, 2009.

[20] Q. Zhang and S. Sun, "Multiple-view multiple-learner active learning," *Pattern Recognition*, vol. 43, no. 9, pp. 3113–3119, 2010.

[21] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[22] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: a review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[23] Y. Zhang and M. J. Er, "Sequential active learning using meta-cognitive extreme learning machine," *Neurocomputing*, vol. 173, no. P3, pp. 835–844, 2016.

[24] B. Du, Z. Wang, L. Zhang et al., "Exploring representativeness and informativeness for active learning," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 14–26, 2017.

[25] Y. Gu, S. C. Chiu, and Z. Jin, "Active learning combining uncertainty and diversity for multi-class image classification," *IET Computer Vision*, vol. 9, no. 3, pp. 400–407, 2015.

[26] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 1936–1949, 2014.

[27] G. B. Huang, H. Zhou, X. Ding et al., "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems Man & Cybernetics Part B*, vol. 42, no. 2, pp. 513–529, 2012.

[28] A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos, "Scalable active learning for multiclass image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2259–2273, 2012.

[29] J. Alcalá-Fdez, A. Fernández, J. Luengo et al., "KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, pp. 255–287, 2011.

[30] D. Dua and E. Karra Taniskidou, *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml/, 2017.

[31] J. Cao, K. Zhang, M. Luo, C. Yin, and X. Lai, "Extreme learning machine and adaptive sparse representation for image classification," *Neural Networks*, vol. 81, pp. 91–102, 2016.

[32] M. Tangermann, K. R. Müller, A. Aertsen et al., "Review of the BCI competition IV," *Frontiers in Neuroscience*, vol. 6, p. 55, 2012.

[33] M. Meng, J. Zhu, Q. She et al., "Two-level feature extraction method for multi-class motor imagery EEG," *Acta Automatica Sinica*, vol. 42, no. 12, pp. 1915–1922, 2016.

[34] J. S. Lim, S. Lee, and H. S. Pang, "Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimations," *Neural Computing & Applications*, vol. 22, no. 3-4, pp. 569–576, 2013.

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 4854895, 20 pages
https://doi.org/10.1155/2020/4854895

Review Article

Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation

Aram M. Ahmed,[1,2] **Tarik A. Rashid**,[3] **and Soran Ab. M. Saeed**[2]

[1]*International Academic Office, Kurdistan Institution for Strategic Studies and Scientific Research, Sulaymaniyah 46001, Iraq*
[2]*Information Technology, Sulaimani Polytechnic University, Sulaymaniyah 46001, Iraq*
[3]*Computer Science and Engineering, University of Kurdistan Hewler, Erbil 44001, Iraq*

Correspondence should be addressed to Aram M. Ahmed; aramahmed@kissr.edu.krd

Received 25 July 2019; Revised 15 December 2019; Accepted 20 December 2019; Published 22 January 2020

Academic Editor: Juan A. Gómez-Pulido

This paper presents an in-depth survey and performance evaluation of cat swarm optimization (CSO) algorithm. CSO is a robust and powerful metaheuristic swarm-based optimization approach that has received very positive feedback since its emergence. It has been tackling many optimization problems, and many variants of it have been introduced. However, the literature lacks a detailed survey or a performance evaluation in this regard. Therefore, this paper is an attempt to review all these works, including its developments and applications, and group them accordingly. In addition, CSO is tested on 23 classical benchmark functions and 10 modern benchmark functions (CEC 2019). The results are then compared against three novel and powerful optimization algorithms, namely, dragonfly algorithm (DA), butterfly optimization algorithm (BOA), and fitness dependent optimizer (FDO). These algorithms are then ranked according to Friedman test, and the results show that CSO ranks first on the whole. Finally, statistical approaches are employed to further confirm the outperformance of CSO algorithm.

1. Introduction

Optimization is the process by which the optimal solution is selected for a given problem among many alternative solutions. One key issue of this process is the immensity of the search space for many real-life problems, in which it is not feasible for all solutions to be checked in a reasonable time. Nature-inspired algorithms are stochastic methods, which are designed to tackle these types of optimization problems. They usually integrate some deterministic and randomness techniques together and then iteratively compare a number of solutions until a satisfactory one is found. These algorithms can be categorized into trajectory-based and population-based classes [1]. In trajectory-based types, such as a simulated annealing algorithm [2], only one agent is searching in the search space to find the optimal solution, whereas, in the population-based algorithms, also known as swarm Intelligence, such as particle swarm optimization (PSO) [3], multiple agents are searching and communicating with each other in a decentralized manner to find the optimal solution. Agents usually move in two phases, namely, exploration and exploitation. In the first one, they move on a global scale to find promising areas, while in the second one, they search locally to discover better solutions in those promising areas found so far. Having a trade-off between these two phases, in any algorithm, is very crucial because biasing towards either exploration or exploitation would degrade the overall performance and produce undesirable results [1]. Therefore, more than hundreds of swarm intelligence algorithms have been proposed by researchers to achieve this balance and provide better solutions for the existing optimization problems.

Cat swarm optimization (CSO) is a swarm Intelligence algorithm, which was originally invented by Chu et al. in 2006 [4, 5]. It is inspired by the natural behavior of cats, and it has a novel technique in modeling exploration and exploitation phases. It has been successfully applied in various optimization fields of science and engineering. However, the

literature lacks a recent and detailed review of this algorithm. In addition, since 2006, CSO has not been compared against novel algorithms, i.e., it has been mostly compared with PSO algorithm while many new algorithms have been introduced since then. So, a question, which arises, is whether CSO competes with the novel algorithms or not? Therefore, experimenting CSO on a wider range of test functions and comparing it with new and robust algorithms will further reveal the potential of the algorithm. As a result, the aims of this paper are as follows: firstly, provide a comprehensive and detailed review of the state of art of CSO algorithm (see Figure 1), which shows the general framework for conducting the survey; secondly, evaluate the performance of CSO algorithm against modern metaheuristic algorithms. These should hugely help researchers to further work in the domain in terms of developments and applications.

The rest of the paper is organized as follows. Section 2 presents the original algorithm and its mathematical modeling. Section 3 is dedicated to reviewing all modified versions and variants of CSO. Section 4 summarizes the hybridizing CSO algorithm with ANN and other non-metaheuristic methods. Section 5 presents applications of the algorithm and groups them according to their disciplinary. Section 6 provides performance evaluation, where CSO is compared against dragonfly algorithm (DA) [6], butterfly optimization algorithm (BOA) [7], and fitness dependent optimizer (FDO) [8]. Finally, Section 7 provides the conclusion and future directions.

2. Original Cat Swarm Optimization Algorithm

The original cat swarm optimization is a continuous and single-objective algorithm [4, 5]. It is inspired by resting and tracing behaviours of cats. Cats seem to be lazy and spend most of their time resting. However, during their rests, their consciousness is very high and they are very aware of what is happening around them. So, they are constantly observing the surroundings intelligently and deliberately and when they see a target, they start moving towards it quickly. Therefore, CSO algorithm is modeled based on combining these two main deportments of cats.

CSO algorithm is composed of two modes, namely, tracing and seeking modes. Each cat represents a solution set, which has its own position, a fitness value, and a flag. The position is made up of M dimensions in the search space, and each dimension has its own velocity; the fitness value depicts how well the solution set (cat) is; finally, the flag is to classify the cats into either seeking or tracing mode. Thus, we should first specify how many cats should be engaged in the iteration and run them through the algorithm. The best cat in each iteration is saved into memory, and the one at the final iteration will represent the final solution.

2.1. General Structure of the Algorithms. The algorithm takes the following steps in order to search for optimal solutions:

(1) Specify the upper and lower bounds for the solution sets.

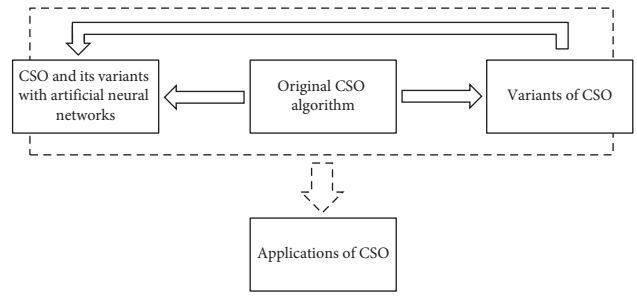


Figure 1: General framework for conducting the survey.

(2) Randomly generate N cats (solution sets) and spread them in the M dimensional space in which each cat has a random velocity value not larger than a predefined maximum velocity value.

(3) Randomly classify the cats into seeking and tracing modes according to MR. MR is a mixture ratio, which is chosen in the interval of [0, 1]. So, for example, if a number of cats N is equal to 10 and MR is set to 0.2, then 8 cats will be randomly chosen to go through seeking mode and the other 2 cats will go through tracing mode.

(4) Evaluate the fitness value of all the cats according to the domain-specified fitness function. Next, the best cat is chosen and saved into memory.

(5) The cats then move to either seeking or tracing mode.

(6) After the cats go through seeking or tracing mode, for the next iteration, randomly redistribute the cats into seeking or tracing modes based on MR.

(7) Check the termination condition; if satisfied; terminate the program; otherwise, repeat Step 4 to Step 6.

2.2. Seeking Mode. This mode imitates the resting behavior of cats, where four fundamental parameters play important roles: seeking memory pool (SMP), seeking range of the selected dimension (SRD), counts of dimension to change (CDC), and self-position considering (SPC). These values are all tuned and defined by the user through a trial-and-error method.

SMP specifies the size of seeking memory for cats, i.e., it defines number of candidate positions in which one of them is going to be chosen by the cat to go to, for example, if SMP was set to 5, then for each and every cat, 5 new random positions will be generated and one of them will be selected to be the next position of the cat. How to randomize the new positions will depend on the other two parameters that are CDC and SRD. CDC defines how many dimensions to be modified which is in the interval of [0, 1]. For example, if the search space has 5 dimensions and CDC is set to 0.2, then for each cat, four random dimensions out of the five need to be modified and the other one stays the same. SRD is the mutative ratio for the selected dimensions, i.e., it defines the amount of mutation and modifications for those dimensions that were selected by CDC. Finally, SPC is a Boolean value, which specifies whether the current position of a cat will

be selected as a candidate position for the next iteration or not. So, for example, if the SPC flag is set to true, then for each cat, we need to generate (SMP-1) number of candidates instead of SMP number as the current position is considered as one of them. Seeking mode steps are as follows:

(1) Make as many as SMP copies of the current position of Cat_k.

(2) For each copy, randomly select as many as CDC dimensions to be mutated. Moreover, randomly add or subtract SRD values from the current values, which replace the old positions as shown in the following equation:

$$Xjd_{\text{new}} = (1 + \text{rand} * \text{SRD}) * Xjd_{\text{old}}, \quad (1)$$

where Xjd_{old} is the current position; Xjd_{new} is the next position; j denotes the number of a cat and d denotes the dimensions; and rand is a random number in the interval of [0, 1].

(3) Evaluate the fitness value (FS) for all the candidate positions.

(4) Based on probability, select one of the candidate points to be the next position for the cat where candidate points with higher FS have more chance to be selected as shown in equation (2). However, if all fitness values are equal, then set all the selecting probability of each candidate point to be 1.

$$\text{Pi} = \frac{\left|\text{FS}_i - \text{FS}_b\right|}{\text{FS}_{\max} - \text{FS}_{\min}}, \quad \text{where } 0 < i < j. \quad (2)$$

If the objective is minimization, then $\text{FS}_b = \text{FS}_{\max}$; otherwise, $\text{FS}_b = \text{FS}_{\min}$.

2.3. Tracing Mode. This mode copies the tracing behavior of cats. For the first iteration, random velocity values are given to all dimensions of a cat's position. However, for later steps, velocity values need to be updated. Moving cats in this mode are as follows:

(1) Update velocities ($V_{k,d}$) for all dimensions according to equation (3).

(2) If a velocity value outranged the maximum value, then it is equal to the maximum velocity.

$$V_{k,d} = V_{k,d} + r_1 c_1 \left(X_{\text{best},d} - X_{k,d}\right). \quad (3)$$

(3) Update position of Cat_k according to the following equation:

$$X_{k,d} = X_{k,d} + V_{k,d}. \quad (4)$$

Refer to Figure 2 which recaps the whole algorithm in a diagram.

3. Variants of CSO

In the previous section, the original CSO was covered; this section briefly discusses all other variants of CSO found in the literature. Variants may include the following points: binary or multiobjective versions of the algorithm, changing parameters, altering steps, modifying the structure of the algorithm, or hybridizing it with other algorithms. Refer to Table 1, which presents a summary of these modifications and their results.

3.1. Discrete Binary Cat Swarm Optimization Algorithm (BCSO). Sharafi et al. introduced the BCSO Algorithm, which is the binary version of CSO [9]. In the seeking mode, the SRD parameter has been substituted by another parameter called the probability of mutation operation (PMO). However, the proceeding steps of seeking mode and the other three parameters stay the same. Accordingly, the dimensions are selected using the CDC and then PMO will be applied. In the tracing mode, the calculations of velocity and position equations have also been changed into a new form, in which the new position vector is composed of binary digits taken from either current position vector or global position vector (best position vector). Two velocity vectors are also defined in order to decide which vector (current or global) to choose from.

3.2. Multiobjective Cat Swarm Optimization (MOCSO). Pradhan and Panda proposed multiobjective cat swarm optimization (MOCSO) by extending CSO to deal with multiobjective problems [10]. MOCSO is combined with the concept of the external archive and Pareto dominance in order to handle the nondominated solutions.

3.3. Parallel Cat Swarm Optimization (PCSO). Tsai and pan introduced parallel cat swarm optimization (PCSO) [11]. This algorithm improved the CSO algorithm by eliminating the worst solutions. To achieve this, they first distribute the cats into subgroups, i.e., subpopulations. Cats in the seeking mode move as they do in the original algorithm. However, in the tracing mode, for each subgroup, the best cat will be saved into memory and will be considered as the local best. Furthermore, cats move towards the local best rather than the global best. Then, in each group, the cats are sorted according to their fitness function from best to worst. This procedure will continue for a number of iterations, which is specified by a parameter called ECH (a threshold that defines when to exchange the information of groups). For example, if ECH was equal to 20, then once every 20 iterations, the subgroups exchange information where the worst cats will be replaced by a randomly chosen local best of another group. These modifications lead the algorithm to be computationally faster and show more accuracy when the number of iteration is fewer and the population size is small.

3.4. CSO Clustering. Santosa and Ningrum improved the CSO algorithm and applied it for clustering purposes [12]. The main

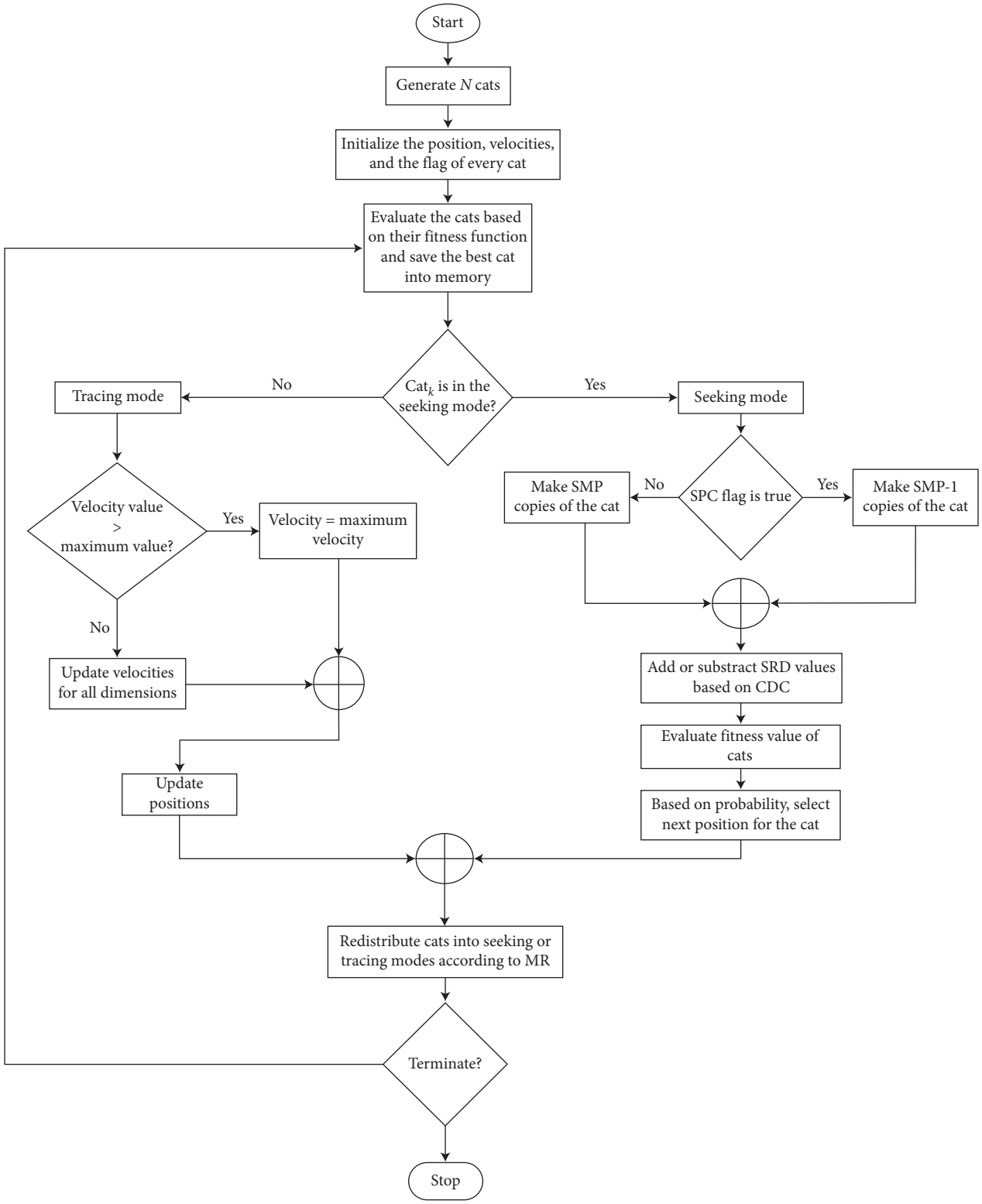


Figure 2: Cat swarm optimization algorithm general structure.

goal was to use CSO to cluster the data and find the best cluster center. The modifications they did were two main points: firstly, removing the mixture ratio (MR) and hence forcing all the cats to go through both seeking and tracing mode. This is aimed at shortening the time required to find the best cluster center. Secondly, always setting the CDC value to be 100%, instead of 80% as in the original CSO, in order to change all dimensions of the candidate cats and increase diversity.

Table 1: Summary of the modified versions of the CSO algorithm.

Comparison of	With	Testing field	Performance	Reference
CSO (original)	PSO and weighted-PSO	Six test functions	Better	[4, 5]
BCSO	GA, BPSO, and NBPSO	Four test functions (sphere, Rastrigin, Ackley, and Rosenbrock)	Better	[9]
MOCSO	NSGA-II	Cooperative spectrum sensing in cognitive radio	Better	[10]
PCSO	CSO and weighted-PSO	Three test functions (Rosenbrock, Rastrigrin, and Griewank)	Better when the number of iteration is fewer and the population size is small	[11]
CSO clustering	*K*-means and PSO clustering	Four different clustering datasets (Iris, Soybean, Glass, and Balance Scale)	More accurate but slower.	[12]
EPCSO	PCSO, PSO-LDIW, PSO-CREV, GCPSO, MPSO-TVAC, CPSO-H6, PSO-DVM	Five test functions and aircraft schedule recovery problem	Better	[13]
AICSO	CSO	Three test functions (Rastrigrin, Griewank, and Ackley)	Better	[14]
ADCSO	CSO	Six test functions (Rastrigrin, Griewank, Ackley, axis parallel, Trid10, and Zakharov)	Better except for Griewank test function.	[15]
Enhanced HCSO	PSO	Motion estimation block-matching	Better	[16, 17]
ICSO	PSO	Motion estimation block-matching	Better	[17]
OL-ICSO	*K*-median, PSO, CSO, and ICSO	ART1, ART2, Iris, CMC, Cancer, and Wine datasets	Better	[18]
CQCSO	QCSO, CSO, PSO, and CPSO	Five test functions (Schaffer, Shubert, Griewank, Rastrigrin, and Rosenbrock) and multipeak maximum power point tracking for a photovoltaic array under complex conditions	Better	[19]
ICSO	CSO and PSO	The 69-bus test distribution system	Better	[20]
ICSO	CSO, BCSO, AICSO, and EPCSO	Twelve test functions (sphere, Rosenbrock, Rastrigin, Griewank, Ackley, Step, Powell, Schwefel, Schaffer, Zakharov's, Michalewicz, quartic) and five real-life clustering problems (Iris, Cancer, CMC, Wine, and Glass)	Better	[21]
Hybrid PCSOABC	PCSO and ABC	Five test functions	Better	[22]
CSO-GA-PSO_{SVM}	CSO + SVM (CSO_{SVM})	66 feature points from each face of CK + (Cohn Kanade) dataset	Better	[23]
Hybrid CSO-based algorithm	GA, EA, SA, PSO, and AFS	School timetabling test instances	Better	[24]
Hybrid CSO-GA-SA	SLPA and CFinder	Seven datasets (Karate, Dolphin, Polbooks, Football, Net-Science, Power, Indian Railway)	Better	[25]
MCSO	CSO	Nine datasets from UCI	Better	[26]
MCSO	CSO	Eight dataset	Better	[27]
NMCSO	CSO, PSO	Sixteen benchmark functions	Better	[28]
ICSO	CSO	Ten datasets from UCI	Better	[29]
cCSO	DE, PSO, CSO	47 benchmark functions	Better	[30]
BBCSO	Binary particle swarm optimization (BPSO), binary genetic algorithm (BGA), binary CSO	0/1 Knapsack optimization problem	Better	[31]
CSO-CS	N/A	VRP instances from http://neo.lcc.uma.es/vrp/	N/A	[32]

3.5. Enhanced Parallel Cat Swarm Optimization (EPCSO). Tsai et al. further improved the PCSO Algorithm in terms of accuracy and performance by utilizing the orthogonal array of Taguchi method and called it enhanced parallel cat swarm optimization (EPCSO) [13]. Taguchi methods are statistical methods, which are invented by Japanese Engineer Genichi Taguchi. The idea is developed based on "ORTHOGONAL ARRAY" experiments, which improves the engineering productivity in the matters of cost, quality, and performance. In their proposed algorithm, the seeking mode of EPCSO is the same as the original CSO. However, the tracing mode has adopted the Taguchi orthogonal array. The aim of this is to improve the computational cost even when the number of agents increases. Therefore, two sets of candidate velocities will be created in the tracing mode. Then, based on the orthogonal array, the experiments will be run and accordingly the position of cats will be updated. Orouskhani et al. [14] added some partial modifications to EPCSO in order to further improve it and make it fit their application. The modifications were changing the representation of agents from the coordinate to a set; adding a newly defined cluster flag; and designing custom-made fitness function.

3.6. Average-Inertia Weighted CSO (AICSO). Orouskhani et al. introduced an inertia value to the velocity equation in order to achieve a balance between exploration and exploitation phase. They experimented that (w) value is better to be selected in the range of [0.4, 0.9] where at the beginning of the operation, it is set to 0.9, and as the iteration number moves forward, (w) value gradually becomes smaller until it reaches 0.4 at the final iteration. Large values of (w) assist global search; whereas small values of (w) assist the local search. In addition to adding inertia value, the position equation was also reformed to a new one, in which averages of current and previous positions, as well as an average of current and previous velocities, were taken in the equation [14].

3.7. Adaptive Dynamic Cat Swarm Optimization (ADCSO). Orouskhani et al. further enhanced the algorithm by introducing three main modifications [15]. Firstly, they introduced an adjustable inertia value to the velocity equation. This value gradually decreases as the dimension numbers increase. Therefore, it has the largest value for dimension one and vice versa. Secondly, they changed the constant (C) to an adjustable value. However, opposite to the inertia weight, it has the smallest value for dimension one and gradually increases until the final dimension where it has the largest value. Finally, they reformed the position equation by taking advantage of other dimensions' information.

3.8. Enhanced Hybrid Cat Swarm Optimization (Enhanced HCSO). Hadi and Sabah proposed a hybrid system and called it enhanced HCSO [16, 17]. The goal was to decrease the computation cost of the block matching process in video editing. In their proposal, they utilized a fitness calculation strategy in seeking mode of the algorithm. The idea was to avoid calculating some areas by deciding whether or not to do the calculation or estimate the next search location to move to. In addition, they also introduced the inertia weight to the tracing mode.

3.9. Improvement Structure of Cat Swarm Optimization (ICSO). Hadi and Sabah proposed combining two concepts together to improve the algorithm and named it ICSO. The first concept is parallel tracing mode and information exchanging, which was taken from PCSO. The second concept is the addition of an inertia weight to the position equation, which was taken from AICSO. They applied their algorithm for efficient motion estimation in block matching. Their goal was to enhance the performance and reduce the number of iterations without the degradation of the image quality [17].

3.10. Opposition-Based Learning-Improved CSO (OL-ICSO). Kumar and Sahoo first proposed using Cauchy mutation operator to improve the exploration phase of the CSO algorithm in [34]. Then, they introduced two more modifications to further improve the algorithm and named it opposition-based learning-improved CSO (OL-ICSO). They improved the population diversity of the algorithm by adopting opposition-based learning method. Finally, two heuristic mechanisms (for both seeking and tracing mode) were introduced. The goal of introducing these two mechanisms was to improve the diverse nature of the populations and prevent the possibility of falling the algorithm into the local optima when the solution lies near the boundary of the datasets and data vectors cross the boundary constraints frequently [18].

3.11. Chaos Quantum-Behaved Cat Swarm Optimization (CQCSO). Nie et al. improved the CSO algorithm in terms of accuracy and avoiding local optima trapping. They first introduced quantum-behaved cat swarm optimization (QCSO), which combined the CSO algorithm with quantum mechanics. Hence, the accuracy was improved and the algorithm avoided trapping in the local optima. Next, by incorporating a tent map technique, they proposed chaos quantum-behaved cat swarm optimization (CQCSO) algorithm. The idea of adding the tent map was to further improve the algorithm and again let the algorithm to jump out of the possible local optima points it might fall into [19].

3.12. Improved Cat Swarm Optimization (ICSO). In the original algorithm, cats are randomly selected to either go into seeking mode or tracing mode using a parameter called MR. However, Kanwar et al. changed the seeking mode by forcing the current best cat in each iteration to move to the seeking mode. Moreover, in their problem domain, the decision variables are firm integers while solutions in the original cat are continuous. Therefore, from selecting the best cat, two more cats are produced by flooring and ceiling its value. After that, all probable combinations of cats are produced from these two cats [20].

3.13. Improved Cat Swarm Optimization (ICSO). Kumar and Singh made two modifications to the improved CSO algorithm and called it ICSO [21]. They first improved the tracing mode by modifying the velocity and updating position equations. In the velocity equation, a random uniformly distributed vector and two adaptive parameters were added to tune global and local search movements. Secondly, a local search method was combined with the algorithm to prevent local optima problem.

3.14. Hybrid PCSOABC. Tsai et al. proposed a hybrid system by combining PCSO with ABC algorithms and named is hybrid PCSOABC [22]. The structure simply included running PCSO and ABC consecutively. Since PCSO performs faster with a small population size, the algorithm first starts with a small population and runs PCSO. After a predefined number of iterations, the population size will be increased and the ABC algorithm starts running. Since the proposed algorithm was simple and did not have any adjustable feedback parameters, it sometimes provided worse solutions than PCSO. Nevertheless, its convergence was faster than PCSO.

3.15. CSO-GA-PSOSVM. Vivek and Reddy proposed a new method by combining CSO with particle swarm intelligence (PSO), genetic algorithm (GA), and support vector machine (SVM) and called it CSO-GA-PSOSVM [23]. In their method, they adopted the GA mutation operator into the seeking mode of CSO in order to obtain divergence. In addition, they adopted all GA operators as well as PSO subtraction and addition operators into the tracing mode of CSO in order to obtain convergence. This hybrid metaheuristic system was then incorporated with the SVM classifier and applied on facial emotion recognition.

3.16. Hybrid CSO-Based Algorithm. Skoullis et al. introduced three modifications to the algorithm [24]. Firstly, they combined CSO with a local search refining procedure. Secondly, if the current cat is compared with the global best cat and their fitness values were the same, the global best cat will still be updated by the current cat. The aim of this is to achieve more diversity. Finally, cats are individually selected to go into either seeking mode or tracing mode.

3.17. Hybrid CSO-GA-SA. Sarswat et al. also proposed a hybrid system by combining CSO, GA, and SA and then incorporating it with a modularity-based method [25]. They named their algorithm hybrid CSO-GA-SA. The structure of the system was very simple and straight forward as it was composed of a sequential combination of CSO, GA, and SA. They applied the system to detect overlapping community structures and find near-optimal disjoint communities. Therefore, input datasets were firstly fed into CSO algorithm for a predefined number of iterations. The resulted cats were then converted into chromosomes and henceforth GA was applied on them. However, GA may fall into local optima, and to solve this issue, SA was applied afterward.

3.18. Modified Cat Swarm Optimization (MCSO). Lin et al. combined a mutation operator as a local search procedure with CSO algorithm to find better solutions in the area of the global best [26]. It is then used to optimize the feature selection and parameters of the support vector machine. Additionally, Mohapatra et al. used the idea of using mutation operation before distributing the cats into seeking or tracing modes [27].

3.19. Normal Mutation Strategy-Based Cat Swarm Optimization (NMCSO). Pappula et al. adopted a normal mutation technique to CSO algorithm in order to improve the exploration phase of the algorithm. They used sixteen benchmark functions to evaluate their proposed algorithm against CSO and PSO algorithms [28].

3.20. Improved Cat Swarm Optimization (ICSO). Lin et al. improved the seeking mode of CSO algorithm. Firstly, they used crossover operation to generate candidate positions. Secondly, they changed the value of the new position so that SRD value and current position have no correlations [29]. It is worth mentioning that there are four versions of CSO referenced in [17, 20, 21, 29], all having the same name (ICSO). However, their structures are different.

3.21. Compact Cat Swarm Optimization (CCSO). Zhao introduced a compact version of the CSO algorithm. A differential operator was used in the seeking mode of the proposed algorithm to replace the original mutation approach. In addition, a normal probability model was used in order to generate new individuals and denote a population of solutions [30].

3.22. Boolean Binary Cat Swarm Optimization (BBCSO). Siqueira et al. worked on simplifying the binary version of CSO in order to increase its efficiency. They reduced the number of equations, replaced the continues operators with logic gates, and finally integrated the roulette wheel approach with the MR parameter [31].

3.23. Hybrid Cat Swarm Optimization-Crow Search (CSO-CS) Algorithm. Pratiwi proposed a hybrid system by combining CSO algorithm with crow search (CS) algorithm. The algorithm first runs CSO algorithm followed by the memory update technique of the CS algorithm and then new positions will be generated. She applied her algorithm on vehicle routing problem [32].

4. CSO and its Variants with Artificial Neural Networks

Artificial neural networks are computing systems, which have countless numbers of applications in various fields. Earlier neural networks were used to be trained by conventional methods, such as the backpropagation algorithm. However, current neural networks are trained by nature-

inspired optimization algorithms. The training could be optimizing the node weights or even the network architectures [35]. CSO has also been extensively combined with neural networks in order to be applied in different application areas. This section briefly goes over those works, in which CSO is hybridized with ANN and similar methods.

4.1. CSO + ANN + OBD. Yusiong proposes combining ANN with CSO algorithm and optimal brain damage (OBD) approach. Firstly, the CSO algorithm is used as an optimization technique to train the ANN algorithm. Secondly, OBD is used as a pruning algorithm to decrease the complexity of ANN structure where less number of connections has been used. As a result, an artificial neural network was obtained that had less training errors and high classification accuracy [36].

4.2. ADCSO + GD + ANFIS. Orouskhani et al. combined ADCSO algorithm with gradient descent (GD) algorithm in order to tweak parameters of the adaptive network-based fuzzy inference system (ANFIS). In their method, the antecedent and consequent parameters of ANFIS were trained by CSO algorithm and GD algorithm consecutively [37].

4.3. CSO + SVM. Abed and Al-Asadi proposed a hybrid system based on SVM and CSO. The system was applied to electrocardiograms signals classification. They used CSO for the purpose of feature selection optimization and enhancing SVM parameters [38]. In addition, Lin et al. and Wang and Wu [39, 40] also combined CSO with SVM and applied it to a classroom response system.

4.4. CSO + WNN. Nanda proposed a hybrid system by combining wavelet neural network (WNN) and CSO algorithm. In their proposal, the CSO algorithm was used to train the weights of WNN in order to obtain the near-optimal weights [41].

4.5. BCSO + SVM. Mohamadeen et al. built a classification model based on BCSO and SVM and then applied it in a power system. The use of BCSO was to optimize SVM parameters [42].

4.6. CCSO + ANN. Wang et al. proposed designing an ANN that can handle randomness, fuzziness, and accumulative time effect in time series concurrently. In their work, the CSO algorithm was used to optimize the network structure and learning parameters at the same time [43].

4.7. CSO/PSO + ANN. Chittineni et al. used CSO and PSO algorithms to train ANN and then applied their method on stock market prediction. Their comparison results showed that CSO algorithm performed better than the PSO algorithm [44].

4.8. CS-FLANN. Kumar et al. combined the CSO algorithm with functional link artificial neural network (FLANN) to develop an evolutionary filter to remove Gaussian noise [45].

5. Applications of CSO

This section presents the applications of CSO algorithm, which are categorized into seven groups, namely, electrical engineering, computer vision, signal processing, system management and combinatorial optimization, wireless and WSN, petroleum engineering, and civil engineering. A summary of the purposes and results of these applications is provided in Table 2.

5.1. Electrical Engineering. CSO algorithm has been extensively applied in the electrical engineering field. Hwang et al. applied both CSO and PSO algorithms on an electrical payment system in order to minimize electricity costs for customers. Results indicated that CSO is more efficient and faster than PSO in finding the global best solution [46]. Economic load dispatch (ELD) and unit commitment (UC) are significant applications, in which the goal is to reduce the total cost of fuel is a power system. Hwang et al. applied the CSO algorithm on economic load dispatch (ELD) of wind and thermal generators [47]. Faraji et al. also proposed applying binary cat swarm optimization (BCSO) algorithm on UC and obtained better results compared to the previous approaches [48]. UPFC stands for unified power flow controller, which is an electrical device used in transmission systems to control both active and reactive power flows. Kumar and Kalavathi used CSO algorithm to optimize UPFC in order to improve the stability of the system [49]. Lenin and Reddy also applied ADCSO on reactive power dispatch problem with the aim to minimize active power loss [50]. Improving available transfer capability (ATC) is very significant in electrical engineering. Nireekshana et al. used CSO algorithm to regulate the position and control parameters of SVC and TCSC with the aim of maximizing power transfer transactions during normal and contingency cases [51]. The function of the transformers is to deliver electricity to consumers. Determining how reliable these transformers are in a power system is essential. Mohamadeen et al. proposed a classification model to classify the transformers according to their reliability status [42]. The model was built based on BCSO incorporation with SVM. The results are then compared with a similar model based on BPSO. It is shown that BCSO is more efficient in optimizing the SVM parameters. Wang et al. proposed designing an ANN that can handle randomness, fuzziness, and accumulative time effect in time series concurrently [43]. In their work, the CSO algorithm has been used to optimize the network structure and learning parameters at the same time. Then, the model was applied to two applications, which were individual household electric power consumption forecasting and Alkaline-surfactant-polymer (ASP) flooding oil recovery index forecasting in oilfield development. The current source inverter (CSI) is a conventional kind of power inverter topologies. Hosseinnia and Farsadi combined

Table 2: The purposes and results of using CSO algorithm in various applications.

Purpose	Results	Ref.
CSO applied on electrical payment system in order to minimize electricity cost for customers	CSO outperformed PSO	[46]
CSO applied on economic load dispatch (ELD) of wind and thermal generator	CSO outperformed PSO	[47]
BCSO applied on unit commitment (UC)	CSO outperformed LR, ICGA, BF, MILP, ICA, and SFLA	[48]
Applied CSO algorithm on UPFC to increase the stability of the system	IEEE 6-bus and 14-bus networks were used in the simulation experiments and desirable results were achieved	[49]
Applied ADCSO on reactive power dispatch problem to minimize active power loss	IEEE 57-bus system was used in the simulation experiments, in which ADCSO outperformed 16 other optimization algorithms	[50]
Applied CSO algorithm to regulate the position and control parameters of SVC and TCSC to improve available transfer capability (ATC)	IEEE 14-bus and IEEE 24-bus systems were used in the simulation experiments, in which the system provided better results after adopting CSO	[51]
Building a classification model based on BCSO and SVM to classify the transformers according to their reliability status.	The model performed better compared to a similar model, which was based on BPSO and VSM	[42]
Applied CSO to optimize the network structure and learning parameters of an ANN model named CPNN-CSO, which is used to predict household electric power consumption	CPNN-CSO outperformed ANFIS and similar methods with no CSO such as PNN and CPNN	[43]
Applied CSO and selective harmonic elimination (SHE) algorithm on current source inverter (CSI)	CSO successfully optimized the switching parameters of CSI and hence minimized the total harmonic distortion	[52]
Applied both CSO, PCSO, PSO-CFA, and ACO-ABC on distributed generation units on distribution networks	IEEE 33-bus and IEEE 69-bus distribution systems were used in the simulation experiments and CSO outperformed the other algorithms	[53]
Applied MCSO on MPPT to achieve global maximum power point (GMPP) tracking	MCSO outperformed PSO, MPSO, DE, GA, and HC algorithms	[54]
Applied BCSO to optimize the location of phasor measurement units and reduce the required number of PMUs	IEEE 14-bus and IEEE 30-bus test systems were used in the simulation. BCSO outperformed BPSO, generalized integer linear programming, and effective data structure-based algorithm	[55]
Used CSO algorithm to identify the parameters of single and double diode models in solar cell system	CSO outperformed PSO, GA, SA, PS, Newton, HS, GGHS, IGHS, ABSO, DE, and LMSA	[56]
Applied CSO and SVM to classify students' facial expression	The results show 100% classification accuracy for the selected 9 face expressions	[39]
Applied CSO and SVM to classify students' facial expression	The system achieved satisfactory results	[40]
Applied CSO-GA-PSOSVM to classify students' facial expression	The system achieved 99% classification accuracy	[23]
Applied CSO, HCSO and ICSO in block matching for efficient motion estimation	The system reduced computational complexity and provided faster convergence	[16, 17, 57]
Used CSO algorithm to retrieve watermarks similar to the original copy	CSO outperformed PSO and PSO time-varying inertia weight factor algorithms	[58, 59]
Sabah used EHCSO in an object-tracking system to obtain further efficiency and accuracy	The system yielded desirable results in terms of efficiency and accuracy	[60]
Used BCSO as a band selection method for hyperspectral images	BCSO outperformed PSO	[61]
Used CSO and multilevel thresholding for image segmentation	CSO outperformed PSO	[62]
Used CSO and multilevel thresholding for image segmentation	PSO outperformed CSO	[63]
Used CSO, ANN and wavelet entropy to build an AUD identification system.	CSO outperformed GA, IGA, PSO, and CSPSO	[64]
Used CSO and FLANN to remove the unwanted Gaussian noises from CT images	The proposed system outperformed mean filter and adaptive Wiener filter.	[45]
Used CSO with L-BFGS-B technique to register nonrigid multimodal images	The system yielded satisfactory results	[65]
Used CSO in image enhancement to optimize parameters of the histogram stretching technique	PSO outperformed CSO	[66]
Used CSO algorithm for IIR system identification	CSO outperformed GA and PSO	[67]

Table 2: Continued.

Purpose	Results	Ref.
Applied CSO to do direct and inverse modeling of linear and nonlinear plants	CSO outperformed GA and PSO	[68]
Used CSO and SVM for electrocardiograms signal classification	Optimizing SVM parameters using CSO improved the system in terms of accuracy	[38]
Applied CSO to increase reliability in a task allocation system	CSO outperformed GA and PSO	[69, 70]
Applied CSO on JSSP	The benchmark instances were taken from OR-Library. CSO yielded desirable results compared to the best recorded results in the dataset reference.	[71]
Applied BCSO on JSSP	ACO outperformed CSO and cuckoo search algorithms	[72]
Applied CSO on FSSP	Carlier, Heller, and Reeves benchmark instances were used, CSO can solve problems of up to 50 jobs accurately	[73]
Applied CSO on OSSP	CSO performs better than six metaheuristic algorithms in the literature.	[74]
Applied CSO on JSSP	CSO performs better than some conventional algorithms in terms of accuracy and speed.	[75]
Applied CSO on bag-of-tasks and workflow scheduling problems in cloud systems	CSO performs better than PSO and two other heuristic algorithms	[76]
Applied CSO on TSP and QAP	The benchmark instances were taken from TSPLIB and QAPLIB. The results show that CSO outperformed the best results recorded in those dataset references.	[77]
Comparison between CSO, cuckoo search, and bat-inspired algorithm to solve TSP problem	The benchmark instances are taken from STPLIB. The results show that CSO falls behind the other algorithms	[78]
Applied CSO and MCSO on workflow scheduling in cloud systems	CSO performs better than PSO	[79]
Applied BCSO on workflow scheduling in cloud systems	BCSO performs better than PSO and BPSO	[80]
Applied BCSO on SCP	BCSO performs better than ABC	[81]
Applied BCSO on SCP	BCSO performs better than binary teaching-learning-based optimization (BTLBO)	[82, 83]
Used a CSO as a clustering mechanism in web services.	CSO performs better than K-means	[84]
Applied hybrid CSO-GA-SA to find the overlapping community structures.	Very good results were achieved. Silhouette coefficient was used to verify these results in which was between 0.7 and 0.9	[25]
Used CSO to optimize the network structures for pinning control	CSO outperformed a number of heuristic methods	[85]
Applied CSO with local search refining procedure to address high school timetabling problem	CSO outperformed genetic algorithm (GA), evolutionary algorithm (EA), simulated annealing (SA), particle swarm optimization (PSO) and artificial fish swarm (AFS).	[24]
BCSO with dynamic mixture ratios to address the manufacturing cell design problem	BCSO can effectively tackle the MCDP problem regardless of the scale of the problem	[86]
Used CSO to find the optimal reservoir operation in water resource management	CSO outperformed GA	[87]
Applied CSO to classify the the feasibility of small loans in banking systems	CSO resulted in 76% of accuracy in comparison to 64% resulted from OLR procedure.	[88]
Used CSO, AEM and RPT to build a groundwater management systems	CSO outperformed a number of metaheuristic algorithms in addressing groundwater management problem	[89]
Applied CSO to solve the multidocument summarization problem	CSO outperformed harmonic search (HS) and PSO	[90]
Used CSO and (RPCM) to address groundwater resource management	CSO outperformed a similar model based on PSO	[91]
Applied CSO-CS to solve VRPTW	CSO-CS successfully solves the VRPTW problem. The results show that the algorithm convergences faster by increasing population and decreasing *cdc* parameter.	[32]

Table 2: Continued.

Purpose	Results	Ref.
Applied CSO and K-median to detect overlapping community in social networks	CSO and K-median provides better modularity than similar models based on PSO and BAT algorithm	[92]
Applied MOCSO, fitness sharing, and fuzzy mechanism on CR design	MOCSO outperformed MOPSO, NSGA-II and MOBFO	[93, 94]
Applied CSO and five other metaheuristic algorithms to design a CR engine	CSO outperformed the GA, PSO, DE, BFO and ABC algorithms	[95]
Applied EPCSO on WSN to be used as a routing algorithm	EPCSO outperformed AODV, a ladder diffusion using ACO and a ladder diffusion using CSO.	[33]
Applied CSO on WSN in order to solve optimal power allocation problem	PSO is marginally better for small networks. However, CSO outperformed PSO and cuckoo search algorithm	[96]
Applied CSO on WSN to optimize cluster head selection	The proposed system outperformed the existing systems by 75%.	[97]
Applied CSO on CR based smart grid communication network to optimize channel allocation	The proposed system obtains desirable results for both fairness-based and priority-based cases	[98]
Applied CSO in WSN to detect optimal location of sink nodes	CSO outperformed PSO in reducing total power consumption.	[99, 100]
Applied CSO on time modulated concentric circular antenna array to minimize the sidelobe level of antenna arrays and enhance the directivity	CSO outperformed RGA, PSO and DE algorithms	[101]
Applied CSO to optimize the radiation pattern controlling parameters for linear antenna arrays.	CSO successfully tunes the parameters and provides optimal designs of linear antenna arrays.	[102]
Applied Cauchy mutated CSO to make linear aperiodic arrays, where the goal was to reduce sidelobe level and control the null positions	The proposed system outperformed both CSO and PSO	[103]
Applied CSO and analytical formula-based objective function to optimize well placements	CSO outperformed DE algorithm	[104]
Applied CSO to optimize well placements considering oilfield constraints during development.	CSO outperformed GA and DE algorithms	[105]
CSO applied to optimize the network structure and learning parameters of an ANN model, which is used to predict an ASP flooding oil recovery index	The system successfully forecast the ASP flooding oil recovery index	[42]
Applied CSO to build an identification model to detect early cracks in beam type structures	CSO yields a desirable accuracy in detecting early cracks	[106]

selective harmonic elimination (SHE) in corporation with CSO algorithm and then applied it on current source inverter (CSI) [52]. The role of the CSO algorithm was to optimize and tune the switching parameters and minimize total harmonic distortion. El-Ela et al. [53] used CSO and PCSO to find the optimal place and size of distributed generation units on distribution networks. Guo et al. [54] used MCSO algorithm to propose a novel maximum power point tracking (MPPT) approach to obtain global maximum power point (GMPP) tracking. Srivastava et al. used BCSO algorithm to optimize the location of phasor measurement units and reduce the required number of PMUs [55]. Guo et al. used CSO algorithm to identify the parameters of single and double diode models in solar cell models [56].

5.2. Computer Vision. Facial emotion recognition is a biometric approach to identify human emotion and classify them accordingly. Lin et al. and Wang and Wu [39, 40] proposed a classroom response system by combining the CSO algorithm with support vector machine to classify student's facial expressions. Vivek and Reddy also used CSO-GA-PSOSVM algorithm for the same purpose [23]. Block matching in video processing is computationally expensive and time consuming. Hadi and Sabah used CSO algorithm in block matching for efficient motion estimation [57]. The aim was to decrease the number of positions that needs to be calculated within the search window during the block matching process, i.e., to enhance the performance and reduce the number of iterations without the degradation of the image quality. The authors further improved their work and achieved better results by replacing the CSO algorithm with HCSO and ICSO in [16, 17], respectively. Kalaiselvan et al. and Lavanya and Natarajan [58, 59] used CSO Algorithm to retrieve watermarks similar to the original copy. In video processing, object tracking is the process of determining the position of a moving object over time using a camera. Hadi and Sabah used EHCSO in an object-tracking system for further enhancement in terms of efficiency and accuracy [60]. Yan et al. used BCSO as a band selection method for hyperspectral images [61]. In computer vision, image segmentation refers to the process of dividing an image into multiple parts. Ansar and Bhattacharya and Karakoyun et al. [62, 63] proposed using CSO algorithm incorporation with the concept of multilevel thresholding for image segmentation purposes. Zhang et al. combined

wavelet entropy, ANN, and CSO algorithm to develop an alcohol use disorder (AUD) identification system [64]. Kumar et al. combined the CSO algorithm with functional link artificial neural network (FLANN) to remove the unwanted Gaussian noises from CT images [45]. Yang et al. combined CSO with L-BFGS-B technique to register nonrigid multimodal images [65]. Çam employed CSO algorithm to tune the parameters in the histogram stretching technique for the purpose of image enhancement [66].

5.3. Signal Processing. IIR filter stands for infinite impulse response. It is a discrete-time filter, which has applications in signal processing and communication. Panda et al. used CSO algorithm for IIR system identification [67]. The authors also applied CSO algorithm as an optimization mechanism to do direct and inverse modeling of linear and nonlinear plants [68]. Al-Asadi combined CSO Algorithm with SVM for electrocardiograms signal classification [38].

5.4. System Management and Combinatorial Optimization. In parallel computing, optimal task allocation is a key challenge. Shojaee et al. [69, 70] proposed using CSO algorithm to maximize system reliability. There are three basic scheduling problems, namely, open shop, job shop, and flow shop. These problems are classified as NP-hard and have many real-world applications. They coordinate assigning jobs to resources at particular times, where the objective is to minimize time consumption. However, their difference is mainly in having ordering constraints on operations. Bouzidi and Riffi applied the BCSO algorithm on job scheduling problem (JSSP) in [71]. They also made a comparative study between CSO and two other metaheuristic algorithms, namely, cuckoo search (CS) algorithm and the ant colony optimization (ACO) for JSSP in [72]. Then, they used the CSO algorithm to solve flow shop scheduling (FSSP) [73] and open shop scheduling problems (OSSP) as well [74]. Moreover, Dani et al. also applied CSO algorithm on JSSP in which they used a nonconventional approach to represent cat positions [75]. Maurya and Tripathi also applied CSO algorithm on bag-of-tasks and workflow scheduling problems in cloud systems [76]. Bouzidi and Riffi applied CSO algorithm on the traveling salesman problem (TSP) and the quadratic assignment problem (QAP), which are two combinatorial optimization problems [77]. Bouzidi et al. also made a comparative study between CSO algorithm, cuckoo search algorithm, and bat-inspired algorithm for addressing TSP [78]. In cloud computing, minimizing the total execution cost while allocating tasks to processing resources is a key problem. Bilgaiyan et al. applied CSO and MCSO algorithms on workflow scheduling in cloud systems [79]. In addition, Kumar et al. also applied BCSO on workflow scheduling in cloud systems [80]. Set cover problem (SCP) is considered as an NP-complete problem. Crawford et al. successfully applied the BCSO Algorithm to this problem [81]. They further improved this work by using Binarization techniques and selecting different parameters for each test example sets [82, 83]. Web services provide a standardized communication between applications over the web which have many important applications. However, discovering appropriate web services for a given task is challenging. Kotekar and Kamath used a CSO-based approach as a clustering algorithm to group service documents according to their functionality similarities [84]. Sarswat et al. applied Hybrid CSO-GA-SA to detect the overlapping community structures and find the near-optimal disjoint communities [25]. Optimizing the problem of controlling complex network systems is critical in many areas of science and engineering. Orouskhani et al. applied CSO algorithm to address a number of problems in optimal pinning controllability and thus optimized the network structure [85]. Skoullis et al. combined the CSO algorithm with local search refining procedure and applied it on high school timetabling problem [24]. Soto et al. combined BCSO with dynamic mixture ratios to organize the cells in manufacturing cell design problem [86]. Bahrami et al. applied a CSO algorithm on water resource management where the algorithm was used to find the optimal reservoir operation [87]. Kencana et al. used CSO algorithm to classify the feasibility of small loans in banking systems [88]. Majumder and Eldho combined the CSO algorithm with the analytic element method (AEM) and reverse particle tracking (RPT) to model novel groundwater management systems [89]. Rautray and Balabantaray used CSO algorithm to solve the multidocument summarization problem [90]. Thomas et al. combined radial point collocation meshfree (RPCM) approach with CSO algorithm to be used in the groundwater resource management [91]. Pratiwi created a hybrid system by combining the CSO algorithm and crow search (CS) algorithm and then used it to address the vehicle routing problem with time windows (VRPTW) [32]. Naem et al. proposed a modularity-based system by combining the CSO algorithm with K-median clustering technique to detect overlapping community in social networks [92].

5.5. Wireless and WSN. The ever-growing wireless devices push researchers to use electromagnetic spectrum bands more wisely. Cognitive radio (CR) is an effective dynamic spectrum allocation in which spectrums are dynamically assigned based on a specific time or location. Pradhan and Panda in [93, 94] combined MOCSO with fitness sharing and fuzzy mechanism and applied it on CR design. They also conducted a comparative analysis and proposed a generalized method to design a CR engine based on six evolutionary algorithms [95]. Wireless sensor network (WSN) refers to a group of nodes (wireless sensors) that form a network to monitor physical or environmental conditions. The gathered data need to be forwarded among the nodes and each node requires having a routing path. Kong et al. proposed applying enhanced parallel cat swarm optimization (EPCSO) algorithm in this area as a routing algorithm [33]. Another concern in the context of WSN is minimizing the total power consumption while satisfying the performance criteria. So, Tsiflikiotis and Goudos addressed this problem which is known as optimal power allocation problem, and for that, three metaheuristic algorithms were

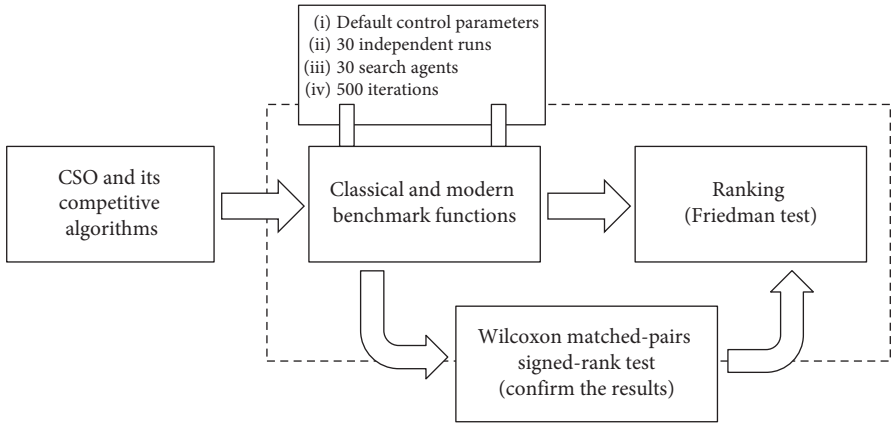


Figure 3: General framework of the performance evaluation process.

presented and compared [96]. Moreover, Pushpalatha and Kousalya applied CSO in WSN for optimizing cluster head selection which helps in energy saving and available bandwidth [97]. Alam et al. also applied CSO algorithm in a clustering-based method to handle channel allocation (CA) issue between secondary users with respect to practical constraints in the smart grid environment [98]. The authors of [99, 100] used the CSO algorithm to find the optimal location of sink nodes in WSN. Ram et al. applied CSO algorithm to minimize the sidelobe level of antenna arrays and enhance the directivity [101]. Ram et al. used CSO to optimize controlling parameters of linear antenna arrays and produce optimal designs [102]. Pappula and Ghosh also used Cauchy mutated CSO to make linear aperiodic arrays, where the goal was to reduce sidelobe level and control the null positions [103].

5.6. Petroleum Engineering. CSO algorithm has also been applied in the petroleum engineering field. For example, it was used as a good placement optimization approach by Chen et al. in [104, 105]. Furthermore, Wang et al. used CSO algorithm as an ASP flooding oil recovery index forecasting approach [43].

5.7. Civil Engineering. Ghadim et al. used CSO algorithm to create an identification model that detects early cracks in building structures [106].

6. Performance Evaluation

Many variants and applications of CSO algorithm were discussed in the above sections. However, benchmarking these versions and conducting a comparative analysis between them were not feasible in this work. This is because: firstly, their source codes were not available. Secondly, different test functions or datasets have been used during their experiments. In addition, since the emergence of CSO algorithm, many novel and powerful metaheuristic algorithms have been introduced. However, the literature lacks a comparative study between CSO algorithm and these new algorithms. Therefore, we conducted an experiment, in which the original CSO algorithm was compared against three new and robust algorithms, which were dragonfly algorithm (DA) [6], butterfly optimization algorithm (BOA) [7], and fitness dependent optimizer (FDO) [8]. For this, 23 traditional and 10 modern benchmark functions were used (see Figure 3), which illustrates the general framework for conducting the performance evaluation process. It is worth mentioning that for four test functions, BOA returned imaginary numbers and we set "N/A" for them.

6.1. Traditional Benchmark Functions. This group includes the unimodal and multimodal test functions. Unimodal test functions contain one single optimum while multimodal test functions contain multiple local optima and usually a single global optimum. F1 to F7 are unimodal test functions (Table 3), which are employed to experiment with the global search capability of the algorithms. Furthermore, F8 to F23 are multimodal test functions, which are employed to experiment with the local search capability of the algorithms. Refer to [107] for the detailed description of unimodal and multimodal functions.

6.2. Modern Benchmark Functions (CEC 2019). These set of benchmark functions, also called composite benchmark functions, are complex and difficult to solve. The CEC01 to CEC10 functions as shown in Table 3 are of these types, which are shifted, rotated, expanded, and combined versions of traditional benchmark functions. Refer to [108] for the detailed description of modern benchmark functions.

The comparison results for CSO and other algorithms are given in Table 3 in the form of mean and standard deviations. For each test function, the algorithms are executed for 30 independent runs. For each run, 30 search agents were searching over the course of 500 iterations. Parameter settings are set as defaults for all algorithms, and nothing was changed.

It can be noticed from Table 3 that the CSO algorithm is a competitive algorithm for the modern ones and provides very satisfactory results. In order to perceive the overall performance of the algorithms, they are ranked as shown in Table 4 according to different benchmark function groups. It

Table 3: Comparison results of CSO algorithm with modern metaheuristic algorithms.

	CSO		DA		BOA		FDO		$f_{\min}$
Functions	AV	STD	AV	STD	AV	STD	AV	STD	
F1	$3.50E-14$	$6.34E-14$	15.24805	23.78914	$1.01E-11$	$1.66E-12$	$2.13E-23$	$1.06E-22$	0
F2	$2.68E-08$	$2.61E-08$	1.458012	0.869819	$4.65E-09$	$4.63E-10$	0.047175	0.188922	0
F3	$7.17E-09$	$1.16E-08$	136.259	151.9406	$1.08E-11$	$1.71E-12$	$2.39E-06$	$1.28E-05$	0
F4	0.010352	0.007956	3.262584	2.112636	$5.25E-09$	$5.53E-10$	$4.93E-08$	$9.09E-08$	0
F5	8.587858	0.598892	374.9048	691.5889	8.935518	0.02146	21.58376	39.66721	0
F6	1.151759	0.431511	12.07847	17.97414	1.04685	0.346543	$7.15E-22$	$2.80E-21$	0
F7	0.026026	0.015039	0.035679	0.023538	0.001513	0.00056	0.612389	0.299315	0
F8	−2855.11	359.1697	−2814.14	432.944	NA	NA	−10502.1	15188.77	-418.9829×5
F9	24.01772	6.480946	26.53478	11.20011	28.6796	20.17813	7.940883	4.110302	0
F10	3.754226	1.680534	2.827344	1.042434	$3.00E-09$	$1.16E-09$	$7.76E-15$	$2.46E-15$	0
F11	0.355631	0.19145	0.680359	0.353454	$1.35E-13$	$6.27E-14$	0.175694	0.148586	0
F12	1.900773	1.379549	2.083215	1.436402	0.130733	0.084891	7.737715	4.714534	0
F13	1.160662	0.53832	1.072302	1.327413	0.451355	0.138253	4.724571	6.448214	0
F14	0.998004	$3.39E-07$	1.064272	0.252193	1.52699	0.841504	2.448453	1.766953	1
F15	0.001079	0.00117	0.005567	0.012211	0.000427	$9.87E-05$	0.001492	0.003609	0.00030
F16	−1.03162	$1.53E-05$	−1.03163	$4.76E-07$	NA	NA	−1.00442	0.149011	−1.0316
F17	0.304253	$1.81E-06$	0.304251	0	0.310807	0.004984	0.397887	$5.17E-15$	0.398
F18	3.003667	0.004338	3.000003	$1.22E-05$	3.126995	0.211554	3	$2.37E-07$	3
F19	−3.8625	0.00063	−3.86262	0.00037	NA	NA	−3.86015	0.003777	−3.86
F20	−3.30564	0.045254	−3.25226	0.069341	NA	NA	−3.06154	0.380813	−3.32
F21	−9.88163	0.90859	−7.28362	2.790655	−4.44409	0.383552	−4.19074	2.664305	−10.1532
F22	−10.2995	0.094999	−8.37454	2.726577	−4.1496	0.715469	−4.89633	3.085016	−10.4028
F23	−10.0356	1.375583	−6.40669	2.892797	−4.12367	0.859409	−4.03276	2.517357	−10.5363
CEC01	$1.58E+09$	$1.71E+09$	$3.8E+10$	$4.03E+10$	58930.69	11445.72	4585.278	20707.63	1
CEC02	19.70367	0.580672	83.73248	100.1326	18.91597	0.291311	4	$3.28E-09$	1
CEC03	13.70241	$2.35E-06$	13.70263	0.000673	13.70321	0.000617	13.7024	$1.68E-11$	1
CEC04	179.1984	55.37322	371.2471	420.2062	20941.5	7707.688	33.08378	16.81143	1
CEC05	2.671378	0.171923	2.571134	0.304055	6.176949	0.708134	2.13924	0.087218	1
CEC06	11.21251	0.708359	10.34469	1.335367	11.83069	0.771166	12.13326	0.610499	1
CEC07	365.2358	164.997	534.3862	240.0417	1043.895	215.3575	120.4858	13.82608	1
CEC08	5.499615	0.484645	5.86374	0.51577	6.337199	0.359203	6.102152	0.769938	1
CEC09	6.325862	1.295848	8.501541	16.90603	2270.616	811.4442	2	$2.00E-10$	1
CEC10	21.36829	0.06897	21.29284	0.176811	21.4936	0.079492	2.718282	$4.52E-16$	1

Table 4: Ranking of CSO algorithm compared to the modern metaheuristic algorithms.

Test functions	Ranking of CSO	Ranking of DA	Ranking of BOA	Ranking of FDO
F1	2	4	3	1
F2	2	4	1	3
F3	2	4	1	3
F4	3	4	1	2
F5	1	4	2	3
F6	3	4	2	1
F7	2	3	1	4
F8	2	3	4	1
F9	2	3	4	1
F10	4	3	2	1
F11	3	4	1	2
F12	2	3	1	4
F13	3	2	1	4
F14	1	2	3	4
F15	2	4	1	3
F16	1	2	4	3
F17	3	4	2	1
F18	3	2	4	1
F19	2	3	4	1
F20	1	2	4	3
F21	1	2	3	4

Table 4: Continued.

Test functions	Ranking of CSO	Ranking of DA	Ranking of BOA	Ranking of FDO
F22	1	2	4	3
F23	1	2	3	4
Cec01	3	4	2	1
Cec02	3	4	2	1
Cec03	2	3	4	1
Cec04	2	3	4	1
Cec05	3	2	4	1
Cec06	2	1	3	4
Cec07	2	3	4	1
Cec08	1	2	4	3
Cec09	2	3	4	1
Cec10	3	2	4	1
Total	70	97	91	72
Overall ranking	**2.121212**	2.939394	2.757576	2.181818
F1–F7 subtotal	15	27	**11**	17
F1–F7 ranking	2.142857	3.857143	**1.571429**	2.428571
F8–F23 subtotal	**32**	43	45	40
F8–F23 ranking	**2**	2.6875	2.8125	2.5
CEC01–CEC10 subtotal	23	27	35	**15**
CEC01–CEC10 ranking	2.3	2.7	3.5	**1.5**

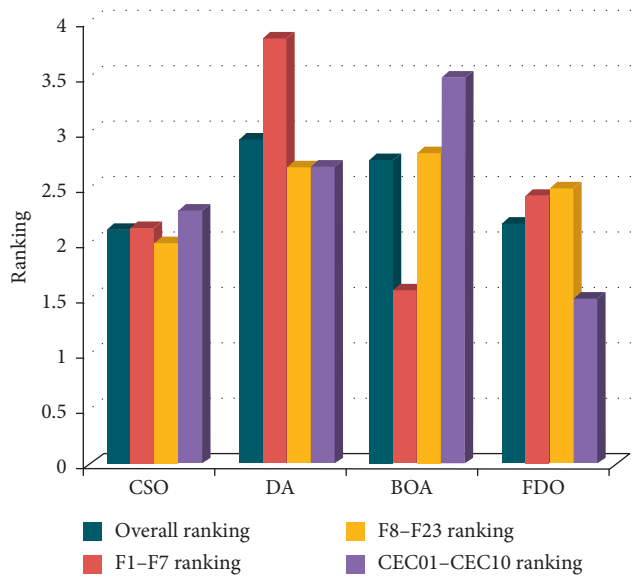


Figure 4: Ranking of algorithms according to different groups of test functions.

can be seen that CSO ranks first in the overall ranking and multimodal test functions. Additionally, it ranks second in unimodal and CEC test functions (see Figure 4). These results indicate the effectiveness and robustness of the CSO algorithm. That being said, these results need to be confirmed statistically. Table 5 presents the Wilcoxon matched-pairs signed-rank test for all test functions. In more than 85% of the results, P value is less than 0.05%, which proves that the results are significant and we can reject the null hypothesis that there is no difference between the means. It is worth mentioning that the performance of CSO can be further evaluated by comparing it against other new algorithms such as donkey and smuggler optimization algorithm [109], modified grey wolf optimizer [110], BSA and its variants [111], WOA and its variants [112], and other modified versions of DA [113].

7. Conclusion and Future Directions

Cat swarm optimization (CSO) is a metaheuristic optimization algorithm proposed originally by Chu et al. [5] in 2006. Henceforward, many modified versions and applications of it have been introduced. However, the literature lacks a detailed survey in this regard. Therefore, this paper firstly addressed this gap and presented a comprehensive review including its developments and applications.

CSO showed its ability in tackling different and complex problems in various areas. However, just like any other

Table 5: Wilcoxon matched-pairs signed-rank test.

Test functions	CSO vs. DA	CSO vs. BOA	CSO vs. FDO
F1	<0.0001	<0.0001	<0.0001
F2	<0.0001	<0.0001	0.0003
F3	<0.0001	<0.0001	0.2286
F4	<0.0001	<0.0001	<0.0001
F5	<0.0001	0.0879	0.0732
F6	0.0008	0.271	<0.0001
F7	0.077	<0.0001	<0.0001
F8	0.586	N/A	<0.0001
F9	0.2312	0.3818	<0.0001
F10	0.0105	<0.0001	<0.0001
F11	<0.0001	<0.0001	0.0002
F12	0.4	<0.0001	<0.0001
F13	<0.0001	<0.0001	0.0185
F14	0.4	<0.0001	0.0003
F15	0.0032	0.0004	0.9515
F16	<0.0001	N/A	<0.0001
F17	<0.0001	<0.0001	<0.0001
F18	<0.0001	<0.0001	<0.0001
F19	0.2109	N/A	0.6554
F20	0.0065	N/A	<0.0001
F21	0.0057	<0.0001	<0.0001
F22	0.1716	<0.0001	<0.0001
F23	<0.0001	<0.0001	<0.0001
cec01	<0.0001	<0.0001	<0.0001
cec02	0.001	<0.0001	<0.0001
cec03	0.0102	<0.0001	<0.0001
cec04	0.0034	<0.0001	<0.0001
cec05	0.1106	<0.0001	<0.0001
cec06	0.0039	0.0007	<0.0001
cec07	0.0002	<0.0001	<0.0001
cec08	0.0083	<0.0001	<0.0001
cec09	0.115	<0.0001	<0.0001
cec10	0.0475	<0.0001	<0.0001

metaheuristic algorithm, CSO algorithm possesses strengths and weaknesses. The tracing mode resembles the global search process while the seeking mode resembles the local search process. This algorithm enjoys a significant property for which these two modes are separated and independent. This enables researchers to easily modify or improve these modes and hence achieve a proper balance between exploration and exploitation phases. In addition, fast convergence is another strong point of this algorithm, which makes it a sensible choice for those applications that require quick responses. However, the algorithm has a high chance of falling into local optima, known as premature convergence, which can be considered as the main drawback of the algorithm.

Another concern was the fact that CSO algorithm was not given a chance to be compared against new algorithms since it has been mostly measured up against PSO and GA algorithms in the literature. To address this, a performance evaluation was conducted to compare CSO against three new and robust algorithms. For this, 23 traditional benchmark functions and 10 modern benchmark functions were used. The results showed the outperformance of CSO algorithm, in which it ranked first in general. The significance of these results was also confirmed by statistical methods. This indicates that CSO is still a competitive algorithm in the field.

In the future, the algorithm can be improved in many aspects; for example, different techniques can be adapted to the tracing mode in order to solve the premature convergence problem or transforming MR parameter is static in the original version of CSO. Transforming this parameter into a dynamic parameter might improve the overall performance of the algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK, 2010.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[3] J. Kennedy, "Particle swarm optimization," *Encyclopedia of Machine Learning*, pp. 760–766, Springer, Berlin, Germany, 2010.

[4] S. C. Chu and P. W. Tsai, "Computational intelligence based on the behavior of cats," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp. 163–173, 2007.

[5] S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimization," in *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pp. 854–858, Springer, Guilin, China, August 2006.

[6] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[7] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, vol. 23, no. 3, pp. 715–734, 2019.

[8] J. M. Abdullah and T. Ahmed, "Fitness dependent optimizer: inspired by the bee swarming reproductive process," *IEEE Access*, vol. 7, pp. 43473–43486, 2019.

[9] Y. Sharafi, M. A. Khanesar, and M. Teshnehlab, "Discrete binary cat swarm optimization algorithm," in *Proceedings of the 2013 3rd International Conference on Computer, Control & Communication (IC4)*, pp. 1–6, IEEE, Karachi, Pakistan, September 2013.

[10] P. M. Pradhan and G. Panda, "Solving multiobjective problems using cat swarm optimization," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2956–2964, 2012.

[11] P. W. Tsai, J. S. Pan, S. M. Chen, B. Y. Liao, and S. P. Hao, "Parallel cat swarm optimization," in *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*, vol. 6, pp. 3328–3333, IEEE, Kunming, China, July 2008.

[12] B. Santosa and M. K. Ningrum, "Cat swarm optimization for clustering," in *Proceedings of the 2009 International Conference of Soft Computing and Pattern Recognition*, pp. 54–59, IEEE, December 2009.

[13] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on the Taguchi

method," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6309–6319, 2012.

[14] M. Orouskhani, M. Mansouri, and M. Teshnehlab, "Average-inertia weighted cat swarm optimization," in *Proceedings of the International Conference in Swarm Intelligence*, pp. 321–328, Springer, Chongqing, China, June 2011.

[15] M. Orouskhani, Y. Orouskhani, M. Mansouri, and M. Teshnehlab, "A novel cat swarm optimization algorithm for unconstrained optimization problems," *International Journal of Information Technology and Computer Science*, vol. 5, no. 11, pp. 32–41, 2013.

[16] I. Hadi and M. Sabah, "Enhanced hybrid cat swarm optimization based on fitness approximation method for efficient motion estimation," *International Journal of Hybrid Information Technology*, vol. 7, no. 6, pp. 345–364, 2014.

[17] I. Hadi and M. Sabah, "Improvement cat swarm optimization for efficient motion estimation," *International Journal of Hybrid Information Technology*, vol. 8, no. 1, pp. 279–294, 2015.

[18] Y. Kumar and G. Sahoo, "An improved cat swarm optimization algorithm based on opposition-based learning and Cauchy operator for clustering," *Journal of Information Processing Systems*, vol. 13, no. 4, pp. 1000–1013, 2017.

[19] X. Nie, W. Wang, and H. Nie, "Chaos quantum-behaved cat swarm optimization algorithm and its application in the PV MPPT," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 1583847, 11 pages, 2017.

[20] N. Kanwar, N. Gupta, K. R. Niazi, and A. Swarnkar, "Improved cat swarm optimization for simultaneous allocation of DSTATCOM and DGs in distribution systems," *Journal of Renewable Energy*, vol. 2015, Article ID 189080, 10 pages, 2015.

[21] Y. Kumar and P. K. Singh, "Improved cat swarm optimization algorithm for solving global optimization problems and its application to clustering," *Applied Intelligence*, vol. 48, no. 9, pp. 2681–2697, 2017.

[22] P. W. Tsai, J. S. Pan, P. Shi, and B. Y. Liao, "A new framework for optimization based-on hybrid swarm intelligence," *Handbook of Swarm Intelligence*, pp. 421–449, Springer, Berlin, Germany, 2011.

[23] T. V. Vivek and G. R. Reddy, "A hybrid bioinspired algorithm for facial emotion recognition using CSO-GA-PSO-SVM," in *Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 472–477, IEEE, April 2015.

[24] V. I. Skoullis, I. X. Tassopoulos, and G. N. Beligiannis, "Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm," *Applied Soft Computing*, vol. 52, pp. 277–289, 2017.

[25] A. Sarswat, V. Jami, and R. M. Guddeti, "A novel two-step approach for overlapping community detection in social networks," *Social Network Analysis and Mining*, vol. 7, no. 1, p. 47, 2017.

[26] K.-C. Lin, Y.-H. Huang, J. C. Hung, and Y.-T. Lin, "Feature selection and parameter optimization of support vector machines based on modified cat swarm optimization," *International Journal of Distributed Sensor Networks*, vol. 11, no. 7, Article ID 365869, 2015.

[27] P. Mohapatra, S. Chakravarty, and P. K. Dash, "Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system," *Swarm and Evolutionary Computation*, vol. 28, pp. 144–160, 2016.

[28] L. Pappula and D. Ghosh, "Cat swarm optimization with normal mutation for fast convergence of multimodal functions," *Applied Soft Computing*, vol. 66, pp. 473–491, 2018.

[29] K.-C. Lin, K.-Y. Zhang, Y.-H. Huang, J. C. Hung, and N. Yen, "Feature selection based on an improved cat swarm optimization algorithm for big data classification," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 3210–3221, 2016.

[30] M. Zhao, "A novel compact cat swarm optimization based on differential method," *Enterprise Information Systems*, vol. 2018, pp. 1–25, 2018.

[31] H. Siqueira, E. Figueiredo, M. Macedo, C. J. Santana, C. J. Bastos-Filho, and A. A. Gokhale, "Boolean binary cat swarm optimization algorithm," in *Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–6, IEEE, Guadalajara, Mexico, November 2018.

[32] A. B. Pratiwi, "A hybrid cat swarm optimization-crow search algorithm for vehicle routing problem with time windows," in *Proceedings of the 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pp. 364–368, IEEE, Yogyakarta, Indonesia, November 2017.

[33] L. Kong, J.-S. Pan, P.-W. Tsai, S. Vaclav, and J.-H. Ho, "A balanced power consumption algorithm based on enhanced parallel cat swarm optimization for wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 11, no. 3, Article ID 729680, 2015.

[34] Y. Kumar and G. Sahoo, "An improved cat swarm optimization algorithm for clustering," *Computational Intelligence in Data Mining-Volume 1*, vol. 2015, pp. 187–197, Springer, New Delhi, India.

[35] A. Baldominos, Y. Saez, and P. Isasi, "Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning," *Complexity*, vol. 2019, Article ID 2952304, 16 pages, 2019.

[36] J. P. T. Yusiong, "Optimizing artificial neural networks using cat swarm optimization algorithm," *International Journal of Intelligent Systems and Applications*, vol. 5, no. 1, pp. 69–80, 2012.

[37] M. Orouskhani, M. Mansouri, Y. Orouskhani, and M. Teshnehlab, "A hybrid method of modified cat swarm optimization and gradient descent algorithm for training ANFIS," *International Journal of Computational Intelligence and Applications*, vol. 12, no. 2, Article ID 1350007, 2013.

[38] H. A. Al-Asadi, "New hybrid (SVMs-CSOA) architecture for classifying electrocardiograms signals," *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, vol. 4, no. 5, 2015.

[39] K. C. Lin, R. W. Lin, S. J. Chen, C. R. You, and J. L. Chai, "The classroom response system based on affective computing," in *Proceedings of the 2010 3rd IEEE International Conference on Ubi-Media Computing*, pp. 190–197, IEEE, Jinhua, China, July 2010.

[40] W. Wang and J. Wu, "Notice of retraction emotion recognition based on CSO&SVM in e-learning," in *Proceedings of the Seventh International Conference on Natural Computation*, vol. 1, pp. 566–570, IEEE, Shanghai, China, July 2011.

[41] S. J. Nanda, "A WNN-CSO model for accurate forecasting of chaotic and nonlinear time series," in *Proceedings of the 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pp. 1–5, IEEE, Kozhikode, India, February 2015.

[42] K. I. Mohamadeen, R. M. Sharkawy, and M. M. Salama, "Binary cat swarm optimization versus binary particle swarm optimization for transformer health index determination," in *Proceedings of the 2014 International Conference on Engineering and Technology (ICET)*, pp. 1–5, IEEE, Cairo, Egypt, April 2014.

[43] B. Wang, S. Xu, X. Yu, and P. Li, "Time series forecasting based on cloud process neural network," *International Journal of Computational Intelligence Systems*, vol. 8, no. 5, pp. 992–1003, 2015.

[44] S. Chittineni, V. Mounica, K. Abhilash, S. C. Satapathy, and P. P. Reddy, "A comparative study of CSO and PSO trained artificial neural network for stock market prediction," in *Proceedings of the International Conference on Computational Science, Engineering and Information Technology*, pp. 186–195, Springer, Tirunelveli, India, September 2011.

[45] M. Kumar, S. K. Mishra, and S. S. Sahu, "Cat swarm optimization based functional link artificial neural network filter for Gaussian noise removal from computed tomography images," *Applied Computational Intelligence and Soft Computing*, vol. 2016, Article ID 6304915, 6 pages, 2016.

[46] J. C. Hwang, J. C. Chen, J. S. Pan, and Y. C. Huang, "CSO and PSO to solve optimal contract capacity for high tension customers," in *Proceedings of the 2009 International Conference on Power Electronics and Drive Systems (PEDS)*, pp. 246–251, IEEE, Taipei, Taiwan, November 2009.

[47] J. C. Hwang, J. C. Chen, J. S. Pan, and Y. C. Huang, "CSO algorithm for economic dispatch decision of hybrid generation system," in *Proceedings of the 10th WSEAS International Conference on Applied Informatics and Communications, and 3rd WSEAS International Conference on Biomedical Electronics and Biomedical Informatics*, pp. 81–86, World Scientific and Engineering Academy and Society (WSEAS), Taipei, Taiwan, August 2010.

[48] I. Faraji, A. Z. Bargabadi, and Z. Hejrati, *Application of Binary Cat Swarm Optimization Algorithm for Unit Commitment problem*, in *The First National Conference on Meta-Heuristic Algorithms and Their Applications in Engineering and Science*, Fereydunkenar, Iran, August 2014.

[49] G. N. Kumar and M. S. Kalavathi, "Dynamic load models for voltage stability studies with a solution of UPFC using CSO," *International Journal of Computer Applications*, vol. 116, no. 10, pp. 27–32, 2015.

[50] K. Lenin and B. R. Reddy, "Reduction of active power loss by using adaptive cat swarm optimization," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 2, no. 3, pp. 111–118, 2014.

[51] T. Nireekshana, G. Kesava Rao, and S. Sivanaga Raju, "Available transfer capability enhancement with FACTS using cat swarm optimization," *Ain Shams Engineering Journal*, vol. 7, no. 1, pp. 159–167, 2016.

[52] H. Hosseinnia and M. Farsadi, "Utilization cat swarm optimization algorithm for selected harmonic elemination in current source inverter," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 6, no. 4, pp. 888–896, 2015.

[53] A. A. El-Ela, R. A. El-Sehiemy, A. M. Kinawy, and E. S. Ali, "Optimal placement and sizing of distributed generation units using different cat swarm optimization algorithms," in *Proceedings of the 2016 Eighteenth International Middle East Power Systems Conference (MEPCON)*, pp. 975–981, IEEE, Cairo, Egypt, December 2016.

[54] L. Guo, Z. Meng, Y. Sun, and L. Wang, "A modified cat swarm optimization based maximum power point tracking method for photovoltaic system under partially shaded condition," *Energy*, vol. 144, pp. 501–514, 2018.

[55] A. Srivastava and S. Maheswarapu, "Optimal PMU placement for complete power system observability using binary cat swarm optimization," in *Proceedings of the 2015 International Conference on Energy Economics and Environment (ICEEE)*, pp. 1–6, IEEE, Greater Noida, India, March 2015.

[56] L. Guo, Z. Meng, Y. Sun, and L. Wang, "Parameter identification and sensitivity analysis of solar cell models with cat swarm optimization algorithm," *Energy Conversion and Management*, vol. 108, pp. 520–528, 2016.

[57] I. Hadi and M. Sabah, "A novel block matching algorithm based on cat swarm optimization for efficient motion estimation," *International Journal of Digital Content Technology and Its Applications*, vol. 8, no. 6, p. 33, 2014.

[58] G. Kalaiselvan, A. Lavanya, and V. Natrajan, "Enhancing the performance of watermarking based on cat swarm optimization method," in *Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pp. 1081–1086, IEEE, Chennai, India, June 2011.

[59] A. Lavanya and V. Natarajan, "Analyzing the performance of watermarking based on swarm optimization methods," *Advances in Computing and Information Technology*, Springer, pp. 167–176, Berlin, Germany.

[60] I. Hadi and M. Sabah, "An enriched 3D trajectory generated equations for the most common path of multiple object tracking," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 6, pp. 53–76, 2015.

[61] L. Yan, X. Yan-Qiu, and W. Li-Hai, "Hyperspectral dimensionality reduction of forest types based on cat swarm algorithm," *The Open Automation and Control Systems Journal*, vol. 7, no. 1, 2015.

[62] W. Ansar and T. Bhattacharya, "A new gray image segmentation algorithm using cat swarm optimization," in *Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 1004–1008, IEEE, Chennai, India, April 2016.

[63] M. Karakoyun, N. A. Baykan, and M. Hacibeyoglu, "Multilevel thresholding for image segmentation with swarm optimization algorithms," *International Research Journal of Electronics & Computer Engineering*, vol. 3, no. 3, p. 1, 2017.

[64] Y. D. Zhang, Y. Sui, J. Sun, G. Zhao, and P. Qian, "Cat swarm optimization applied to alcohol use disorder identification," *Multimedia Tools and Applications*, vol. 77, no. 17, pp. 22875–22896, 2018.

[65] F. Yang, M. Ding, X. Zhang, W. Hou, and C. Zhong, "Non-rigid multi-modal medical image registration by combining L-BFGS-B with cat swarm optimization," *Information Sciences*, vol. 316, pp. 440–456, 2015.

[66] H. B. Çam, S. Akçakoca, A. Elbir, H. O. Ilhan, and N. Aydın, "The performance evaluation of the cat and particle swarm optimization techniques in the image enhancement," in *Proceedings of the 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, pp. 1–4, IEEE, Istanbul, Turkey, April 2018.

[67] G. Panda, P. M. Pradhan, and B. Majhi, "IIR system identification using cat swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12671–12683, 2011.

[68] G. Panda, P. M. Pradhan, and B. Majhi, "Direct and inverse modeling of plants using cat swarm optimization," *Handbook of Swarm Intelligence*, pp. 469–485, Springer, Berlin, Germany, 2011.

[69] R. Shojaee, H. R. Faragardi, S. Alaee, and N. Yazdani, "A new cat swarm optimization-based algorithm for reliability-

oriented task allocation in distributed systems," in *Proceedings of the 6th International Symposium on Telecommunications (IST)*, pp. 861–866, IEEE, Tehran, Iran, November 2012.

[70] R. Shojaee, H. R. Faragardi, and N. Yazdani, "From reliable distributed system toward reliable cloud by cat swarm optimization," *International Journal of Information and Communication*, vol. 5, no. 4, pp. 9–18, 2013.

[71] A. Bouzidi and M. E. Riffi, "Cat swarm optimization to solve job shop scheduling problem," in *Proceedings of the 2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, pp. 202–205, IEEE, Tetuan, Morocco, October 2014.

[72] A. Bouzidi and M. E. Riffi, "A comparative study of three population-based metaheuristics for solving the JSSP," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pp. 235–243, Springer, Cham, Switzerland, 2017.

[73] A. Bouzidi and M. E. Riffi, "Cat swarm optimization to solve flow shop scheduling problem," *Journal of Theoretical & Applied Information Technology*, vol. 72, no. 2, 2015.

[74] A. Bouzidi, M. E. Riffi, and M. Barkatou, "Cat swarm optimization for solving the open shop scheduling problem," *Journal of Industrial Engineering International*, vol. 15, no. 2, pp. 367–378, 2019.

[75] V. Dani, A. Sarswat, V. Swaroop, S. Domanal, and R. M. Guddeti, "Fast convergence to near optimal solution for job shop scheduling using cat swarm optimization," in *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence*, pp. 282–288, Springer, Kolkata, India, December 2017.

[76] A. K. Maurya and A. K. Tripathi, "Deadline-constrained algorithms for scheduling of bag-of-tasks and workflows in cloud computing environments," in *Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications*, pp. 6–10, ACM, Hong Kong, China, March 2018.

[77] A. Bouzidi and M. E. Riffi, "Discrete cat swarm optimization algorithm applied to combinatorial optimization problems," in *Proceedings of the 2014 5th Workshop on Codes, Cryptography and Communication Systems (WCCCS)*, pp. 30–34, IEEE, El Jadida, Morocco, November 2014.

[78] S. Bouzidi, M. E. Riffi, and A. Bouzidi, "Comparative analysis of three metaheuristics for solving the travelling salesman problem," *Transactions on Machine Learning and Artificial Intelligence*, vol. 5, no. 4, 2017.

[79] S. Bilgaiyan, S. Sagnika, and M. Das, "Workflow scheduling in cloud computing environment using cat swarm optimization," in *Proceedings of the 2014 IEEE International Advance Computing Conference (IACC)*, pp. 680–685, IEEE, Gurgaon, India, February 2014.

[80] B. Kumar, M. Kalra, and P. Singh, "Discrete binary cat swarm optimization for scheduling workflow applications in cloud systems," in *Proceedings of the 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, pp. 1–6, IEEE, Ghaziabad, India, February 2017.

[81] B. Crawford, R. Soto, N. Berríos et al., "A binary cat swarm optimization algorithm for the non-unicost set covering problem," *Mathematical Problems in Engineering*, vol. 2015, Article ID 578541, 8 pages, 2015.

[82] B. Crawford, R. Soto, N. Berrios, and E. Olguin, "Solving the set covering problem using the binary cat swarm optimization metaheuristic. World academy of science, engineering and technology," *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, vol. 10, no. 3, pp. 104–108, 2016.

[83] B. Crawford, R. Soto, N. Berrios, and E. Olguín, "Cat swarm optimization with different binarization methods for solving set covering problems," in *Artificial Intelligence Perspectives in Intelligent Systems*, pp. 511–524, Springer, Cham, Switzerland, 2016.

[84] S. Kotekar and S. S. Kamath, "Enhancing service discovery using cat swarm optimisation based web service clustering," *Perspectives in Science*, vol. 8, pp. 715–717, 2016.

[85] Y. Orouskhani, M. Jalili, and X. Yu, "Optimizing dynamical network structure for pinning control," *Scientific Reports*, vol. 6, no. 1, Article ID 24252, 2016.

[86] R. Soto, B. Crawford, A. Aste Toledo, C. Castro, F. Paredes, and R. Olivares, "Solving the manufacturing cell design problem through binary cat swarm optimization with dynamic mixture ratios," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 4787856, 16 pages, 2019.

[87] M. Bahrami, O. Bozorg-Haddad, and X. Chu, "Application of cat swarm optimization algorithm for optimal reservoir operation," *Journal of Irrigation and Drainage Engineering*, vol. 144, no. 1, Article ID 04017057, 2017.

[88] E. N. Kencana, N. Kiswanti, and K. Sari, "The application of cat swarm optimisation algorithm in classifying small loan performance," *Journal of Physics: Conference Series*, vol. 893, Article ID 012037, 2017.

[89] P. Majumder and T. I. Eldho, "A new groundwater management model by coupling analytic element method and reverse particle tracking with cat swarm optimization," *Water Resources Management*, vol. 30, no. 6, pp. 1953–1972, 2016.

[90] R. Rautray and R. C. Balabantaray, "Cat swarm optimization based evolutionary framework for multi document summarization," *Physica A: Statistical Mechanics and its Applications*, vol. 477, pp. 174–186, 2017.

[91] A. Thomas, P. Majumdar, T. I. Eldho, and A. K. Rastogi, "Simulation optimization model for aquifer parameter estimation using coupled meshfree point collocation method and cat swarm optimization," *Engineering Analysis with Boundary Elements*, vol. 91, pp. 60–72, 2018.

[92] A. A. Naem, L. M. El Bakrawy, and N. I. Ghali, "A hybrid cat optimization and K-median for solving community detection," *Asian Journal of Applied Sciences*, vol. 5, no. 5, 2017.

[93] P. M. Pradhan and G. Panda, "Pareto optimization of cognitive radio parameters using multiobjective evolutionary algorithms and fuzzy decision making," *Swarm and Evolutionary Computation*, vol. 7, pp. 7–20, 2012.

[94] P. M. Pradhan and G. Panda, "Cooperative spectrum sensing in cognitive radio network using multiobjective evolutionary algorithms and fuzzy decision making," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1022–1036, 2013.

[95] P. M. Pradhan and G. Panda, "Comparative performance analysis of evolutionary algorithm based parameter optimization in cognitive radio engine: a survey," *Ad Hoc Networks*, vol. 17, pp. 129–146, 2014.

[96] A. Tsiflikiotis and S. K. Goudos, "Optimal power allocation in wireless sensor networks using emerging nature-inspired algorithms," in *Proceedings of the 2016 5th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4, IEEE, Thessaloniki, Greece, May 2016.

[97] A. Pushpalatha and G. Kousalya, "A prolonged network life time and reliable data transmission aware optimal sink

relocation mechanism," *Cluster Computing*, vol. 22, no. S5, pp. 12049–12058, 2018.

[98] S. Alam, A. N. Malik, I. M. Qureshi, S. A. Ghauri, and M. Sarfraz, "Clustering-based channel allocation scheme for neighborhood area network in a cognitive radio based smart grid communication," *IEEE Access*, vol. 6, pp. 25773–25784, 2018.

[99] V. Snasel, L. Kong, P. Tsai, and J. S. Pan, "Sink node placement strategies based on cat swarm optimization algorithm," *Journal of Network Intelligence*, vol. 1, no. 2, pp. 52–60, 2016.

[100] P. W. Tsai, L. Kong, S. Vaclav, J. S. Pan, V. Istanda, and Z. Y. Hu, "Utilizing cat swarm optimization in allocating the sink node in the wireless sensor network environment," in *Proceedings of the 2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN)*, pp. 166–169, IEEE, Matsue, Japan, May 2016.

[101] G. Ram, D. Mandal, R. Kar, and S. P. Ghoshal, "Cat swarm optimization as applied to time-modulated concentric circular antenna array: analysis and comparison with other stochastic optimization methods," *IEEE Transactions on Antennas and Propagation*, vol. 63, no. 9, pp. 4180–4183, 2015.

[102] G. Ram, D. Mandal, S. P. Ghoshal, and R. Kar, "Optimal array factor radiation pattern synthesis for linear antenna array using cat swarm optimization: validation by an electromagnetic simulator," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 4, pp. 570–577, 2017.

[103] L. Pappula and D. Ghosh, "Synthesis of linear aperiodic array using cauchy mutated cat swarm optimization," *AEU-–International Journal of Electronics and Communications*, vol. 72, pp. 52–64, 2017.

[104] H. Chen, Q. Feng, X. Zhang, S. Wang, W. Zhou, and Y. Geng, "Well placement optimization using an analytical formula-based objective function and cat swarm optimization algorithm," *Journal of Petroleum Science and Engineering*, vol. 157, pp. 1067–1083, 2017.

[105] C. Hongwei, F. Qihong, Z. Xianmin, W. Sen, Z. Wensheng, and L. Fan, "Well placement optimization with cat swarm optimization algorithm under oil field development constraints," *Journal of Energy Resources Technology*, vol. 141, no. 1, Article ID 012902, 2019.

[106] R. Hassannejad, S. Tasoujian, and M. R. Alipour, "Breathing crack identification in beam-type structures using cat swarm optimization algorithm," *Modares Mechanical Engineering*, vol. 15, no. 12, pp. 17–24, 2016.

[107] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[108] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, *The 100-Digit Challenge: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization*, Nanyang Technological University, Singapore, 2018.

[109] A. S. Shamsaldin, T. A. Rashid, R. A. Agha, N. K. Al-Salihi, and M. Mohammadi, "Donkey and smuggler optimization algorithm: a collaborative working approach to path finding," *Journal of Computational Design and Engineering*, vol. 6, no. 4, pp. 562–583, 2019.

[110] T. A. Rashid, D. K. Abbas, and Y. K. Turel, "A multi hidden recurrent neural network with a modified grey wolf optimizer," *PLoS One*, vol. 14, no. 3, Article ID e0213237, 2019.

[111] B. A. Hassan and T. A. Rashid, "Operational framework for recent advances in backtracking search optimisation algorithm: a systematic review and performance evaluation," *Applied Mathematics and Computation*, vol. 370, Article ID 124919, 2019.

[112] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A systematic and meta-analysis survey of whale optimization algorithm," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 8718571, 25 pages, 2019.

[113] C. M. Rahman and T. A. Rashid, "Dragonfly algorithm and its applications in applied science survey," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 9293617, 21 pages, 2019.

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 2710561, 19 pages
https://doi.org/10.1155/2020/2710561

Research Article

A Dendritic Neuron Model with Adaptive Synapses Trained by Differential Evolution Algorithm

Zhe Wang[1], **Shangce Gao**[1], **Jiaxin Wang**[1], **Haichuan Yang**[1] and **Yuki Todo**[2]

[1]*Faculty of Engineering, University of Toyama, Toyama-Shi 930-8555, Japan*
[2]*School of Electrical and Computer Engineering, Kanazawa University, Kanazawa-Shi 920-1192, Japan*

Correspondence should be addressed to Shangce Gao; gaosc@eng.u-toyama.ac.jp

Received 22 July 2019; Revised 12 November 2019; Accepted 30 December 2019; Published 17 January 2020

Guest Editor: Eduardo Rodriguez-Tello

A dendritic neuron model with adaptive synapses (DMASs) based on differential evolution (DE) algorithm training is proposed. According to the signal transmission order, a DNM can be divided into four parts: the synaptic layer, dendritic layer, membrane layer, and somatic cell layer. It can be converted to a logic circuit that is easily implemented on hardware by removing useless synapses and dendrites after training. This logic circuit can be designed to solve complex nonlinear problems using only four basic logical devices: comparators, AND (conjunction), OR (disjunction), and NOT (negation). To obtain a faster and better solution, we adopt the most popular DE for DMAS training. We have chosen five classification datasets from the UCI Machine Learning Repository for an experiment. We analyze and discuss the experimental results in terms of the correct rate, convergence rate, ROC curve, and the cross-validation and then compare the results with a dendritic neuron model trained by the backpropagation algorithm (BP-DNM) and a neural network trained by the backpropagation algorithm (BPNN). The analysis results show that the DE-DMAS shows better performance in all aspects.

1. Introduction

The human brain consists of billions of neurons, and a single neuron cell is constituted by a cell body, an axon, a cell membrane, and a dendrite. Dendrites occupy more than 90 percent of the nerve cell organization and have a pivotal role in a human's learning process. The first artificial neuron was originally proposed by MuCulloch and Pitts in 1943 [1]. This model is an abstract and simplified model that was constructed according to the structure and working principle of a biological neuron membrane based on mathematics and algorithms called threshold logic.

The perceptron is a method for pattern recognition, which was first created by Rosenblatt in 1958 [2, 3]. It was the first artificial neural network model, laying the foundation for the neural network model. However, in Minsky Papert's analysis of Rosenbatt's single-layer perceptron from a mathematical perspective [4], the artificial neural network was criticized with an example of the XOR operation. The problem of how an intelligent system independently learns from an environment is not well solved, and the development of artificial neural networks (ANNs) has deteriorated. In the mid-1980s, scholars began to explore the inner logic of knowledge discovery in depth and discovered that inductive logic, especially incomplete induction logic, is a reasonable way to discover knowledge. Rumelhart et al. surprisingly discovered that the backpropagation error (BP) [5], which was invented by Werbos more than 10 years ago, can effectively solve the learning problems of hidden nodes in multilayer networks. It is not correct to accept Minsky's assertion that there may be no effective learning methods for multilayer networks. Since then, people's enthusiasm for ANN research has been rekindled.

However, researchers have argued that the use of McCulloch and Pitts's neuron is inadvisable because it disregards the dendritic structure in a real biology neuron. Koch and Segev [6, 7] proposed that the interaction between synapses and the action at the turning point of a branch can be approximated as logic operation. In recent years, several dendritic computing models considering the functions of dendrites in a neuron have been proposed in the literature. A dendritic morphological neural network (DMNN) which is based on the traditional morphological neural networks [8, 9] is proposed for solving classification problems [10] and 3D object recognition tasks [11]. A nonlinear dendritic neuron model equipped with binary

synapses [11] is demonstrated to be capable of learning temporal features of spike input patterns. Most recently, a dendritic neuron model (DNM) with nonlinear synapses has been proposed [12–14]. Different from DMNN, DNM only considers a single neuron rather than the network of a couple of neurons and has shown great information processing capacity [15–19]. The DNM uses a pruning technique derived from an interesting biological phenomenon: in the early stages of neuron triggering, the selective removal of unnecessary synapses and dendrites does not cause neuron cell death [20, 21]. The DNM subtly solves nonlinear problems that cannot be well handled by the Koch model [22, 23]. The DNM has four layers in its structure. The input signal is triggered in the synaptic layer and then sequentially received by the dendritic layer. The membrane layer collects the output from each branch of the dendritic layer and sends the results to the somatic cell layer. By the pruning function of the DNM, the precise dendritic structure and morphology are simplified. After training, all mature neurons are approximately replaced by a logic circuit that consists of comparators, AND gates, OR gates, and NOT gates.

In this study, we use a dendritic neuron model with adaptive synapses (DMASs). Recent advances in neurobiology have highlighted the importance of dendritic calculation. In 2019, Beaulieu-Laroche and his team [24] discovered that dendrites are always active when the cell body of a neuron is active, which implies that the dendritic synapse has a role in the neural computing process. Based on this biophysical hypothesis, we develop a synaptic adaptable neuron network without parameters that need to be artificially adjusted. All synaptic layer parameters will be trained by the learning algorithm. The effectiveness of adaptive synapses will be proved in Section 4.3. Thus, we have to consider additional aspects in the choice of learning algorithms.

With the emergence of various new optimization algorithms, how to train an ANN has been discussed [25]. BP is very effective as an ANN training algorithm and can solve some nonlinear problems [5]. However, BP has certain limitations; for example, falling into a local minimum is easy, the convergence speed is slow, and it is prone to overfitting [26]. Differential evolution (DE) has been employed to train DMAS in our research. DE was first proposed by Storn and Price in 1997 [27]. It is a biological-inspired, population-based global optimization algorithm. Due to its simple concept, easy implementation, fast convergence, and excellent robustness, it has been more extensively utilized than other mainstream evolutionary algorithms, such as the genetic algorithm (GA) [28, 29], the evolutionary strategy (ES) [30, 31], and particle swarm optimization (PSO) [32] in recent years. DE is similar to the GA and ES but differs from them because a unique differential evolution operator is referenced in DE. DE has proven to be superior to many algorithms [33–35]. Because of these characteristics and the advantages of DE, it has been recognized by scholars in the field of ANNs [36, 37]. Also, DE has been applied in dendrite morphological neural networks [38].

Five realistic classifications problems are considered in our research to validate our model (DE-DMAS) : iris, BUPA liver disorders, breast cancer, glass, and Australian credit approval (ACA). All the datasets are preprocessed as the binary-classification problem. These five datasets have undergone preprocessing, including outlier repair to fill in missing values. We compare the experimental results of DE-DMAS, BP-DNM, and BPNN for these five datasets. Experimental results show that DMAS outperforms its peers in terms of test accuracy, sensitivity, specificity, receiver operating characteristic (ROC), and cross-validation.

The remainder of this paper is organized as follows: Section 2 introduces the structure of our model (DMAS). The learning algorithm (DE) is explained in Section 3. The experimental method is designed in Section 4. Section 5 presents the analysis and discussion of the experimental results. The conclusions are provided in Section 6.

2. Dendritic Neuron Model with Adaptive Synapses

DMAS is applied in our research. The neuron model includes four layers: the adaptive synaptic layer, dendritic layer, membrane layer, and somatic cell layer. In this section, we detail the structure and principle of these four layers.

2.1. Adaptive Synaptic Layer. The synaptic layer receives and computes the input signal and sends the calculated results to the dendritic layer. Once the input signal exceeds the threshold, synapses will be fired. To simulate this process, we design a synaptic layer with a sigmoid faction as in the following equation:

$$Y_{i,m} = \frac{1}{1 + e^{-k\left(w_{im}x_i - q_{im}\right)}}, \tag{1}$$

where x_i is the input and $Y_{i,m}$ is the output of the m-th ($m = 1, 2, 3, \ldots, M$) branch of dendrites. The i in $i = 1, 2, 3, \ldots, I$ represents the number of inputs that have been normalized into [0, 1] from the dataset. I also represents the number of synapses on each dendrite. k is a tunable parameter which denotes the connection strength between presynaptic and postsynaptic neurons. To reduce the parameters that need to be adjusted in our study, k will be used as the training object. Due to the nature of the sigmoid function, this step has a minimal effect on the function. w_{im} and q_{im} are objects that also need to be trained by the learning algorithm; their values will be set initially within [−2, 2]. Because the synaptic layer works with these three training objects and inputs and no artificial adjustment parameters are needed, this synapse has an adaptive function [39]. The threshold θ_{im} is an important indicator for synapses and is calculated by the following equation:

$$\theta_{im} = \frac{q_{im}}{w_{im}}. \tag{2}$$

After the synapse has been activated by the sigmoid function, it can adopt one of the 4 different states according to different ranges of w_{im} and q_{im}. These states are described as the direct-connecting state (●), opposite-connecting state (▬), constant-1 state (①), and constant-0 state (⓪), as shown

in Figure 1. According to the change of the values of w_{im} and q_{im}, the four states are divided into the following six cases.

Case (a): direct-connecting state, when $w_{im} > q_{im} > 0$. In this state, if the value of the input x_{im} is greater than θ_{im}, the value of the output approximately equals 1; otherwise, it equals 0. For example, when $w_{im} = 1.0$ and $q_{im} = 0.5$, the function can be shown in Figure 2(a), where the X-axis represents the value of the input x and the Y-axis represents the value of the output. Since the range of input is [0, 1], we only need to pay attention to the area between the two dashed lines.

Case (b): opposite-connecting state, e.g., when $0 > q_{im} > w_{im}$. In this state, if the value of the input x_{im} is less than θ_{im}, the value of the output approximately equals 1; otherwise, it equals 0. A synapse in this state works as a logic NOT operation. For example, when $w_{im} = -1.0$ and $q_{im} = -0.5$, the function diagram is as shown in Figure 2(b).

Case (c1): constant-1 state when $w_{im} > 0 > q_{im}$. For example, when $w_{im} = 1.0$ and $q_{im} = -0.5$, the function diagrams are as shown in Figure 2(c1).

Case (c2): constant-1 state when $0 > w_{im} > q_{im}$. For example, when $w_{im} = -1.0$ and $q_{im} = -1.5$, the function diagrams are as shown in Figure 2(c2). In cases (c1) and (c2), regardless of the value of the input $x_{i,m}$, the output remains 1.

Case (d1): constant-0 state when $q_{im} > w_{im} > 0$. For example, when $w_{im} = 1.0$ and $q_{im} = 1.5$, the function diagrams are as shown in Figure 2(d1).

Case (d2): constant-0 when state $q_{im} > 0 > w_{im}$. For example, when $w_{im} = -1.0$ and $q_{im} = 0.5$, the function diagrams are as shown in Figure 2(d2). In cases (d1) and (d2), regardless of the value of the input x_{im}, the output remains 0.

2.2. Dendritic Layer. The outputs of the synaptic layer are calculated by the dendritic layer using multiplication. Because the sigmoid function is employed, the outputs are approximately equal to either 1 or 0. The outputs of the dendritic layer are also approximately equal to either 1 or 0. The dendrites work the same as a logic AND operation. The equation is

$$Z_m = \prod_{i=1}^{I} Y_{im}. \tag{3}$$

2.3. Membrane Layer. The membrane accepts the output of the dendritic layer as the input and linearly sums the values. The summation can be approximately simulated with logic OR operations. The equation is

$$V = \sum_{m=1}^{M} Z_m. \tag{4}$$

2.4. Somatic Cell Layer. The somatic cell layer will receive the signal from the membrane. The signal is calculated using the sigmoid function as follows:

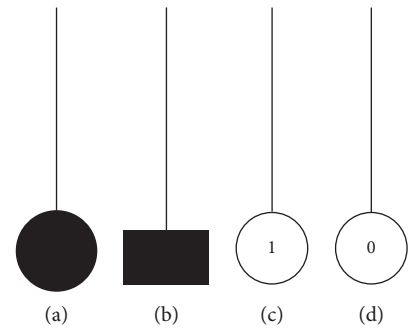


Figure 1: Four connecting states: (a) direct-connecting state; (b) opposite-connecting state; (c) constant-1 state; (d) constant-0 state.

$$O = \frac{1}{1 + e^{-k_{\text{soma}}(V - \theta_{\text{soma}})}}, \tag{5}$$

where k_{soma} and θ_{soma} are set to 10 and 0.5, which were suggested to be the most promising setting in our previous papers [40, 41].

2.5. Simplified Model. We pruned the synapses and dendrites to obtain our simplified model. The synapse receiving the input signal is activated and converted into the constant-1, constant-0, direct-connecting, or opposite-connecting state. The activated signal is transmitted to the dendrites. These signals are multiplied in the dendrites, enter the membrane, and are received by the soma. When a synapse is converted into the constant-1 state, we will remove this synapse since 1 multiplied by any number is equal to the number itself. When a synapse on a dendrite is converted to the constant-0 state, we will remove this dendrite since 0 multiplied by any number equals 0. An example is shown in Figure 3. In the upper left diagram, dendrite (2) has a constant-0 state synapse (c) and an opposite-connecting state synapse (d). Because synapse (c) is in the constant-0 state, dendrite (2) is removed, including the other synapses, as shown in the lower left diagram. We refer to this step as dendrite pruning. In the lower left diagram, a constant-1 state synapse (a) and a direct-connecting state synapse (b) exist on dendrite (1). Since synapse (a) is in the constant-1 state, this synapse is removed. We refer to this step as synapse pruning. The diagram on the right shows the simplified model after pruning, and only dendrite (1) and synapse (b) remain.

3. Learning Algorithm

As previously mentioned, we use the three parameters of the synaptic layer as training objects. This space is a vast search space. We use the DE algorithm as the learning algorithm of DMAS. Since DE is excellent in global optimization [42], it can find the optimal solution faster in the immense search space.

DE demonstrates a fixed number of vectors that are randomly initialized in the search space. The new vectors evolve over time to explore the minimum of the objective function. In the process of evolution, arithmetical operators are combined with the operators of mutation, hybridization, and selection. A randomly generated starting population will be evolved to an optimal solution. DE has numerous strategies

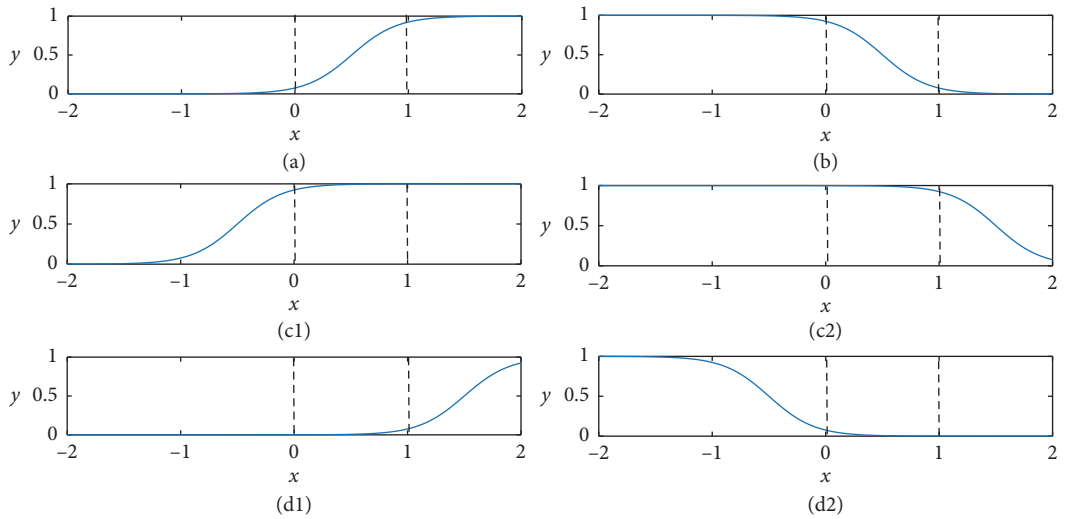


Figure 2: The synapse function figures for the four states: (a) direct-connecting state; (b) opposite-connecting state; (c1) constant-1 state; (c2) constant-1 state; (d1) constant-0 state; (d2) constant-0 state.

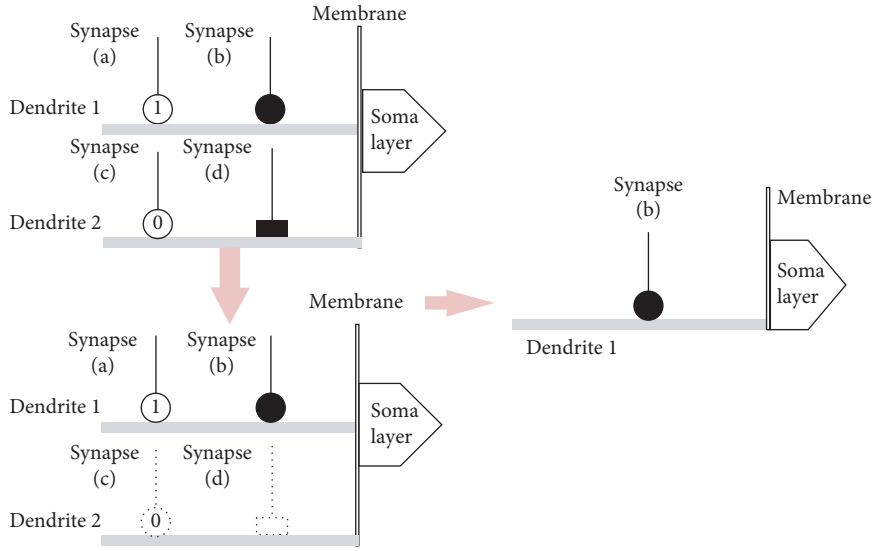


Figure 3: Simplified model.

[43], and we used DE/rand/1/bin in this study. Other strategies include DE/best/1/exp and DE/rand/2/exp, and the preliminary experimental results had suggested that they were slightly inferior than DE/rand/1/bin because it has the simplest structure [44, 45]. Next, we will explain how DE works.

Step 1. Parameter setup. Select the population size P and restrict the boundary. Confirm the cross-probability CR, the impact factor F [46, 47], and the termination criterion of the maximum number of generations (G).

Step 2. Initialization of the population. Set the generation $g = 0$. Initialize a population containing P individuals. The attributes of each individual include weights w, thresholds q and a k, described as D. The number of weights w and thresholds q equals the number of hidden layers (M) multiplied by the number of inputs (I) provided by the dataset. Thus, the population is considered to be a vector matrix of P rows and D ($D = 2 \times M \times I + 1$) columns. Each value of the weights w and thresholds q is initialized as a random real number in the range $[-2, 2]$. The value of the k is randomly initialized in the range $[1, 10]$. The following equation shows the content of the population:

$$\begin{cases} x_p \longrightarrow (x_1, x_2, x_3, \ldots, x_P), \\ x_p = \{w_{(1\times1)p}, q_{(1\times1)p}, \ldots, w_{(1\times M)p}, q_{(1\times M)p}, \ldots, \\ \quad w_{(I\times M)p}, q_{(I\times M)p}, k_p\}. \end{cases} \tag{6}$$

Step 3. Evaluation of the population. DE can be employed as a training algorithm for DE-DMAS. DE-DMAS can also be regarded as the evaluation function of DE. Therefore, the evaluation becomes a calculation of the mean square error (MSE), which will be formally

defined in equation (12). In the experiment, the MSE of DMAS is the fitness at each step. Each up-to-date generation will be evaluated after the next mutation, crossover, and selection operations. In our research, the maximum number of generations is set to 1000. Thus, the evaluation function will be run 1001 times.

Step 4. Mutation operation. The mutation operation produces a mutation operator ($v_{i,g}$). The process of production is shown in the following equation:

$$v_{i,g} = x_{r_1,g} + F\left(x_{r_2,g} - x_{r_3,g}\right), \tag{7}$$

where $x_{r_1,g}$, $x_{r_2,g}$, and $x_{r_3,g}$ are randomly chosen from the population of this generation. If all individuals are regarded as points in the search space, then the mutation operation can be interpreted as follows: $v_{i,g}$ is a new point after $x_{r_1,g}$ moves in the direction of $x_{r_3,g}$ to $x_{r_2,g}$ by F times the Euclidean distance between $x_{r_2,g}$ and $x_{r_3,g}$.

Step 5. Crossover operation. The crossover operation combines the mutation operator with the target individual, resulting in a new individual. DE involves two methods of crossover: binomial crossover and exponential crossover. Zaharie analyzed the performance of binomial crossover and exponential crossover [48] and suggested that exponential crossover is more affected by population size than binomial crossover. Binomial crossover is applied in our research. The following equation shows the crossover function:

$$u_{j,i,g} = \begin{cases} v_{j,i,g,} & \text{if } \text{rand}_j \le \text{CR or } j = j_{\text{rand}}, \\ x_{j,i,g}, & \text{otherwise.} \end{cases} \tag{8}$$

Generate the random number $\text{rand}_j \in [0, 1]$ for each dimension of each individual. If rand_j is less than CR in one dimension, then the target individual $x_{i,g}$ is replaced by the mutation operator $v_{i,g}$ in this dimension; otherwise, it remains the same as the target individual $x_{i,g}$. Before this step, to ensure that the target individual hybridizes in at least in one dimension, a random integer $j_{\text{rand}} \in \{1, 2, 3, \ldots, D\}$ is generated. When $j = j_{\text{rand}}$, the target individual must hybridize in the j-th dimension.

Step 6. Selection operation. DE employs the mutation operator and the crossover operator to generate a son population and applies a one-to-one selection to compare the son individuals with the corresponding parent individuals. The better individuals are saved to the next-generation population. In DE-DMAS, the one-to-one selection operation can be described as follows:

$$x_{i,g+1} = \begin{cases} u_{i,g}, & \text{if } \text{MSE}_{u_i} \le \text{MSE}_{x_i}, \\ x_{i,g}, & \text{otherwise.} \end{cases} \tag{9}$$

Since DE employs a one-to-one selection method, the algorithm can ensure that the elitism will not be lost during the evolution process. In addition, one-to-one selection operation has a better ability to maintain population diversity than sequencing or competitive bidding selection [44]. The following Algorithm 1 summarizes the above steps, where two functions rnd_int and rnd_real return random integer and real numbers in the specified range, respectively.

4. Experimental Design

To achieve the best performance of the proposed method, it is first necessary to confirm the parameters. DE-DMAS has six main parameters. The parameters can be divided into fixed parameters and adjustable parameters. The best adjustable parameters are determined using the Taguchi method for each dataset [49], which is detailed in Section 4.2. In Section 4.3, we will prove the adaptability of synapses as mentioned above. Finally, DE-DMAS is compared with BP-DNM and BPNN, which are introduced in Sections 4.4 and Section 4.5, respectively. The five datasets adopted in our research are introduced in the following sections.

4.1. Dataset. The five datasets, which are obtained from UCI, are extensively applied in artificial intelligence research. The datasets have been standardized by maximum minimization to [0, 1] in our research. Their detailed introduction and summary are provided in Table 1.

The iris data were provided by Fisher in July 1988 [50–52]. The data have three classes: Iris Setosa, Iris Versicolour, and Iris Virginica. Each class has 50 instances. We chose one of the instances as the experimental standard; thus, the data are divided into two categories. The selected 50 instances are divided into one class, and the other 100 instances are divided into another class. Each instance has four attributes: sepal length, sepal width, petal length, and petal width. In our research, we use the class Iris Versicolour as the output, and thus, it becomes a binary classification. Because of the limitations of the single neuron model, the DE-DMAS can only solve binary classification problems. So we apply the iris dataset as a binary classification problem.

The liver disorders dataset was provided by Richard S. Forsyth in "None known other than what is shown in the PC/BEAGLE User's Guide." It has been applied in [40, 53]. The dataset has 345 instances. Each instance has six attributes, which include five kinds of blood tests and average daily alcohol consumption. The liver dataset has two classifications: 164 healthy disorders and 181 unhealthy disorders.

The breast cancer data were provided by Dr. William Wolberg in July 1992 [54, 55]. It has been applied in [41]. The 699 instances of these data consist of 458 benign instances and 241 malignant instances. Breast cancer data can be divided into two classes. The breast cancer data include 9 attributes, such as clump thickness, uniformity of cell size and shape, and marginal adhesion.

The glass identification database was provided by B. Herman in September 1987. It has been applied in [56]. The glass data include 163 window glass instances and 51 nonwindow glass instances for a total of 214 instances. The attributes of the glass data include various element contents

```
Initial population;
Calculate the fitness of the first generation;
for g = 1 to G do
    randomly uniformly select r1 ≠ r2 ≠ r3 ≠ i;
    j_rand = rnd_int(1, D);
    for j = 1 to D do
        if rnd_real_j[0, 1) < CR or j == j_rand then
            U_{i,g}(j) = X_{r1,g}(j) + F × (X_{r2,g}(j) − X_{r3,g}(j));
        else
            U_{i,g}(j) = X_{i,g}(j);
    for i = 1 to P do
        Calculate the fitness of the new individual U_i;
        if U_{i,g} performs better than X_{i,g} then
            X_{i,g+1} = U_{i,g};
```

Algorithm 1: Differential evolution algorithm for DNM.

(Na, Mg, Al, Si, K, Ca, Ba, and Fe) and the refractive index (RI). The instances can be classified by these 9 attributes.

The ACA data indicate whether the applicants are creditworthy. It has been applied in [57, 58]. The credit history of the applicants classifies the data into two classes. These data provide information about 690 applicants. The applicants include 307 people who are creditworthy and 383 people who have no credit. The information that can be considered as the attributes of the ACA dataset consist of 8 categorical records and 6 numerical records.

4.2. Optimal Parameter Settings. Three parameters, F, CR, and NP, are mentioned in the DE learning algorithm. The number of hidden layers is an important parameter in DE-DMAS, namely, M. For different datasets, M should be suitably determined. The parameter ranges in DE-DMAS are shown in Table 2.

Typically, we need to experiment with all combinations of parameters to obtain the optimal parameters. However, four parameters exist, and each parameter has three choices. Thus, we should perform 81 (3^4) different experiments, which will be time-consuming. To ensure the credibility of the experimental results, we should repeat every different experiment 30 times. Because this approach is time-consuming, we should reduce the number of different experiments. Taguchi's method is a kind of method to efficiently obtain the optimal parameters [59, 60]. This method is primarily employed using orthogonal arrays. According to the previously mentioned parameter ranges, four parameter trials containing three datasets are available. Thus, the $L_9(3^4)$ orthogonal array has been applied in the optimal parameter experiments of the five datasets. The instances of each dataset have been divided into 70% for training and 30% for testing. The orthogonal experiments of each dataset have been repeated 30 times. The epoch of each orthogonal experiment is set to 1000. The orthogonal experimental result of the iris dataset is shown in Table 3. The result of the liver dataset is shown in Table 4. The result of the glass dataset is shown in Table 5. The result of the cancer dataset is shown in Table 6. The result of the ACA dataset is shown in Table 7. The last column displays the average correct rate of 30 test experiments. We obtain the most optimal parameters by a comprehensive analysis of the mean and variance. The bold font indicates the optimal combination of parameters. The optimal parameters for all datasets are shown in Table 8.

Table 1: Dataset introduction.

Dataset name	No. of instances	No. of input attributes	No. of class 1 instances	No. of class 2 instances
Iris	150	4	50	100
Liver	345	6	164	181
Cancer	699	9	458	241
Glass	214	9	51	163
ACA	690	14	307	383

Table 2: Parameter ranges in DE-DMAS.

Dataset	NP	CR	F	M
Iris	10, 30, 60	0.3, 0.6, 0.9	0.3, 0.6, 0.9	4, 8, 12
Liver	10, 30, 60	0.3, 0.6, 0.9	0.3, 0.6, 0.9	6, 12, 18
Cancer	10, 30, 60	0.3, 0.6, 0.9	0.3, 0.6, 0.9	9, 18, 27
Glass	10, 30, 60	0.3, 0.6, 0.9	0.3, 0.6, 0.9	9, 18, 27
ACA	10, 30, 60	0.3, 0.6, 0.9	0.3, 0.6, 0.9	16, 32, 48

4.3. Adaptability of Synapses. In order to demonstrate the adaptability of the synaptic layer in DE-DMAS, we carried out a confront analysis. We removed the hyperparameter k from the population of DE and set it to 1, 5, and 10, respectively. The five datasets were randomly divided into two parts: 70% for training and 30% for testing. The parameters except k were set as the same as these in Table 8. Then, we did 30 independent experiments for them. We recorded all test accuracy results and compared the results in terms of mean and standard deviation, as shown in Table 9. From it, we found that the adaptive k which was learned by DE generally performed better than these fixed values. Additionally, the Friedman test [61] gave the statistical analysis results for the accuracies. In this case, the lower the value of the Friedman test, the better the performance. The result of the Friedman test is shown in Table 10. Based on the above results, it is evident that the adaptive synapse is beneficial for DNM.

4.4. Comparison with BPNN. In this section, we compared DE-DMAS with the most popular model BPNN. To make the comparison relatively fair, the number of adjusted weights and thresholds (D_{BPNN} and $D_{\mathrm{DE-OMAS}}$ shown in equations (10) and (11), respectively) in both models should be arranged nearly the same because these numbers generally determine the size of the model and the computational complexity although the two models have different architectures:

Table 3: Orthogonal array for parameters of the iris dataset.

Experimental runs	Parameters (levels)				Accuracy (%)
	Branch (3)	F (3)	CR (3)	NP (3)	
1	4	0.3	0.3	10	92.51 ± 4.61
2	8	0.3	0.6	30	94.49 ± 5.46
3	12	0.3	0.9	60	94.96 ± 2.47
4	12	0.6	0.3	30	94.20 ± 4.61
5	4	0.6	0.6	10	**96.74 ± 2.78**
6	8	0.6	0.9	60	93.62 ± 5.32
7	8	0.9	0.3	60	94.20 ± 4.47
8	12	0.9	0.6	10	95.50 ± 4.90
9	4	0.9	0.9	30	94.00 ± 2.74

Table 4: Orthogonal array for parameters of the liver dataset.

Experimental runs	Parameters (levels)				Accuracy (%)
	Branch (3)	F (3)	CR (3)	NP (3)	
1	6	0.3	0.3	10	70.60 ± 4.81
2	12	0.3	0.6	30	**73.59 ± 6.66**
3	18	0.3	0.9	60	72.17 ± 3.86
4	18	0.6	0.3	30	66.92 ± 6.80
5	6	0.6	0.6	10	71.85 ± 4.75
6	12	0.6	0.9	60	68.33 ± 6.91
7	12	0.9	0.3	60	66.28 ± 7.36
8	18	0.9	0.6	10	68.58 ± 7.74
9	6	0.9	0.9	30	69.61 ± 4.88

Table 5: Orthogonal array for parameters of the glass dataset.

Experimental runs	Parameters (levels)				Accuracy (%)
	Branch (3)	F (3)	CR (3)	NP (3)	
1	9	0.3	0.3	10	93.75 ± 2.65
2	18	0.3	0.6	30	94.37 ± 4.13
3	27	0.3	0.9	60	**94.42 ± 2.51**
4	27	0.6	0.3	30	94.06 ± 4.44
5	9	0.6	0.6	10	93.12 ± 4.29
6	18	0.6	0.9	60	93.02 ± 4.07
7	18	0.9	0.3	60	93.54 ± 3.66
8	27	0.9	0.6	10	92.81 ± 4.03
9	9	0.9	0.9	30	94.01 ± 2.66

Table 6: Orthogonal array for parameters of the cancer dataset.

Experimental runs	Parameters (levels)				Accuracy (%)
	Branch (3)	F (3)	CR (3)	NP (3)	
1	9	0.3	0.3	10	**96.12 ± 1.17**
2	18	0.3	0.6	30	96.12 ± 1.65
3	27	0.3	0.9	60	95.80 ± 1.26
4	27	0.6	0.3	30	95.39 ± 1.97
5	9	0.6	0.6	10	96.09 ± 1.64
6	18	0.6	0.9	60	96.19 ± 1.90
7	18	0.9	0.3	60	96.20 ± 2.00
8	27	0.9	0.6	10	95.81 ± 1.55
9	9	0.9	0.9	30	95.84 ± 1.31

Table 7: Orthogonal array for parameters of the ACA dataset.

Experimental runs	Parameters (levels)				Accuracy (%)
	Branch (3)	F (3)	CR (3)	NP (3)	
1	16	0.3	0.3	10	84.41 ± 5.00
2	32	0.3	0.6	30	85.81 ± 2.41
3	48	0.3	0.9	60	85.28 ± 2.28
4	48	0.6	0.3	30	85.18 ± 1.67
5	16	0.6	0.6	10	85.58 ± 2.14
6	32	0.6	0.9	60	85.70 ± 1.64
7	32	0.9	0.3	60	84.89 ± 2.26
8	48	0.9	0.6	10	84.41 ± 5.73
9	16	0.9	0.9	30	**86.18 ± 1.98**

Table 8: Parameters for DE-DMAS

Dataset	NP	CR	F	M
Iris	0.6	0.6	10	4
Liver	0.3	0.6	30	12
Cancer	0.3	0.3	10	9
Glass	0.3	0.9	60	27
ACA	0.9	0.9	30	16

$$D_{\mathrm{BPNN}} = (\text{input} \times \text{hidden}) + (\text{hidden} \times \text{output}) + \text{hidden bias} + \text{output bias}, \tag{10}$$

$$D_{\mathrm{DE-DMAS}} = (\text{input} \times \text{hidden}) + 1. \tag{11}$$

However, the larger the number of weights is, the more the occupied computing resources are. In our research, to demonstrate the excellent performance of DE-DMAS for five datasets, the structure of DE-DMAS should be set smaller than that of BPNN. Because the input and output of each dataset are fixed, they are given the same number of weights by adjusting their number of hidden layers. In the previous section, we have configured this parameter (the number of hidden layers) for DE-DMAS. We configure the BPNN with the number of hidden layers according to the above principles. The structures of BPNN and DE-DMAS for the five datasets are shown in Table 11. The learning rate is set to be 0.1 according to the experience.

Table 9: Demonstrate on the adaptability of synapse in terms of test accuracy.

Dataset	$k=1$	$k=5$	$k=10$	Adaptive
Iris	80.81 ± 6.66	94.15 ± 2.83	95.11 ± 2.50	96.74 ± 2.78
Liver	69.39 ± 3.57	69.81 ± 4.12	69.39 ± 5.56	73.59 ± 6.66
Cancer	92.76 ± 3.35	92.97 ± 3.38	93.59 ± 2.11	96.12 ± 1.17
Glass	91.04 ± 2.84	93.33 ± 3.23	85.48 ± 2.02	94.42 ± 2.51
ACA	85.44 ± 2.99	85.56 ± 1.70	85.86 ± 2.53	86.18 ± 1.98

Table 10: Demonstrate on the adaptability of synapse by the Friedman test.

Dataset	$k=1$	$k=5$	$k=10$	Adaptive
Iris	4	2.33	2	1.67
Liver	2.77	2.73	2.55	1.95
Cancer	3.05	2.95	2.73	1.27
Glass	2.55	1.98	3.93	1.53
ACA	2.48	2.73	2.52	2.27

4.5. Comparison with BP-DNM. In order to compare BP-DNM and DE-DMAS fairly, the three common parameters of k_{soma}, θ_{soma}, and the number of neurons in hidden layers (M) are set to be the same. According to the experience, the learning rate is set to 0.01, and the value of k is set to 3. BP-DNM is also a single neuron model with synaptic nonlinearities. It has been proven to have outstanding performance in the liver [40], cancer [41], and ACA [58]. We will show the performs of BP-DNM when it has the same structure as DE-DMAS. We will use multiple objective methods to demonstrate the performances of DE-DMAS and BP-DNM on the five datasets and make a discussion.

5. Experimental Result Analysis

The comparison experiment of DE-DMAS vs BPNN and DE-DMAS vs BP-DNM is set up as follows: (1) the instances of the five datasets are divided into 30% for testing and 70% for training randomly, (2) the number of iterations is set to 1000, and (3) all experiments are run using Matlab 2018a.

5.1. Convergence Comparison. We use the value of the mean square error (MSE) to represent the degree of convergence. The smaller the value is, the better the convergence is. We calculate the value of MSE after each iteration in the DE-DMAS, BP-DNM, and BPNN training process and record it. We employ the following equation to calculate the value of the MSE:

$$\text{MSE} = \frac{1}{2N}\sum_{i=1}^{N}\left(O_i - T_i\right)^2, \tag{12}$$

where N represents the number of training instances and O_i and T_i represent the output and the teacher signal of the i-th training instance, respectively.

We perform 30 training sessions for DE-DMAS, BP-DNM, and BPNN. We randomly select 70% of the instances as the input for each training. We draw two graphs to analyze the convergence effect of DE-DMAS, BP-DNM, and BPNN for the five training datasets. In the first figure, the ordinate represents the mean value of the MSE for 30 training sessions, and the abscissa represents the number of iterations. A total of 1000 MSE values are recorded from the start of initialization for the DE-DMAS, BP-DNM, and BPNN. We can evaluate the speed of convergence by the degree of the curve drop. In Figure 4, we note that curve of DE-DMAS is falling faster than the comparators. These figures show that DE-DMAS has an advantage in convergence speed.

Table 11: Structures of DE-DMAS and BPNN for the five datasets.

Dataset	Method	No. of inputs	No. of branches	No. of outputs	No. of adjusted weights
Iris	DE-DMAS	4	4	1	33
	BPNN	4	28	1	169
Liver	DE-DMAS	6	12	1	145
	BPNN	6	18	1	145
Cancer	DE-DMAS	9	9	1	163
	BPNN	9	18	1	199
Glass	DE-DMAS	9	27	1	487
	BPNN	27	45	1	496
ACA	DE-DMAS	14	16	1	449
	BPNN	14	30	1	481

We record the value of MSE in the final iteration for 30 times in the training sessions. We use a box-and-whisker plot [62] to represent the value of the MSE, as shown in Figure 5. In this figure, the ordinate represents the value of the MSE. The horizontal line from the top to the bottom of each box represents the maximum, 3/4 median, median, 1/4 median, and minimum. The 1/4 median, median, and 3/4 median represent the value of MSE at 25%, 50%, and 75%, respectively, after sorting. The +sign represents an outlier, which is a value that exceeds twice the standard deviation. The lines corresponding to the maximum, 3/4 median line, median, 1/4 median, and minimum for DE-DMAS are below those of BP-DNM and BPNN for the five datasets. Many outliers exist in the boxes of BPNN and BP-DNM. The outliers above the maximum represent falling into a local minimum during training. On the contrary, there is no outlier above the maximum in the boxes of DE-DMAS. The results show that the convergence effect of DE-DMAS is better than that of BP-DNM and BPNN.

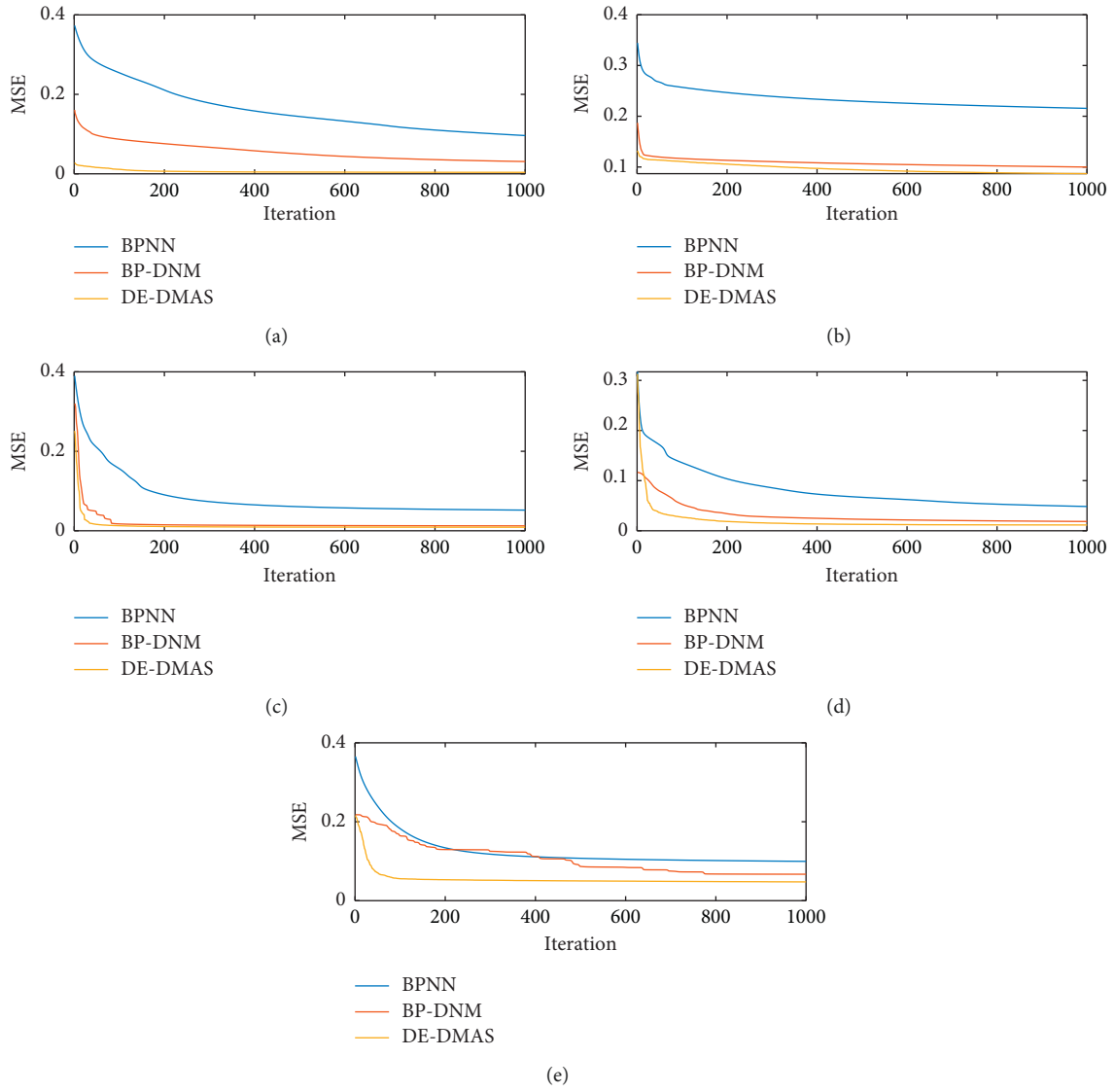


Figure 4: Convergence graphs for the five datasets: (a) iris; (b) liver; (c) cancer; (d) glass; (e) ACA.

5.2. Accuracy Comparison. We compare DE-DMAS, BP-DNM, and BPNN in terms of the test accuracy, sensitivity, specificity, and receiver operating characteristic (ROC) curve [63], which is a method for objectively analyzing the performance of classifiers. To draw the ROC curve, we collected the output (O) of the five datasets tested by DE-DMAS, BP-DNM, and BPNN. T represents the corresponding teacher signals. We convert the output O from a real number to an integer of 0 or 1. For a two-category problem, the instances are divided into positive and negative classes. The actual classification has four situations:

(1) If an instance is in the positive class and is predicted to be in the positive class, then it is a true classification (true positive (TP))

(2) If an instance is in the positive class but is predicted to be in the negative class, then it is a false-negative classification (false negative (FN))

(3) If an instance is in the negative class but is predicted to be in the positive class, then it is a false-positive classification (false positive (FP))

(4) If an instance is in the negative class and is predicted to be in the negative class, then it is a true-negative classification (true negative (TN))

The true-positive rate (TPR), which represents the proportion of actual positive instances in the positive class predicted by the classifier to all positive instances, equals the sensitivity. The false-positive rate (FPR), which represents the proportion of actual negative instances in the positive class predicted by the classifier to all negative instances, equals 1-specificity. The ROC curve is drawn with the FPR (1 – specificity) as the x-axis and the TPR (sensitivity) as the y-axis. The AUC is the area under the ROC curve. The value of AUC is between 0.0 and 1.0 since the ROC curve is drawn in an square area. The greater the values of the sensitivity, specificity, and AUC are, the better the performance of the

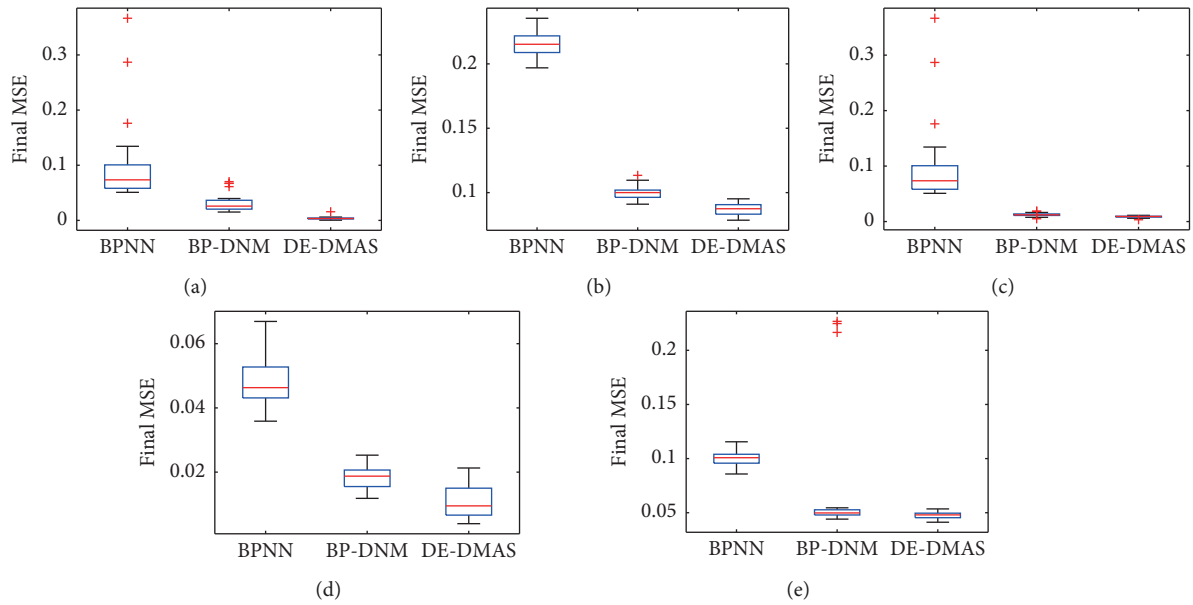


Figure 5: Box-and-whisker plots for the final MSE: (a) iris; (b) liver; (c) cancer; (d) glass; (e) ACA.

classifier is. These terms are defined in Table 12. We calculate the accuracy, sensitivity, specificity, and AUC based on these terms using the following equations:

$$\begin{aligned}
\text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}},\\
\text{sensitivity} &= \frac{\text{TP}}{(\text{TP} + \text{FN})},\\
\text{specificity} &= \frac{\text{TN}}{(\text{TN} + \text{FP})},\\
\text{AUC}(\%) &= \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \times 100.
\end{aligned} \tag{13}$$

We plot the ROC curves of the five datasets to compare DE-DMAS with BP-DNM and BPNN, as shown in Figure 6. The DE-DMAS curves are above the BP-DNM and BPNN curves. The sensitivity, specificity, and AUC, which can be determined from the numerical values, are shown in Table 13. The test accuracy is the average of 30 experiments, which we represent by the mean and variance in Table 13. DE-DMAS exhibits higher values than BP-DNM and BPNN for these four assessment levels. All test results prove the superiority of DE-DMAS.

5.3. Cross-Validation. In order to facilitate the comparison performance, four different experimental train-to-test ratios were adopted and the four multifold cross-validation ($K \times$ CV) methods include tenfold CV (90–10%, ×10), fivefold CV (80–20%, ×5), fourfold CV (75–25%, ×4), and twofold CV(50–50%, ×2). Here, the train-to-test ratio represents the ratio between sample size for training and testing. With $K \times$ CV ($K = 2, 4, 5, 10$), the whole dataset is randomly divided into K and mutually exclusive subsets with approximately equal sample size. In $K \times$ CV, the method is utilized on the training subsets, and the testing error is measured on the testing subset. The procedure is repeated for a total of K trials, each time using a different subset for testing. The performance of the model is evaluated by the mean of the squared error through testing over all trails of the experiment. Compared with the single-fold validation method, $K \times$ CV has an advantage of minimizing the correlation deviation of random sampling of training samples, but its disadvantage lies in that it may need too much computation since the model has to be trained K times. We select four kinds of BPNN with different learning rates and number (No.) of branch as Table 14, namely, BPNN1, BPNN2, BPNN3, and BPNN4. BP-DNM and DE-DMAS used the above four types of training-to-test ratios. Five datasets were applied to each type of training-to-test ratios for each model, and 30 independent experiments were performed. Finally, we compared the mean and standard deviation of their test accuracy results. Table 15 shows the cross-validation results of the iris dataset. Table 16 shows the cross-validation results of the liver dataset. Table 17 shows the cross-validation results of the cancer dataset. Table 18 shows the cross-validation results of the glass dataset. Table 19 shows the cross-validation results of the ACA dataset. Bold fonts in these tables indicate the top two according to the standard deviation and the mean. DE-DMAS only in the CV5 of the cancer dataset is not bold font. So we can conclude that DE-DMAS is excellent.

Table 12: Description of terms.

Teacher signal	Real output		Row total
	Positive (1)	Negative (0)	
Positive (1)	TP	FN	TP + FN
Negative (0)	FP	TN	FP + TN
Column total	TP + F	FN + TN	N = TP + TN + FP + FN

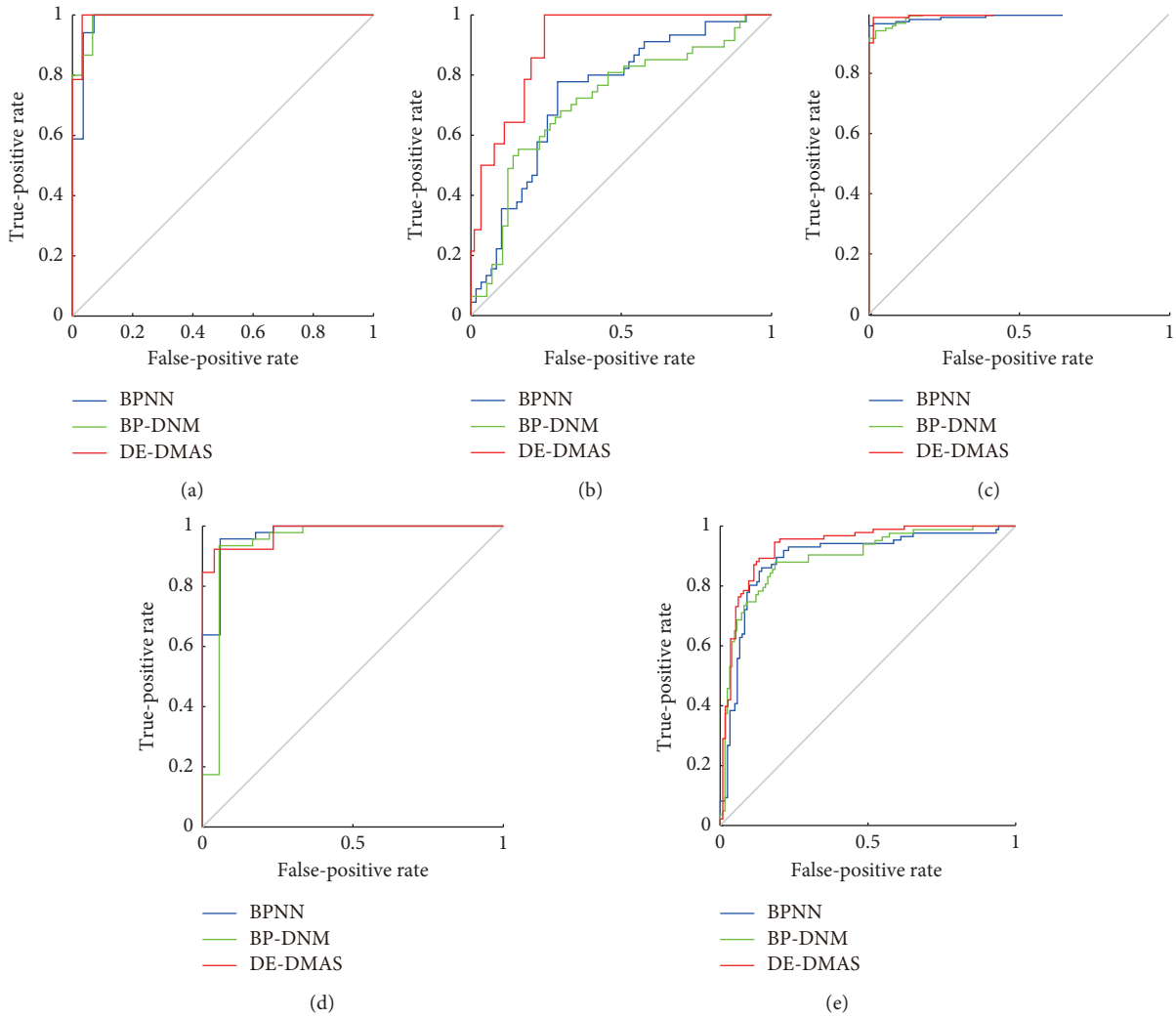


Figure 6: ROC analysis for the five datasets: (a) iris ROC; (b) liver ROC; (c) cancer ROC; (d) glass ROC; (e) ACA ROC.

Table 13: Rate results for the five datasets.

Dataset	Method	Test accuracy	Sensitivity	Specificity	AUC
Iris	DE-DMAS	96.74 ± 2.78	100	96.88	98.44
	BP-DNM	91.78 ± 5.87	91.16	86.67	95.65
	BPNN	85.93 ± 10.89	90.00	91.43	90.71
Liver	DE-DMAS	73.59 ± 6.66	66.67	83.87	75.27
	BP-DNM	68.62 ± 5.24	52.50	.79.69	66.09
	BPNN	59.94 ± 6.33	57.89	72.73	65.31
Cancer	DE-DMAS	96.12 ± 1.17	96.45	98.55	97.50
	BP-DNM	96.33 ± 1.43	97.04	94.67	95.85
	BPNN	93.76 ± 11.06	95.89	96.88	96.38
Glass	DE-DMAS	94.42 ± 2.51	98.57	92.85	96..67
	BP-DNM	91.87 ± 3.88	84.62	96.08	90.35
	BPNN	92.50 ± 3.34	97.92	81.25	89.58
ACA	DE-DMAS	86.18 ± 1.98	88.04	86.09	87.07
	BP-DNM	83.66 ± 9.26	85.54	81.45	83.50
	BPNN	85.57 ± 2.28	87.10	78.07	82.58

5.4. Simplified Model. As previously mentioned, we remove the useless dendrites and synapses by the pruning function. The entire simplification process for the iris dataset is shown in Figure 7. First, initialize the structure of neurons with four dendrites, as shown in Figure 7(a). Synapses on these dendrites receive the inputs X_1, X_2, X_3, and X_4. The

Table 14: No. of models.

No. of BPNN	Learning rate	No. of branch
BPNN1	0.1	30
BPNN2	0.08	60
BPNN3	0.08	30
BPNN4	0.06	60

Table 15: Cross-validation for the iris.

Model	CV10	CV5	CV4	CV2
DE-DMAS	**94.44 ± 5.59**	**94.56 ± 5.90**	**95.00 ± 3.61**	**93.38 ± 5.09**
BP-DNM	93.33 ± 10.79	93.22 ± 6.64	91.93 ± 4.98	86.80 ± 9.16
BPNN1	90.67 ± 7.50	91.42 ± 5.62	89.06 ± 9.41	91.29 ± 7.182
BPNN2	**93.55 ± 10.72**	**93.35 ± 7.92**	**92.80 ± 7.82**	**91.82 ± 7.97**
BPNN3	91.10 ± 9.85	88.77 ± 9.20	90.27 ± 9.89	86.71 ± 13.43
BPNN4	90.44 ± 8.69	89.89 ± 9.07	88.80 ± 9.59	89.07 ± 9.72

Table 16: Cross-validation for the liver.

Model	CV10	CV5	CV4	CV2
DE-DMAS	**70.20 ± 7.98**	**71.93 ± 4.79**	**71.40 ± 0.40**	**70.12 ± 3.10**
BP-DNM	58.82 ± 7.93	**69.42 ± 5.68**	**68.88 ± 5.57**	**66.44 ± 4.81**
BPNN1	58.76 ± 9.45	59.18 ± 5.72	59.43 ± 7.34	60.44 ± 4.30
BPNN2	**63.14 ± 8.20**	58.72 ± 6.11	61.47 ± 6.72	58.83 ± 5.05
BPNN3	58.00 ± 8.33	58.55 ± 5.80	61.47 ± 6.72	59.37 ± 8.50
BPNN4	60.19 ± 8.94	61.11 ± 6.09	60.92 ± 8.45	60.71 ± 5.56

Table 17: Cross-validation for cancer.

Model	CV10	CV5	CV4	CV2
DE-DMAS	**96.38 ± 2.36**	96.05 ± 1.74	**96.11 ± 1.12**	**95.90 ± 0.70**
BP-DNM	95.71 ± 2.17	96.02 ± 1.84	95.75 ± 1.64	95.70 ± 1.06
BPNN1	91.48 ± 11.65	92.26 ± 11.80	**96.29 ± 1.67**	94.29 ± 1.08
BPNN2	**96.57 ± 2.51**	**96.28 ± 1.54**	94.33 ± 1.52	93.71 ± 2.44
BPNN3	93.45 ± 12.00	95.52 ± 1.45	95.73 ± 1.35	93.53 ± 11.54
BPNN4	94.05 ± 12.53	**96.30 ± 1.34**	96.10 ± 1.75	**96.07 ± 1.04**

Table 18: Cross-validation for glass.

Model	CV10	CV5	CV4	CV2
DE-DMAS	**95.24 ± 3.75**	**96.05 ± 1.74**	**96.01 ± 1.36**	**95.70 ± 0.70**
BP-DNM1	**92.53 ± 4.95**	92.40 ± 3.23	**91.58 ± 3.16**	91.12 ± 2.77
BPNN1	91.59 ± 5.55	90.39 ± 4.44	91.05 ± 3.19	91.53 ± 2.22
BPNN2	91.90 ± 4.36	**92.56 ± 4.12**	90.99 ± 4.12	90.31 ± 5.24
BPNN3	91.75 ± 4.83	89.61 ± 4.56	91.42 ± 4.28	89.50 ± 3.85
BPNN4	90.00 ± 6.77	92.02 ± 4.08	90.00 ± 5.36	**91.78 ± 3.72**

Table 19: Cross-validation for ACA.

Model	CV10	CV5	CV4	CV2
DE-DMAS	**84.98 ± 3.95**	**86.79 ± 2.70**	**85.59 ± 2.81**	**85.78 ± 1.39**
BP-DNM	83.09 ± 8.52	82.00 ± 10.62	83.06 ± 10.51	82.07 ± 10.36
BPNN1	**85.94 ± 4.34**	85.20 ± 2.66	**85.04 ± 2.63**	85.39 ± 1.61
BPNN2	84.12 ± 3.97	85.11 ± 2.60	83.51 ± 3.41	**86.13 ± 1.28**
BPNN3	83.51 ± 4.64	85.48 ± 2.79	84.00 ± 3.49	85.35 ± 1.18
BPNN4	84.16 ± 4.48	**86.11 ± 2.15**	83.51 ± 33.30	85.09 ± 1.55

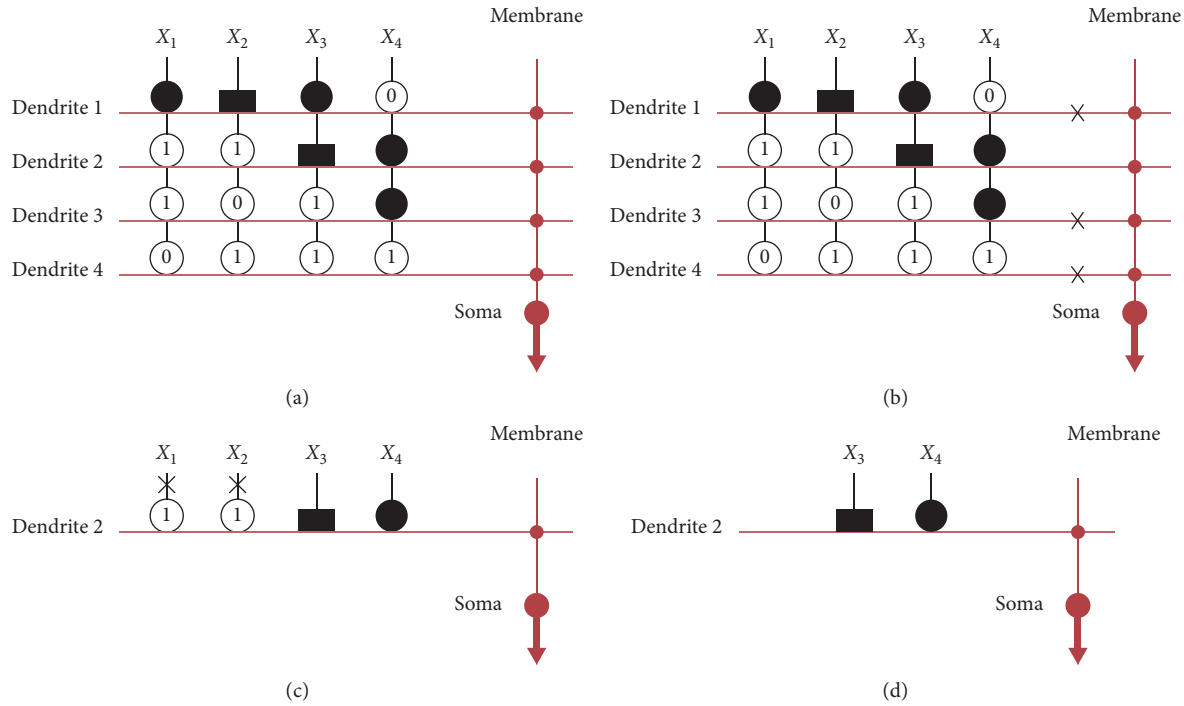


Figure 7: Structure simplification process for the iris dataset.

synapses are activated and converted to the direct-connecting state (●), opposite-connecting state (━), constant-1 state(①), or constant-0 state (⓪) after learning. Second, remove all useless dendrites by the dendrite pruning function; that is, if at least one synapse on a dendrite is in the constant-0 state, remove the dendrites. In Figure 7(b), we denote removed dendrite 1, dendrite 3, and dendrite 4 with the symbol ✖. After dendrite pruning, only dendrite 2 remains, as shown in Figure 7(c). Then, remove all unnecessary synapses by the synapse pruning function; that is, remove the synapses with the constant-1 state by the symbol ✖ in the Figure 7(c). Figure 7(d) shows the simplified structure for the iris dataset. This structure is simplified from 4 layers of dendrites and 4 inputs to only dendrite 2 and the 2 inputs X_3 and X_4.

Liver dataset has 12 layers of dendrites and 6 inputs, as shown in Figure 8(a). In Figure 8(b), we denote removed (dendrites 1, 3, 4, 5, 6, 7, 8, 9, 10, and 12) with the symbol ✖. After dendrite pruning, dendrites 2 and 11 remain, as shown in Figure 8(c). After synapse pruning, X_1, X_3, X_4, and X_6 have been remained, and the others have been removed by the symbol ✖. Figure 8(d) shows the simplified structure for the liver dataset. This structure is simplified from 12 layers of dendrites and 6 inputs to only dendrites 2 and 11 and the 4 inputs X_1, X_3, X_4, and X_6.

Cancer dataset has 9 layers of dendrites and 9 inputs, as shown in Figure 9(a). In Figure 9(b), we denote removed (dendrites 1, 2, 3, 4, 6, 7, and 9) with the symbol ✖. After dendrite pruning, dendrites 5 and 8 remain, as shown in Figure 9(c). After synapse pruning, X_1, X_2, X_3, X_5, and X_6 have been remained, and the others have removed by the symbol ✖. Figure 9(d) shows the simplified structure for the cancer dataset. This structure is simplified from 9 layers of dendrites and 9 inputs to only dendrites 5 and 8 and the 5 inputs X_1, X_2, X_3, X_5, and X_6.

Glass dataset has 27 layers of dendrites and 9 inputs, as shown in Figure 10(a). In Figure 10(b), we denote removed (dendrites 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, and 27) with the symbol ✖. After dendrite pruning, dendrites 6 and 24 remain, as shown in Figure 10(c). After synapse pruning, X_1, X_3, X_4, and X_8 have been remained, and the others have been removed by the symbol ✖. Figure 10(d) shows the simplified structure for the liver dataset. This structure is simplified from 12 layers of dendrites and 6 inputs to only dendrites 6 and 24 and the 4 inputs X_1, X_3, X_4, and X_8.

ACA dataset has 16 layers of dendrites and 14 inputs, as shown in Figure 11(a). In Figure 11(b), we denote removed (dendrites 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 14, 15, and 16) with the symbol ✖. After dendrite pruning, dendrites 10 and 13 remain, as shown in Figure 11(c). After synapse pruning, X_3, X_7, X_8, X_{10}, X_{12}, and X_{13} have remained, and the others have been removed by the symbol ✖. Figure 11(d) shows the simplified structure for the liver dataset. This structure is simplified from 16 layers of dendrites and 6 inputs to only dendrites 10 and 13 and the 6 inputs X_3, X_7, X_8, X_{10}, X_{12}, and X_{13}.

As shown in these figures, we have obtained the final simplified models of the five datasets. After simplifying the models, the structures of the models have been reduced by more than 90%, which indicates that we can use simpler logic to solve the real problem. The problem used to be solved with more than hundreds of logic components but can now be solved by only a few dozen simple logic components, such as comparators, AND gates, OR gates, and NOT gates. This change substantially reduces the labor and time costs. Logic circuits of the five datasets have been drawn

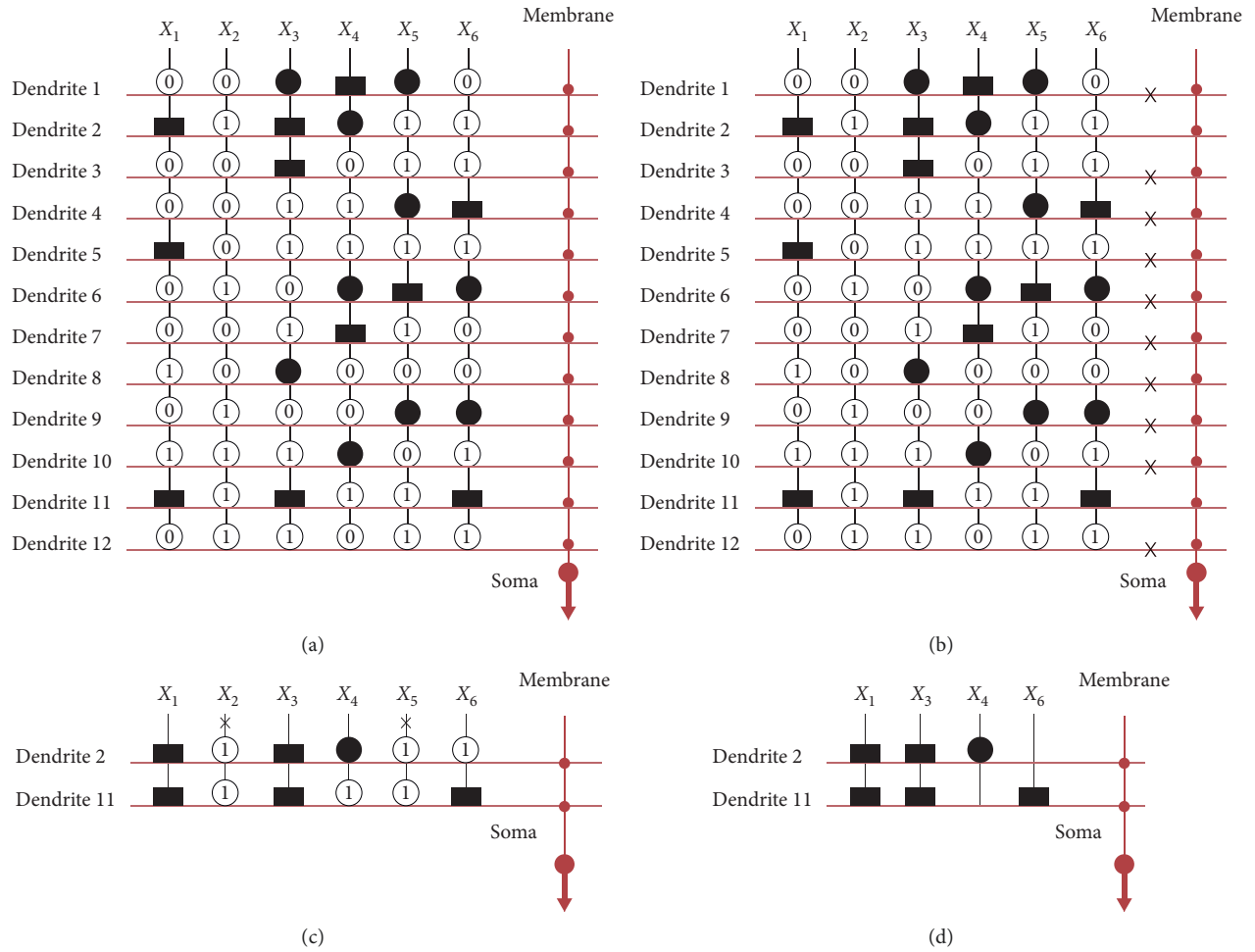


Figure 8: Structure simplification process for the liver dataset.

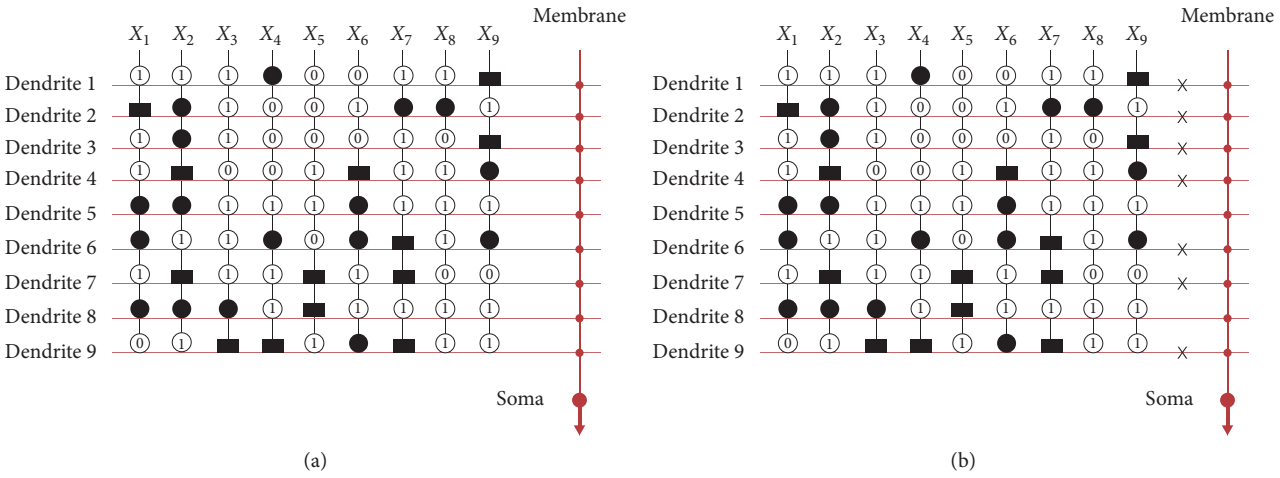


Figure 9: Continued.

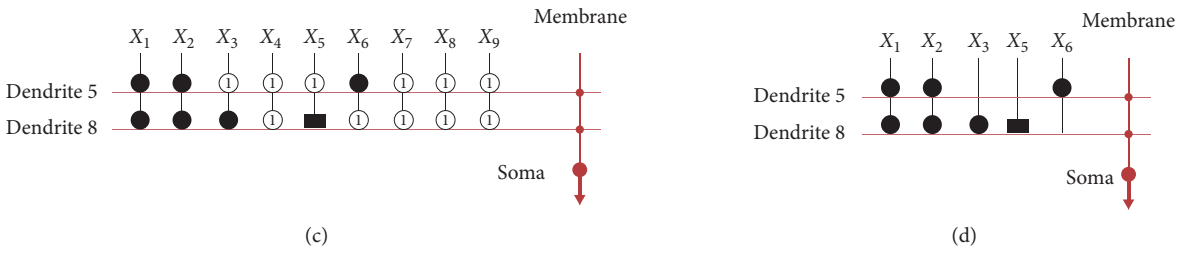


Figure 9: Structure simplification process for the cancer dataset.

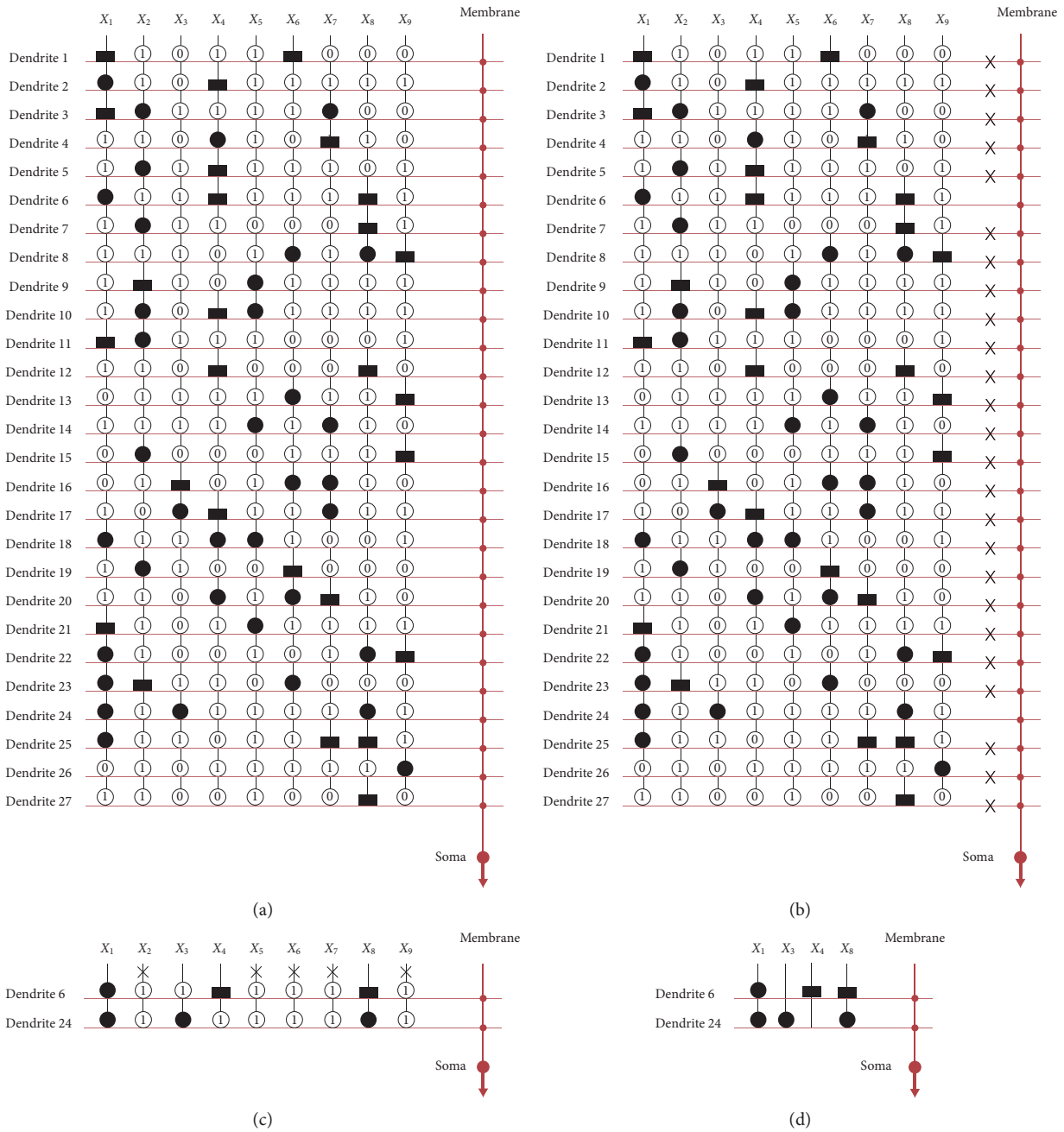


Figure 10: Structure simplification process for the glass dataset.

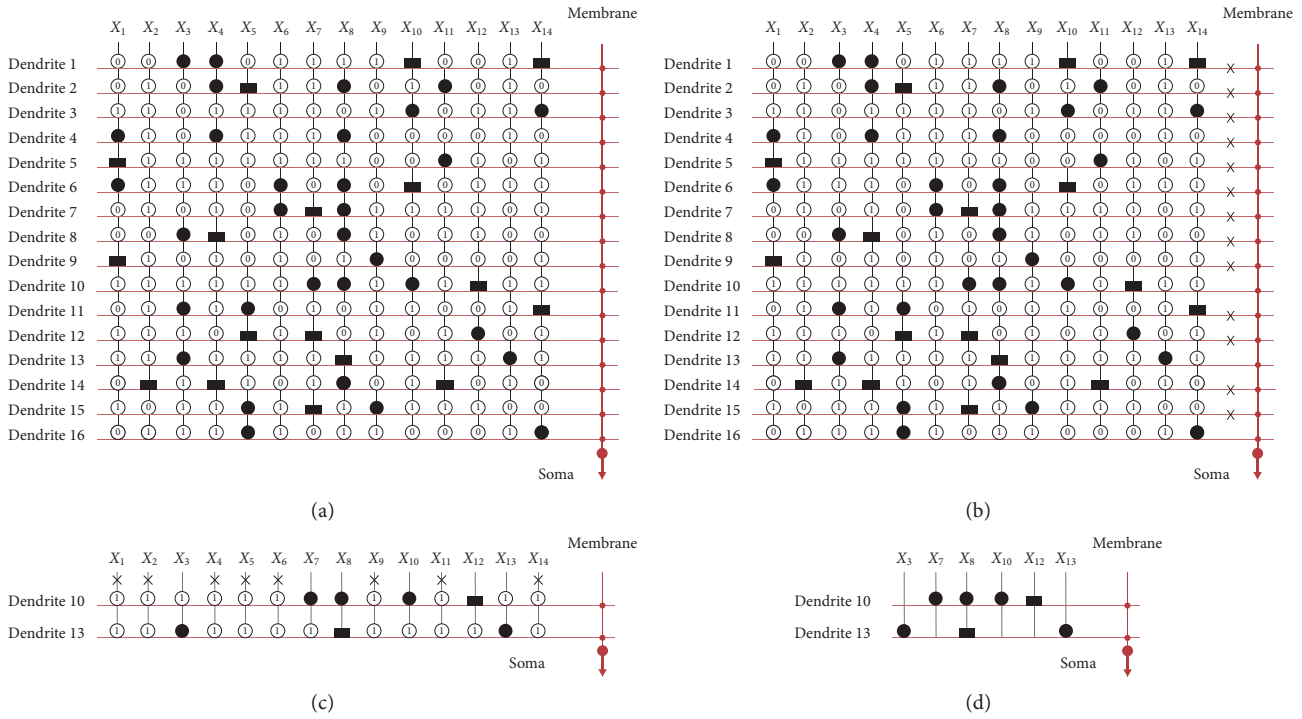


Figure 11: Structure simplification process for the ACA dataset.

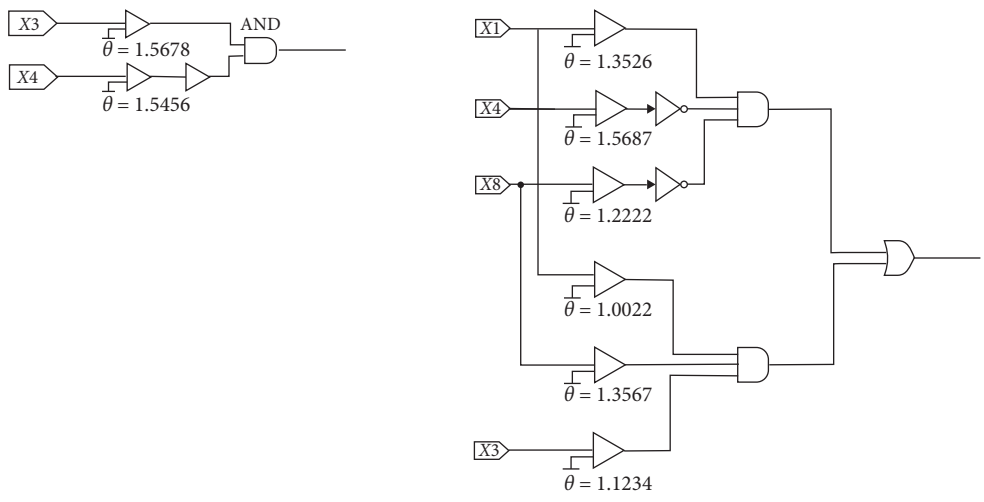


Figure 12: Continued.

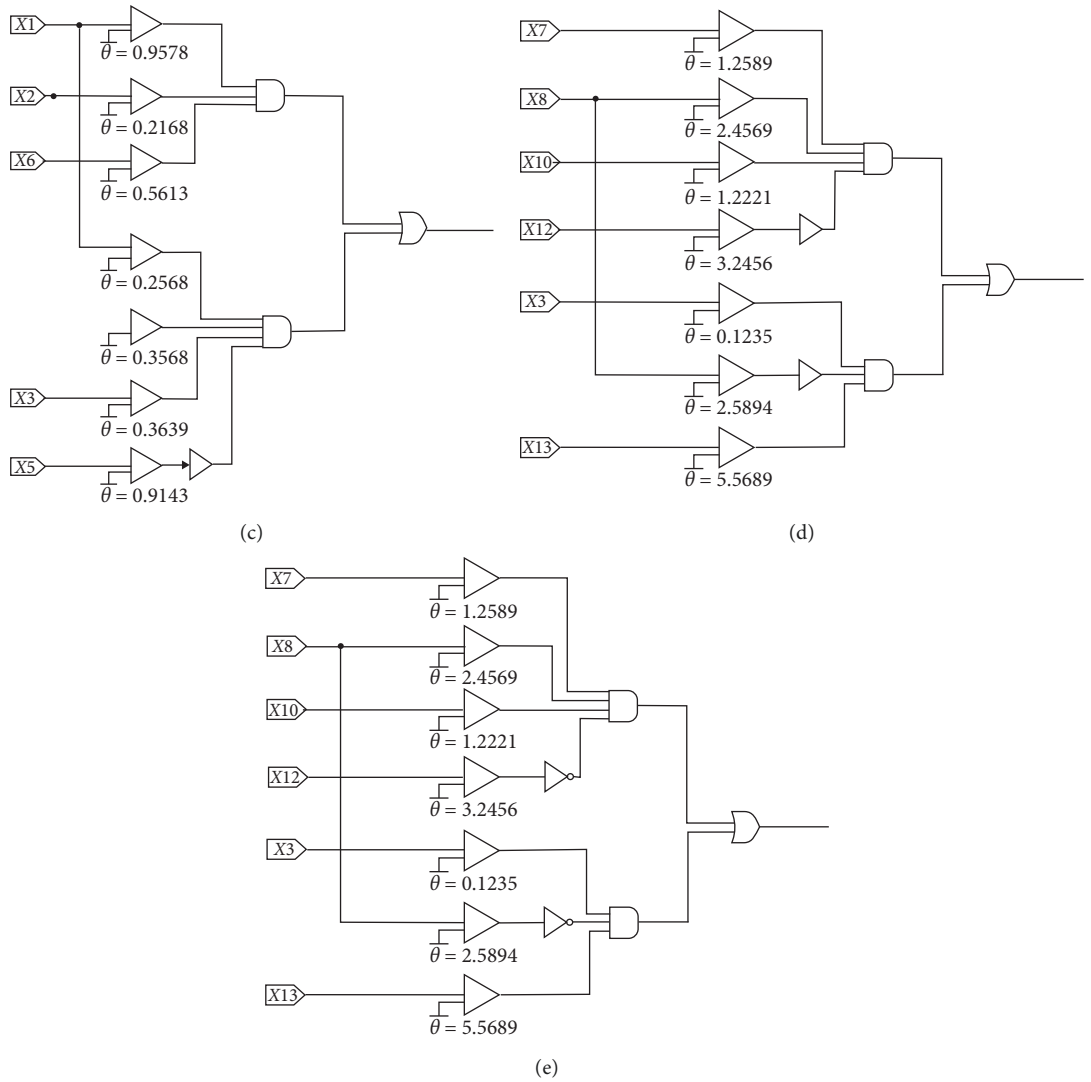


Figure 12: Logic circuits obtained by the proposed method for five datasets: (a) iris; (b) liver; (c) cancer; (d) glass; (e) ACA.

in Figure 12. The value of θ in the comparator in the figure is the value of θ in the corresponding synapse after training. The standardized input was calculated by these logic components to get the expected output as 0 or 1.

6. Conclusion

To improve the calculation ability of the dendritic neuron model (DNM), a dendritic neuron model with adaptive synapses trained by differential evolution algorithm (DE-DMAS) is proposed, which shows enhanced performance in the simulation based on UCI datasets. A comparison with the classic BPNN and BP-DNM is carried out in terms of the test accuracy, sensitivity, specificity, and ROC and cross-validation. DE-DMAS shows its superiority in all the results, and DE-DMAS as a single neuron model is found to substantially outperform BPNN and BP-DNM.

DMAS has further access to the real biological neuron with a self-pruning ability. This function can eliminate branches from the dendrite morphology depending on the continuum values. It hence reduces the computational load by evolving and simplifying the dendritic structure without affecting the computational result. Simplified dendritic structure can be implemented in the logic circuit with comparator, OR gate, AND gate, and NOT gate. It makes it possible to solve real problems with less cost.

To highlight the contribution of this work, a self-adaptive synapse is for the first time proposed in the paper. Its utility is proved by the Friedman test as summarized in Section 4.3. The ability of adaptive synapse not only has stronger robustness but also reduces a parameter in DNM and improves the performance of DNM.

The dendrite plays a pivotal role in the computing process. A single DE-DMAS neuron model can only deal with dichotomies problems (i.e., binary classification problems), which is its main limitation. But all the current neural networks are made up of multiple single neuron models which can only deal with dichotomies. This paper aims at proposing the DE-DMAS model instead of the network structure, so it is only a single one. Nevertheless, it is worth pointing out that variants of DE-DMAS can be developed for solving multiclass classification problems. For

example, by using softmax function (together with the cross entropy), the multiclass classification problem can be approximately transformed into dichotomies problems, and the one-hot-encoding strategy based on several DE-DMAS neuron models can be used to calculate the information entropy. The reason why we choose multiple datasets that can be classified into two categories for experiments is that we want to more intuitively reflect the ability of a single neuron model, rather than the network of several ones. Further research will focus on DMAS's adjustment to make it adaptive to the deep learning structure. We also believe that this model has considerable potential in fields of electronic design, such as VLSI and biomedical science.

Data Availability

The classification dataset could be downloaded freely at https://archive.ics.uci.edu/ml/index.php.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the JSPS KAKENHI (Grant no. JP19K12136).

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] N. Rochester, J. Holland, L. Haibt, and W. Duda, "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *IEEE Transactions on Information Theory*, vol. 2, no. 3, pp. 80–93, 1956.

[3] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[4] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA, 2017.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Technical Report, California Univ San Diego La Jolla Inst for Cognitive Science, La Jolla, CA, USA, 1985.

[6] C. Koch, "Computation and the single neuron," *Nature*, vol. 385, no. 6613, pp. 207–210, 1997.

[7] C. Koch and I. Segev, "The role of single neurons in information processing," *Nature Neuroscience*, vol. 3, no. 11, pp. 1171–1177, 2000.

[8] J. L. Davidson and F. Hummer, "Morphology neural networks: an introduction with applications," *Circuits, Systems, and Signal Processing*, vol. 12, no. 2, pp. 177–210, 1993.

[9] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 4, pp. 709–717, IEEE, Vienna, Austria, August 1996.

[10] H. Sossa and E. Guevara, "Efficient training for dendrite morphological neural networks," *Neurocomputing*, vol. 131, pp. 132–142, 2014.

[11] H. Sossa and E. Guevara, "Modified dendrite morphological neural network applied to 3D object recognition," in *Mexican Conference on Pattern Recognition*, pp. 314–324, Springer, Berlin, Germany, 2013.

[12] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Networks*, vol. 60, pp. 96–103, 2014.

[13] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, 2016.

[14] Y. Tang, J. Ji, Y. Zhu, S. Gao, Z. Tang, and Y. Todo, "A differential evolution-oriented pruning neural network model for bankruptcy prediction," *Complexity*, vol. 2019, Article ID 8682124, 21 pages, 2019.

[15] X. Qian, Y. Wang, S. Cao, Y. Todo, and S. Gao, "Mr2DNM: a novel mutual information-based dendritic neuron model," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 7362931, 13 pages, 2019.

[16] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowledge-Based Systems*, vol. 105, pp. 214–224, 2016.

[17] W. Chen, J. Sun, S. Gao, J.-J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to Japan," *IEICE Transactions on Information and Systems*, vol. 100, no. 1, pp. 190–202, 2017.

[18] J. Ji, S. Song, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "Approximate logic neuron model trained by states of matter search algorithm," *Knowledge-Based Systems*, vol. 163, pp. 120–130, 2019.

[19] Y. Yu, Y. Wang, S. Gao, and Z. Tang, "Statistical modeling and prediction for tourism economy using dendritic neural network," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 7436948, 9 pages, 2017.

[20] L. Luo and D. D. M. O'Leary, "Axon retraction and degeneration in development and disease," *Annual Review of Neuroscience*, vol. 28, no. 1, pp. 127–156, 2005.

[21] P. Hagmann, O. Sporns, N. Madan et al., "White matter maturation reshapes structural connectivity in the late developing human brain," *Proceedings of the National Academy of Sciences*, vol. 107, no. 44, pp. 19067–19072, 2010.

[22] C. Koch, T. Poggio, and V. Torres, "Retinal ganglion cells: a functional interpretation of dendritic morphology," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 298, no. 1090, pp. 227–263, 1982.

[23] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing," *Proceedings of the National Academy of Sciences*, vol. 80, no. 9, pp. 2799–2802, 1983.

[24] L. Beaulieu-Laroche, E. H. S. Toloza, N. J. Brown, and M. T. Harnett, "Widespread and highly correlated somato-dendritic activity in cortical layer 5 neurons," *Neuron*, vol. 103, no. 2, pp. 235–241, 2019.

[25] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.

[26] X. G. Wang, Z. Tang, H. Tamura, M. Ishii, and W. D. Sun, "An improved backpropagation algorithm to avoid the local minima problem," *Neurocomputing*, vol. 56, pp. 455–460, 2004.

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

[28] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[29] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, vol. 89, pp. 762–767, Detroit, MI, USA, August 1989.

[30] S. Gao, S. Song, J. Cheng, Y. Todo, and M. Zhou, "Incorporation of solvent effect into multi-objective evolutionary algorithm for improved protein structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 4, pp. 1365–1378, 2018.

[31] S. Song, S. Gao, X. Chen, D. Jia, X. Qian, and Y. Todo, "AIMOES: archive information assisted multi-objective evolutionary strategy for ab initio protein structure prediction," *Knowledge-Based Systems*, vol. 146, pp. 58–72, 2018.

[32] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, Boston, MA, USA, 2010.

[33] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 2, pp. 1980–1987, IEEE, Portland, OR, USA, June 2004.

[34] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, In press.

[35] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution algorithm for multivalued logic networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 233–246, 2020.

[36] B. Subudhi and D. Jena, "A differential evolution based neural network approach to nonlinear system identification," *Applied Soft Computing*, vol. 11, no. 1, pp. 861–871, 2011.

[37] E. Bas, "The training of multiplicative neuron model based artificial neural networks with differential evolution algorithm for forecasting," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 6, no. 1, pp. 5–11, 2016.

[38] F. Arce, E. Zamora, H. Sossa, and R. Barrón, "Differential evolution training algorithm for dendrite morphological neural networks," *Applied Soft Computing*, vol. 68, pp. 303–313, 2018.

[39] M. J. Tyre and E. Von Hippel, "The situated nature of adaptive learning in organizations," *Organization Science*, vol. 8, no. 1, pp. 71–83, 1997.

[40] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Transactions on Information and Systems*, vol. 98, no. 7, pp. 1365–1376, 2015.

[41] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree for liver disorders," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 12, no. 1, pp. 105–115, 2017.

[42] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pp. 485–492, ACM, Seattle, WC, USA, July 2006.

[43] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[44] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[45] Y. Yu, S. Gao, Y. Wang, and Y. Todo, "Global optimum-based search differential evolution," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 379–394, 2019.

[46] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, vol. 10, no. 10, pp. 293–298, 2002.

[47] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 506–513, IEEE, Scotland, UK, September 2005.

[48] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Applied Soft Computing*, vol. 9, no. 3, pp. 1126–1138, 2009.

[49] G. Taguchi, R. Jugulum, and S. Taguchi, *Computer-Based Robust Engineering: Essentials for DFSS*, ASQ Quality Press, Milwaukee, WI, USA, 2004.

[50] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[51] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification and Scene Analysis*, Wiley, New York, NY, USA, 1973.

[52] B. V. Dasarathy, "Nosing around the neighborhood: a new system structure and classification rule for recognition in partially exposed environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 67–71, 1980.

[53] J. McDermott and R. S. Forsyth, "Diagnosing a disorder in a classification benchmark," *Pattern Recognition Letters*, vol. 73, pp. 41–43, 2016.

[54] O. L. Mangasarian and W. H. Wolberg, *Cancer Diagnosis via Linear Programming,*Technical Report, University of Wisconsin-Madison Department of Computer Sciences, Madison, WI, USA, 1990.

[55] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the National Academy of Sciences*, vol. 87, no. 23, pp. 9193–9196, 1990.

[56] I. W. Evett and J. S. Ernest, *Rule Induction in Forensic Science*, Central Research Establishment. Home Office Forensic Science Service, Aldermaston, UK, 1998.

[57] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221–234, 1987.

[58] Y. Tang, J. Ji, S. Gao, H. Dai, Y. Yu, and Y. Todo, "A pruning neural network model in credit classification analysis," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 9390410, 22 pages, 2018.

[59] J. F. C. Khaw, B. S. Lim, and L. E. N. Lim, "Optimal design of neural networks using the Taguchi method," *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.

[60] W. Yang and Y. Tarng, "Design optimization of cutting parameters for turning operations based on the Taguchi method," *Journal of Materials Processing Technology*, vol. 84, no. 1–3, pp. 122–129, 1998.

[61] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[62] C. H. Yu, "Exploratory data analysis," *Methods*, vol. 2, pp. 131–160, 1977.

[63] N. R. Cook, "Statistical evaluation of prognostic versus diagnostic models: beyond the ROC curve," *Clinical Chemistry*, vol. 54, no. 1, pp. 17–23, 2008.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 3238574, 16 pages
https://doi.org/10.1155/2019/3238574

Research Article

A Db-Scan Binarization Algorithm Applied to Matrix Covering Problems

José García,[1] **Paola Moraga**,[1] **Matias Valenzuela**,[1] **Broderick Crawford**,[1] **Ricardo Soto**,[1] **Hernan Pinto**,[1] **Alvaro Peña**,[1] **Francisco Altimiras**,[1] **and Gino Astorga**[2]

[1]*Pontificia Universidad Católica de Valparíso, 2362807 Valparaíso, Chile*
[2]*Universidad de Valparaíso, 2361864 Valparaíso, Chile*

Correspondence should be addressed to José García; jose.garcia@pucv.cl

Received 17 June 2019; Accepted 4 August 2019; Published 16 September 2019

Academic Editor: Cornelio Yáñez-Márquez

The integration of machine learning techniques and metaheuristic algorithms is an area of interest due to the great potential for applications. In particular, using these hybrid techniques to solve combinatorial optimization problems (COPs) to improve the quality of the solutions and convergence times is of great interest in operations research. In this article, the db-scan unsupervised learning technique is explored with the goal of using it in the binarization process of continuous swarm intelligence metaheuristic algorithms. The contribution of the db-scan operator to the binarization process is analyzed systematically through the design of random operators. Additionally, the behavior of this algorithm is studied and compared with other binarization methods based on clusters and transfer functions (TFs). To verify the results, the well-known set covering problem is addressed, and a real-world problem is solved. The results show that the integration of the db-scan technique produces consistently better results in terms of computation time and quality of the solutions when compared with TFs and random operators. Furthermore, when it is compared with other clustering techniques, we see that it achieves significantly improved convergence times.

1. Introduction

In recent years, different optimization methods based on evolutionary concepts have been explored. These methods have been used to solve complex problems, therein obtaining interesting levels of performance [1–3], and many such methods have been inspired by concepts extracted from abstractions of natural or social phenomena. These abstractions can be interpreted as search strategies according to an optimization perspective [4]. These algorithms are inspired, for example, by the collective behavior of birds, e.g., the cuckoo search algorithm [5]; the movement of fish, e.g., the artificial fish swarm algorithm (AFSA) [6]; particle movement, e.g., particle swarm optimization (PSO) [7]; the social interactions of bees (ABC) [8]; and the process of musical creation, as in the search for harmony (HS) [9] and in genetic algorithms (GA) [10], among others. Many of these algorithms work naturally in continuous spaces.

On the other hand, lines of research that allow robust algorithms associated with the solution of combinatorial optimization problems (COPs) to be obtained are of great interest in the areas of computer science and operations research. This interest is currently mainly related to decision making in complex systems. Many of these decisions require the evaluation of a very large combination of elements in addition to having to solve a COP to find a feasible and satisfactory result. Examples of COPs can be found in the areas of logistics, finance, transportation, biology, and many others. Depending on the definition of the problem, many COPs can be classified as NP-hard. Among the most successful ways to address such problems, one common method is to simplify the model to attempt to solve instances of small to medium size using exact techniques. Large problems are usually addressed by heuristic or metaheuristic algorithms.

The idea of hybridization between metaheuristic techniques and methods from other areas aims to obtain more

robust algorithms in terms of solution quality and convergence times. State-of-the-art proposals for hybridization mainly include the following: (i) *mateheuristics*, which combines heuristics or metaheuristics with mathematical programming [11]; (ii) *hybrid heuristics*, which corresponds to the integration of different heuristic or metaheuristic methods [12]; (iii) *simheurístics*, which combines simulation and metaheuristics [13]; and (iv) the hybridization between metaheuristics and machine learning [14]. Machine learning can be considered as a set of algorithms that enable the identification of significant, potentially useful information and learning through the use of data. In this work, useful information will be obtained using the data generated by a continuous metaheuristic algorithm through the use of the db-scan unsupervised learning technique to obtain robust binarizations of this algorithm.

Within the lines of this discussion, we aim to provide the following contributions:

(i) A novel automatic learning binarization algorithm is proposed to allow metaheuristics commonly defined and used in continuous optimization to efficiently address COPs. This algorithm uses the db-scan unsupervised learning technique to perform the binarization process. The selected metaheuristics are particle swarm optimization (PSO) and cuckoo search (CS). Their selection is based on the fact that they are commonly used in continuous optimization and enable a method for adjusting their parameters in continuous spaces.

(ii) These hybrid metaheuristics are applied to the well-known set covering problem (SCP). This problem has been studied extensively in the literature, and therefore, there known instances where we can clearly evaluate the contribution of the db-scan binarization operator. On the other hand, the SCP has numerous practical real-world applications such as vehicle routing, railways, airline crew scheduling, microbial communities, and pattern finding [15–18].

(iii) Random operators are designed to study the contribution of the db-scan binarization algorithm in the binarization process. Additionally, the behavior of db-scan binarization is studied, comparing it with binarization methods that use k-means and transfer functions (TFs). Finally, the binarizations obtained by db-scan are used to solve a real-world problem.

The remainder of this article is structured as follows. Section 2 describes the SCP and some of its applications. In Section 3, a state-of-the-art hybridization between the areas of machine learning and metaheuristics is provided, and the main binarization methods are described. Later, in Section 4, the proposed db-scan algorithm is detailed. The contributions of the db-scan operator are provided in Section 5. Additionally, in this section, the db-scan technique is studied by comparing it with other binarization techniques that use k-means and TFs as a binarization mechanism. In Section 6, a real-world application problem is solved. Finally, in Section 7, conclusions and some future lines of research are given.

2. Set Covering Problem

SCP is one of the oldest and most-studied optimization problems and is well known to be NP-hard [19]. Nevertheless, different solution algorithms have been developed. There exist exact algorithms that generally rely on the branch-and-bound and branch-and-cut methods to obtain optimal solutions [20, 21]. These methods, however, struggle to solve SCP instances that grow exponentially with the problem size. Even medium-sized problem instances often become intractable and can no longer be solved using exact algorithms. To overcome this issue, different heuristics have been proposed [22, 23].

For example, [22] presented a number of greedy algorithms based on a Lagrangian relaxation (called Lagrangian heuristics). Caprara et al. [24] introduced relaxation-based Lagrangian heuristics applied to the SCP. Metaheuristics, e.g., genetic algorithms [25], simulated annealing [26], and ant colony optimization [27], have also been applied to solve the SCP. More recently, swarm-based metaheuristics, such as the cat swarm [28], cuckoo search [29], artificial bee colony [8], and black hole [30] metaheuristics, have also been proposed.

The SCP has many practical applications in engineering, e.g., vehicle routing, railways, airline crew scheduling, microbial communities, and pattern finding [15, 16, 18, 31].

The SCP can be formally defined as follows. Let $A = (a_{ij})$ be an $n \times m$ zero-one matrix, where a column j covers a row i if $a_{ij} = 1$, and a column j is associated with a nonnegative real cost c_j. Let $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$ be the row and column set of A, respectively. The SCP consists of searching a minimum cost subset $S \subset J$ for which every row $i \in I$ is covered by at least one column $j \in J$, i.e.,

$$\text{minimize } f(x) = \sum_{j=1}^{m} c_j x_j, \tag{1}$$

$$\text{subject to } \sum_{j=1}^{m} a_{ij} x_j \geq 1, \quad \forall i \in I, \text{ and } x_j \in \{0, 1\}, \forall j \in J, \tag{2}$$

where $x_j = 1$ if $j \in S$ and $x_j = 0$, otherwise.

3. Related Work

3.1. Related Binarization Work. A series of metaheuristic algorithms designed to work in continuous spaces have been developed. Particle swarm optimization (PSO) and cuckoo search (CS) are two of the most commonly used metaheuristic algorithms. On the other hand, the existence of a large number of $\mathcal{NP}$-hard combinatorial problems motivates the investigation of robust mechanisms that allow these continuous algorithms to be adapted to discrete versions.

In a review of the state-of-the-art binarization techniques [32], two approximations were identified. The first approach considers general methods of binarization. In those general methods, there is a mechanism that allows the transformation of any continuous metaheuristic into a binary one without

altering the metaheuristic operators. In this approach, the main frameworks used are TFs and angle modulation. The second approach corresponds to binarizations in which the method of operating metaheuristics is specifically altered. Under this second approach, notable techniques include quantum binary and set-based approaches.

3.1.1. Transfer Functions. The simplest and most widely used binarization method corresponds to TFs. TFs were introduced by [33] to generate binary versions of PSO. This algorithm considers each solution as a particle. The particle has a position given by a solution in an iteration and a velocity that corresponds to the vector obtained from the difference of the particle position between two consecutive iterations. The TF is a very simple operator and relates the velocity of the particles in PSO with a transition probability. The TF takes values from $\mathbb{R}^n$ and generates transition probability values in $[0,1]^n$. The TFs force the particles to move in a binary space. Depending on the function's shape, they are usually classified as S-shape [34] and V-shape functions [1]. Once the function produces a value between 0 and 1, the next step is to use a rule that allows obtaining 0 or 1. For this, well-defined rules are applied that use the concepts of complement, elite, and random, among others.

3.1.2. Angle Modulation. This method is based on the family of trigonometric functions shown in equation (3). These functions have four parameters responsible for controlling the frequency and displacement of the trigonometric function:

$$g_i\left(x_j\right) = \sin\left(2\pi\left(x_j - a_i\right)b_i \cos\left(2\pi\left(x_j - a_i\right)c_i\right)\right) + d_i. \quad (3)$$

The first time this method was applied to binarizations was in PSO. In this case, binary PSO was applied to benchmark functions. Assume a given binary problem of dimension n, and let $X = (x_1, x_2, \ldots, x_n)$ be a solution. We start with a four-dimensional search space. Each dimension represents a coefficient of equation (3). Then, every solution (a_i, b_i, c_i, d_i) is associated with a trigonometric function g_i. For each element x_j, the following rule is applied:

$$b_{ij} = \begin{cases} 1, & \text{if } g_i\left(x_j\right) \geq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Then, for each initial solution of 4 dimensions (a_i, b_i, c_i, d_i), the function g_i, which is shown in equation (3), is applied and then equation (4) is utilized. As a result, a binary solution of dimension n, $(b_{i1}, b_{i2}, \ldots, b_{\text{in}})$, is obtained. This is a feasible solution for our n-binary problem. The angle modulation method has been applied to network reconfiguration problems [35] using a binary PSO method, to an antenna position problem using an angle modulation binary bat algorithm [36], and to a multiuser detection technique [37] using a binary adaptive evolutionary algorithm.

3.1.3. Quantum Binary Approach. There are three main types of algorithms in research that integrates the areas of evolutionary computation (EC) and quantum computation [38].

(1) Quantum evolutionary algorithms: these methods correspond to the design of EC algorithms to be applied in a quantum computing environment

(2) Evolutionary-based quantum algorithms: these algorithms attempt to automate the generation of new quantum algorithms using evolutionary algorithms

(3) Quantum-inspired evolutionary algorithms: this category uses quantum computing concepts to strengthen EC algorithms

In particular, the quantum binary approach is a type of quantum-inspired evolutionary algorithm. Specifically, this approach adapts the concepts of q-bits and superposition used in quantum computing applied to traditional computers.

In the quantum binary approach, each feasible solution has a position $X = (x_1, x_2, \ldots, x_n)$ and a quantum q-bit vector $Q = [Q_1, Q_2, \ldots, Q_n]$. Q represents the probability of x_j taking the value 1. For each dimension j, a random number between [0, 1] is generated and compared with Q_j: if $\text{rand} < Q_j$, then $x_j = 1$; otherwise, $x_j = 0$. The upgrade mechanism of the Q vector is specific for each metaheuristic.

The main difficulty that general binarization frameworks face is related to the concept of spatial disconnect [39]. A spatial disconnect originates when nearby solutions generated by metaheuristics in the continuous space are not transformed into nearby solutions when applying the binarization process. Roughly speaking, we can think of a loss of the continuity of the framework. The spatial disconnect phenomenon consequently alters the properties of exploration and exploitation, and therefore the precision and convergence of the metaheuristics decrease. A study was conducted on how the TFs affect the exploration and exploitation properties in [40]. For angle modulation, a study was conducted in [39].

On the other hand, specific binarization algorithms that modify the operators of the metaheuristic are susceptible to problems such as Hamming cliffs, loss of precision, search space discretization, and the curse of dimensionality [39]. This was studied by [41] and for the particular case of PSO by [42]. In the latter, the authors observed that the parameters of the Binary PSO change the speed behavior of the original metaheuristic.

3.2. Hybridizing Metaheuristics with Machine Learning. Machine learning concerns algorithms that are capable of learning from a dataset [43]. This learning can be supervised, unsupervised, or semisupervised. Usually, these algorithms are used in problems of regression, classification, transformation, dimensionality reduction, time series, anomaly detection, and computational vision [44], among others. On the other hand, metaheuristics correspond to a broad family of algorithms designed to solve complex problems without the need for a deep adaptation of their mechanism when

changing problems. They are incomplete techniques and generally have a set of parameters that must be adjusted for proper operation.

When a state-of-the-art integration between metaheuristic and machine learning algorithms is developed, the integration is considered bidirectional. This means that there are studies whereby metaheuristic algorithms contribute to improving the performance of machine learning algorithms, and there are investigations where machine learning algorithms improve the convergence and quality of metaheuristic algorithms. In the case of metaheuristics that improve the performance of machine learning algorithms, we see that integration is found in all areas. In classification problems, we find that such algorithms have been used mainly in feature selection, feature extraction, and the tuning of parameters. In [45], a genetic-SVM algorithm was developed to improve the recognition of breast cancer through image analysis. In this algorithm, genetic algorithms were used for the extraction of characteristics. A multiverse optimizer algorithm was used in [46] to perform the feature selection and parameter tuning of SVM on a robust system architecture. The training of feed-forward neural networks was addressed using an improved monarch butterfly algorithm in [47]. In [48], a geotechnical problem was addressed by integrating a firefly algorithm with the least squares support vector machine technique. For the case of regression problems, we found in [49] an application in the prediction of the compressive strength of high-performance concrete using metaheuristic-optimized least squares support vector regression. The improved prediction of stock prices was addressed in [50], therein integrating metaheuristics and artificial neural networks. Additionally, in [51], a stock price prediction technique was developed using a sliding-window metaheuristic-optimized machine learning regression applied to Taiwan's construction companies. In [52], using a firefly version, the least squares vector regression parameters were optimized with the aim of improving the accuracy of the prediction in engineering design. In the case of unsupervised learning techniques, we find that metaheuristics have contributed significantly to clustering techniques. For example, in [53], an evolutionary-based clustering algorithm that combines a metaheuristic with a kernel intuitionistic fuzzy c-means method was proposed with the aim of designing clustering solutions to apply them to different types of datasets. Also in clustering, centroid search is a problem type that suffers a large algorithmic complexity. This problem consists of the search for centroids with the objective of grouping the set of objects studied in an improved manner. Because this problem is NP-hard, approximation methods have been proposed. For example, an improved artificial bee colony algorithm was developed in [54] with the goal of solving the energy efficiency clustering problem in a wireless sensor network. In [55], a mathematical model and a clustering search metaheuristic were developed for addressing the helicopter transportation planning of oil and gas production platform employees.

On the other hand, when looking for the contributions of machine learning techniques in metaheuristic algorithms, two main lines of research can be distinguished. The first line of research corresponds to specific integrations. In these integrations, machine learning techniques are integrated through a specific operator in one of the modules that establish the metaheuristic. The second line of research explores general integrations, where the machine learning techniques work as a selector of different metaheuristic algorithms, therein choosing the most appropriate for each instance. A metaheuristic, in addition to its evolution mechanism, usually uses solution initiation operators, solution perturbation, population management, binarization, the tuning of parameters, and local search operators, among others. The specific integrations explore the machine learning application on some of these operators. For the case of parameter tuning in [56], the parameter tuning of a chess rating system was implemented. Based on decision trees and using fuzzy logic, a semiautomatic parameter tuning algorithm was designed in [57]. Another relevant area of research is related to the design of binary versions of algorithms that work naturally in continuous spaces. This line of research aims to apply these binary versions in combinatorial problems. In this area, we find in [2] the application of unsupervised learning techniques to perform the binarization process. In [29], the percentile concept was explored in the process of generating binary algorithms. Additionally, in [17], the big data Apache spark framework was applied to manage the size of the solution population to improve the convergence times and quality of results. The randomness mechanism is frequently used for the initialization of the solutions of a metaheuristic. However, machine learning has been used to improve the solution initialization stage. In [58], case-based reasoning was used to initialize a genetic algorithm and apply it to the weighted-circle design problem. Hopfield neural networks were used in [59] to initiate solutions of a genetic algorithm that was used to solve an economic dispatch problem.

When analyzing the methods found in the literature addressing general integrations of machine learning algorithms on metaheuristics, we find three main groups: algorithm selection, hyperheuristics, and cooperative strategies. The objective of algorithm selection is to choose from a set of algorithms and a group of associated characteristics for each instance of the problem an algorithm that performs best for similar instances. In the hyperheuristics strategy, the goal is to automate the design of heuristics or metaheuristic methods to address a wide range of problems. Finally, cooperative strategies consist of combining algorithms in a parallel or sequential manner to obtain more robust methods. The cooperation can be completed by sharing the complete solution or partially when only part of the solution is shared. In [60], the berth scheduling problem at bulk terminals was addressed by algorithm selection techniques. The problem of nurse rostering through a tensor-based hyperheuristic algorithm was addressed in [61]. Finally, a distributed framework based on agents was proposed in [62]. In this case, each agent corresponds to a metaheuristic, and it has the ability to adapt through direct cooperation. This framework was applied to the problem of permutation flow stores.

4. Binary Db-Scan Algorithm

The binary db-scan algorithm is composed of five operators. The first operator, which will be detailed in Section 4.1, initializes the solutions. After the solutions are started, the next step is to verify if the maximum iteration criterion is met. When the criterion has not been met, the binary db-scan operator is executed. This operator continuously executes the metaheuristics and then clusters the solutions considering the absolute value of the velocity of the solutions. The details of this operator are described in Section 4.2. Subsequently, using the clusters generated by the db-scan operator, the transition operator will proceed to binarize the solutions generated by the continuous metaheuristics. When points are identified by db-scan as outliers, a transition operator for outliers is applied. The transition operator and the outlier operator are described in Section 4.3. Finally, when the obtained solutions do not satisfy all the restrictions, the repair operator described in Section 4.4 is applied. Additionally, a heuristic operator is used in the initiation and repair of the solutions. This operator is detailed in Section 4.5. The flow diagram of the binary db-scan algorithm is shown in Figure 1.

4.1. Initiation Operator. This operator attempts to initiate the solutions that the binary db-scan algorithm will use. The first step, the SelectRandomColumn() function select a column randomly. Then, the operator asks if the row coverage constraint is fulfilled. When the constraint is not met, the initiating operator calls the heuristic operator. This heuristic operator receives the list of columns that currently has the solution and returns a new column to be incorporated. The details of the heuristic operator are described in Section 4.5. The call to the heuristic operator is executed until all rows are covered. The procedure for initiating the solutions is shown in Algorithm 1.

4.2. Binary Db-Scan Operator. The goal of the binary db-scan operator is to group the different solutions obtained by the execution of the continuous metaheuristics. When considering solutions as particles, we will understand the position of the particle as the location of the solution in the search space. On the other hand, the velocity represents the transition vector of the particle from iteration t to iteration $t+1$.

To perform the clustering, the density-based spatial clustering of applications with noise (db-scan) algorithm is used. Db-scan is a data grouping algorithm proposed in 1996 by [63]. Db-scan uses the concept of density to perform the clustering: given a set of S points in a metric space, db-scan groups the points with many nearby neighbors, marking as outliers those that are alone in low-density regions. Db-scan requires two parameters: a radius ϵ and a minimum number of neighbors δ. The db-scan algorithm can be divided into the following steps:

(i) Find the points in the ϵ neighborhood of every point and identify the core points with more than δ neighbors

(ii) Find the connected components of core points on the neighbor graph, ignoring all noncore points

(iii) Assign each noncore point to a nearby cluster if the cluster is an ϵ neighbor; otherwise, assign it to noise

Let Mh be a swarm intelligence continuous metaheuristic and List$P(t)$ be the position list of the solutions given by Mh at iteration t. The binary db-scan operator has input parameters Mh and List$P(t)$ and aims to cluster the solutions given by Mh. The first step of the operator is to iterate the list List$P(t)$ using Mh to obtain the list of positions List$P(t+1)$ at iteration $t+1$. Subsequently, using List$P(t)$ and List$P(t+1)$, we obtain the list with transition velocities List$V(t+1)$.

Let $v^p(t+1) \in \text{List}V(t+1)$ be the velocity vector in the transition between t and $t+1$ corresponding to particle p. This vector has n dimensions, where n depends on the number of columns that the problem possesses. Let $v_i^p(t+1) \in v^p(t+1)$ be the value for dimension i of the vector $v^p(t+1)$. Then, List$V_i(t+1)$ corresponds to the list of absolute values of $v_i^p(t+1), \forall v^p(t+1) \in \text{List}V(t+1)$. Next, we apply db-scan to the list List$V_i(t+1)$, thereby obtaining the number of clusters nClusters$(t+1)$ and the cluster to which each $v_i(t+1)$ belongs ListV_iClusters$(t+1)$, where $\text{abs}(v_i(t+1)) \in \text{List}V_i(t+1)$. The procedure for the binary db-scan operator is shown in Algorithm 2.

4.3. Transition Operator. The db-scan operator returns the number of clusters and a list with the cluster identifier to which each element belongs: $v_i^p \in \text{List}V_i(t+1)$. The purpose of the transition operator is to binarize the solutions generated by Mh and clustered by the binary db-scan operator. To perform the binarization, we must consider that the identifier $\text{Id}(J) \in \mathbb{Z}$ of the clusters will be assigned in the following manner: a value of 0 will be assigned to the cluster that has v_i with the lowest absolute value. Let $v_j \in J$ and $v_i \in I$ be elements of clusters J and I, respectively, and $\text{abs}(v_j) > \text{abs}(v_i)$; then, $\text{Id}(J) > \text{Id}(I)$. The value of Id will be consecutive integers and if $J \neq I \Longrightarrow \text{Id}(J) \neq \text{Id}(I)$. Finally, for the cases identified by db-scan as outliers, we have $(\text{Id}(\text{Ol}) = -1$, where $\text{Ol} \in$ outliers). Then, each cluster will be assigned a transition probability given by equation (5). In this equation, α corresponds to the initial transition coefficient, and β corresponds to the transition probability coefficient.

$$P_{tr}(J) = \alpha + \beta \frac{\text{Id}(J)}{T}, \tag{5}$$

where T is the total number of clusters not considering outliers.

Finally, to execute the binarization process, consider $x(t)$ as the position of a particle in iteration t. Let $x_i(t)$ be the value of the dimension i for the particle $x(t)$, and let $v_i^x(t+1)$ be the velocity of the particle $x(t)$ in the i

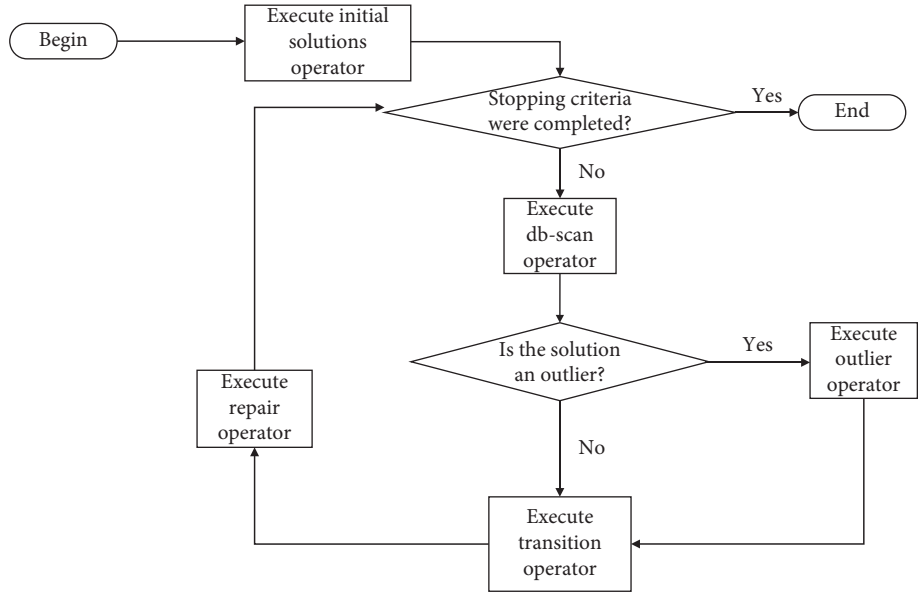


Figure 1: A general flow chart of the binary db-scan algorithm.

(1) **Function** Initiation ()
(2) **Input**
(3) **Output** Solution S
(4) $S_{\text{ini}} \longleftarrow$ SelecRandomColumn ()
(5) **while** All row are not covered **do**
(6) $\quad S_{\text{ini}}$.append (Heuristic (S_{ini}))
(7) **end while**
(8) $S \longleftarrow S_{\text{ini}}$
(9) **return** S

Algorithm 1: Initialization operator.

(1) **Function** BinaryDbscan (Mh, List$P(t)$)
(2) **Input** Mh, List$P(t)$
(3) **Output** nClusters$(t+1)$, ListV_iClusters$(t+1)$
(4) List$P(t+1) \longleftarrow$ applyMh (Mh (List$P(t)$))
(5) List$V(t+1) \longleftarrow$ getVelocityList (List$P(t)$, List$P(t+1)$)
(6) List$V_i(t+1) \longleftarrow$ getClusterList (List$V(t+1)$)
(7) nClusters$(t+1)$, ListV_iClusters$(t+1) \longleftarrow$ applyDbscan (List$V_i(t+1)$)
(8) **return** nClusters$(t+1)$, ListV_iClusters$(t+1)$

Algorithm 2: Binary db-scan operator.

dimension to transform $x(t)$ from iteration t to iteration $t+1$. Additionally, consider that $v_i^x(t+1) \in J$, where J is one of the clusters identified by the binary db-scan operator. Then, we use equation (6) to generate the binary position of the particles in iteration $t+1$.

$$x_i(t+1) := \begin{cases} \widehat{x}_i(t), & \text{if rand} < P_{tr}(J) \text{ where } v_i^x(t+1) \in J, \\ x_i(t), & \text{otherwise.} \end{cases} \tag{6}$$

When $v_i^x(t+1) \in$ outliers, the procedure is as follows: From the complete list of outliers, the $v_i^x(t+1)$ are ordered, starting with the solution with the best fitness and proceeding to those with the worst performance. The top 20% of solutions in terms of fitness is chosen, and a transition value of α is applied. For the remaining elements, a transition value of $\alpha + \beta$ is applied. Finally, once the transition operator is applied, a repair operator is used, as described in Section 4.4 for solutions that do not satisfy some of the restrictions. The details of the transition operator are shown in Algorithm 3.

4.4. Repair Operator. The repair operator is executed after the execution of the transition operator. In the event that the

(1) **Function** Transition (List$P(t)$, List$X(t)$, nClusters$(t+1)$)
(2) **Input** List$P(t)$, ListV_iClusters$(t+1)$, nClusters$(t+1)$
(3) **Output** List Binary$P(t+1)$
(4) **for** $x_i(t)$, $v_i^x(t+1)$ in (List$P(t)$, List$V_i(t+1)$) **do**
(5) **if** $v_i^x(t+1)$ not in ouliers **then**
(6) $P_{tr}(x_i) \longleftarrow$ getTransitionProbabily (ListV_iClusters$(t+1)$, nClusters$(t+1)$) –equation (5)
(7) **else**
(8) $P_{tr}(x_i) \longleftarrow$ getOutlierTransitionProbabily (ListV_iClusters$(t+1)$, nClusters$(t+1)$)
(9) **end if**
(10) List Binary$P(t+1)$.append $(x_i(t+1)) \longleftarrow$ getBinaryPosition ($P_{tr}(x_i(t))$, ListV_iClusters$(t+1)$) –equation (6)
(11) **end for**
(12) **for** $x(t+1)$ in List Binary$P(t+1)$ **do**
(13) List Binary$P(t+1)[x(t+1)] \longleftarrow$ Repair $(x(t+1))$
(14) **end for**
(15) **return** List Binary$P(t+1)$

Algorithm 3: Transition algorithm.

coverage condition of the rows is not met, the repair operator uses the heuristic operator to add new columns. After all the rows are covered, we verify that there are no groups of columns that cover the same rows. The details of the repair operator are shown in Algorithm 4.

4.5. Heuristic Operator. When a solution needs to be started or repaired, a heuristic operator is used that selects a new element. As an input parameter, the operator considers the solution S_{in}, which needs to be completed. In the case of being a new solution, $S_{\text{in}} = \varnothing$. With the list of columns belonging to S_{in}, we obtain the set of rows R not covered by the solution. With the set of rows not covered and using equation (7), we obtain in line 4 the best 10 rows to be covered. With this list of rows (listRows) online 5, we obtain the list of the best columns according to the heuristic shown in equation (8). Finally, in line 6, we randomly obtain the column to incorporate. The details of the heuristic operator are shown in Algorithm 5.

$$\text{Weight row}(i) = \frac{1}{L_i}, \tag{7}$$

where L_i is the sum of all ones in row i.

$$\text{Weight column}(j) = \frac{c_j}{\left|R \cap M_j\right|}, \tag{8}$$

where M_j is the set of rows covered by Col$j$.

5. Results

To determine the contribution of the db-scan algorithm to the binarization process, three groups of experiments are performed. The first group compares the db-scan algorithm with two random operators, as detailed in Section 5.2. The second group considers comparing db-scan with the k-means clustering technique. The results are shown in Section 5.3, and the details of the k-means technique can be found in [1]. Finally, the third group is shown in Section 5.4 and compares the binarization performed by db-scan with the binarization using TFs. The latter is a technique widely used in the binarization of continuous algorithms. Additionally, in Section 5.1, we describe the methodology used to define the parameters of the utilized algorithms.

CS [5] and PSO [7] were the selected algorithms. These algorithms are chosen for three reasons. Both algorithms are quite simple to parameterize; thus, the study can focus on the binarization technique rather than the parameterization. On the other hand, both algorithms have satisfactorily resolved nonlinear optimization problems [17, 32, 64–66]. Finally, simplified theoretical convergence models for both PSO [39] and CS [67] have been developed.

For the evaluation of the db-scan algorithm, instances E, F, G, and H, which correspond to the most difficult instances from Beasley's OR library, were used. For the execution of the instances, we used a PC with Windows 10 and an Intel Core i7-8550U processor with 16 GB of RAM. The algorithm was programmed in Python 3.7. To perform the statistical analysis in this study, the nonparametric Wilcoxon signed-rank test and violin charts were used. The analysis was performed by comparing the dispersion, median, and interquartile ranges of the distributions.

5.1. Parameter Settings. To obtain the parameters necessary to generate the binary algorithms db-scan-PSO and db-scan-CS, the methodology proposed in [1, 2] was selected. This methodology uses 4 measures defined in equations (9) to (12) to determine the best configuration. To be able to compare the different configurations, there are 4 measures, which are located on a radar chart, and the area under the curve is calculated for each configuration. The configuration with the largest area is selected.

(1) The percentage deviation of the best value obtained in the ten executions compared with the best known value (see equation (9))

$$b\text{Solution} = 1 - \frac{\text{KnownBestValue} - \text{BestValue}}{\text{KnownBestValue}}. \tag{9}$$

```
(1) Function Repair (x(t + 1))
(2) Input Input solution x(t + 1)
(3) Output The repaired solution x_rep(t + 1)
(4) while needRepair (x(t + 1)) = = True do
(5)     x(t + 1).append (Heuristic (x(t + 1)))
(6) end while
(7) x_rep(t + 1) ⟵ deleteRepeatedItem (x(t + 1))
(8) return x_rep(t + 1)
```

Algorithm 4: Repair algorithm.

```
(1) Function Heuristic (S_in)
(2) Input Input solution S_in
(3) Output The new column C_out
(4) listRows ⟵ getBestRows (S_in, N = 10)
(5) listcolumsnOut ⟵ getBestColumns (ListRows, M = 5)
(6) columnOut ⟵ getColumn (listcolumnsOut)
(7) return columnOut
```

Algorithm 5: Heuristic operator.

(2) The percentage deviation of the worst value obtained in the ten executions compared with the best known value (see equation (10))

$$w\text{Solution} = 1 - \frac{\text{KnownBestValue} - \text{WorstValue}}{\text{KnownBestValue}}. \quad (10)$$

(3) The percentage deviation of the average value obtained in the ten executions compared with the best known value (see equation (11))

$$a\text{Solution} = 1 - \frac{\text{KnownBestValue} - \text{AverageValue}}{\text{KnownBestValue}}. \quad (11)$$

(4) The convergence time for the best value in each experiment normalized according to equation (12)

$$n\text{Time} = 1 - \frac{\text{convergenceTime} - \text{minTime}}{\text{maxTime} - \text{minTime}}. \quad (12)$$

For PSO, the coefficients c_1 and c_2 are set to 2. ω is linearly decreased from 0.9 to 0.4. For the parameters used by db-scan, the minimum number of neighbors (min Pts) is estimated as a percentage of the number of particles (N). Specifically, if N = 50 and min Pts = 10, then the minimum number of neighbors for the point to be considered within a cluster is 5. To select the parameters, problems $E.1$, $F.1$, $G.1$, and $H.1$ were chosen. The parameter settings are shown in Tables 1 and 2. In both tables, the column labeled Value represents the selected value, and the column labeled Range corresponds to the set of scanned values.

5.2. Contribution of Db-Scan Binary Operator. This section attempts to understand the contribution of the db-scan operator when compared with two random operators. The random operator models the situation whereby the transition probability does not depend on the velocity of the particle, unlike the db-scan operator, where the velocity strongly influences the cluster in which it will be located. Two random operators were considered. The first case (naive) is whereby each point has the same probability of transition and therefore is independent of the velocity. In the experiment, two conditions were considered for the random operator. First, the operator N random-0.25 has a fixed probability of 0.25; second, the operator N random-0.5 uses a fixed transition probability of 0.5. To make the comparison, CS was used. The second random operator, C random-5, additionally includes the concept of clusters. In this second operator, 5 clusters are defined, where 5 corresponds to, on average, the clusters obtained by db-scan when executing the different instances. Subsequent to each cluster, a transition probability of the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ is assigned without repetitions. Finally, each particle randomly assigns a cluster.

The results obtained using the N random operator are shown in Table 3 and Figure 2. When we analyzed the best and average indicators shown in the table, the superiority of the results obtained by db-scan over those obtained by the N random-0.5 and N random-0.25 operators was observed. This difference is consistent in all instances. The Wilcoxon test indicates that the difference is significant. When analyzing the violin charts, we see that the dispersion, interquartile range, and median are substantially more robust when using the db-scan operator. This experiment is a strong indicator that, in the binarization process, i.e., the assignment of a transition probability to a particle, it is critical to consider the behavior of the particle in the search space. This allows us to obtain better behaving methods.

For the C random operator, the results are shown in Table 4 and Figure 3. In this experiment, the PSO and CS algorithms were used. When we analyzed the results of the

Table 1: Parameter setting for PSO Algorithm.

Parameters	Description	Value	Range
α	Initial transition coefficient	0.1	[0.08, 0.1, 0.12]
β	Transition probability coefficient	0.6	[0.5, 0.6, 0.7]
N	Number of particles	50	[30, 40, 50]
ϵ	ϵ db-scan parameter	0.4	[0.3, 0.4, 0.5]
minPts	Point db-scan parameter	10%	[10, 12, 14]
Iteration number	Maximum iterations	800	[600, 700, 800]

Table 2: Parameter setting for CS algorithm.

Parameters	Description	Value	Range
α	Transition probability coefficient	0.1	[0.08, 0.1, 0.12]
β	Transition probability coefficient	0.5	[0.5, 0.6, 0.7]
N	Number of particles	50	[30, 40, 50]
ϵ	ϵ db-scan parameter	0.4	[0.3, 0.4, 0.5]
minPts	Point db-scan parameter	12%	[10, 12, 14]
γ	Step length	0.01	[0.009, 0.01, 0.011]
κ	Levy distribution parameter	1.5	[1.4, 1.5, 1.6]
Iteration number	Maximum iterations	800	[600, 700, 800]

Table 3: Comparison between db-scan and Nrandom operators.

Instance	Best known	db − scan-CS			Nrandom-0.25-CS			Nrandom-0.5-CS		
		Best	Avg	Time (s)	Best	Avg	Time (s)	Best	Avg	Time (s)
$E.1$	29	29	29.0	12.1	29	30.4	7.7	29	30.7	8.2
$E.2$	30	30	30.2	11.8	31	32.6	8.1	30	32.4	7.8
$E.3$	27	27	27.3	12.9	28	29.4	6.6	28	29.8	8.1
$E.4$	28	28	28.0	11.5	29	30.3	6.5	28	30.7	8.3
$E.5$	28	28	28.0	11.4	29	29.8	6.7	28	30.1	8.2
$F.1$	14	14	14.0	12.7	15	16.1	9.1	15	16.9	14.1
$F.2$	15	15	15.2	13.1	16	17.8	8.7	16	18.1	15.3
$F.3$	14	14	14.1	12.6	15	15.4	9.3	15	15.5	14.8
$F.4$	14	14	14.0	12.9	15	16.2	9.4	15	16.2	14.9
$F.5$	13	13	13.2	13.2	14	15.7	8.9	14	15.9	14.1
$G.1$	176	176	177.1	73.1	183	187.4	54.6	184	189.1	60.3
$G.2$	154	156	156.6	72.6	162	167.1	57.3	161	166.3	61.2
$G.3$	166	168	168.4	70.3	174	179.4	58.6	173	178.4	59.7
$G.4$	168	169	169.7	68.9	173	177.2	56.6	174	178.2	60.5
$G.5$	168	168	168.2	72.1	172	176.7	54.1	171	177.8	58.1
$H.1$	63	64	64.8	65.3	68	72.3	52.7	68	73.1	54.9
$H.2$	63	63	63.6	68.1	69	73.1	55.3	68	73.5	53.1
$H.3$	59	60	60.9	69.7	64	68.4	57.2	64	67.9	58.9
$H.4$	58	59	59.2	70.3	63	66.3	56.6	62	67.1	60.4
$H.5$	55	55	55.2	69.3	61	64.2	55.3	60	64.9	59.1
Average	67.1	67.5	67.84	41.2	70.5	73.29	31.97	70.1	73.63	35.0
Wilcoxon p − value					1.03e−4	8.84e−5		5.20e−4	8.85e−5	

table, it is observed that db-scan has a better behavior than C random in both algorithms. When analyzing Figure 3, it is observed that the median, interquartile range, and dispersion measures obtain better results with the db-scan operator. Additionally, we should note that C random achieves a better performance than N random, which suggests that assigning random transition probabilities by groups is more appropriate than assigning them randomly.

5.3. K-Means Algorithm Comparison. K-means is another clustering technique that was used in [2] to binarize continuous swarm intelligence algorithms and applied to the knapsack problem. The objective of this section is to compare the behavior of the binarization used by db-scan with that used by k-means. The k-means technique, unlike db-scan, is necessary to define the number of clusters. On the other hand, the computational complexity of k-means once the number of clusters (k) and the dimension (d) of the points are fixed is $O(n^{dk+1}\log n)$, where n is the number of points to be clustered. The computational complexity of db-scan is $O(n\log n)$. In this experiment, the quality of the solutions and their execution times are compared. For the case of k-means, $k = 5$. In the case of db-scan, the number of

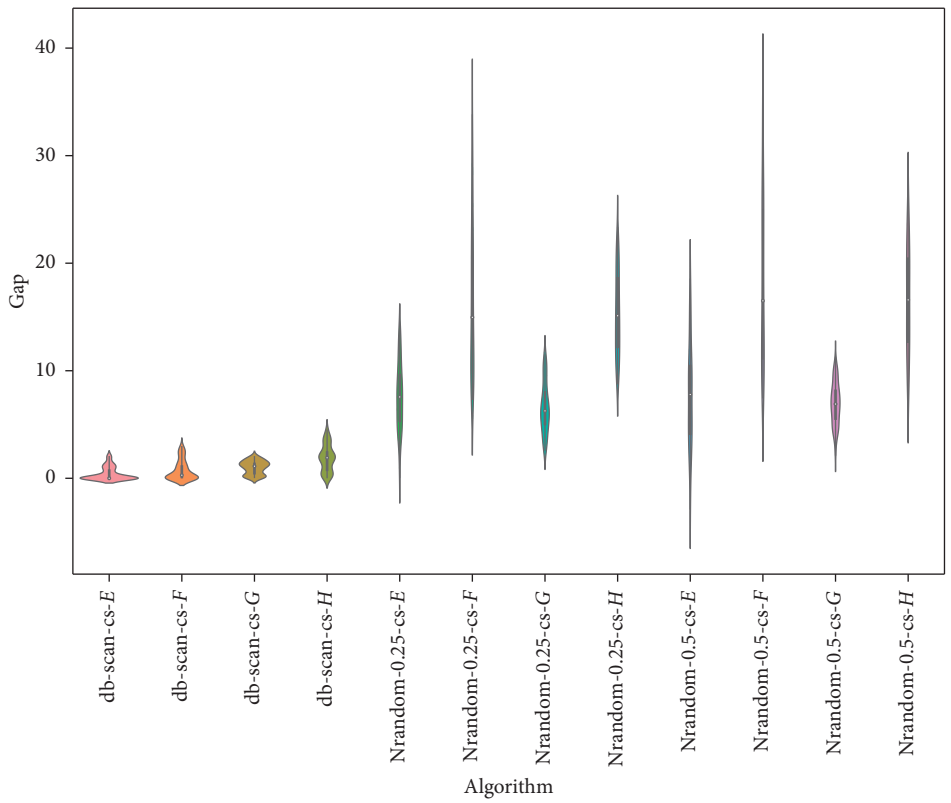


Figure 2: Gap comparison between db-scan and *N*random algorithms for the SCP dataset.

Table 4: Comparison between db-scan and *C*random operators.

Instance	Best known	Crandom-5.PSO			db − scan-PSO			Crandom-5.CS			db − scan-CS		
		Best	Avg	Time	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time (*s*)
E.1	29	29	29.9	11.1	29	29.0	13.4	29	29.8	10.6	29	29.0	12.1
E.2	30	30	31.1	10.8	30	30.1	13.7	31	31.6	10.9	30	30.2	11.8
E.3	27	28	28.7	10.6	27	27.5	14.1	28	28.5	9.8	27	27.3	12.9
E.4	28	29	29.9	10.1	28	28.1	12.9	29	29.6	10.2	28	28.0	11.5
E.5	28	28	28.7	10.5	28	28.3	13.2	28	28.4	10.4	28	28.0	11.4
F.1	14	15	15.5	10.9	14	14.1	12.8	15	15.7	11.3	14	14.0	12.7
F.2	15	16	16.8	11.5	15	15.4	13.5	16	16.8	12.1	15	15.2	13.1
F.3	14	14	14.9	11.9	14	14.4	13.7	15	15.9	10.9	14	14.1	12.6
F.4	14	15	15.8	12.1	14	14.1	13.1	15	15.7	11.2	14	14.0	12.9
F.5	13	14	14.7	11.4	13	13.4	13.4	14	15.1	11.4	13	13.2	13.2
G.1	176	180	183.9	68.2	176	176.8	81.3	181	184.2	67.2	176	177.1	73.1
G.2	154	160	163.8	69.1	156	156.8	77.4	160	164.1	64.3	156	156.6	72.6
G.3	166	171	174.6	68.7	168	168.9	79.8	172	175.3	65.1	168	168.4	70.3
G.4	168	172	175.1	68.4	169	170.1	78.1	172	174.9	66.3	169	169.7	68.9
G.5	168	173	176.4	67.1	169	169.6	81.2	172	175.8	64.8	168	168.2	72.1
H.1	63	68	70.6	65.8	64	64.5	74.2	68	70.4	61.4	64	64.8	65.3
H.2	63	68	71.2	67.2	64	64.3	73.2	68	71.7	59.7	63	63.6	68.1
H.3	59	63	66.1	68.1	60	60.4	72.1	62	65.4	62.3	60	60.9	69.7
H.4	58	63	65.9	65.7	59	59.8	76.5	63	66.1	61.8	59	59.2	70.3
H.5	55	58	61.5	63.2	55	55.2	74.6	59	62.3	60.2	55	55.2	69.3
Average	67.1	69.7	71.76	39.12	67.6	68.04	45.11	69.85	71.87	37.09	67.5	67.84	41.2
Wilcoxon p − value		3.65*e*−4	8.84*e*−5					1.58*e*−4	8.82*e*−5				

clusters is variable. For comparison, the same dataset as in the previous experiment is used. In Table 5, the results of the binarization for CS and PSO are shown using the *k*-means and db-scan operators. When we observe the best and average indicators, we see that their values are very similar for both the implementation with *k*-means and the implementation with db-scan. Moreover, when we use the Wilcoxon test, we see that the small differences are not significant.

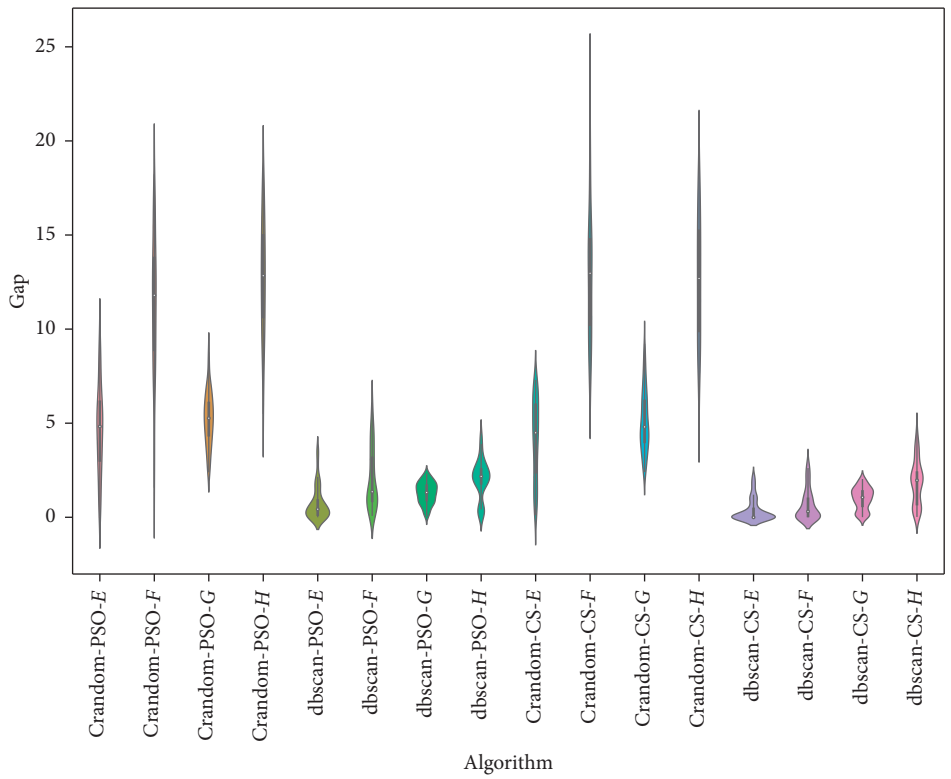


Figure 3: Gap comparison between db-scan and Crandom algorithms for the SCP dataset.

Table 5: Comparison between db-scan and k-means operators.

Instance	Best known	k-means.PSO			db − scan-PSO			k-means.CS			db − scan-CS		
		Best	Avg	Time	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time (s)
$E.1$	29	29	29.2	17.1	29	29.0	13.4	29	29.1	18.1	29	29.0	12.1
$E.2$	30	30	30.1	18.1	30	30.1	13.7	30	30.2	17.9	30	30.2	11.8
$E.3$	27	27	27.6	16.8	27	27.5	14.1	27	27.1	19.1	27	27.3	12.9
$E.4$	28	28	28.3	17.3	28	28.1	12.9	28	28.2	16.4	28	28.0	11.5
$E.5$	28	28	28.6	17.9	28	28.3	13.2	28	28.2	16.9	28	28.0	11.4
$F.1$	14	14	14.1	17.5	14	14.1	12.8	14	14.1	19.1	14	14.0	12.7
$F.2$	15	15	15.4	18.1	15	15.4	13.5	15	15.3	17.2	15	15.2	13.1
$F.3$	14	14	14.5	18.4	14	14.4	13.7	14	14.2	17.3	14	14.1	12.6
$F.4$	14	14	14.1	17.3	14	14.1	13.1	14	14.3	17.7	14	14.0	12.9
$F.5$	13	13	13.3	17.8	13	13.4	13.4	13	13.0	18.1	13	13.2	13.2
$G.1$	176	176	176.5	98.5	176	176.8	81.3	176	176.8	102.7	176	177.1	73.1
$G.2$	154	156	157.1	95.5	156	156.8	77.4	156	156.9	96.5	156	156.6	72.6
$G.3$	166	168	168.6	93.4	168	168.9	79.8	169	169.7	99.1	168	168.4	70.3
$G.4$	168	169	170.4	103.2	169	170.1	78.1	169	169.4	97.4	169	169.7	68.9
$G.5$	168	168	170.0	101.8	169	169.6	81.2	168	168.4	96.3	168	168.2	72.1
$H.1$	63	64	64.7	99.7	64	64.5	74.2	63	63.6	101.3	64	64.8	65.3
$H.2$	63	63	63.5	101.2	64	64.3	73.2	64	64.5	99.8	63	63.6	68.1
$H.3$	59	60	60.3	96.6	60	60.4	72.1	60	60.8	97.4	60	60.9	69.7
$H.4$	58	59	59.7	97.3	59	59.8	76.5	59	59.7	99.5	59	59.2	70.3
$H.5$	55	55	55.3	98.2	55	55.2	74.6	55	55.4	95.1	55	55.2	69.3
Average	67.1	67.5	68.07	58.09	67.6	68.04	45.11	67.55	67.94	58.14	67.5	67.84	41.2
Wilcoxon p − value		0.16	0.42					0.56	0.21				

However, when we analyze the execution times, we see that db-scan improves the times obtained by k-means. When we compare the interquartile range and the dispersion shown in Figure 4, we see that the results are very similar. Considering that k-means handles a fixed number of clusters and given that, in the case of db-scan, this can be variable, the quality of the solutions is not affected significantly.

5.4. Transfer Function Comparison. In this section, we detail the experiments that allow us to evaluate the behavior of

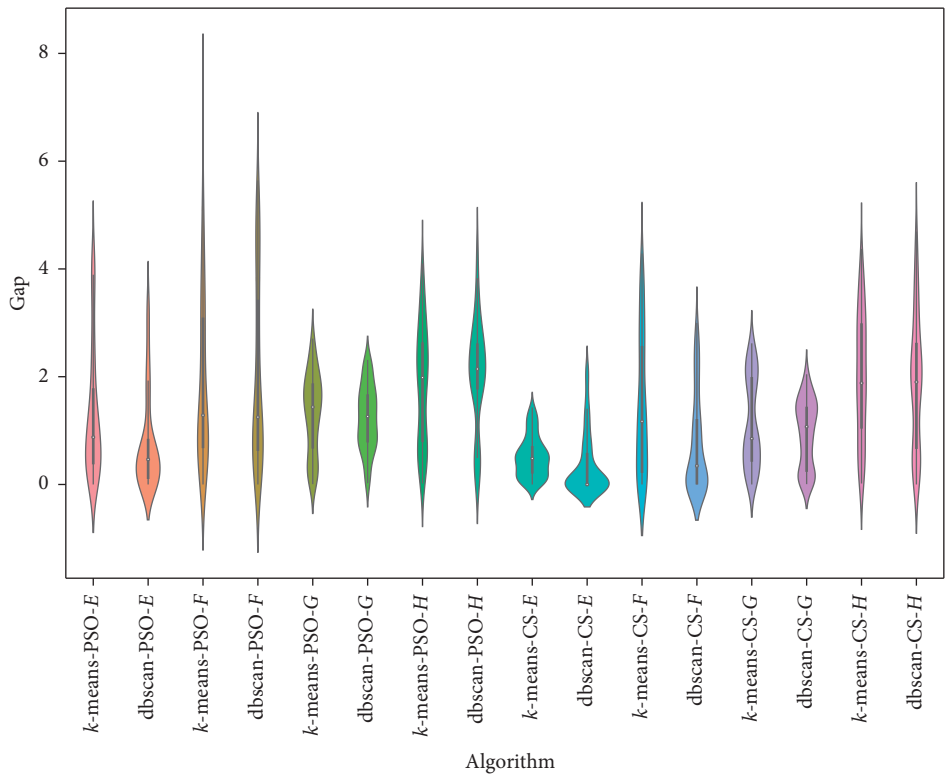


Figure 4: Gap comparison between db-scan and k-means algorithms for the SCP dataset.

binarization using db-scan with respect to the TF. The TF is a general binarization mechanism that, instead of using the cluster concept to assign a transition probability, uses functions that map $\mathbb{R}^n$ in the space $(0,1)^n$. Usually, two families of functions are used: v-shape $((e^{\tau|x_i^d|}-1)/(e^{\tau|x_i^d|}+1))$ and s-shape $(1/(e^{-\tau x_i^d}+1))$ functions. For more details about TFs, we recommended [32].

In our case, we used the v-shape function with the parameter $\tau = 2.5$ for both the CS and PSO algorithms. The methodology used to determine the family and the parameter τ corresponds to the same detailed in Section 5.1. The results are shown in Table 6 and Figure 5. For the TFs, 2000 iterations were considered for the experiment. From Table 6, it is observed when analyzing the best and average indicators that the binary algorithms obtained through db-scan achieve better performance than those obtained with the TF. When performing the Wilcoxon test, the obtained differences are significant. When we look at Figure 5, we see that the dispersion and interquartile ranges are considerably improved when using db-scan. We should note that the TF assigns a particular value of the transition probability to each solution based on a function, unlike db-scan, uses assignment by groups of solutions.

6. Real-World Application

The crew scheduling problem (CSP) is related to building the work schedules of crews necessary to cover a planned timetable. The CSP is studied in operations research and is usually related to the airline industry, transit companies, and railways, among others. In this section, we are interested in using the binarizations obtained from applying the db-scan algorithm to the CSP.

The CSP, due to its difficulty, needs to be decomposed in several stages, where each stage has a given computational complexity. The literature contains variations of the CSP. These variations consider integration with other problems or the inclusion of new restrictions. For example, in the CPS in [68], attendance rates were studied. A CSP integration with the vehicle scheduling problem was developed in [69]. In [70], an application of the CSP with fairness preferences was explored. The crew pairing and fleet assignment problems were studied in [71].

The CSP starts with a timetable of services that must be executed with a certain frequency. On the other hand, the service needs to be executed in a certain time window. A service consists of a sequence of trips, where a trip has the following attributes: a start time, an end time, a departure station, an arrival station, and a crew that delivers the service. In terms of the above attributes, each trip is assigned a cost. When we consider a period of time and a crew, a roster must be generated. Then, the CSP consists of finding a subset of rosters that covers all trips at the minimum cost. The problem can be divided into two phases. The first phase corresponds to the generation of a pairing. A pairing is defined as a set of trips that are assigned to a single crew in a short period of time. In this pairing phase, a large number of pairings is generated that satisfy the constraints of the problem. A match must start and end at the same depot, and a cost must be associated. The second phase corresponds to the pairing optimization. At this stage, a selection is made of the best subset of

Table 6: Comparison between *db*-scan and *TF* operators.

Instance	Best known	TF-PSO			db − scan-PSO			TF-CS			db − scan-CS		
		Best	Avg	Time	Best	Avg	Time	Best	Avg	Time	Best	Avg	Time (*s*)
E.1	29	29	30.8	47.4	29	29.0	13.4	29	29.7	37.2	29	29.0	12.1
E.2	30	30	30.7	41.5	30	30.1	13.7	30	31.3	36.5	30	30.2	11.8
E.3	27	28	30.1	39.8	27	27.5	14.1	28	29.2	38.3	27	27.3	12.9
E.4	28	29	29.8	45.7	28	28.1	12.9	29	29.7	37.7	28	28.0	11.5
E.5	28	29	29.6	44.2	28	28.3	13.2	29	30.1	34.1	28	28.0	11.4
F.1	14	14	14.9	46.1	14	14.1	12.8	14	14.9	39.5	14	14.0	12.7
F.2	15	15	15.1	49.2	15	15.4	13.5	15	15.2	43.2	15	15.2	13.1
F.3	14	14	14.6	49.3	14	14.4	13.7	14	14.9	47.1	14	14.1	12.6
F.4	14	14	14.7	45.2	14	14.1	13.1	14	14.8	46.3	14	14.0	12.9
F.5	13	14	14.9	41.4	13	13.4	13.4	14	14.7	44.1	13	13.2	13.2
G.1	176	177	178.4	286.4	176	176.8	81.3	177	177.9	324.4	176	177.1	73.1
G.2	154	157	158.3	301.3	156	156.8	77.4	158	159.1	351.3	156	156.6	72.6
G.3	166	169	170.2	314.5	168	168.9	79.8	170	171.4	346.7	168	168.4	70.3
G.4	168	169	170.7	322.1	169	170.1	78.1	169	171.2	358.1	169	169.7	68.9
G.5	168	169	170.5	303.1	169	169.6	81.2	169	169.9	354.2	168	168.2	72.1
H.1	63	64	65.1	265.2	64	64.5	74.2	64	65.1	286.8	64	64.8	65.3
H.2	63	64	65.3	246.4	64	64.3	73.2	64	65.7	279.4	63	63.6	68.1
H.3	59	60	61.8	298.1	60	60.4	72.1	61	62.1	277.2	60	60.9	69.7
H.4	58	59	60.3	293.7	59	59.8	76.5	60	60.6	298.1	59	59.2	70.3
H.5	55	56	57.4	300.1	55	55.2	74.6	56	57.2	305.2	55	55.2	69.3
Average	67.1	68.0	69.16	169.03	67.6	68.04	45.11	68.2	69.23	179.27	67.5	67.84	41.2
Wilcoxon p − value		$4.6e-3$	$1.2e-4$					$1.05e-3$	$1.3e-4$				

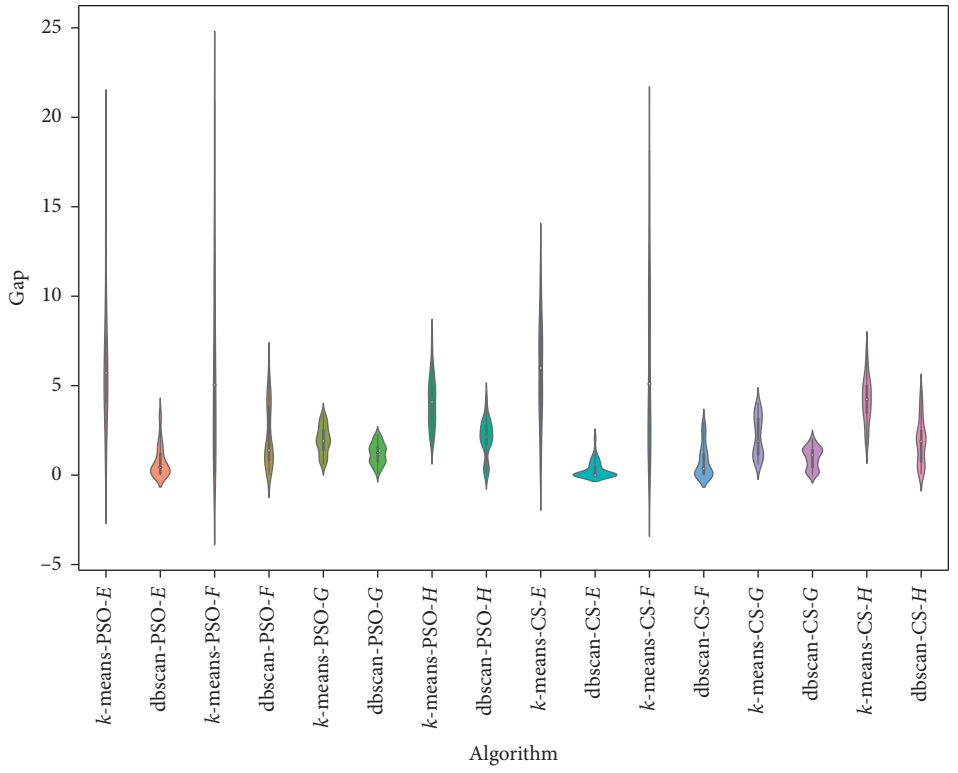


Figure 5: Gap comparison between db-scan and TF algorithms for the SCP dataset.

all generated pairings to ensure that all trips are covered at a minimum cost. The modeling of this phase follows an approach based on the solution to set covering or set partitioning problems. In this work, we use a dataset on which the pairs were generated; therefore, we focus our efforts on performing the pairing optimization phase. To verify our algorithm, 7 datasets associated with real-world crew scheduling problems were used. These datasets come

Table 7: Railway crew scheduling problems.

Instance	Row	Col	Density (%)	Best known	db − scan-PSO (Best)	db − scan-PSO (Avg)	Time (*s*)	db − scan-CS (Best)	db − scan-CS (Avg)	Time (*s*)
Rail507	507	63009	1.2	174	175	179.4	135.1	174	176.8	127.1
Rail516	516	47311	1.3	182	184	185.9	146.7	183	185.1	151.3
Rail582	582	55515	1.2	211	214	216.3	202.8	214	215.9	198.6
Rail2536	2536	1081841	0.4	690	694	698.1	1225.1	693	698.2	1202.1
Rail2586	2586	920683	0.4	944	948	952.7	1201.5	949	951.2	1301.8
Rail4284	4284	1092610	0.2	1062	1067	1070.9	3154.1	1067	1070.4	3015.3
Rail4872	4872	968672	0.2	1527	1533	1542.8	3700.5	1535	1544.2	3682.1
Average				684.29	687.85	692.3	1395.11	687.86	691.69	1382.61

from an application from the Italian railways and have been provided by Ceria et al. [72]. Table 7 shows the datasets and their results. When we analyzed the table, we observed that although the problems were larger than the previous problems, the performances of the db − scan—PSO and db − scan— binarizations were adequate. In the case of db − scan—PSO, the gap for the best value was 0.52%, and, on average, it was 1.17%. For db − scan—CS, the gap for the best value was 0.52%, and, on average, it was 1.08%.

7. Conclusions

In this article, an algorithm was proposed that uses the db-scan technique with the goal of binarizing continuous swarm intelligence metaheuristics. To evaluate the proposed algorithm, as a first step, two random operators were designed with the objective of identifying the contribution of db-scan to the binarization process. Subsequently, the proposed db-scan algorithm was compared with two binarization techniques. The first technique is based on the clustering concept and uses the *k*-means technique, where the number of clusters is fixed. The second technique uses TFs as a binarization mechanism. In the comparison with the binarization technique that uses the concept of i-means, the results were very similar. Those results were confirmed with the Wilcoxon test, which showed no significant differences between the two techniques. However, we must emphasize that the execution times of db-scan were shorter than those of *k*-means. One point to consider is that the different methods of generating the clusters do not affect the quality of the solutions. In the case of k-means, a fixed number of clusters, generated based on the proximity of the points, is defined. For db-scan, the number of clusters is variable and is generated based on the proximity and density of points. In comparison with the TFs, we observed that there is a significant difference in favor of db-scan. This suggests that it is more efficient in the binarization process to assign transition probabilities to groups than to assign them individually. The application of machine learning to metaheuristic algorithms is a line of research that has several aspects. We see that machine learning techniques can learn and help to understand under which conditions a metaheuristic algorithm performs efficiently. However, these techniques can be applied to other operators, such as perturbation operators, when a metaheuristic algorithm is trapped in a local minimum, and operators that control the population of a swarm intelligence algorithm to improve the intensification and diversification properties. Additionally, appealing to the no-free-lunch theorem, it would be interesting to evaluate these algorithms when including machine learning tuning applied to other combinatorial problems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

José García was supported by the grant CONICYT/FONDECYT/INICIACION/11180056, Broderick Crawford was supported by the grant CONICYT/FONDECYT/REGULAR/1171243, and Ricardo Soto was supported by the grant CONICYT/FONDECYT/REGULAR/1190129.

References

[1] J. García, B. Crawford, R. Soto, and G. Astorga, "A clustering algorithm applied to the binarization of swarm intelligence continuous metaheuristics," *Swarm and Evolutionary Computation*, vol. 44, pp. 646–664, 2019.

[2] J. García, B. Crawford, R. Soto, C. Castro, and F. Paredes, "A k-means binarization framework applied to multidimensional knapsack problem," *Applied Intelligence*, vol. 48, no. 2, pp. 357–380, 2018.

[3] R. Munoz, R. Olivares, C. Taramasco et al., "Using black hole algorithm to improve eeg-based emotion recognition," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 3050214, 21 pages, 2018.

[4] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation*, vol. 16, pp. 1–18, 2014.

[5] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing, NaBIC 2009*, pp. 210–214, IEEE, Coimbatore, India, December 2009.

[6] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 965–997, 2014.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Micro Machine and Human Science MHS'95*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[8] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Erciyes University, Engineering Faculty," Computer Engineering Department, Kayseri, Turkey, 2005.

[9] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[10] D. E. Golberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addion Wesley, Boston, MA, USA, 1989.

[11] M. Caserta and S. Voß, "Matheuristics: hybridizing metaheuristics and mathematical programming," in *Metaheuristics: Intelligent Problem Solving*, Springer, Berlin, 2009.

[12] E.-G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *Annals of Operations Research*, vol. 240, no. 1, pp. 171–215, 2016.

[13] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, "A review of simheuristics: extending metaheuristics to deal with stochastic combinatorial optimization problems," *Operations Research Perspectives*, vol. 2, pp. 62–72, 2015.

[14] L. Calvet, J. de Armas, D. Masip, and A. A. Juan, "Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs," *Open Mathematics*, vol. 15, no. 1, pp. 261–280, 2017.

[15] E. Balas and M. W. Padberg, "Set partitioning: a survey," *SIAM Review*, vol. 18, no. 4, pp. 710–760, 1976.

[16] J. Borneman, M. Chrobak, G. Della Vedova, A. Figueroa, and T. Jiang, "Probe selection algorithms with applications in the analysis of microbial communities," *Bioinformatics*, vol. 17, no. 1, pp. S39–S48, 2001.

[17] J. García, F. Altimiras, A. Peña, G. Astorga, and O. Peredo, "A binary cuckoo search big data algorithm applied to large-scale crew scheduling problems," *Complexity*, vol. 2018, Article ID 8395193, 15 pages, 2018.

[18] E. Boros, P. L. Hammer, T. Ibaraki, and A. Kogan, "Logical analysis of numerical data," *Mathematical Programming*, vol. 79, no. 1–3, pp. 163–190, 1997.

[19] M. R. Gary and D. S. Johnson, "Computers and intractability," in *A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.

[20] E. Balas and M. C. Carrera, "A dynamic subgradient-based branch-and-bound procedure for set covering," *Operations Research*, vol. 44, no. 6, pp. 875–890, 1996.

[21] J. E. Beasley, "An algorithm for set covering problem," *European Journal of Operational Research*, vol. 31, no. 1, pp. 85–93, 1987.

[22] J. E. Beasley, "A Lagrangian heuristic for set-covering problems," *Naval Research Logistics*, vol. 37, no. 1, pp. 151–164, 1990.

[23] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.

[24] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem," *Operations Research*, vol. 47, no. 5, pp. 730–743, 1999.

[25] B. Yelbay, Ş. İ. Birbil, and K. Bülbül, "The set covering problem revisited: an empirical study of the value of dual information," *Journal of Industrial & Management Optimization*, vol. 11, no. 2, pp. 575–594, 2015.

[26] M. J. Brusco, L. W. Jacobs, and G. M. Thompson, "A morphing procedure to supplement a simulated annealing heuristic for cost-andcoverage-correlated set-covering problems," *Annals of Operations Research*, vol. 86, pp. 611–627, 1999.

[27] C. Valenzuela, B. Crawford, R. Soto, E. Monfroy, and F. Paredes, "A 2-level metaheuristic for the set covering problem," *International Journal of Computers Communications & Control*, vol. 7, no. 2, pp. 377–387, 2014.

[28] B. Crawford, R. Soto, N. Berríos et al., "A binary cat swarm optimization algorithm for the non-unicost set covering problem," *Mathematical Problems in Engineering*, vol. 2015, Article ID 578541, 8 pages, 2015.

[29] J. García, B. Crawford, R. Soto, and G. Astorga, "A percentile transition ranking algorithm applied to binarization of continuous swarm intelligence metaheuristics," in *Proceedings of the International Conference on Soft Computing and Data Mining*, pp. 3–13, Springer, Senai, Malaysia, January 2018.

[30] J. García, B. Crawford, R. Soto, and P. García, "A multi dynamic binary black hole algorithm applied to set covering problem," in *Proceedings of the International Conference on Harmony Search Algorithm*, pp. 42–51, Springer, Bilbao, Spain, February 2017.

[31] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*, Wiley, New York, NY, USA, 1972.

[32] B. Crawford, R. Soto, G. Astorga, J. García, C. Castro, and F. Paredes, "Putting continuous metaheuristics to work in binary search spaces," *Complexity*, vol. 2017, Article ID 8404231, 19 pages, 2017.

[33] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Computational Cybernetics and Simulation*, pp. 4104–4108, IEEE, Orlando, FL, USA, October 1997.

[34] Y. Yang, Y. Mao, P. Yang, and Y. Jiang, "The unit commitment problem based on an improved firefly and particle swarm optimization hybrid algorithm," in *Proceedings of the Chinese Automation Congress (CAC)*, pp. 718–722, IEEE, Changsha, Hunan, China, November 2013.

[35] W. Liu, L. Liu, and D. Cartes, "Angle Modulated Particle Swarm Optimization Based Defensive Islanding of Large Scale Power Systems," in *Proceedings of the 2007 IEEE Power Engineering Society Conference and Exposition in Africa-PowerAfrica*, pp. 1–8, Johannesburg, South Africa, 2007.

[36] D. Zakaria and D. A. Cartes, "Angle Modulated Particle Swarm Optimization Based Defensive Islanding of Large Scale Power Systems," *Computer Science and Its Applications*, vol. 456, pp. 3–14, 2015.

[37] D. Swagatam, M. Rohan, and K. Rupam, "Multi-user detection in multi-carrier cdma wireless broadband system using a binary adaptive differential evolution algorithm," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 1245–1252, GECCO, New York, NY, USA, July 2013.

[38] G. Zhang, "Quantum-inspired evolutionary algorithms: a survey and empirical study," *Journal of Heuristics*, vol. 17, no. 3, pp. 303–351, 2011.

[39] B. J. Leonard, A. P. Engelbrecht, and C. W. Cleghorn, "Critical considerations on angle modulated particle swarm optimisers," *Swarm Intelligence*, vol. 9, no. 4, pp. 291–314, 2015.

[40] S. Saremi, S. Mirjalili, and A. Lewis, "How important is a transfer function in discrete heuristic algorithms," *Neural Computing and Applications*, vol. 26, no. 3, pp. 625–640, 2015.

[41] G. Pampara, *Angle modulated population based algorithms to solve binary problems*, PhD thesis, University of Pretoria, Pretoria, South Affrica, 2012.

[42] E. Chen, J. Li, and X. Liu, "In search of the essential binary discrete particle swarm," *Applied Soft Computing*, vol. 11, no. 3, pp. 3260–3269, 2011.

[43] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, Berlin, Germany, 2006.

[44] J. García, C. Pope, and F. Altimiras, "A distributed-means segmentation algorithm applied to lobesia botrana recognition," *Complexity*, vol. 2017, Article ID 5137317, 14 pages, 2017.

[45] H. Kaur, J. Virmani, Kriti, and S. Thakur, "Chapter 10—a genetic algorithm-based metaheuristic approach to customize a computer-aided classification system for enhanced screen film mammograms," in *U-Healthcare Monitoring Systems, Advances in Ubiquitous Sensing Applications for Healthcare*, N. Dey, A. S. Ashour, S. J. Fong, and S. Borra, Eds., pp. 217–259, Academic Press, Cambridge, MA, USA, 2019.

[46] H. Faris, M. A. Hassonah, A. M. Al-Zoubi, S. Mirjalili, and I. Aljarah, "A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture," *Neural Computing and Applications*, vol. 30, no. 8, pp. 2355–2369, 2018.

[47] H. Faris, I. Aljarah, and S. Mirjalili, "Improved monarch butterfly optimization for unconstrained global search and neural network training," *Applied Intelligence*, vol. 48, no. 2, pp. 445–464, 2018.

[48] J.-S. Chou and J. P. P. Thedja, "Metaheuristic optimization within machine learning-based classification system for early warnings related to geotechnical problems," *Automation in Construction*, vol. 68, pp. 65–80, 2016.

[49] A.-D. Pham, N.-D. Hoang, and Q.-T. Nguyen, "Predicting compressive strength of high-performance concrete using metaheuristic-optimized least squares support vector regression," *Journal of Computing in Civil Engineering*, vol. 30, no. 3, article 06015002, 2015.

[50] M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru, "Integrating metaheuristics and artificial neural networks for improved stock price prediction," *Expert Systems with Applications*, vol. 44, pp. 320–331, 2016.

[51] J.-S. Chou and T.-K. Nguyen, "Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3132–3142, 2018.

[52] J.-S. Chou and A.-D. Pham, "Nature-inspired metaheuristic optimization in least squares support vector regression for obtaining bridge scour information," *Information Sciences*, vol. 399, pp. 64–80, 2017.

[53] R. J. Kuo, T. C. Lin, F. E. Zulvia, and C. Y. Tsai, "A hybrid metaheuristic and kernel intuitionistic fuzzy c-means algorithm for cluster analysis," *Applied Soft Computing*, vol. 67, pp. 299–308, 2018.

[54] P. S. Mann and S. Singh, "Energy efficient clustering protocol based on improved metaheuristic in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 83, pp. 40–52, 2017.

[55] R. d. A. Rosa, A. M. Machado, G. M. Ribeiro, and G. R. Mauri, "A mathematical model and a clustering search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas," *Computers & Industrial Engineering*, vol. 101, pp. 303–312, 2016.

[56] N. Vecek, M. Mernik, B. Filipic, and M. Xrepinsek, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms," *Information Sciences*, vol. 372, pp. 446–469, 2016.

[57] J. Ries and P. Beullens, "A semi-automated design of instance-based fuzzy parameter tuning for metaheuristics based on decision tree induction," *Journal of the Operational Research Society*, vol. 66, no. 5, pp. 782–793, 2015.

[58] Z.-q. Li, H.-l. Zhang, J.-h. Zheng, M.-j. Dong, Y.-f. Xie, and Z.-j. Tian, "Heuristic evolutionary approach for weighted circles layout," in *Proceedings of the International Symposium on Information and Automation*, pp. 324–331, Springer, Guangzhou, China, June 2010.

[59] T. Yalcinoz and H. Altun, "Power economic dispatch using a hybrid genetic algorithm," *IEEE Power Engineering Review*, vol. 21, no. 3, pp. 59-60, 2001.

[60] A. D. de León, E. Lalla-Ruiz, B. Melián-Batista, and J. Marcos Moreno-Vega, "A machine learning-based system for berth scheduling at bulk terminals," *Expert Systems with Applications*, vol. 87, pp. 170–182, 2017.

[61] S. Asta, E. Ö.tae, and T. Curtois, "A tensor based hyper-heuristic for nurse rostering," *Knowledge-based Systems*, vol. 98, pp. 185–199, 2016.

[62] S. Martin, D. Ouelhadj, P. Beullens, E. Ozcan, A. A. Juan, and E. K. Burke, "A multi-agent based cooperative approach to scheduling and routing," *European Journal of Operational Research*, vol. 254, no. 1, pp. 169–178, 2016.

[63] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Kdd Knowledge Discovery and Data Mining*, vol. 96, pp. 226–231, 1996.

[64] M. Jiang, J. Luo, D. Jiang, J. Xiong, H. Song, and J. Shen, "A cuckoo search-support vector machine model for predicting dynamic measurement errors of sensors," *IEEE Access*, vol. 4, pp. 5030–5037, 2016.

[65] Y. Zhou, N. Wang, and W. Xiang, "Clustering hierarchy protocol in wireless sensor networks using an improved pso algorithm," *IEEE Access*, vol. 5, pp. 2241–2253, 2017.

[66] C. Mao, R. Lin, C. Xu, and Q. He, "Towards a trust prediction framework for cloud services based on pso-driven neural network," *IEEE Access*, vol. 5, pp. 2187–2199, 2017.

[67] X.-S. He, F. Wang, Y. Wang, and X.-S. Yang, "Global convergence analysis of cuckoo search using markov theory," in *Nature-Inspired Algorithms and Applied Optimization*, Springer, Berlin, Germany, 2018.

[68] K. Hoffmann and U. Buscher, "Valid inequalities for the arc flow formulation of the railway crew scheduling problem with attendance rates," *Computers & Industrial Engineering*, vol. 127, pp. 1143–1152, 2019.

[69] M. Horváth and T. Kis, "Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price," *Central European Journal of Operations Research*, vol. 27, no. 1, pp. 39–67, 2019.

[70] S. Jütte, D. Müller, and U. W. Thonemann, "Optimizing railway crew schedules with fairness preferences," *Journal of Scheduling*, vol. 20, no. 1, pp. 43–55, 2017.

[71] O. Ö. Özener, M. Örmeci Matoğlu, G. Erdoğan, M. Haouari, and H. Sözer, "Solving a large-scale integrated fleet assignment and crew pairing problem," *Annals of Operations Research*, vol. 253, no. 1, pp. 477–500, 2017.

[72] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," *Mathematical Programming*, vol. 81, no. 2, pp. 215–228, 1998.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 4589060, 19 pages
https://doi.org/10.1155/2019/4589060

Research Article

A Neural Network-Inspired Approach for Improved and True Movie Recommendations

Muhammad Ibrahim,[1] Imran Sarwar Bajwa ,[1] Riaz Ul-Amin,[2] and Bakhtiar Kasi[2]

[1]*Department of Computer Science & IT, Islamia University of Bahawalpur, Bahawalpur, Pakistan*
[2]*Faculty of Information and Communication Technologies, BUITEMS, Quetta, Pakistan*

Correspondence should be addressed to Imran Sarwar Bajwa; imran.sarwar@iub.edu.pk

Received 15 April 2019; Revised 28 May 2019; Accepted 17 June 2019; Published 4 August 2019

Guest Editor: Ricardo Soto

In the last decade, sentiment analysis, opinion mining, and subjectivity of microblogs in social media have attracted a great deal of attention of researchers. Movie recommendation systems are the tools, which provide valuable services to the users. The data available online are growing gradually because the online activities of users or viewers are increasing day by day. Because of this, big data, analytics, and computational issues have raised. Therefore, we have to improve recommendations services upon the traditional one to make the recommendation system significant and efficient. This article presents the solution for these issues by producing the significant and efficient recommendation services using multivariates (ratings, votes, Twitter likes, and reviews) of movies from multiple external resources which are fetched by the web bot and managed by the Apache Hadoop framework in a distributed manner. Reviews are analyzed by a deep semantic analyzer based on the recurrent neural network (RNN/LSTM attention) with user movie attention (UMA) to produce the emotion. The proposed recommender evaluates multivariates and produces a more significant movie recommendation list according to the taste of the user on a mobile app in an efficient way.

1. Introduction

"Recommendation systems" are services that use Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques to provide the empirical solutions of the recommendations for various application frameworks and services [1]. Recommendation systems enables mobile apps and web applications to make the perception intelligently about the selection of different items, movies [2], hotels [3], food [4], tourism [5], books [6], TV shows [7], YouTube videos [8], health [9], etc. Community trends polarize towards music, movies, or videos. For music or movies or videos, a huge amount of stream is available online, but which one of them will be watched is still a rising question. Music or movie recommendation systems still have challenges like the playlist, magnitude, security, privacy, recommendation, and session. Therefore, MRSs become a domain of music information retrieval (MIR) [10–13]. Now, the society has changed, and community trends highly depend on mobile app usage. Several products are enriched by the usage of a mobile app. So mobile app recommendation systems are essential for suitable selection of recommended items [14–16]. Most of the recommender systems are univariate and use ratings and reviews or tweets [17], and other few are bivariate (sentiment score and likes) [18–20]. This work is state of the art and uses the multivariate matrix, which makes the decision using a dynamic approach for suggesting the movie according to the relative taste of the users. The term "multivariate" means involving many variables like a qualitative variable (semantic score) and quantitative variables (Twitter likes, rating, and votes) of movies from three movie sites for significant recommendation [21]. Our work is on extremity grouping of movie reviews, where an opinionated report is labeled with semantic emotions of the microblog text or reviews and emotions [22] using a semantic parser based on the recurrent neural network (RNN/LSTM) [23, 24]. A drawback is that change of a user's review about a movie may affect the user's preference. The nature of reviews influenced by the choice of words uses multilingual dictionaries. Some

recommendation systems use linked movie databases, including Trovacinema, Google Places, and Netflix, and Wikipedia provides linked data and ontologies for descriptions about the movie [25–27]. Using the shallow machine learning models for solving the NLP problems is handcrafted and time-consuming. Nowadays, word embedding, neural-based models achieve success and popularity by producing a better result as compared to traditional machine learning logistic regression, SVM, and KNN.

Artificial neural networks are the mathematical models that are inspired by human neural networks. They have three simple layers: input, output, and hidden layers, or sometimes only two layers: input and output layers. The input layer is connected to the hidden layer via a lean weight. The hidden layer output combines via the activation function $h = \phi(w_i \cdot x_i)$. In the ANN, like the biological neural network, neurons are the nodes, while synopses are the edges. Each artificial neuron has an activation function in the ANN. There are several activation functions like sigmoid which ranges from 0 to 1, hyperbolic function which ranges from −1 to 1, and softmax function whose output in categorical distribution and ReLu function is a feedforward neural network. The ANN is not an algorithm; it is a framework for several machine learning algorithms to solve a complex work. Therefore, we can say that it is a collection of neurons or networks of neurons (https://en.wikipedia.org/wiki/Artificial_neural_network). The recurrent neural network (RNN/LSTM) processes the sequence semantically, which is the basic structure of deep neural networks. Several NLP tasks are performed by RNNs/LSTM attention. In this work, we used the hierarchical neural network (HNN) based on LSTM attention, which impaled the global user and movie information via word and sentence-level attention for document representation. The user's reviews and movie features at the word and sentence level are taken for semantic analysis of reviews, which play a major role in the process of true recommendations. Global user information represents the personal behavior and the movie feature represents a movie genre or a movie profile or linked data which are useful for semantic extraction of movie reviews [28]. In natural language (word sequence), each word or sentence is related to another one and requires to be understood semantically. A huge amount of data are available online on web contents (ratings, reviews, likes, votes, smiley, images, and stars) that can be fetched by a web bot or web agent or crawler, which are all same terms used interchangeably. Web content (ratings, reviews, likes, votes, smiley, images, and stars) is useful for recommendation services. These contents are evaluated and make the perception about users, and items make the recommendation for others [29, 30]. The hot issues of big data like computational complexity are managed by using Map-Reduce and Apache Mahout in NoSQL [31, 32] distributed environment which reduce computation complexity by clustering and horizontal scaling instead of empowered single machine [33]. Because user frequency and data volume gradually increase, it is difficult to manage these huge data by a single machine. Sparsity can be reduced by factorization [34]. Movie recommendation systems provide services to users using content-based filtering algorithms [35], collaborative filtering [36], and some combined forms to make a hybrid filtering algorithm [37]. We used implicate rating to handle the cold start problem [38], an implication managed by the server The multivariate movie recommender provides the services to users to watch the movies according to their profile or history (previously watched or rated). Therefore, there is a need to improve recommendation systems for significant recommendation services. We developed a pilot version for these problems, which consists of a mobile app, a web scraper, and a multivariate recommender to provide the significant services for movie recommendation in an efficient way.

This work is arranged as follows: related works are discussed in Section 2, the recurrent multivariate movie recommendation system model is explained in Section 3, recurrent multivariate movie recommendation system implementation is given in Section 4, experiments and results are discussed in Section 5, and evaluation of the system is done in Section 6. The conclusion of this paper is presented in Section 7 and future work with more parameters in Section 8.

2. Literature Review

Sentiment analysis deals with the user's comments, reviews, likeness, ratings, etc. to retrieve the sentiment and opinions of users. The microblog text sentiment analysis is based on the NLP methodology to retrieve suitable YouTube videos and movies and campaigns for smoking cessation, pharmacovigilance, politics of elections, advertisement of pizza, journalistic inquiry, and influenza prevention for public health [39–45]. The CNN and RNN are two major categories of deep neural networks (DNNs). Sequential and hierarchal structures deal with the RNN and CNN, respectively. Both the CNN and RNN can be supervised, semisupervised, and unsupervised. The deep learning algorithm also involves in propagation and weight update activities. RNNs are based on multiple layers: input, hidden, and output layers, while CNNs have input, hidden, and pooling layers. The CNN is efficient for pattern recognition in hierarchal data classification. However, the RNN deals with linear data to be semantically analyzed and classified in NLP; in the CNN, the window size is limited, so the RNN is very useful if reviews from the microblog are very large [46, 47]. Recommendation frameworks were presented as agents of the second class, being characterized as frameworks that "... enable individuals to settle on decisions dependent on the conclusions of other individuals." [48]. Early data-sharing frameworks had a place with the primary class and depended on text-based classification or separation, which works by choosing important things as per many literary catchphrases [49]. Recommender frameworks propose "things important to clients dependent on their unequivocal and verifiable inclinations, the inclinations of different clients, and client and thing traits." [50]. The recommendation system is finding the right product according to the taste of the customer by filtering the fact through the likeness value [51]. Suggestions utilize the assessments of a community of clients to help

people in that community all the more adequately distinguish the content of enthusiasm from a possibly overpowering set of decisions [52]. Recommendation by demographics which groups the users as per the traits of their personnel file, besides, creates proposals dependent on classes of the statistic. A premature precedent is a generalization-based Grundy system, which has been made to bolster book searching in a library [53]. The recommendation is reliant on the computation of utility of each item for a user' utility capacity (http://www.eqo.info). Recommendation by knowledge proposes things dependent on legitimate inductions about a user's inclinations. A learning portrayal or a rule about how a thing meets a specific client requirement is important (http://www.findme.com.ph). By applying preference-based collaborative filtering, a recommender system intend to foresee majority of estimation of likeness, where a few users may provide inconspicuous views as well [54]. There are two types of architecture for the recommendation systems: One is centralized and situated at a specific location [55]. Another one is geographically distributed and situated at different locations [56]. There are three types of recommendation modes by which the system will be initiated: The first one is the push mode in which suggestions are pushed to the user while he is not associating with the system by email [48]. The second one is the pull mode in which suggestions are generated but are displayed to the user just when he permits or unequivocally asks for it [57]. Push and pull modes are the active mode in which the recommender is initiated. The third one is the passive mode in which suggestions are generated as a feature of the customary framework activity, for instance, an item suggestion with reference to a user's preference [58]. A user's preference of items can be determined by using the linear adaptive function multiattribute utility theory (MAUT) [59]. Cosine similarity determined by cosine vector comparability is one of the well-known measurements of insight since it notionally considers just the edge of two vectors without the size. The collaboration between the search item and the other item that is rated by users can be measured by the angle of their vectors; if the angle is 90°, then the value of cosine similarity is zero, which means the item is irrelevant. If the angle between cosine vectors is nearly about zero, then the value of cosine similarity is one, which means the product is relevant (https://en.wikipedia.org/wiki/Cosine_similarity) [60]. There are three major classes of collaborative filtering: (1) collaborative filtering (CF) in which users and items' profile data are required to make a decision for recommendation [61], (2) content-based filtering on the description of the content of items and user preference information (explicate or implicate) for recommendation [62], and (3) combining various filtering techniques to handle scalability, sparsity, and cold start problem and other big data issues of the recommendation system to get better outcomes [63].

3. Multivariate Movie Recommendation Model

The multivariate approach is (see Figure 1) based on three modules: mobile app, multivariate recommender, and web scraper. Users can get the recommendation services through a mobile application. The mobile app module provides the information such as the user's query, profile, and history to the recommender module. The recommendation is made for both registered and unregistered users of the mobile app. The recommendation module is based on the deep learning NLP module and computation module. The NLP module preprocesses the fetched qualitative data (user's reviews) of microblogs using a tokenizer, stemmer, and POStagger and then semantically analyzes the reviews and extracts the semantic emotions about movies. Semantic parser work is based on the deep machine learning algorithm recurrent neural network (RNN/LSTM attention) with user movie attention (UMA). Semantic emotion is classified into five major classes: (i) Highly Favorable, (ii) Favorable, (iii) Averagely Favorable, (iv) Unfavorable, and (v) Highly Unfavorable, on the bases of their relative semantic scores. While the computation module normalized the quantitative data (Twitter likes, votes, and ratings), normalized scores and semantic emotional scores were evaluated to generate the recommended movie list. The recommended movie list consists of five medals and their popularity such as Platinum: "Highly Popular," Gold: "Popular," Silver: "Averagely Popular," Bronze: "Unpopular," and Copper: "Highly Unpopular." The recommended movie list is generated according to users' taste and preference. A web scraper fetched data (reviews, Twitter likes, votes, and ratings) from external data source sites (CinemaBlend, Moviefone, Rotten Tomatoes, and Twitter) and stored them in the NoSQL database for computation. Users' feedback about a movie and app is useful for generating the recommended list and evaluation of system reliability.

3.1. NLP Module. NLP has the capability to understand natural language. Users share their opinions and reviews from the microblog that help in making a decision. Positivity, negativity, and neutrality are extracted by opinion mining, whereas emotions are extracted by semantic analysis. In our work, the NLP module determines the semantic emotion of the movie's reviews by the LSTM-attention machine learning algorithm. This semantics is one of the parameters in multivariates used to make a recommendation. This methodology for semantics is depicted as follows:

(i) The module fetches the reviews from microblogs related to movies such as CinemaBlend, Moviefone, and Rotten Tomatoes

(ii) The module preprocesses the microblog text or reviews using a sentence splitter, tokenizer, and stemmer/lemmatizer

(iii) The module determines the sense of the word to strength the sentiment using SenticNet

(iv) Semantic parsing based on attention is done to construct a parse tree to identify the syntactic tree as the emotion of the sentence

(v) RNN/LSTM-user movie attention (UMA) machine learning algorithm is used to classify the reviews

3.2. Preprocessing. It is estimated that more than 80% of data are unstructured and not in an organized manner.

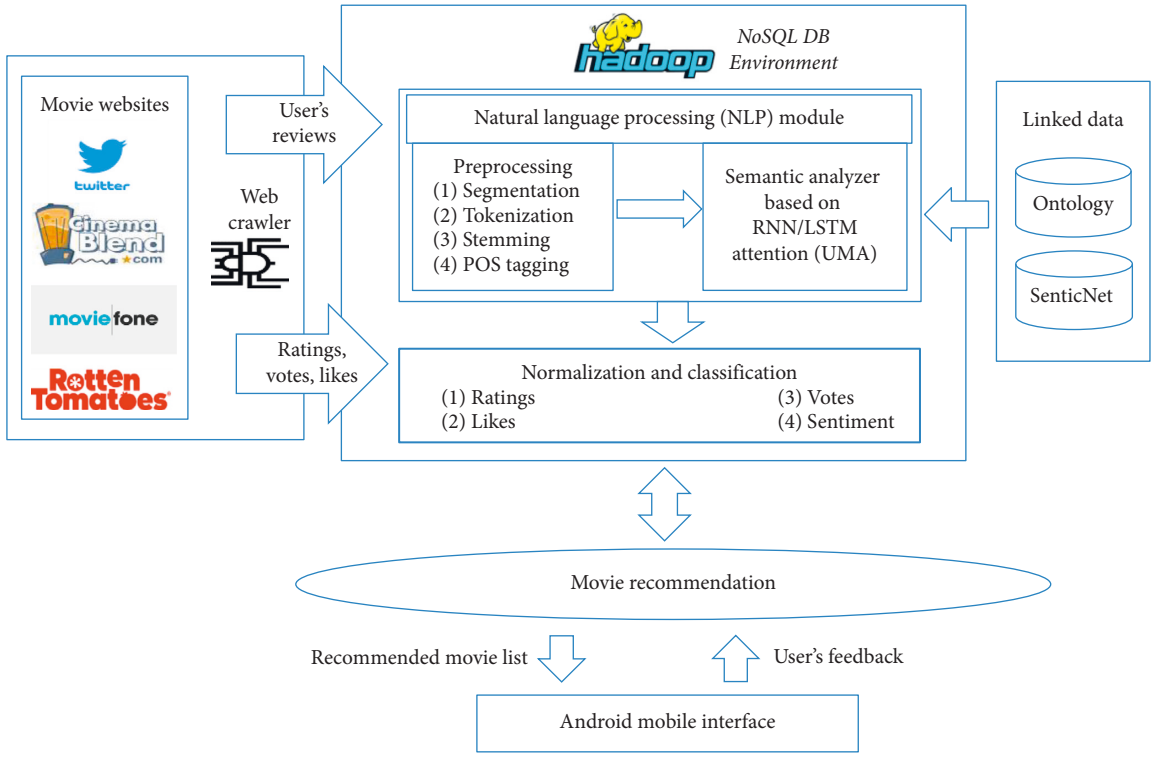


Figure 1: Architecture of the multivariate movie recommendation system.

Preprocessing of text is cleaning or normalization of text/reviews. Stemming or lemmatization and tokenization are done to reduce the sparsity and shrink the feature space. Semantic analysis has to face some challenges such as short text, misspelling, grammatical mistake, slang, unusual terms, tags, white spaces, noise, and emoji. Text is a sequence of words, while word is a meaningful sequence of characters. However, the question is how to find out the boundaries of words. Words are identified by spaces or punctuation in English. However, a compound word is a set of words which have no spaces in German, for example, ("childhood memories description of an unforgettable event") ⟶ ("Kindheitserinnerungen Beschreibung eines unvergesslichen Ereignisses"), while there are no spaces at all in Japanese like this ("childhoodmemoriesdescriptionofanunforgettableevent").

3.2.1. Tokenization. The process of splitting the text stream into units is called tokenization. Units refer to tokens. For example, "This movie is so riddled" is a character string which is tokenized as [This] [movie] [is] [so] [riddled]. Splitting the input sequence into tokens has some problems. Splitting by white space has a problem that different tokens are tokenized into similar words, while the same words may have similar meanings (https://NLTK.Tokenize.WhiteSpaceTokenizer). Splitting by punctuation in which some punctuation are not meaningful is like "An apostrophe problem" (https://NLTK.Tokenize.WordPunctTokenizer). Splitting comes up with the set of rules that generate a more meaning full result (https://NLTK.Tokenize.TreeBankWordTokenizer).

3.2.2. Stemming (Lemmatization). The stemmer stemmed the words like the Porter stemmer, which stemmed the English words "looked" as "look" with a morphological production rule, for example, [("SSES ⟶ SS"): ("Caresses ⟶ caress")], [("IES ⟶ I"): ("Ponies ⟶ Poni")], [("SS ⟶ SS"): ("Caress ⟶ Caress")], and [("S ⟶ S"): ("Cats ⟶ Cat")], but due to stemming of nonwords, the same plural word can be stemmed to singular and irregular forms. These are produced like (Wolves ⟶ wolv), (Feet ⟶ Feet). The WordNet database is looked up for lemmas to solve this type of problem. It solves some specific problems but not all, like (Wolves ⟶ wolf) and (Feet ⟶ Foot) (https://NLTK.Stem.WordNetlemmatizer).

3.2.3. POS-Tag Generation. POS tags are determined for all the tokens by Treebank POStagger. Treebank Project 1 represents 36 POS tags (http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html). For example, the POStag of string "Unwatchable I made it through 20 minutes I think" is [Unwatchable/VB] [I/PRP] [made/VBD] [it/PRP] [through/IN] [20/CD] [minutes/NNS] [I/PRP] [think/VBP].

3.2.4. Word Sense Disambiguation (WSD). WSD is the issue of deciding the "sense" of a word. A lexicon controls a word and its conceivable faculties. Bar-Hillel, 1960, presented the example ["Little John was looking for his toy box. Finally, he found it. The box was in the pen. John was very happy."]. In the previous string, a word "pen" has different senses according to WordNet. "Pen" word defines an "ink flow from a point to write"; here, pen is defined as an "arena of

cattle" and as a "bird's family." In the assessment of the movie's reviews, SenticNet is utilized to indicate their degrees of polarity, antagonism, and impartiality. The SenticNet score of the terms and its recurrence are determined to get the general supposition of the reviews (https://sentic.net) [64].

3.3. Parsing. In NLP, parsing is the process of determining the structure of a sentence by analyzing its essential words based on an underlying syntax. The Stanford parser is used to construct the parse tree that determines the syntactic structure relative to grammar (language). Parsing can refer to various things. Shallow parsing or chunking is the process of grouping the words into noun phrases (NP). Stuff can also be grouped into VP (verb phrases) and PP (prepositional phrases) using grammar like (S $\longrightarrow$ NP | VP), (NP $\longrightarrow$ Det-Noun), (NP $\longrightarrow$ ProperNoun), and (VP $\longrightarrow$ Verb | NP). In contrast, dependency parsing determines the dependencies between the words and their type. For example, spaCy + displaCy for parsing and rendering is used to produce a more semantic result.

3.3.1. RNN/LSTM. Neural networks are represented by RNN/LSTM cells [65]. Typically, in Birdseye, RNN/LSTM is a chain of several copies of the same static network, as shown in Figure 2. From input, the sequence of copies of networks is working in a single timestep. In addition, networks are linked with each other via their hidden states h. So we can say that every copy network has its own inputs as the copy network is unfolded or unrolled. Let the sequence be represented as x_1, x_2, $x_3 \ldots x_n$ and each timestep be represented as $x_t \in x_1 \ldots x_n$. At timestep t, h_t is a hidden layer and f is used to calculate the hidden state: $h_t = f(h_{t-1}, x_t)$. A word is represented by a timestep in the long sequence. For example, the given string is represented as a sequence in the mathematical form: "it is a good movie" $\longrightarrow$ ["it," "is," "a," "good," "movie"]. And the timestep ($t = 0, 1, 2, \ldots$) for the string "it" is represented as x_0, "is" as x_1, "a" as x_2, "good" as x_3, and "movie" as x_4. If $t = 1$, then x_t = "is" $\longrightarrow$ "current timestep to event" and x_{t-1} = "it" $\longrightarrow$ "previous time stamp to event":

$$
\begin{aligned}
X &= \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}, \\
\text{input gate at time } t\text{:}\quad i_t &= \sigma\left(W_i \cdot X + b_i\right), \\
\text{forget gate at time } t\text{:}\quad f_t &= \sigma\left(W_f \cdot X + b_f\right), \\
\text{candidate state at time } t\text{:}\quad \widetilde{C}_t &= \tanh\left(W_c \cdot X + b_c\right), \\
\text{final memory cell:}\quad C_t &= f_t * c_{t-1} + i_t * \widetilde{C}_t, \\
\text{output gate:}\quad o_t &= \sigma\left(W_o \cdot X + b_o\right), \\
h_t &= o_t * \tanh\left(C_t\right).
\end{aligned}
\tag{1}
$$

At the $i_t s$ input gate, the decision on which information should be remembered or rid of is made by the sigmoid function σ. It produces a 0 or 1 value: 0 means forget, while 1 means remember in the cell state. Sigmoid function at the input gate takes a decision on which value should be updated, and the new candidate value information is represented by tanh function $\widetilde{C}_t$. Output gate sigmoid function decides which part of information should be produced, and then tanh function produces the value between 1 and −1.

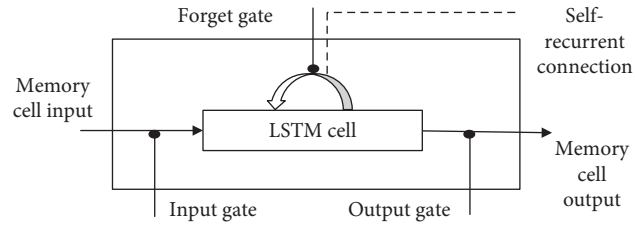


Figure 2: LSTM cell architecture.

The sequential semantic information is preserved in the recurrent neural network's hidden states. In the hidden state (h_t), the semantic information of the input sequence is preserved. When a new input is experienced and again delivered to be the subsequent input, then semantic information is altered. Passing the information from one to another network helping to find out the correlation among the words from the sequence is represented as a long-term dependency.

3.3.2. LSTM-Based Sequence Labeling. Predicates from a given input sequence are marked, and the label arguments corresponding to every predicate are identified. For example, in the given sentence "I watched the movie," the predicate (watched) is marked, and labels corresponding to the predicate are "I," "the," and "movie" as an agent, null, and theme, respectively. Multiple predicates may present in a sentence, and different labels may be marked to the same word for every predicate. Concatenating pretrained ones (Word2vec) generates vectors of every word. The 1-bit flag represents the predicate in the specific training unit to confirm that the network deals with every predicate separately and serves it into the LSTM layer to the word context. With the predicate, any one word is labeled to take the dot product of its hidden state. A softmax function is applied over it. The probability of a sentence is calculated as follows:

$$
\begin{aligned}
P(X) &= \prod_{k=1}^{n} P\left(x_i \mid x_1, x_2, x_3 \ldots x_{t-1}\right), \\
X_{l,r} &= \mathrm{ReLU}\left(x_l \cdot x_r\right).
\end{aligned}
\tag{2}
$$

Here, the role label r is calculated by the weight matrix parameter using ReLU function and predicate lemma and the role depicted by taking the dot product of vectors to embedding.

3.3.3. Neural Sentiment Classification (NSC). Document-level sentiment classification is measured by neural sentiment classification (NSC) based on hierarchical LSTM attention with user movie attention (UMA) (see Figure 3) that is represented by the user's global information and movie features [28]. Let a review $d \in D$ with sentences, each sentence $(s_1, s_2, \ldots, s_n)$ of a particular review $s_i \in d$, a user $u \in U$, and a movie $m \in M$ review corpus (users and their movie set). Moreover, l_i is the length of the i-th sentence,

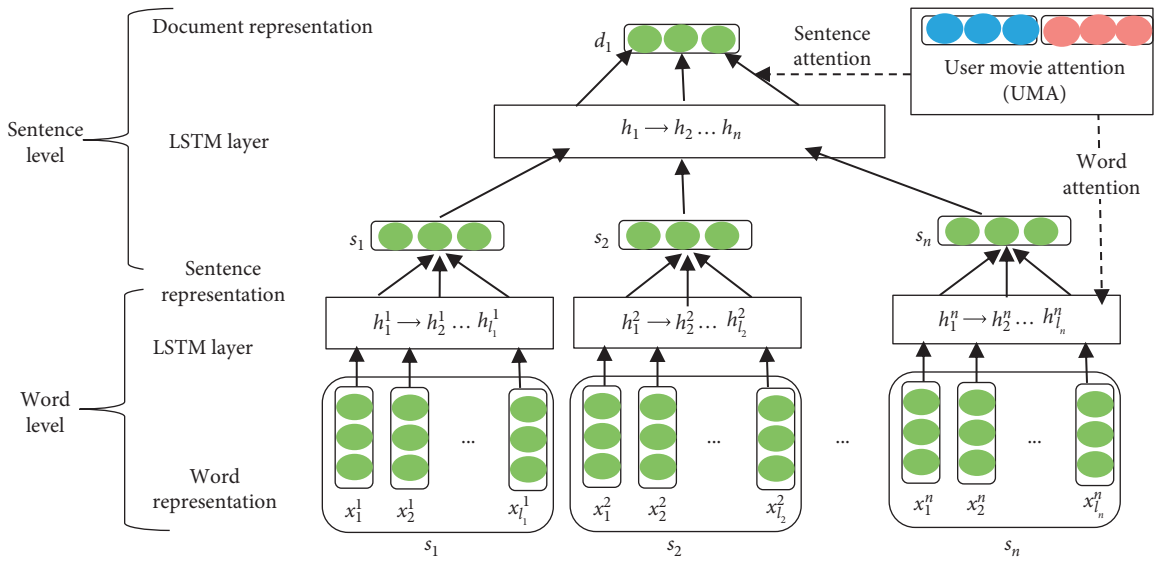


Figure 3: LSTM-attention (UMA) architecture.

while s_i consists of l_i words as $x_1^i, x_2^i, \ldots, x_{l_i}^i$. Predicting the semantic rating of documents is done according to their text information. Firstly, in word-level low-dimensional semantic space, each word x_j^i is mapped to its embedding $x_j^i \in \mathbb{R}^d$ in a sentence. Every step has a given input word x_j^i, the current cell state c_j^i, and the hidden state h_j^i that may be updated with the preceding cell state c_{j-1}^i. Then, the hidden state h_{j-1}^i is represented. The document representation architecture is presented as follows:

$$\begin{aligned}\begin{bmatrix} i_j^i \\ f_j^i \\ o_j^i \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix}\left(W \cdot \left[h_{j-1}^i, x_j^i\right] + b\right),\\ \widehat{c}_j^i &= \tanh\left(W \cdot \left[h_{j-1}^i, x_j^i\right] + b\right),\\ c_j^i &= f_j^i \odot c_j^i + i_j^i \odot \widehat{c}_j^i,\\ h_j^i &= o_j^i \odot \tanh\left(c_j^i\right).\end{aligned} \tag{3}$$

Sigmoid activation function and gate activation functions are represented as σ and i, f, and o, respectively, while elementwise multiplication is represented as $\odot$. Training parameters needed for training are represented as x and b. The feed hidden states $[h_1^i, h_2^i, \ldots, h_{l_i}^i]$ are represented to a mediocre pooling layer to acquire the representation of the s_i sentence. Sentences are embedded at the sentence level $(s_1, s_2, \ldots, s_n)$ into the LSTM; after that, document representation d is acquired via a mediocre pooling layer in a similar way as follows:

$$\begin{aligned}\begin{bmatrix} i_i \\ f_i \\ o_i \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix}\left(W \cdot \left[h_{i-1}, s_i\right] + b\right),\\ \widetilde{C}_i &= \tanh\left(W \cdot \left[h_{i-1}, s_i\right] + b\right),\\ C_i &= f_i \odot c_{i-1} + i_i \odot \widetilde{C}_i,\\ h_i &= o_i \odot \tanh\left(C_i\right).\end{aligned} \tag{4}$$

Here, training parameters needed for training are represented as s and b. The feed hidden states $[h_1, h_2, \ldots, h_n]$ are represented to a mediocre pooling layer to acquire the d_i document representation.

3.3.4. User Movie Attention (UMA). At various levels, a necessary component is extracted by using user movie attention (UMA) for sentiment classification. UMA is applied at the word level to construct a sentence and sentence level to generate a document. Obviously, sentence meaning may not be represented by all words for several users and movies. In spite of feeding hidden states at the word level to an average pooling layer, user movie attention (UMA) is used to extract user/movie relative words, which are essential to sentence meaning. Informative words are aggregated to produce the representation of the sentence. Formally, weighted hidden states generate the enhanced sentence as follows:

$$\begin{aligned} s_i &= \sum_{j=1}^{l_i} a_j^i h_j^i,\\ d_i &= \sum_{i=1}^{n} a_i h_i.\end{aligned} \tag{5}$$

Importance of the jth word is measured by a_j^i for the current user and movie. Each user u and movie m are embedded continuous and real-valued vectors $u \varepsilon \mathbb{R}^{d_u}$ and $m \varepsilon \mathbb{R}^{d_m}$, while user and movie embedding is represented as d_u and d_m dimensions, respectively. Moreover, for every hidden state, the attention weight a_j^i is presented as follows:

$$a_j^i = \frac{\exp\left(e\left(h_j^i, u, m\right)\right)}{\sum_{k=1}^{l_i} \exp\left(e\left(h_k^i, u, m\right)\right)}. \tag{6}$$

For the sentence level,

$$a_i = \frac{\exp\left(e\left(h_i, u, m\right)\right)}{\sum_{i=1}^{n} \exp\left(e\left(h_i, u, m\right)\right)}. \tag{7}$$

Importance of words for sentence representation as well as document representation is presented by e score function as follows:

$$e\left(h_j^i, u, m\right) = v^{\mathrm{T}} \tanh\left(W_h h_{ij} + W_u u + W_m m + b\right). \tag{8}$$

For the sentence level,

$$e\left(h_i, u, m\right) = v^{\mathrm{T}} \tanh\left(W_h h_i + W_u u + W_m m + b\right), \tag{9}$$

where v is a weight vector and v^{T} represents its transpose, while W_h, W_u, and W_m are weight matrices. Meaning of every document varies for different users and movies by the sentence, which provides the hints. So in the sentence level, usage of attention a with the u user and m movie vector at the word level to select informative sentences to generate document representation d is presented as follows:

$$d = \sum_{i=1}^{n} \beta_i h_i. \tag{10}$$

In the sentence level, the β_i weight of the h_i hidden state is measured similar to word attention. The higher level representation of document d is generated by hierarchical extraction from words and sentences in the document. So, for sentiment classification of the document, it is used as features. tanh activation function is used at the nonlinear layer for current document representation in the target space of C classes:

$$\widehat{d} = \tanh\left(W_c d + b_c\right). \tag{11}$$

tanh activation function is used at an absolute layer to get sentiment distribution of the document:

$$d_c = \frac{\exp\left(\widehat{d}_c\right)}{\sum_{k=1}^{C} \exp\left(\widehat{d}_k\right)}. \tag{12}$$

Sentiment classes and prediction probability of sentiment class C are represented as C and p_c, respectively. During the training, loss function for optimization is measured by error cross-entropy between the distribution of Gold sentiment and distribution of our model sentiment as follows:

$$L = \sum_{d \varepsilon D} \sum_{c=1}^{C} p_c^g\left(d\right) \cdot \log\left(p_c\left(d\right)\right). \tag{13}$$

Here, Gold probability of sentiment class C and training document are represented as p_c^g and d, respectively, while reality-based truth is one and others are zero.

Some nomenclatures used in our mathematical model are presented in Table 1.

Table 2 presents the emotion class.

3.4. Computation and Classification. Sentiment analysis determines the emotions of reviews. Firstly, the aggregated sentiment score of each document from each site for the j-th movie is computed. Then, the qualitative score and then aggregated quality scores for Twitter likes are computed to get the final score for the recommendation of movies and generate the popularity class relative to the final score:

$$\begin{aligned}
C^{j,1} &= \sum_{\substack{i=0 \\ k=1}}^{n} dc_i^{j,k}, \\
C^{j,2} &= \sum_{\substack{i=0 \\ k=2}}^{n} dc_i^{j,k}, \\
C^{j,3} &= \sum_{\substack{i=0 \\ k=3}}^{n} dc_i^{j,k}, \\
q^{j,1} &= \left[\left(R^{j,1}\right) + \left(V^{j,1}\right) + \left(C^{j,1}\right)\right], \\
q^{j,2} &= \left[\left(R^{j,2}\right) + \left(V^{j,2}\right) + \left(C^{j,2}\right)\right], \\
q^{j,3} &= \left[\left(R^{j,3}\right) + \left(V^{j,3}\right) + \left(C^{j,3}\right)\right], \\
Q^j &= \left[\left(q^{j,1}\right) + \left(q^{j,2}\right) + \left(q^{j,3}\right)\right], \\
Q^j &= \sum_{k=1}^{3} q^{j,k},
\end{aligned} \tag{14}$$

j-th movie final multivariate emotional score E

$$= \log\left(\gamma\left[\left(Q^j\right) + (\mathrm{TL})\right]\right),$$

where $\gamma = 0.5$, and

j-th movie popularity score P

$$= \left[\frac{f_{m_j} - \min\left(f_{m_j}\right)}{\max\left(f_{m_j}\right) - \min\left(f_{m_j}\right)} * 10\right]. \tag{15}$$

This mathematical formulation is used to determine the final popularity score using the multivariate model. The emotional value is stretched to 10 scales, by which the popularity status is determined. Every movie is labeled with a medal according to the popularity score, and the algorithm that identifies the medal by using fuzzy logic on behalf of the popularity score to find the popularity of the movies is depicted as follows:

(1) IF multivariate value ≥08, THEN: Platinum: "Highly Popular"

(2) ELSE IF multivariate value ≥06, THEN: Gold: "Popular"

(3) ELSE IF multivariate value ≥04, THEN: Silver: "Average Popular"

(4) ELSE IF multivariate value ≥02, THEN: Bronze: "Unpopular"

(5) ELSE Copper: "Highly Unpopular"

Table 1: Nomenclatures and description.

Nomenclature	Description
d	Document/review
s	Sentence
x	Word
D	Review corpus
m	Movie
l	Length of a sentence
h	Hidden state
S	Total movie sites
b	Biases
TL	Twitter likes
t	Timestep
$C^{j,1}$	j-th movie sentiment at site S_1
$C^{j,2}$	j-th movie sentiment at site S_2
$C^{j,3}$	j-th movie sentiment at site S_3
$R^{j,1}$	j-th movie rating at site S_1
$R^{j,2}$	j-th movie rating at site S_2
$R^{j,3}$	j-th movie rating at site S_3
Q^j	j-th movie total quantitative score
RecS	Final recommendation score
AWAS	Aggregated weighted average sentiment
Multivariate	Multivariate final score
i	Input gate
o	Output gate
f	Forget gate
σ	Activation function
b	Biases
v	Weight vector
v^{T}	Vector transpose
t	Timestep
$\odot$	Multiplication
h_t	Hidden state at t timestep
h_{t-1}	Hidden state at t–1 (previous) timestep
W	Weight matrix for input to hidden layers at t timestep
$\varnothing$	tanh is an activation function
x_j^i	Input at timestep (t)
$V^{j,1}$	j-th movie votes at site S_1
$V^{j,2}$	j-th movie votes at site S_2
$V^{j,3}$	j-th movie votes at site S_3
L	Loss
AS	Aggregated sentiment
WAS	Weighted average sentiment

Table 2: Emotion status.

Semantic score	Emotional class
0.5< and ≤1.00	Highly Favorable
0.00< and ≤0.5	Favorable
−0.5< and ≤0.00	Average Favorable
−1.00< and ≤−0.50	Unfavorable
≤−1.00	Highly Unfavourable

The ranges of the popularity scores and their respective medals and degree of popularity are given in Table 3.

The category represented by a movie genre to classify the movie according to its features, movie recommendation services suggests top 10 popular movies with their category according to the user request and profile history.

Table 3: Popularity scores and their respective medals and popularity status.

Popularity score	Medal rank	Status
0.8–1.0	Platinum	Highly Popular
0.6–0.79	Gold	Popular
0.4–0.59	Silver	Average Popular
0.2–0.39	Bronze	Unpopular
0.0–0.19	Copper	Highly Unpopular

4. Multivariate Movie Recommendation System Implementation

4.1. System Component Interaction. User android application is front end of the system (see Figure 4) by which users can get the web services from the system, and back end is the movie recommendation system in the NoSQL environment with Apache Mahout and Hadoop, which provide the web services to the users as well as a web scraper by which the system fetched the data. Web scraper fetched the data from the external data source on the bases of matching lexicons of the query and movie content.

4.2. NoSQL Environment Implementation

4.2.1. Hadoop Architecture. It is a framework with four fundamental components: (1) HDFS splits the file into many small files and stores them on three servers for fault tolerance constraints as replicas in a distributed file system manner. (2) Map Reduce programming standard is for handling and manipulating big data. (3) Common/Core holds the reference library and services to backing up Hadoop. (4) YARN performs management, computation, and scheduling of resources and tasks.

4.2.2. Apache Mahout. Implementation of collaborative filtering, clustering, and classification is done by Apache Mahout. In the NoSQL environment, Apache Mahout interfaces implement the Hadoop framework and evaluate the performance similarities and neighborhood measures. A multivariate web scraper is implemented and big data are generated.

4.3. Web Scraper. Our web scraper is a scripting program, which surfs the W3, fetches data from different movie websites to extracts the reviews, votes, ratings, and Twitter likes, and stores them in the repository. In addition, it manages and handles scrape data in a NoSQL environment using Hadoop and Apache Mahout. The web scraper (web bot) receives the URLs and matches them with keywords (Meta tags) of the web page. If the keywords are matched, then the web pages are downloaded; otherwise, the irrelevant pages are discarded.

4.4. NLP Tools. Stanford CoreNLP technology tools are used to process the natural language like English. They give the words, relative parts of speech, and identification of

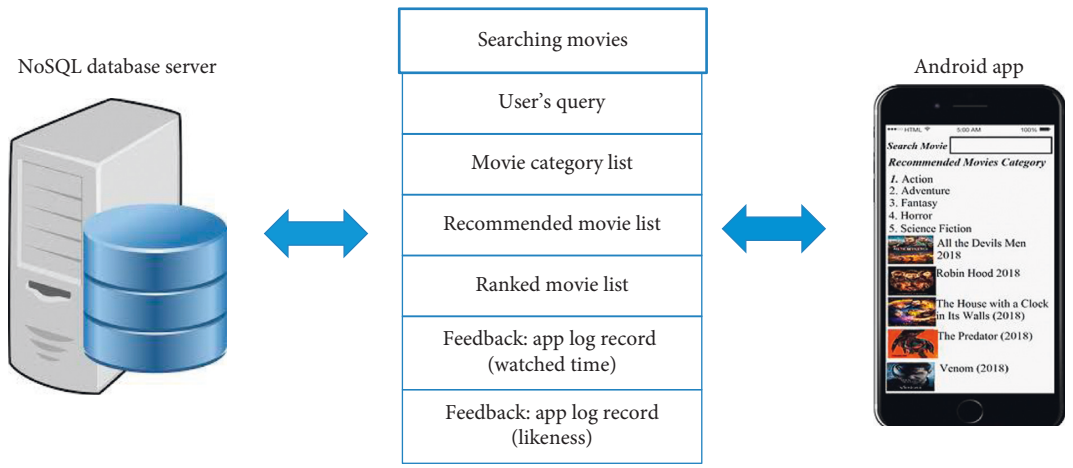


Figure 4: Interaction of components of multivariate movie recommendation.

sentiments. The Stanford CoreNLP framework is the integration of many of Stanford's NLP tools, like POStagger, parser, sentiment analyzer, named "entity recognizer," and pattern learning and information extracting tools from https://stanfordnlp.github.io/CoreNLP.

4.5. Mobile Application Usage

4.5.1. Unregistered Users. Unregistered users can request the movie by the search query to our recommendation system, and the system will respond to that query by content filtering to extract the features or content of a movie from the query. Collaboration is done between the user request and system-generated movies. The system provides the watched window to unregistered users for watching the recommended movies. Unregistered users give feedback by their likeness, and the system uses the feedback for accurate measurement.

4.5.2. Registered Users. If unregistered users sign up, then they can sign in and maintain their profile or history. For registered users, collaborations may be done on the bases of both the query and the history. Registered users may provide feedback to the system by their likeness, and the system uses this feedback for collaborative filtering between liked movies or their movie history and system movies for recommendations of the movie of their choices as well as accurate measurement of the multivariate movie recommendation system. The application also provides a watch window for registered users to watch the recommended movie.

4.6. Cold Start Problem Handling. Collaborative filtering (CF) is done for movie recommendation for registered and unregistered users; but in two cases, the problem may occur:

(i) Case 1: if the registered users request the movie, the system collaborates the requested movie with the system movie and recommends the movies on behalf of user history. Here, one problem arises: if the newly registered users request the movies, then the system recommends the movies according to movies mostly liked by others to solve the cold start problem of newly registered users.

(ii) Case 2: if a new movie arrives for registered or unregistered request, then the system recommends the movies according to the collaboration of new movie trailers, which were mostly liked to solve the cold start problem of newly released movies.

4.7. Similarity Measurement. We use cosine similarity in which there are two vectors for measuring the angle value for similarity manipulation. A smaller angle degree is directly proportional to larger similarity, and vice versa, as shown in Figure 5. It is also known as vector-based similarity. Movie document and search query document correlation is computed where q is the search query document and d is the movie document. The similarity can be calculated by the following equation:

$$\vec{q} \cdot \vec{d} = \|\vec{q}\| \cdot \|\vec{d}\| \cdot \cos\theta,$$
$$\text{sim}(q \cdot d) = \cos\theta = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \cdot \|\vec{d}\|}. \quad (16)$$

5. Experiments and Results

The procedure followed by data preprocessing is NLP procedures applied for sentiment analysis on fetched data, and then the sentiment score is computed by using SenticNet and obtained results are presented as follows.

Table 4 presents the identification of movies and categories.

Table 5 presents the identification of sites, movies, users, and reviews.

Table 6 presents the movie review from the movie website CinemaBlend.

Movie reviews from movie websites Moviefone and Rotten Tomatoes were also fetched, and semantic scores and emotions were computed.

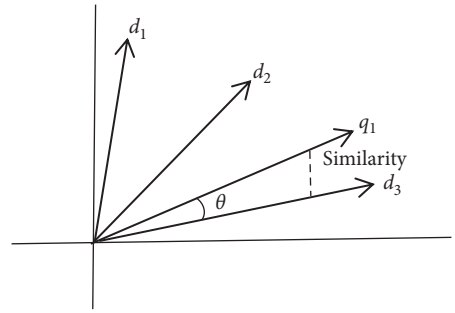


Figure 5: Cosine similarity angle. *Note.* If the angle between vectors is zero degrees, then the cosine similarity value is 1, which means movies are similar or relevant, and if the angle is 90 degrees or above, then the similarity value is zero, which means the movie is irrelevant.

Table 4: Movie ID and movie category ID.

Movie and category IDs		
Movie ID	Movie title	Movie category
m_1	Robin Hood (2018)	Action (c_1)
m_2	The House with a Clock in Its Walls (2018)	Adventure (c_2)
m_3	The Predator (2018)	Fantasy (c_3)
m_4	Venom (2018)	Horror (c_4)
m_5	The Flash	Science fiction (c_5)

Table 5: IDs of sites, movies, user names, and reviews.

ID table					
Site name	Site ID	Movie ID	User name	User ID	Review ID
CinemaBlend	s_1	m_1	Deplorable_me	u_1	d_1
		m_2	Snow gator	u_2	d_2
		m_3	David Curry	u_3	d_3
		m_4	Smedley	u_4	d_4
		m_5	DC villains	u_5	d_5
Moviefone	s_2	m_1	The Guardian Peter Bradshaw	u_6	d_6
		m_2	Snow gator	u_7	d_7
		m_3	Relax ad mike	u_8	d_8
		m_4	Jza Smack	u_9	d_9
		m_5	Clifford De Voe	u_{10}	d_{10}
Rotten Tomatoes	s_3	m_1	Jennifer Heaton	u_{11}	d_{11}
		m_2	Carlos Díaz Reyes	u_{12}	d_{12}
		m_3	Jeffrey Bloomer	u_{13}	d_{13}
		m_4	ugene Bernabe	u_{14}	d_{14}
		m_5	Dee R.	u_{15}	d_{15}

Table 7 presents the movie review tokenization and tagging from movie websites CinemaBlend, Moviefone, and Rotten Tomatoes.

The parsing of movie reviews' tokens taken from CinemaBlend, Moviefone, and Rotten Tomatoes was performed using the Stanford parser. Here, Table 8 presents the sentiment score of movie reviews from CinemaBlend, Moviefone, and Rotten Tomatoes.

Table 9 presents the normalized Twitter likes of movies from Twitter.

Table 10 presents the normalized rating score of movies from CinemaBlend, Moviefone, and Rotten Tomatoes.

Table 11 presents the normalized vote score of movies from CinemaBlend, Moviefone, and Rotten Tomatoes.

Table 12 presents the final score, movie category, medal rank, and genres of movies from CinemaBlend, Moviefone, and Rotten Tomatoes.

Figure 6 presents the multivariate movie ranked recommendation of movies from CinemaBlend, Moviefone, and Rotten Tomatoes.

Figure 7 presents differences in the rating of the movie from CinemaBlend, Moviefone, and Rotten Tomatoes.

Figure 8 presents differences in the votes of the movie from CinemaBlend, Moviefone, and Rotten Tomatoes.

Figure 9 presents differences in the sentiment of the movie from CinemaBlend, Moviefone, and Rotten Tomatoes.

6. Evaluation and Discussion

We evaluate the sentiment classification models as well as recommendation models as follows.

6.1. Sentiment Classification Model Evaluation. For sentiment, classification models are evaluated by accuracy and RMSE, which measure the overall performance of the sentiment classification model and divergence between predicted and truth ground sentiment classes, respectively. We compare the several base sentiment classification methods using three datasets imdb, yulp13, and yulp14, which contain reviews about movies using Stanford CoreNLP. Majority of the baseline sentiment classification models refer to categorization of document sentiments in the training set by an *SVM* classifier with unigram, bigram, and trigram. Text feature extraction including character *n*-gram and -word is done by the SVM classifier. Use of leniency feature is extracted by UPF [66]. Document representation is obtained by AvgWordvec, which nourished into SVM. Feature generation is by SSWE (sentiment-specific word embedding) [67]. Sentence representation is by the RNTN (recursive neural tensor network) [68]. Document classification is by

Table 6: Semantic emotion of movie reviews of users from CinemaBlend.

Reviews of users about movies				
Movie ID	User name	Review ID	Reviews	Semantic emotion
m_1	u_1	d_1	Unwatchable I made it through 20 minutes I think.	Average Favorable
m_2	u_2	d_2	I did not like it. Thought it was uneven and wasted some great talent. Not funny enough, too much turd humor, and think it is a made for USA level of quality with better effects. It is worth seeing for Blanchett. She does steal every scene, and when the sequel happens—and it has made more than enough money for one—I hope she is front and center as the main character. She and Black do have fantastic chemistry.	Average Favorable
m_3	u_3	d_3	Predator 1 and 2 had comedy in it. Shane Black helped write the original (everyone should know that by now). Predators are the most serious movie of the franchise.	Average Favorable
m_4	u_4	d_4	I went to see it today with open expectations (professional reviews bad, viewer reviews good) and thought it was a fun movie. It cracked me up a couple of times.	Highly Favorable
m_5	u_5	d_5	The Flash has done a fantastic job of incorporating classic from the hero's comic book history	Highly Favorable

Table 7: Movie review tokenization and tagging.

		Tags
User ID	Tokens per document	Tagging
u_1	9	Unwatchable/VB I/PRP made/VBD it/PRP through/IN 20/CD minutes/NNS I/PRP think/VBP
u_2	91	Thought/RB it/PRP was/VBD uneven/JJ and/CC wasted/VBD some/DT great/JJ talent/NN ./. Not/RB funny/JJ enough/RB ,/, too/RB much/JJ turd/VBD humor/NN ,/, and/CC think/VBP it/PRP is/VBZ a/DT made/VBN for/IN USA/NNP level/NN of/IN quality/NN with/IN better/JJR effects/NNS ./. It/PRP is/VBZ worth/JJ seeing/VBG for/IN Blanchett/NNP ./.She/PRP does/VBZ steal/VB every/DT scene/NN ,/, and/CC when/WRB the/DT sequel/NN happens/VBZ -/: and/CC it/PRP has/VBZ made/VBN more/RBR than/IN enough/JJ money/NN for/IN one/CD -/: I/PRP hope/VBP she/PRP is/VBZ front/NN and/CC center/NN as/IN the/DT main/JJ character/NN ./. She/PRP and/CC Black/NNP do/VBP have/VB a/DT fantastic/JJ chemistry/NN ./.
u_3	34	Predator/NNP 1/CD &/CC 2/CD had/VBD comedy/NN in/IN it/PRP ./. Shane/NNP Black/NNP helped/VBD write/VB the/DT original/NN -LRB-/-LRB- everyone/NN should/MD know/VB that/DT by/IN now/RB -RRB-/-RRB- ./. Predators/NNS is/VBZ the/DT most/RBS serious/JJ movie/NN of/IN the/DT franchise/NN ./.
u_4	34	I/PRP went/VBD to/TO see/VB it/PRP today/NN with/IN open/JJ expectations/NNS -LRB-/-LRB- professional/JJ reviews/NNS bad/JJ ,/, viewer/CD reviews/NNS good/JJ -RRB-/-RRB- and/CC thought/VBD it/PRP was/VBD a/DT fun/NN movie/NN ./. It/PRP cracked/VBD me/PRP up/IN a/DT couple/NN times/NNS ./.
u_5	17	The/DT Flash/NNP has/VBZ done/VBN a/DT fantastic/JJ job/NN of/IN incorporating/VBG classic/NN from/IN the/DT hero/NN's/POS comic/JJ book/NN history/NN
—	—	—
u_n	—	—

Table 8: Movie reviews' semantic score.

		Aggregated semantic score		
Movie ID	Review ID	CinemaBlend	Moviefone	Rotten Tomatoes
m_1	d_1	0.4	0.6	0.2
m_2	d_2	0.2	0.2	0.4
m_3	d_3	0.2	0.6	0.6
m_4	d_4	0.6	0.2	0.4
m_5	d_5	0.8	0.0	0.0

Table 9: Twitter likes.

	Twitter likes	
Movie name	Unnormalized	Normalized
m_1	366	366
m_2	154	154
m_3	258	258
m_4	3	3
m_5	2196K	2169

paragraph vector: distributed memory model (PVDM) [69], topic modeling, and collaborative filtering JMARS (https://jmars.asu.edu/). Sentiment classification is by vector representation and text preference matrix for the user product neural network (UPNN) [70].

Table 13 presents the comparison between different sentiment classification models using and without using users/product/movies information.

In our approach, the core implementation is neural sentiment classification (NSC) using local user and movie information [71], which provides the significant result, as

Table 10: Movie rating.

Movie ID	Rating					
	Unnormalized			Normalized		
	CinemaBlend (%)	Moviefone (%)	Rotten Tomatoes (%)	CinemaBlend	Moviefone	Rotten Tomatoes
m_1	70	16	39	7	1.60	3.90
m_2	80	55	60	4.0	5.50	6.00
m_3	70	25	49	3.5	2.50	4.90
m_4	40	Nil	39	2.0	Nil	3.90
m_5	73	69	93	7.3	6.90	9.30

Table 11: Normalized movie votes.

Movie ID	Votes					
	Unnormalized			Normalized		
	CinemaBlend	Moviefone	Rotten Tomatoes	CinemaBlend	Moviefone	Rotten Tomatoes
m_1	1.5K	679	4.5K	1500	679	4500
m_2	870	760	6.5K	870	760	6500
m_3	797	890	94	797	890	94
m_4	3.46K	6.9K	910	3460	6900	910
m_5	67	76	8.3K	670	760	8300

Table 12: Final score, movie category, medal rank, and genres.

Movie ID	Final score	Genre category	Medal rank	Recommendation of movie
m_1	1.30	Action	Copper	Highly Unpopular
m_2	4.41	Adventure	Silver	Average Popular
m_3	3.63	Fantasy	Bronze	Unpopular
m_4	4.59	Horror	Silver	Average Popular
m_5	4.47	Science fiction	Silver	Average Popular

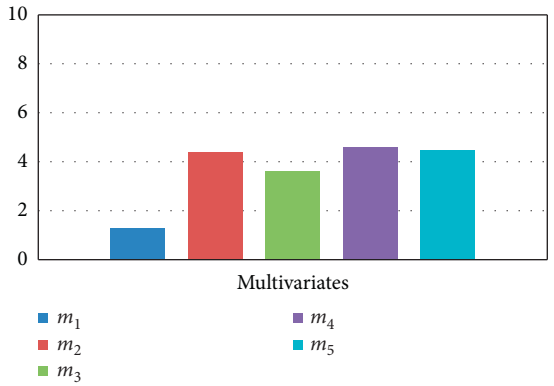


Figure 6: Multivariate movie ranked recommendation.

shown in Table 13, which represents the significant 4% improvement/difference with all the baseline methods, which use the local textual information about users and movies. While using global information about users and movies, the UPNN gains 3% improvement, and our approach NSC-UMA achieves 9% improvement. Our approach uses the vector for embedding the user and movie information, which is suitable for larger datasets, while the UPNN uses the matrix and vector simultaneously. NSC-UMA is considerable for capturing the information from each semantic layer. Therefore, our model incorporates using user movie global information in an efficient and effective way.

Word-level attention and sentence-level attention are considerable to outperform to reflect the semantic information of user and movie characteristics at multiple levels, which leads to introduction of the user movie attention (UMA) in sentiment classification. Furthermore, perceptions of user taste or preferences are more understandable than movie attributes, so both user and movie

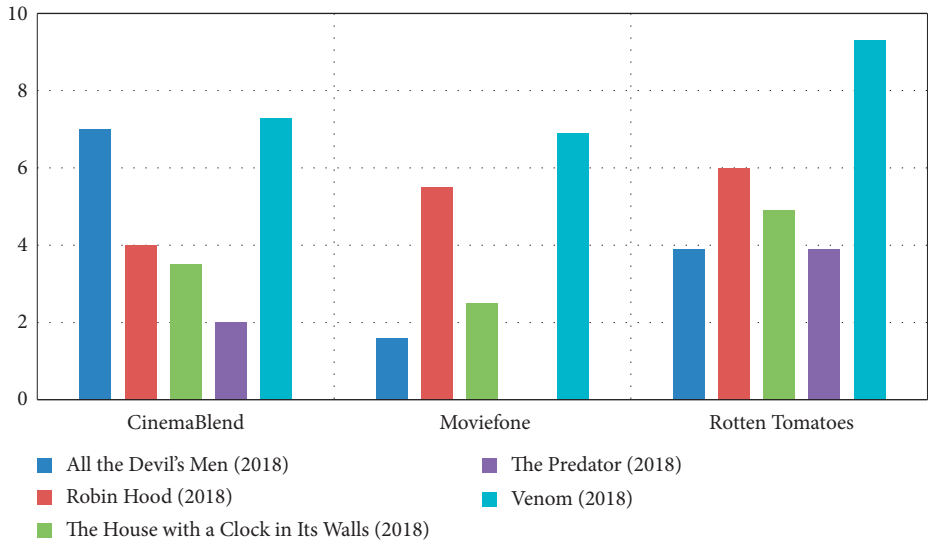


Figure 7: Rating difference.

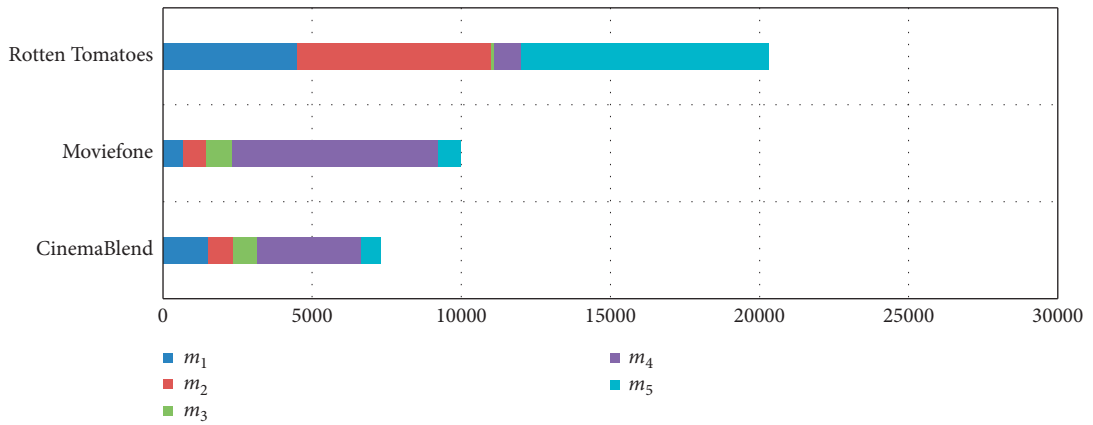


Figure 8: Vote difference.

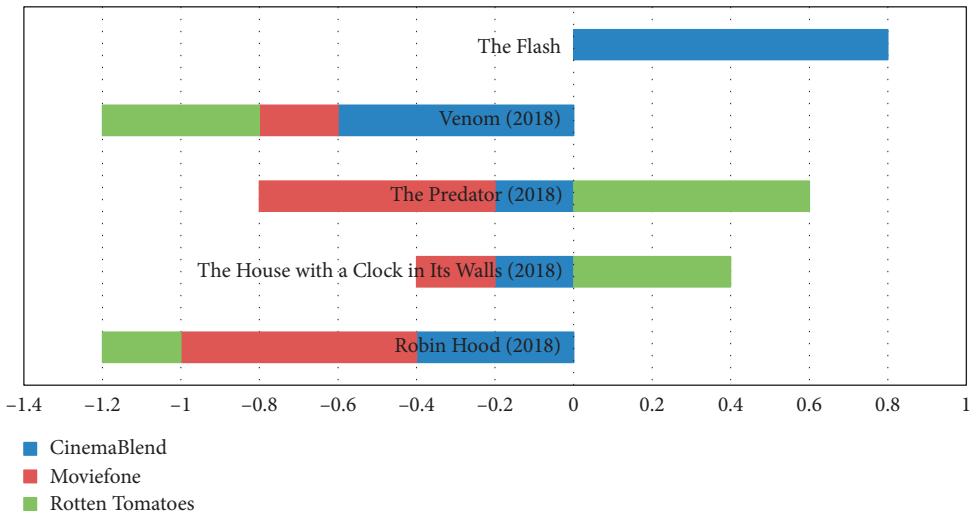


Figure 9: Differences in sentiment scores.

Table 13: Comparison between sentiment classification models.

Classification models	IMDB		Yelp 2013		Yelp 2014	
	Accuracy	RMSE	Accuracy	RMSE	Accuracy	RMSE
Without using user and product information						
Majority	0.196	2.495	0.411	1.060	0.392	1.097
Trigram	0.399	1.783	0.569	0.814	0.577	0.804
Text feature	0.402	1.793	0.556	0.845	0.572	0.800
AvgWordvec + SVM	0.304	1.985	0.526	0.898	0.530	0.893
SSWE + SVM	0.312	1.973	0.549	0.849	0.557	0.851
Paragraph vector	0.341	1.814	0.554	0.832	0.564	0.802
RNTN + recurrent	0.400	1.764	0.574	0.804	0.582	0.821
CNN and without UP (UPNN)	0.405	1.629	0.577	0.812	0.585	0.808
NSC	0.443	1.465	0.627	0.701	0.637	0.686
NSC + LA	**0.487**	**1.381**	**0.631**	**0.706**	**0.630**	**0.715**
Using user and product information						
Trigram + UPF	0.404	1.764	0.570	0.803	0.576	0.789
Text feature + UPF	0.402	1.774	0.561	1.822	0.579	0.791
JMARS	N/A	1.773	N/A	0.985	N/A	0.999
UPNN (CNN)	0.435	1.602	0.596	0.784	0.608	0.764
UPNN (NSC)	0.471	1.443	0.631	0.702	N/A	N/A
NSC + UMA	**0.533**	**1.281**	**0.650**	**0.692**	**0.667**	**0.654**

information is essential to pay attention in the document for semantic information which impacts movie ranking for recommendation.

6.2. Comparative Analysis of Recommendation Models. Table 14 presents the comparison between different models. Major differences which show our work as a novel approach are that the first one is LSTM-UMA for sentiment classification, the second one is the NoSQL distributed environment to deal with the big data issues, the third one is the multivariate (qualitative and quantitative) score fetched by a web bot from three different reliable external data sources, and the fourth one is app features (movie category and popularity).

In [15] which only uses the implicate and explicate ratings, no user and production attention are used by LSTM, adoptive deep learning is not used to determine the preference and taste of users about a movie from microblogs, so we can say that the authors did not use the qualitative data. It does not declare how data are fetched nor categorized with their popularity. In [15, 72–74], multivariates are not used, and the study [73] is just based on microblogs, while the study [74] uses movie feature ratings.

6.3. Results of the Experiments. True positive (recommended interesting movie) predicted by a search divided by actual movies (total movies) is called precision:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{17}$$

True positive (recommended interesting movie to users) predicted by a search divided by predicted movies (totally recommended movies) is called recall:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{18}$$

Figure 10 presents different decisions made by the movie recommender.

If the recommender grows in precision, then recall is declined:

$$F \text{ score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{19}$$

Table 15 presents the comparisons of the parameters of the multivariate recommendation system and other models.

Table 16 presents the evaluation of the multivariate recommender system with other parameters and other works.

The results of F score in our system are compared with other predicting parameters as well as with other recommendation frameworks. The accuracy of the multivariate system is nearly about 98.70%. The true positive rate of the multivariate system is 0.99106, which means the system recommended movies truly interested for users, and the false positive rate is 0.01814, which means the multivariate system did not recommend movies truly not interested for users.

Figure 11 presents differences in different decision parameters (precision, recall, and accuracy) for recommendation of movies from movie sites CinemaBlend, Moviefone, and Rotten tomatoes and other recommendation models.

Figure 11 justifies the difference between our approach and other works [21, 74]. In [21], just finding the polarity of the term is not enough to evaluate the reviews for significant recommendation, and in [74], LSTM is used to determine the user and movie information, while we used NSC-UMA to evaluate the sentiment score of reviews for significant recommendation.

Table 14: Recommendation model comparisons.

Ref.	NLP/ RNN/ LSTM	User preferences	NoSQL	Heterogeneous data	Quantitative score (votes, likes, and ratings)	Qualitative score (analysis of reviews)	Multivariates	Multiple data source sites	Popularity medals	Web bot	Categories	User app
[15]	×	✓	✓	×	✓	×	×	×	×	×	×	✓
[72]	✓	✓	✓	×	×	✓	×	×	×	×	×	✓
[73]	✓	✓	×	×	×	✓	×	×	×	×	×	×
[74]	✓	✓	×	×	×	×	×	×	×	×	×	×
[21]	✓	×	×	✓	✓	×	✓	✓	×	✓	✓	✓
Proposed work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

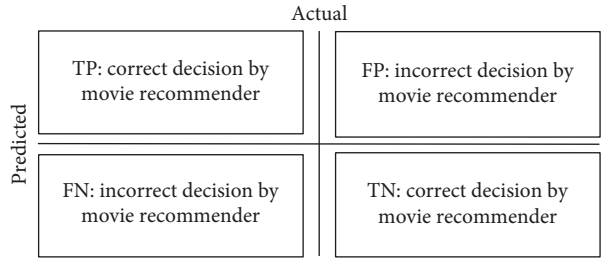


Figure 10: Different decisions made by the movie recommender. TP: the recommended movie is interesting. TN: the recommended movie is uninteresting. FP: the recommended movie is actually interesting. FN: the recommended movie is actually uninteresting.

Table 15: Comparisons between different recommendation decision parameters.

Decision parameters	TP	TN	FP	FN
AS	341	237	207	215
WAS	375	355	95	175
AWAS	406	347	116	131
[74]	525	250	90	135
[21]	442	387	114	57
Multivariate system	554	433	8	5

Table 16: Results of the experiments.

	AS	WAS	AWAS	[74]	[21]	Multivariate system
Precision	0.6223	0.7979	0.7778	0.8537	0.7950	0.9858
Recall	0.6133	0.6818	0.7561	0.7955	0.8858	0.9911
F score	0.6178	0.7353	0.7668	0.8235	0.8379	0.9884
Accuracy	0.5780	0.7300	0.7530	0.7750	0.8290	0.9870

Figure 11: Differences in different decision parameters precision, recall, *F* score, and accuracy and recommendation models.

7. Conclusion

In many movie recommendation systems, suggestion and ranking are done on the bases of only likes or ratings. Furthermore, most of the systems extract data from only one or two sites. Semantics, ratings, or votes for the movie ranking are not reliable and insignificant as they could not provide better recommendation services and there is a huge gap between statistical information (ratings, votes, and likes) and reviews of movie websites; so they are not reliable using from one site as a few of the websites are producing the qualitative score showing high popularity of a movie and other ones are showing low popularity of the same movie. In study [21], deep learning is not used and only word frequency is used. Word frequency is no better way to evaluate the reviews semantically. It only produces the polarity of the term. Therefore, significant values and semantic information are required in an efficient way using the LSTM-attention learning algorithm for better semantic analysis, so semantic emotional value increases the significance of the recommendation system. Document semantic classification is improved by using the user movie attention at the word and sentence levels by the average pooling of word and sentence level to improve the semantic and emotion information

about reviews or document. The reason for applying the attention at the word level is to improve the semantic information of the document as compared to only applying it at the sentence level. Big data issue is covered by adopting the NoSQL environment.

8. Future Work

This work may be enhanced by adding more parameters like session, playlist, users group, session, smiley, tag, context, the feature of movie and video content to improve the work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] K. Ratnaparkhi, *Recommender System for Food in a Restaurant Based on Natural Language Processing and Machine Learning*, National College of Ireland, Dublin, Ireland, 2018.

[2] R. Mukherjee, P. S. Dutta, and S. Sen, *Movies2go—A New Approach to Online Movie Recommendation*, Department of Mathematical & Computer Sciences, University of Tulsa, Tulsa, OK, USA, 2001.

[3] U. Fasahte, D. Gambhir, M. Merulingkar, A. Monde, and A. Pokhare, "Hotel recommendation system, information technology, Atharva College of Engineering, India," *Imperial Journal of Interdisciplinary Research (IJIR)*, vol. 3, no. 11, 2017.

[4] P. M. Chawan, K. Suvarna, Y. Goyal, J. Thakker, and A. Bandiwadekar, "Recommendation system for dining, department of computer engg and info. tech. VJTI, Mumbai, Maharashtra, India," *Journal of Engineering Research and Application*, vol. 8, no. 5, pp. 48–52, 2018.

[5] J. Neidhardt, T. Kuflik, and W. Wörndl, "Special section on recommender systems in tourism," *Information Technology & Tourism*, vol. 19, no. 1–4, pp. 83–85, 2018.

[6] E. U. Okon, B. O. Eke, and P. O. Asagba, "An improved online book recommender system using collaborative filtering algorithm," *International Journal of Computer Applications*, vol. 179, no. 46, pp. 41–48, 2018.

[7] J. A. Xu and K. Araki, "An SVM-based personal recommendation system for TV programs," in *Proceedings of the 12th International Multi-Media Modelling Conference*, Beijing, China, January 2006.

[8] S. Baluja, R. Seth, D. Sivakumar et al., "Video suggestion and discovery for YouTube: taking random walks through the view graph," in *Proceeding of the 17th International Conference on World Wide Web-WWW'08*, Beijing, China, April 2008.

[9] E. Aalipour and M. Ghazisaeedi, "Recommender system introduction for requests of cancer world," *International Journal of Community Medicine and Public Health*, vol. 4, no. 2, pp. 275–280, 2017.

[10] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 95–116, 2017.

[11] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[12] J. Stan, F. Muhlenbach, and C. Largeron, "Recommender systems using social network analysis: challenges and future trends," in *Encyclopedia of Social Network Analysis and Mining*, pp. 1–22, Springer, Berlin, Germany, 2014.

[13] S. K. T. Lam, D. Frankowski, and J. Riedl, *Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems*, GroupLens Research Computer Science and Engineering, University of Minnesota Minneapolis, Minneapolis, MN, USA, 2006.

[14] K.-R. Kim, J.-H. Lee, J.-H. Byeon, and N.-M. Moon, *Recommender System Using the Movie Genre Similarity in Mobile Service*, Department of IT App. Tech. GSV, Hoseo University, Asan, South Korea, 2010.

[15] M.-Y. Hsieh, W.-K. Chou, and K.-C. Li, "Building a mobile movie recommendation service by user rating and APP usage with linked data on Hadoop," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3383–3401, 2017.

[16] K.-R. Kim and N. Moon, "Recommender system design using movie genre similarity and preferred genres in SmartPhone," *Multimedia Tools and Applications*, vol. 61, no. 1, pp. 87–104, 2012.

[17] M. Gao and X. Zhang, *A movie recommender system from tweets data*, January 2019, http://cs229.stanford.edu/proj2015/299_report.pdf.

[18] K. Badge and Jyoti Patil, "A survey on product recommendation systems using social data and microblogging information," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 12, no. 5, pp. 17513–17517, 2017.

[19] Y. Ishida, T. Uchiya, and I. Takumi, "Design and evaluation of a movie recommendation system showing a review for evoking interested," *International Journal of Web Information Systems*, vol. 13, no. 1, pp. 72–84, 2017.

[20] I. Vagliano, D. Monti, and M. Morisio, "SemRevRec: a recommender system based on user reviews and linked data," in *Proceedings of the RecSys 2017 Posters*, Como, Italy, August 2017.

[21] M. Ibrahim and I. Bajwa, "Design and application of a multivariant expert system using apache hadoop framework," *Sustainability*, vol. 10, no. 11, p. 4280, 2018.

[22] F. Ren and C. Quan, "Linguistic-based emotion analysis and recognition for measuring consumer satisfaction: an application of affective computing," *Information Technology and Management*, vol. 13, no. 4, pp. 321–332, 2012.

[23] K. Wakil, R. Bakhtyar, K. Ali, and K. Aladdin, "Improving web movie recommender system based on emotions," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 2, 2015.

[24] Y. Tom, D. Hazarikaz, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," 2018, https://arxiv.org/pdf/1708.02709.

[25] S. Postmus and S. Bhulai, *Recommender System Techniques Applied to Netflix Movie Data*, Vrije Universiteit Amsterdam, Amsterdam, Netherlands, 2018.

[26] P. T. Nguyen, P. Tomeo, T. Di Noia, and E. Di Sciascio, "Content-based recommendations via DBpedia and Freebase: a case study in the music domain," SisInf Lab, Polytechnic University of Bari, Bari, Italy, 2015.

[27] R. Mirizzi, T. Di Noia, A. Ragone, and V. C. Ostuni, "Movie recommendation with DBpedia," Polytechnic University of Bari, Bari, Italy, 2012.

[28] X.-Y. D. ZhenWu, C. Yin, S. Huang, and J. Chen, "Improving review representations with user attention and product attention for sentiment classification," 2018, https://arxiv.org/abs/1801.07861.

[29] R. Ferreira, O. Holanda, J. Melo, I. Ibert, F. Freitas, and E. Costa, *An Agent-Based Semantic Web Blog*, Federal University of Alagoas, Maceió, Brazil, 2012.

[30] W. W. Cohena and W. Fan, "Web-collaborative filtering: recommending music by crawling the Web," *Computer Networks*, vol. 33, no. 1–6, pp. 685–698, 2000.

[31] A. V. Jose and K. M. Jini, "Personalized movie recommender system using rank boosting approach on hadoop," *International Journal for Innovative Research in Science & Technology*, vol. 2, no. 2, 2015.

[32] G. Godhani and M. Dhamecha, "A study on movie recommendation system using parallel map reduce technology," *International Journal of Engineering Development and Research*, vol. 5, no. 1, 2018.

[33] B. Li, Y. Liao, and Q. Zheng, "Precomputed clustering for movie recommendation system in real time," *Journal of Applied Mathematics*, vol. 2014, Article ID 742341, 9 pages, 2014.

[34] L. Zhao, Z. Lu, S. Jialin, and Q. Y. Pan, "Matrix factorization+ for movie recommendation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, Hong Kong University of Science and Technology, Hong Kong Nanyang Technological University, New York, NY, USA, July 2016.

[35] J. Nessel and B. Cimpa, "The movie oracle-content based movie recommendations," in *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, EdgeWorks Software Ltd., Lyon, France, August 2011.

[36] D. Sánchez-Moreno, B. Ana, G. Gil et al., *A Collaborative Filtering Method for Music Recommendation Using Playing Coefficients for Artists and Users*, Department of Computing and Automation, University of Salamanca, Salamanca, Spain, 2016.

[37] K. Soni, R. Goyal, B. Vadera, and S. More, "A three-way hybrid movie recommendation system," *International Journal of Computer Applications*, vol. 160, no. 9, pp. 29–32, 2017.

[38] H. Khalid and S. Wu, "Reducing the cold-start problem by explicit information with mathematical set theory in recommendation systems," *International Journal of Engineering and Computer Science*, vol. 5, no. 8, pp. 17613–17626, 2016.

[39] A. Sarker, R. Ginn, A. Nikfarjam et al., "Utilizing social media data for pharmacovigilance: a review," *Journal of Biomedical Informatics*, vol. 54, pp. 202–212, 2015.

[40] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with Twitter: what 140 characters reveal about political sentiment," *ICWSM*, vol. 10, pp. 178–185, 2010.

[41] W. He, S. Zha, and L. Li, "Social media competitive analysis and text mining: a case study in the pizza industry," *International Journal of Information Management*, vol. 33, no. 3, pp. 464–472, 2013.

[42] E. L. Murnane and S. Counts, "Unraveling abstinence and relapse: smoking cessation reflected in social media," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1345–1354, ACM, Toronto, Canada, April 2014.

[43] N. Diakopoulos, M. Naaman, and F. Kivran-Swaine, "Diamonds in the rough: social media visual analytics for journalistic inquiry," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pp. 115–122, IEEE, Salt Lake City, UT, USA, October 2010.

[44] T. Baldwin, P. Cook, M. Lui, A. MacKinlay, and L. Wang, *How Noisy Social Media Text, How Different Social Media Sources?*, University of Melbourne, Melbourne, Australia, 2013.

[45] C. Corley, D. Cook, A. Mikler, and K. Singh, "Text and structural data mining of influenza mentions in web and social media," *International Journal of Environmental Research and Public Health*, vol. 7, no. 2, pp. 596–615, 2010.

[46] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, https://arxiv.org/abs/1506.00019.

[47] W. Yiny, K. Kanny, Y. Mo, and H. Schützey, "Comparative study of CNN and RNN for natural language processing," 2017, https://arxiv.org/abs/1702.01923.

[48] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[49] J. A. Konstan, "Introduction to recommender systems: algorithms and evaluation," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 1–4, 2004.

[50] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of net news," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, pp. 175–186, ACM, Chapel Hill, NC, USA, October 1994.

[51] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, pp. 210–217, ACM Press/Addison-Wesley Publishing Co., Denver, CO, USA, May 1995.

[52] A. Chang, J. F. Liao, P. C. Chang, C. H. Teng, and M. H. Chen, "Application of artificial immune systems combines collaborative filtering in the movie recommendation system," in *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 277–282, IEEE, Hsinchu, Taiwan, May 2014.

[53] E. Reich, "Users are individuals: individualizing user models," *International Journal of Man-Machine Studies*, vol. 18, no. 3, pp. 199–214, 1983.

[54] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[55] S.-H. Min and I. Han, "Detection of the customer time-variant pattern for improving recommender systems," *Expert Systems with Applications*, vol. 28, no. 2, pp. 189–199, 2005.

[56] B. N. Miller, J. A. Konstan, and J. Riedl, "PocketLens," *ACM Transactions on Information Systems*, vol. 22, no. 3, pp. 437–476, 2004.

[57] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Group lens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[58] Y. H. Cho, J. K. Kim, and S. H. Kim, "A personalized recommender system based on web usage mining and decision tree induction," *Expert Systems with Applications*, vol. 23, no. 3, pp. 329–342, 2002.

[59] R. Schaefer, "Rules for using multi-attribute utility theory for estimating a user's interests," in *Proceedings of the ABIS Workshop, Adaptivität und Benutzermodellierung in*

interaktiven Softwaresystemen, Dortmund, Germany, October 2001.

[60] M. Paul, *Providing Actionable Recommendations: A Movie Recommendation Algorithm with Explanatory Capability*, Joseph Eul Verlag, Siegburg, Germany, 2013.

[61] Y. Zhang, M. Zhang, and Y. Liu, "Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pp. 435–440, Shanghai, China, February 2015.

[62] I. S. Bajwa and S. Iqbal, "A semi supervised semantic analysis of natural language constraints using Drt and Markov logic," *Science International*, vol. 28, no. 3, pp. 2783–2790, 2016.

[63] X. Su and T. M. Khoshgoftaar, *A Survey of Collaborative Filtering Techniques*, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, USA, 2009.

[64] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, LA, USA, February 2018.

[65] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[66] I. Bajwa, H. Ismail, H. Bukhari, and R. Amin, "Automated sentiment analysis of natural language text using machine learning," *Sindh University Research Journal-Surj (Science Series)*, vol. 48, no. 3, 2016.

[67] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1555–1565, Baltimore, MD, USA, June 2014.

[68] R. Socher, A. Perelygin, J. Y. Wu et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the Empirical Methods in Natural Language Processing*, p. 1642, Seattle, WA, USA, October 2013.

[69] U. Ghani, I. Bajwa, and A. Ashfaq, "A fuzzy logic based intelligent system for measuring customer loyalty and decision making," *Symmetry*, vol. 10, no. 12, p. 761, 2018.

[70] K. Yoon, "Convolutional neural networks for sentence classification," in *Proceedings of the EMNLP*, Doha, Qatar, October 2014.

[71] Z. Yang, D. Yang, C. Dyer, X. He, Alex Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, CA, USA, June 2016.

[72] Y. Wang, M. Wang, and W. Xu, "A sentiment-enhanced hybrid recommender system for movie recommendation: a big data analytics framework," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8263704, 9 pages, 2018.

[73] S. Kumar, S. S. Halder, K. De, and P. P. Roy, "Movie recommendation system using sentiment analysis from microblogging data," 2018, https://arxiv.org/abs/1811.10804.

[74] H.-T. Zheng, J.-Y. Chen, N. Liang, A. K. Sangaiah, Y. Jiang, and C.-Z. Zhao, "A deep temporal neural music recommendation model utilizing music and user metadata," *Applied Science*, vol. 9, no. 4, p. 703, 2019.

Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 8097213, 13 pages
https://doi.org/10.1155/2019/8097213

Research Article

Image Processing-Based Detection of Pipe Corrosion Using Texture Analysis and Metaheuristic-Optimized Machine Learning Approach

Nhat-Duc Hoang [1] **and Van-Duc Tran** [2]

[1] *Lecturer, Faculty of Civil Engineering, Institute of Research and Development, Duy Tan University, R.809–No.03 Quang Trung, Da Nang 550000, Vietnam*
[2] *Lecturer, International School, Duy Tan University, 254 Nguyen Van Linh, Danang 550000, Vietnam*

Correspondence should be addressed to Nhat-Duc Hoang; hoangnhatduc@dtu.edu.vn

Received 27 March 2019; Revised 21 May 2019; Accepted 17 June 2019; Published 11 July 2019

Academic Editor: Juan A. Gómez-Pulido

To maintain the serviceability of buildings, the owners need to be informed about the current condition of the water supply and waste disposal systems. Therefore, timely and accurate detection of corrosion on pipe surface is a crucial task. The conventional manual surveying process performed by human inspectors is notoriously time consuming and labor intensive. Hence, this study proposes an image processing-based method for automating the task of pipe corrosion detection. Image texture including statistical measurement of image colors, gray-level co-occurrence matrix, and gray-level run length is employed to extract features of pipe surface. Support vector machine optimized by differential flower pollination is then used to construct a decision boundary that can recognize corroded and intact pipe surfaces. A dataset consisting of 2000 image samples has been collected and utilized to train and test the proposed hybrid model. Experimental results supported by the Wilcoxon signed-rank test confirm that the proposed method is highly suitable for the task of interest with an accuracy rate of 92.81%. Thus, the model proposed in this study can be a promising tool to assist building maintenance agents during the phase of pipe system survey.

1. Introduction

In high-rise building maintenance, an important objective is concerned with the integrity of the water supply system and prevention of water contamination. Cast iron is widely used in water supply and waste disposal systems due to the advantage of high strength. Since stainless steel pipes often fall out of favor in domestic pipework because of their high expenses [1], corrosion is a widely observed type of structural damage.

Corrosion (see Figure 1) can be defined as a chemical process caused by chemical and electrochemical reactions. This phenomenon is typically observed in environmental conditions featuring a high level of moisture. There are different kinds of corrosion such as general corrosion which occurs as uniformly distributed nonprotective flakes of rust and pitting which is a localized point of corrosive attack [2]. Corrosion brings about the destruction of metal pipework surface and consequently leads to reduction in pipe service life and increase in building maintenance cost [3]. In certain case, this defect may strongly affect the health of building occupants due to deterioration of water quality. Thus, corrosion should be identified timely by means of periodic surveys to ensure the integrity of pipe systems and establish cost-effective maintenance strategies.

In Vietnam as well as in many other countries, manual methods performed by human inspectors are commonly employed for condition assessment of water supply/waste disposal systems. As clearly pointed out by Liu et al. [4] and Atha and Jahanshahi [5], these manual approaches are labor intensive and time consuming. Corroded regions can be neglected in positions of pipe system that are difficult to reach and observe visually. Moreover, the processes of data processing and reporting are also very tedious for human

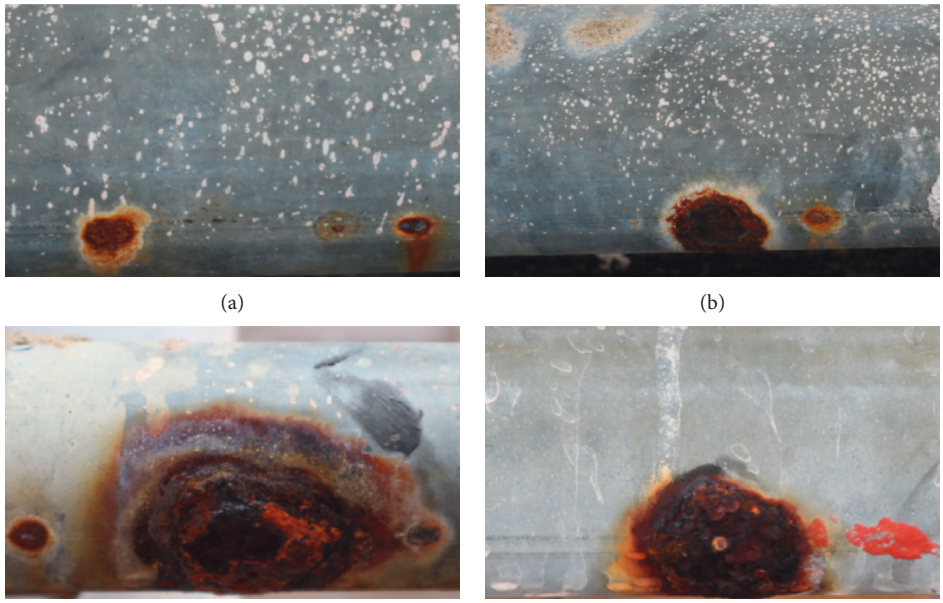


Figure 1: Corroded areas on pipe surface.

technicians. Therefore, there is a practical need to come up with a more productive and accurate method of pipe condition survey.

Although there is a wide range of existing pipe inspection approaches (such as magnetic flux leakage, ultrasonic testing, and external corrosion direct assessment), all of these methods have limitations including high equipment cost, restricted range of inspection, and incapability of detecting small pitting regions [3]. Considering the large amount of pipe systems needed to be surveyed and the limited access to sophisticated equipment in developing countries, there is an urgent need for a productive and low-cost solution for periodic surveys of pipe system condition. Recently, digital image processing has gained a great attention within the field of structural heath monitoring [6, 7].

Particularly, image processing techniques can be effectively employed to investigate the outer surface for detecting defects on pipes or other metal structures including corrosion and cracks [8]. Itzhak et al. [9] relied on statistic measurement of image pixels to quantify pitting corrosion. Choi and Kim [10] identified corrosion based on the morphology of the corroded surface; features of image color, texture, and shape are employed for corrosion recognition. A model for classifying corroded and noncorroded surfaces using texture descriptors obtained from gray level co-occurrence matrix and image color has been proposed in Medeiros et al. [11].

A method based on watershed segmentation has been employed in [12] for rating of corrosion defects; the percentage area of corroded region was used for determining the grade of defects. Idris and Jafar [13] used image filter-based image enhancement and neural network for corrosion inspection. Son et al. [14] proposed a model based on decision tree algorithm for identifying rusted surface area of steel bridge. A model based on image color analysis and K-means clustering for bridge rust identification has been constructed and verified by Liao and Lee [15].

Petricca et al. [16] compared standard computer vision techniques and deep neural network for rust and nonrust detection. Deep neural networks have also been employed for corrosion detection by Liu et al. [4] and Atha and Jahanshahi [5]. Safari and Shoorehdeli [17] applied artificial neural network, Gabor filter, and entropy filter for pipe defect detection. Cheriet et al. [18] incorporated expert knowledge and field data to construct a knowledge-based system for assessing corrosive damage on metallic pipe conduits. Gibbons et al. [19] relied on a Gaussian mixture model for probabilistic classification of corroded and noncorroded areas. Bondada et al. [3] detected and quantitatively assessed corrosion damages on pipelines by computing the mean of saturation value of image pixels; by image analysis, the corroded areas on pipelines can be segmented.

From the above literature, it can be seen that image processing and machine learning have been a feasible alternative for replacing the tedious process of manual survey. Based on a recent review work of Ahuja and Shukla [20], there is an increasing trend of applying computer vision techniques for corrosion detection. Moreover, due to the importance of the research theme, exploring other image processing and machine learning methods used for pipe corrosion detection can be highly meaningful in both academic and practical aspects.

As reported in the literature, although image texture analysis has been applied, few previous studies have employed a combination of image texture descriptors for pipe corrosion recognition. Hence, this study is an attempt to fill this gap in the literature by proposing a method used for analyzing texture of water pipe surface that integrates

statistical measurement of color channels, gray-level co-occurrence matrix, and gray-level run length matrix. Based on the features extracted by the above texture descriptors, the support vector machine (SVM) [21] is employed to categorize image samples into two classes: noncorrosion (negative) and corrosion (positive). SVM is utilized in this study due to the fact that it has been confirmed to be a robust tool for pattern classification in various studies [22–24]. In addition, to optimize the training process of SVM-based corrosion detection model, differential flower pollination (DFP) metaheuristic is employed. A dataset consisting of 2000 image samples has been collected to train and verify the proposed method.

The rest of the study is organized as follows. Section 2 reviews the research material and methods used to construct the water pipe corrosion detection approach. Section 3 reports experimental results and discussions. Section 4 provides several concluding remarks of this study.

2. Material and Methods

2.1. Image Texture Analysis. Identifying corroded areas based on two-dimensional image samples is a challenging task due to the complex and deceptive features of pipe surfaces containing various irregular objects such as dirt and paints. Therefore, using information provided by one pixel is definitely not sufficient for corrosion detection. It is because a pixel having similar color values can belong to both categories of noncorrosion and corrosion. Hence, texture information extracted from a certain region of pipe surface can be used for recognizing the defect of interest. This section of the study describes the employed texture descriptors used for computing the features of water pipe surface.

2.1.1. Statistical Properties of Color Channels. Herein, the statistical properties of three color channels (red, green, and blue) of an image sample can be employed to represent image texture. Thus, an image is described in a RGB color space. It is noted that besides RGB, there are other color spaces such as HSV which can also be useful in the task of corrosion detection. However, in this study, we rely on the original RGB color model obtained from the employed digital camera. Let I be a variable representing the color levels of an image sample. The first-order histogram $P(I)$ is calculated as follows [25]:

$$P_{\mathrm{c}}(I) = \frac{N_{I,\mathrm{c}}}{W \times H}, \tag{1}$$

where c denotes a color channel, $N_{I,\mathrm{c}}$ is the number of pixels with intensity value I of the channel c, and H and W represent the height and width of an image sample, respectively.

Thus, the mean (μ_{c}), standard deviation (σ_{c}), skewness (δ_{c}), kurtosis (η_{c}), entropy (ρ_{c}), and range (Δ_{c}) of color value are calculated as follows:

$$\begin{aligned}
\mu_{\mathrm{c}} &= \sum_{i=0}^{\mathrm{NL}-1} I_{i,\mathrm{c}} \times P_{\mathrm{c}}(I),\\
\sigma_{\mathrm{c}} &= \sqrt{\sum_{i=0}^{\mathrm{NL}-1} \left(I_{i,\mathrm{c}} - \mu_{\mathrm{c}}\right)^2 \times P_{\mathrm{c}}(I)},\\
\delta_{\mathrm{c}} &= \frac{\sum_{i=0}^{\mathrm{NL}-1} \left(I_{i,\mathrm{c}} - \mu_{\mathrm{c}}\right)^3 \times P_{\mathrm{c}}(I)}{\sigma_{\mathrm{c}}^3},\\
\eta_{\mathrm{c}} &= \frac{\sum_{i=0}^{\mathrm{NL}-1} \left(I_{i,\mathrm{c}} - \mu_{\mathrm{c}}\right)^4 \times P_{\mathrm{c}}(I)}{\sigma_{c}^4},\\
\rho_{\mathrm{c}} &= -\sum_{i=0}^{\mathrm{NL}-1} P_{\mathrm{c}}(I) \times \log_2\left(P_{\mathrm{c}}(I)\right),\\
\Delta_{\mathrm{c}} &= \mathrm{Max}\left(I_{\mathrm{c}}\right) - \mathrm{Min}\left(I_{\mathrm{c}}\right),
\end{aligned} \tag{2}$$

where NL = 256 denotes the number of discrete color values.

2.1.2. Gray-Level Co-Occurrence Matrix (GLCM). The GLCM [26] is also a commonly used texture descriptor. To employ this technique, a color image must first be converted to a gray scale one. The GLCM discriminates different image textures based on the repeated occurrence of some gray-level patterns existing in the texture [27]. Let $\delta = (r, \theta)$ be a vector in the polar coordinates of an image sample. For each δ, the joint probability of the pairs of gray levels that occur at the two points separated by the relationship δ is computed [28]. This joint probability is compactly displayed in a GLCM P_δ within which $P_\delta(i, j)$ represents the probability of the two gray levels of i and j occurring according to δ. The original $P_\delta(i, j)$ is often normalized via the following equation:

$$P_\delta^N(i, j) = \frac{P_\delta(i, j)}{S_{\mathrm{P}}}, \tag{3}$$

where $P_\delta^N(i, j)$ denotes the normalized GLCM and S_{P} is the number of pixels.

Based on the suggestion of Haralick et al. [29], four GLCMs with $r = 1$ and $\theta = 0^\circ$, 45°, 90°, and 135° can be established. Accordingly, angular second moment (AM), contrast (CO), correlation (CR), and entropy (ET) for each matrix can be computed to serve as texture descriptors as follows [28, 29]:

$$\begin{aligned}
\mathrm{AM} &= \sum_{i=1}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{g}}} P_\delta^N(i, j)^2,\\
\mathrm{CO} &= \sum_{k=0}^{N_{\mathrm{g}}-1} k^2 \sum_{\substack{i=1\\ |i-j|=k}}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{g}}} P_\delta^N(i, j),\\
\mathrm{CR} &= \frac{\sum_{i=1}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{g}}} i \times j \times P_\delta^N(i, j) - \mu_X \mu_Y}{\sigma_X \sigma_Y},\\
\mathrm{ET} &= -\sum_{i=1}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{g}}} P_\delta^N(i, j) \log\left(P_\delta^N(i, j)\right),
\end{aligned} \tag{4}$$

where N_g is the number of gray-level values and μ_X, μ_Y, σ_X, and σ_Y are the means and standard deviations of the marginal distribution associated with a normalized GLCM [29].

2.1.3. Gray-Level Run Lengths (GLRL). GLRL is a texture description method proposed by Galloway [30]. This method is highly effective in discriminating textures featuring different fineness and has been successfully applied in various fields of study [31, 32]. It is because GLRL is constructed based on the fact that relatively long gray-level runs are observed more frequently in a coarse texture and a fine texture typically has more short runs [33]. A run-length matrix $p(i \cdot j)$ in a certain direction is defined as the number of times that a run length j of gray level i is observed [30].

Using this matrix, the short run emphasis (SRE), long run emphasis (LRE), gray-level nonuniformity (GLN), run length nonuniformity (RLN), and run percentage (RP) [30, 33] can be computed. Additionally, Chu et al. [34] put forward the indices of low gray-level run emphasis (LGRE) and high gray-level run emphasis (HGRE). Dasarathy and Holder [35] proposed to compute the short run low gray-level emphasis (SRLGE), short run high gray-level emphasis (SRHGE), long run low gray-level emphasis (LRLGE), and long run high gray-level emphasis (LRHGE). The above indices are summarized in Table 1. It is noted that one run length matrix is computed for each of direction in the set of $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ and each matrix results in 11 GLRL-based features. Therefore, the total number of features obtained from GLRL matrices is $11 \times 4 = 44$.

2.2. Computational Intelligence Methods

2.2.1. Support Vector Machine (SVM). SVM, described in [21], is a robust pattern recognition method established on the theory of statistical learning. Given the task at hand is to classify a set of input feature x_k into two categories of $y_k = -1$ (noncorrosion) and $y_k = +1$ (corrosion), a SVM model constructs a decision surface that separates the input space into two distinctive regions characterizing the two different two categories. The SVM algorithm aims at identifying a decision boundary so that the gap between classes is as large as possible [36]. In addition, SVM employs the kernel trick to convert a nonlinear classification task into a linear one. A SVM model first maps the input data from the original space to a high-dimensional feature space within which the data can be separated by a hyperplane (see Figure 2).

The SVM training process can be formulated as the following constrained optimization problem [36]:

$$\begin{aligned} &\text{minimize} \quad J_p(w,e) = \frac{1}{2}w^T w + c\frac{1}{2}\sum_{k=1}^{N} e_k^2 \\ &\text{subjected to} \quad y_k\left(w^T\varphi(x_k) + b\right) \geq 1 - e_k, \quad k = 1, \ldots, N, \; e_k \geq 0, \end{aligned} \tag{5}$$

where $w \in R^n$ is a normal vector to the classification hyperplane and $b \in R$ is the model bias; $e_k \geq 0$ is called a slack variable; c denotes a penalty constant; and $\varphi(x)$ is a nonlinear mapping from the input space to the high-dimensional feature space.

During the construction of a SVM model, it is not required to obtain the explicit form of $\varphi(x)$. Instead of that, only the dot product of $\varphi(x)$ in the input space is required and expressed via a kernel function shown as follows:

$$K(x_k, x_l) = \varphi(x_k)^T \varphi(x_l). \tag{6}$$

The radial basis function (RBF) kernel function [37] is often employed for data classification; its functional form is given below:

$$K(x_k, x_l) = \exp\left(-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right), \tag{7}$$

where σ is a free parameter.

Accordingly, a SVM model used for data classification is given compactly as follows:

$$y(x_l) = \operatorname{sign}\left(\sum_{k=1}^{SV} \alpha_k y_k K(x_k, x_l) + b\right), \tag{8}$$

where α_k denotes the solution of the dual form of the aforementioned constrained optimization. SV is the number of support vectors which is the number of $\alpha_k > 0$.

2.2.2. Differential Flower Pollination (DFP). As shown in the previous section, the model training and prediction phases of a SVM model depend on a proper selection of its hyperparameters including the penalty coefficient (c) and the kernel function parameter (σ). The first hyperparameter affects the penalty imposed on data samples deviating from the established decision surface; the later hyperparameter specifies the smoothness of the decision surface. Since this problem of hyperparameter selection can be formulated as an optimization problem [38–40], this study employs the DFP metaheuristic to optimize the model training phase of SVM.

DFP, proposed in [41], is a population-based metaheuristic that combines the advantages of the standard algorithms of differential evolution (DE) [42] and flower pollination algorithm (FPA) [43]. The employed hybrid metaheuristic consists of three main steps: initialization of population members, alteration of member locations, and cost function evaluation. Each member of the DFP metaheuristic is presented as a numerical vector consisting of the two SVM hyperparameters. In the first step, all population members are randomly generated within the feasible domain. In the second step, the location of population members is altered by local and global search phases. In the next step, the cost function of each member is computed and a greedy selection operator is performed to update the location of the DFP's population.

The second step of the DFP includes the FPA-based global pollination operator and the DE-based local pollination operator. A switching probability $p = 0.8$ is used to govern the frequencies of these two operators [43]. The FPA-

Table 1: Texture descriptors using GLRL.

Descriptor	Notation	Equation
Short run emphasis	SRE	$\mathrm{SRE} = 1/N_r \sum_{i=1}^{M} \sum_{j=1}^{N} p(i,j)/j^2$
Long run emphasis	LRE	$\mathrm{LRE} = 1/N_r \sum_{i=1}^{M} \sum_{j=1}^{N} p(i,j) \times j^2$
Gray-level nonuniformity	GLN	$\mathrm{GLN} = 1/N_r \sum_{i=1}^{M} (\sum_{j=1}^{N} p(i,j)^2)$
Run length nonuniformity	RLN	$\mathrm{RLN} = 1/N_r \sum_{j=1}^{N} (\sum_{i=1}^{M} p(i,j)^2)$
Run percentage	RP	$\mathrm{RP} = N_r/N_p$
Low gray-level run emphasis	LGRE	$\mathrm{LGRE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} p(i,j)/i^2$
High gray-level run emphasis	HGRE	$\mathrm{HGRE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} p(i,j) \times i^2$
Short run low gray-level emphasis	SRLGE	$\mathrm{SRLGE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} p(i,j)/(i^2 \times j^2)$
Short run high gray-level emphasis	SRHGE	$\mathrm{SRHGE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} (p(i,j) \times i^2)/j^2$
Long run low gray-level emphasis	LRLGE	$\mathrm{LRLGE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} (p(i,j) \times j^2)/i^2$
Long run high gray-level emphasis	LRHGE	$\mathrm{LRHGE} = 1/N_r \sum_{j=1}^{N} \sum_{i=1}^{M} p(i,j) \times i^2 \times j^2$

Note. M and N are the number of gray levels and the maximum run length, respectively. Let N_r and N_p be the total number of runs and the number of pixels in the image, respectively.

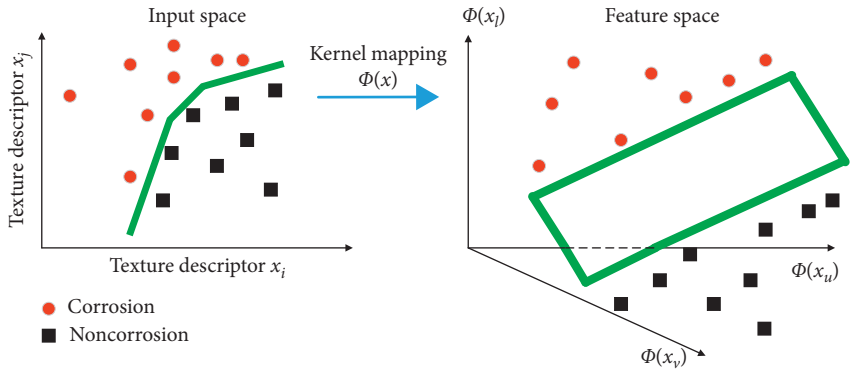


Figure 2: The data classification process of a SVM model.

based global pollination and the DE-based local pollination operators are presented as follows:

(i) The FPA-based global pollination:

$$X_i^{\mathrm{trial}} = X_i^{g_{\mathrm{DFP}}} + L \cdot \left(X_i^{g_{\mathrm{DFP}}} - X_{\mathrm{best}}\right), \tag{9}$$

where g is the index of the current generation, X_i^{trial} is a trial solution, $X_i^{g_{\mathrm{DFP}}}$ denotes a solution of the current population, X_{best} represents the best solution, and L denotes a random number generated from the Lévy distribution [43].

(ii) The DE-based local pollination modifies the current member by creating a mutated flower and a crossed flower according to the following equations:

(a) Creating a mutated flower:

$$X_{i,g_{\mathrm{DFP}}}^{\mathrm{mutated}} = x_{r1,g_{\mathrm{DFP}}} + F \cdot \left(x_{r2,g_{\mathrm{DFP}}} - x_{r3,g_{\mathrm{DFP}}}\right), \tag{10}$$

where $r1$, $r2$, and $r3$ are three random integers and F denotes a mutation scale factor which is drawn from a Gaussian distribution with the mean = 0.5 and the standard deviation = 0.15 [41].

(b) Creating a crossed flower:

$$X_{j,i,g_{\mathrm{DFP}}+1}^{\mathrm{crossed}} = \begin{cases} X_{j,i}^{\mathrm{mutated}}, & \text{if } r \text{ and } j \le \mathrm{Cr} \text{ or } j = \mathrm{rnb}(i), \\ X_{j,i,g_{\mathrm{DFP}}}, & \text{if } r \text{ and } j > \mathrm{Cr} \text{ and } j \ne \mathrm{rnb}(i), \end{cases} \tag{11}$$

where Cr = 0.8 is the crossover probability [44].

2.3. Collected Image Samples. Because SVM is a supervised machine learning algorithm, a dataset consisting of 2000 image samples of pipe surface with the ground truth label has been collected to construct the SVM-based corrosion detection model. It is proper to note that the numbers of image samples in the two labels of noncorrosion (negative class) and corrosion (positive class) are both 1000. The digital image samples have been collected during surveys of several high-rise buildings in Danang city (Vietnam). The used digital camera is the 18-megapixel resolution Canon EOS M10, and the images were manually acquired by human inspectors.

Accordingly, image samples of the two labels of noncorrosion (label = −1) and corrosion (label = +1) have been

prepared for SVM-based classification process. In order to accelerate the texture computation process, the size of image samples has been set to be 50×50 pixels. Hence, image cropping operation is performed to generate the image samples used to train the SVM model. The collected image set is illustrated in Figure 3.

2.4. Proposed Hybridization of Image Processing and Metaheuristic-Optimized SVM for Pipe Corrosion Detection. This section of the study describes the structure of the newly developed hybrid model of image processing and metaheuristic-optimized SVM for pipe corrosion detection. The proposed model, named as MO-SVM-PCD, is a combination of image texture analysis and a metaheuristic-optimized machine learning approach. As mentioned earlier, the statistical measurements of color channels, GLCM, and GLRL are used to extract texture-based features from image samples. The hybrid model relies on SVM to classify data samples into the categories of noncorrosion and corrosion. In addition, the DFP metaheuristic is employed to optimize the SVM-based training and prediction phases. The overall structure of the MO-SVM-PCD model is shown in Figure 4. The model structure can be divided into two separated modules: computation of image texture and data classification based on SVM. The first module is constructed in Visual C#.NET; the second module is developed in MATLAB.

Within the first module, the image texture descriptors based on statistical analysis of color channels, GLCM, and GLRL compute numerical features from image samples. For each of the three color channels (red, green, and blue), six statistical measurements of mean, standard deviation, skewness, kurtosis, entropy, and range are calculated. Hence, the total number of numerical features extracted from the aforementioned statistical indices of an image sample is $6 \times 3 = 18$. Subsequently, the group of features extracted from the four co-occurrence matrices corresponding to the directions of 0°, 45°, 90°, and 135° is computed. Because four indices of the angular second moment, contrast, correlation, entropy are acquired from one co-occurrence matrix, the total number of GLCM-based features is $4 \times 4 = 16$.

In addition, each GLRL matrix yields 11 properties of SRE, LRE, GLN, RLN, RP, LGRE, HGRE, SRLGE, SRHGE, LRLGE, and LRHGE. Thus, as stated earlier, the number of GLRL-based features is $4 \times 11 = 44$. Accordingly, each image sample is characterized by a feature vector having $18 + 16 + 44 = 78$ elements. This module can compute texture of one image for illustration purpose and can extract features from a batch of image samples to construct the training and testing numerical datasets.

When the module of feature computation is accomplished, a dataset consisting of 2000 data samples and 78 input features is ready for further analysis. This dataset has two class outputs: −1 meaning noncorrosion (negative class) and +1 meaning corrosion (positive class). In addition, for standardizing the data ranges and enhancing the data modeling process, the numerical dataset is preprocessed by the Z-score data normalization [45]. The equation of the Z-score data normalization is given as follows:

$$X_{\mathrm{ZN}} = \frac{X_{\mathrm{o}} - m_X}{s_X}, \tag{12}$$

where X_{o} and X_{ZN} represent an original and a normalized input variable, respectively, and m_X and s_X denote the mean and the standard deviation of the original input variable, respectively.

Subsequently, the normalized dataset is randomly divided into two subsets: a training set (70%) and a testing set (30%). The first data subset is employed for model training; the later data subset is reserved for model testing. The training dataset is employed by the SVM-based data classification module to generalize a corrosion recognition model. In addition, DFP is utilized to finetune the SVM model hyperparameters including the penalty coefficient and the RBF kernel parameter. It is worth mentioning that the SVM model operates via the help of the MATLAB's Statistics and Machine Learning Toolbox [46]; in addition, the DFP and the hybridization of DFP and SVM model have been constructed in MATLAB by the authors.

As shown in Figure 4, the two SVM hyperparameters are randomly initialized at the first generation ($g = 1$). Using the local and global pollination operators, the DFP algorithm gradually guides the population of SVM hyperparameters to explore the search space and identify better solutions. Based on the guidance of parameter setting in previous studies [44, 47], the population size and the number of DFP searching generations are selected to be 12 and 100. The feasible domain of the SVM's penalty coefficient and kernel parameter is [1, 100] and [0.1, 100], respectively. In the phase of solution evaluation, the quality of each member in the population is appraised via the following cost function:

$$f_{\mathrm{CF}} = \frac{\sum_{k=1}^{K} 2/\left(\mathrm{PPV}_k + \mathrm{NPV}_k\right)}{K}, \tag{13}$$

where $K = 5$ denotes the number of data folds and PPV and NPV are the positive predictive value and the negative predictive value. PPV and NPV are employed to express the model performance associated with a set of SVM hyperparameters.

PPV and NPV are computed according to the following equations [48]:

$$\begin{aligned} \mathrm{PPV} &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \\ \mathrm{NPV} &= \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}, \end{aligned} \tag{14}$$

where TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative values, respectively.

It is noted that to compute the model's cost function, a K-fold cross validation process with $K = 5$ is employed. Using this cross fold validation, the original dataset is separated into 5 mutually exclusive subsets. Accordingly, the SVM model training and evaluation is repeated 5 times. In each time, 4 subsets are utilized for model training and one subset is used for model validation. The overall model performance is obtained via averaging

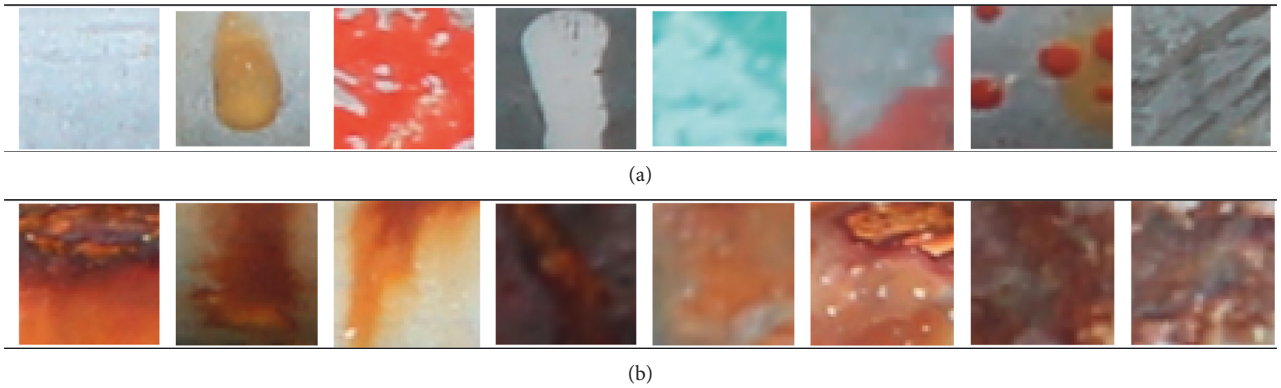


Figure 3: The collected image samples: (a) noncorrosion class and (b) corrosion class.

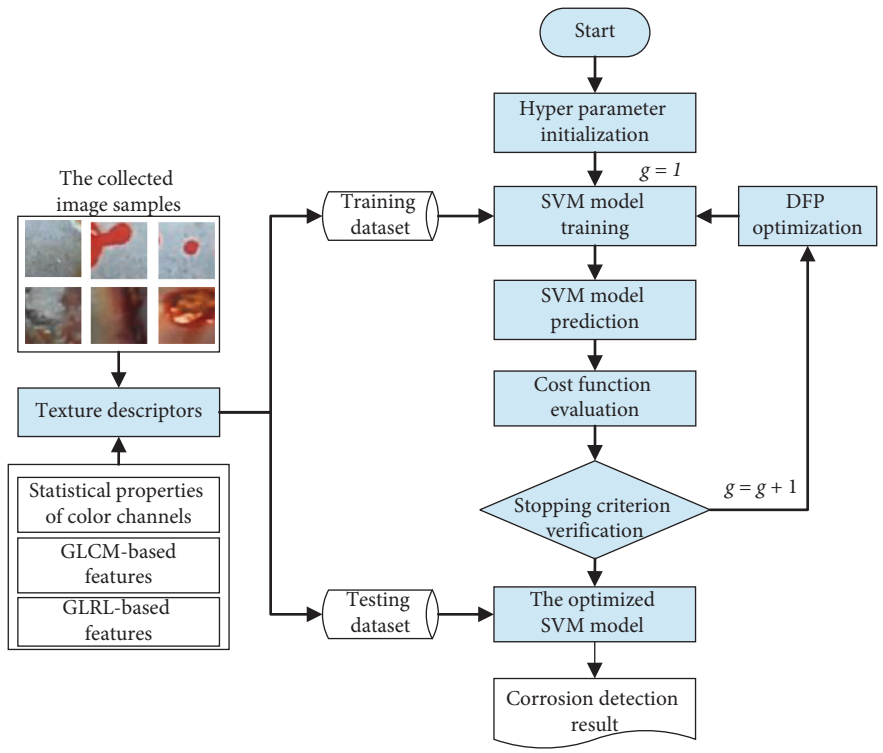


Figure 4: The proposed MO-SVM-PCD model used for pipe corrosion detection.

predictive outcomes of the 5 data folds. This process has been proved to be a robust method for model hyperparameter selection [49]. Notably, in each generation, based on the computed cost function, the location of population members is updated and the stopping criterion is checked to verify whether the current generation number exceeds the allowable value. If the stopping criterion is met, the DFP-based optimization process terminates and the optimized SVM model is ready to predict corrosion status for novel image samples.

3. Experimental Results and Discussion

As stated earlier, the dataset featuring 2000 samples and 78 image texture variables has been separated into the training and testing subset. The training and testing subsets occupy 70% and 30% of the original dataset, respectively. The first subset is used for model training. The second subset is employed for testing the model predictive capability when it predicts corrosion status of novel image samples which has not been encountered in the training subset. Moreover, to reliably assess the model performance and to diminish the randomness caused by the data sampling process, this research work has conducted a random subsampling of the original dataset consisting of 20 runs. In each run, 30% of the data is randomly extracted to constitute the testing subset; the rest of the data is used for model training. Accordingly, the overall model performance is reliably evaluated by averaging prediction results obtained from the repeated data sampling.

In addition to the aforementioned PPV and NPV, the classification accuracy rate (CAR), recall, and F1 score are also used for expressing the model's predictive accuracy. These indices are computed as follows [50]:

$$\text{classification accuracy rate} : \text{CAR} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\%,$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F1 score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \tag{15}$$

where TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative values.

Demonstration of the feature extraction phase which computes image sample texture is provided in Figure 5. Herein, for each image sample, 78 features representing the statistical measurements of image colors, GLCM, and GLRL are attained and used for data classification purpose. In addition, the evolutionary process of the DFP metaheuristic-based SVM model optimization is illustrated in Figure 6 which shows the best and the average cost function values in each generation. The optimal values of the penalty coefficient and the RBF kernel function parameter are found to be 4.30 and 8.86 with the best cost function = 1.08.

The performance of the MO-SVM-PCD in the training and testing phases is reported in Table 2. As shown in this table, the proposed model has attained good predictive accuracy in both phases with CAR >90%. In detail, the MO-SVM-PCD has achieved CAR = 91.17%, PPV = 0.91, recall = 0.92, NPV = 0.92, and F1 score = 0.91 in the testing phase. There is a focus on the MO-SVM-PCD performance in the testing phase because this reflects the generalization capability of the model.

In addition, corrosion detection based on the MO-SVM-PCD for a large-sized image samples can be achieved via a blockwise image separation process. This image separation process is illustrated in Figure 7(a). In this figure, each block corresponds to a sample having the size of 50×50. The classification result for the entire image is carried out by combining the MO-SVM-PCD-based corrosion detection for each image block (see Figure 7(b)). The computational time required to classify one image block is about 4 seconds; therefore, the corrosion detection of the whole large-sized image (800×600 pixels) requires about 768 seconds. It is noted that the detected positive class (corrosion class) samples are highlighted by red squares. As can be seen from this figure, the proposed approach can achieved relatively good classification result. Nevertheless, several positive samples located in the boundary of the corroded area have not been identified correctly.

Furthermore, to better demonstrate the prediction capability of the newly constructed MO-SVM-PCD employed for detecting metal pipe corrosion, its performance has been compared to that of the least squares support vector machine (LSSVM) [51], classification tree (CTree) [52], back-propagation artificial neural network (BPANN) [53], and convolutional neural network (CNN) [54]. The reason for

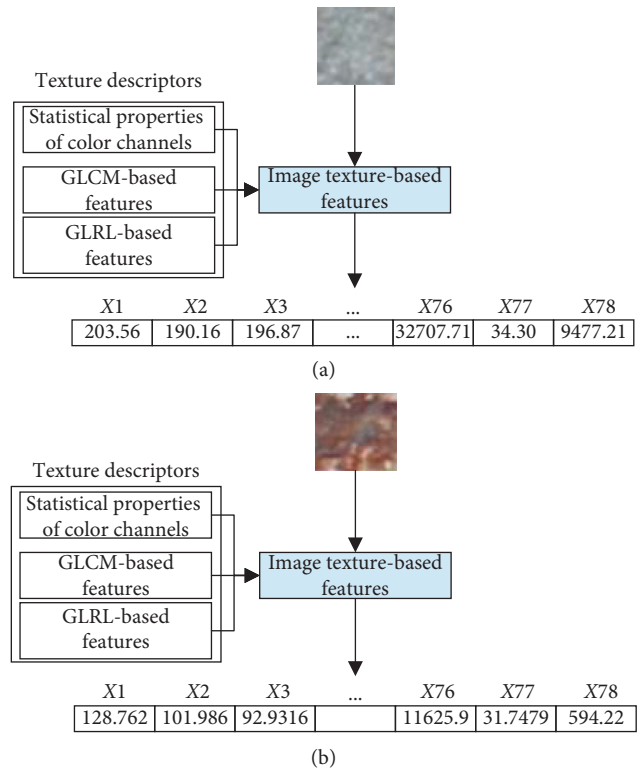


Figure 5: Illustration of the feature extraction process: (a) a noncorrosion case and (b) a corrosion case.

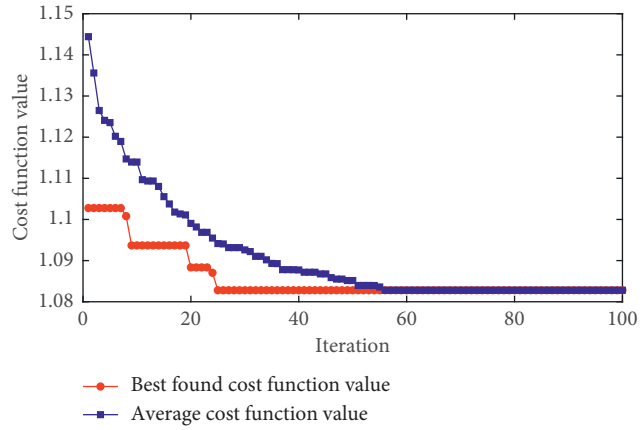


Figure 6: The DFP optimization process.

Table 2: Average prediction performance of the MO-SVM-PCD.

Indices	Training phase	Testing phase
CAR (%)	98.382	92.808
PPV	0.982	0.922
Recall	0.986	0.936
NPV	0.986	0.935
F1 score	0.984	0.929

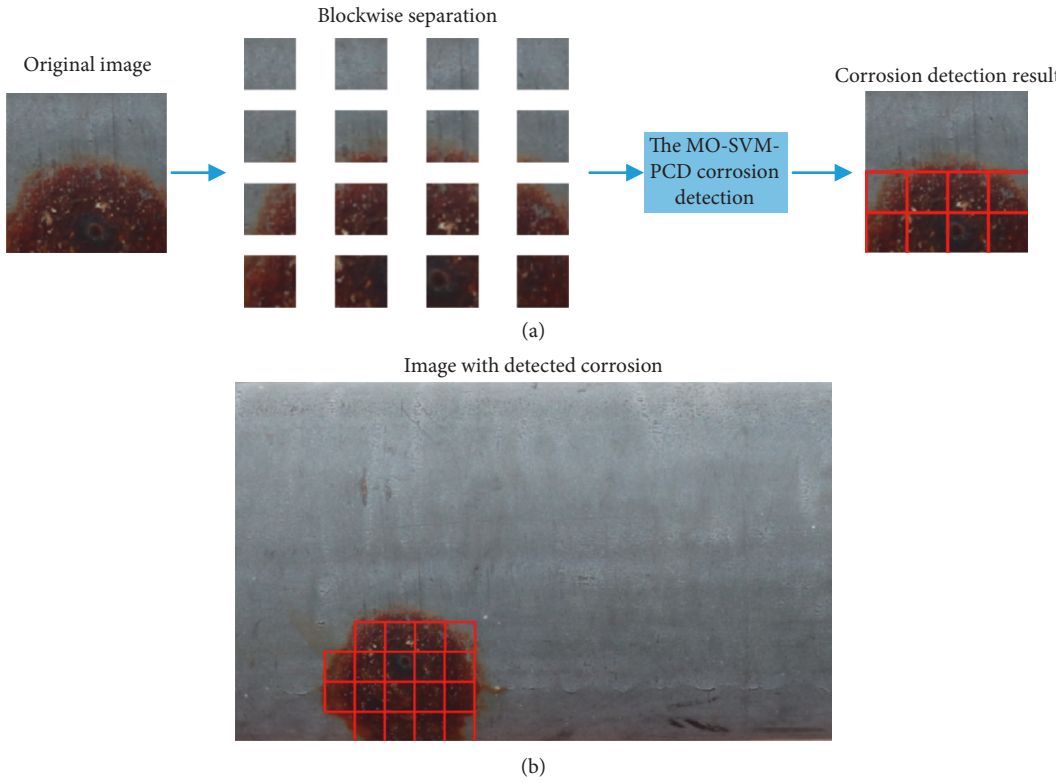


Figure 7: The MO-SVM-PCD-based corrosion detection result: (a) blockwise image separation process and (b) detection outcome with a large-sized image sample.

the selection of these benchmark models is that they have been confirmed to be capable methods for pattern classification by previous studies [5, 40, 55–57].

The LSSVM model is programmed in MATLAB by the authors; its tuning parameters including the regularization coefficient and kernel function parameter are also automatically identified by the DFP metaheuristic. The CTree is developed by the built-in functions provided in the MATLAB Statistics and Machine Learning Toolbox [46]. The BPANN model is programmed in MATLAB environment by the authors. Via experiment, the suitable parameter of minimum leaf size of the employed CTree model has been found to be 2. Based on the suggestions of Heaton [58] and several trial-and-error runs, the number of neuron in the hidden layer of the BPANN model is selected to be $2 \times N_I/3 + N_O = 54$, where $N_I = 78$ is the number of input features and $N_O = 2$ is the number of class labels. Moreover, the learning rate and the number of training epochs of the neural network are set to be 0.1 and 1000, respectively.

In addition, the CNN model employed for corrosion detection is constructed by the MATLAB image processing toolbox [59]; the stochastic gradient descent with momentum (SGDM) and mini-batch mode are used in the model training phase. Via experimental runs, the appropriate configuration of the deep learning method is as follows: input image size is 50×50 pixels. The number of convolution layers is 4. The sizes of the filters are 20×20, 16×16, 8×8, and 4×4 in the 1st, 2nd, 3rd, and 4th convolution layer, respectively. The number of filters in each layer is 36. The batch size is 20% of the training data. In addition, the CNN model has been trained in 1000 epochs. In CNN, the feature extraction phase is automatically performed by convolution layers; therefore, the CNN model does not requires the feature computation done by the three employed image texture descriptors.

The prediction results of all the models obtained from the repeated data sampling with 20 runs are summarized in Table 3 which reports the mean and the standard deviation (Std) of the model performance. Observably, the MO-SVM-PCD has attained the most desired predictive accuracy in terms of CAR, followed by BPANN, LSSVM, CNN, and CTree. The proposed pipe corrosion approach also achieves the highest values of PPV, recall, NPV, and F1 score. The comparison of model performance is graphically displayed in Figure 8.

In addition, the Wilcoxon signed-rank test [60] is utilized in this section to better confirm the statistical significance of the differences in the model performances. This is a nonparametric statistical test commonly employed for model comparison [61]. With the significance level of the test = 0.05, if the p value computed from the Wilcoxon signed-rank test is lower than this significance level, it is able to reject the null hypothesis of insignificant difference in prediction outcomes of the two predictors. Hence, it is confident to conclude that the predictive results of the two pipe corrosion detection models are statistically different. Using the CAR values, the outcome of the Wilcoxon signed-rank tests is reported in Table 4. This test points out that the

Table 3: Corrosion detection result of the machine learning models.

Phase	Indices	MO-SVM-PCD		LSSVM		CTree		BPANN		CNN	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Train	CAR (%)	98.382	0.236	96.432	0.435	97.018	0.532	94.593	1.674	90.890	1.649
	PPV	0.982	0.004	0.937	0.008	0.970	0.006	0.937	0.020	0.891	0.024
	Recall	0.986	0.004	0.996	0.002	0.971	0.008	0.956	0.016	0.933	0.020
	NPV	0.986	0.004	0.995	0.002	0.971	0.007	0.956	0.016	0.930	0.019
	F1	0.984	0.002	0.965	0.004	0.970	0.005	0.947	0.016	0.911	0.016
Test	CAR (%)	92.808	1.094	87.467	1.121	85.825	1.467	88.733	1.721	87.258	1.571
	PPV	0.922	0.015	0.886	0.016	0.854	0.016	0.877	0.022	0.855	0.028
	Recall	0.936	0.017	0.860	0.016	0.864	0.021	0.902	0.026	0.899	0.022
	NPV	0.935	0.016	0.864	0.014	0.863	0.019	0.899	0.024	0.894	0.018
	F1	0.929	0.011	0.873	0.011	0.859	0.015	0.889	0.017	0.876	0.014

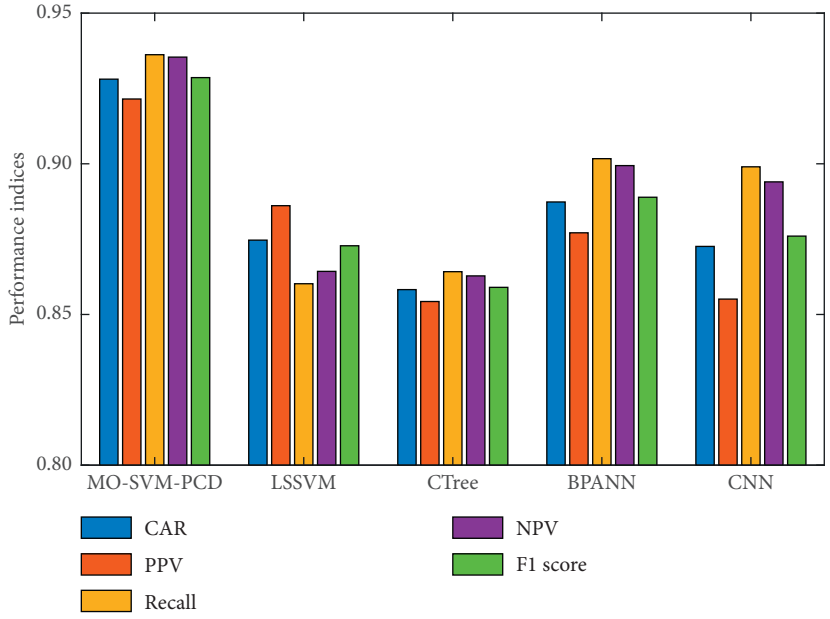


Figure 8: Result comparison.

Table 4: p values obtained from the Wilcoxon signed-rank test.

Models	MO-SVM-PCD	LSSVM	CTree	BPANN	CNN
MO-SVM-PCD	—	0.00009	0.00009	0.00009	0.00009
LSSVM	0.00009	—	0.00427	0.01407	0.48933
CTree	0.00009	0.00427	—	0.00022	0.01185
BPANN	0.00009	0.01407	0.00022	—	0.01954
CNN	0.00009	0.48933	0.01185	0.01954	—

MO-SVM-PCD is statistically better than the LSSVM, CTree, BPANN, and CNN with p values < 0.05. Based on this statistical test, it is able to state that the proposed method is the most suited method for the task of interest.

4. Conclusion

Corrosion is a commonly observed type of pipe defects. Timely detection of corrosion is very crucial to ensure the integrity of the water supply system and avoid water contamination. In addition, information regarding corroded pipe sections obtained during periodic building surveys can significantly help to establish cost-effective maintenance strategies for building owners. This study puts forward an automatic method based on image processing and machine learning for pipe corrosion recognition. Image processing techniques have been employed to extract useful features from images of pipe surface to characterize the corrosion status. In total, 78 features are extracted using three texture descriptors of the statistical properties of image color, GLCM, and GLRL.

The SVM machine learning method integrated with the DFP metaheuristic is utilized to construct a decision boundary used for classifying pipe surface images into two categories of noncorrosion and corrosion. A dataset consisting of 2000 image samples has been used to train and validate the proposed hybrid model of the MO-SVM-PCD. Experimental results supported by the Wilcoxon signed-rank test point out that the newly developed method is superior to other benchmark approaches with an average CAR = 92.81%. Therefore, the newly developed model can be

a useful tool for building maintenance agents to quickly evaluate the status of pipe systems. Further extensions of the current study may include the utilization of other advanced machine learning for data classification, employment of other metaheuristic for model optimization, employment of higher-order statistical features as input to machine learning based classifiers, enhancement of the detection accuracy for image samples located in the boundary of the corroded area, improvement of the computational efficiency of the current method by employing advanced image segmentation techniques, and collection of more image samples to enhance the generalization of the current MO-SVM-PCD model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Supplementary Materials

The supplementary material of this study contains the dataset used to construct the machine learning-based model. The first 78 columns of the dataset are the extracted texture features. The last column is the class label (0 for non-corrosion and 1 for corrosion). (*Supplementary Materials*)

References

[1] E. Fleming, *Construction Technology: An Illustrated Introduction*, Blackwell Publishing Ltd, Hoboken, NJ, USA, 2005.

[2] F. Bonnin-Pascual and A. Ortiz, "Corrosion detection for automated visual inspection, developments in corrosion protection," in *Developments in Corrosion Protection*, pp. 620–632, IntechOpen, London, UK, 2014.

[3] V. Bondada, D. K. Pratihar, and C. S. Kumar, "Detection and quantitative assessment of corrosion on pipelines through image analysis," *Procedia Computer Science*, vol. 133, pp. 804–811, 2018.

[4] L. Liu, E. Tan, Y. Zhen, X. J. Yin, and Z. Q. Cai, "AI-facilitated coating corrosion assessment system for productivity enhancement," in *Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 606–610, Munich, Germany, May 2018.

[5] D. J. Atha and M. R. Jahanshahi, "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection," *Structural Health Monitoring*, vol. 17, no. 5, pp. 1110–1128, 2018.

[6] S. Dorafshan, R. J. Thomas, and M. Maguire, "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete," *Construction and Building Materials*, vol. 186, pp. 1031–1045, 2018.

[7] Y. Jung, H. Oh, and M. M. Jeong, "An approach to automated detection of structural failure using chronological image analysis in temporary structures," *International Journal of Construction Management*, vol. 19, no. 2, pp. 178–185, 2019.

[8] J.-M. Maatta, J. Vanne, T. Hamalainen, and J. Nikkanen, "Generic software framework for a line-buffer-based image processing pipeline," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1442–1449, 2011.

[9] D. Itzhak, I. Dinstein, and T. Zilberberg, "Pitting corrosion evaluation by computer image processing," *Corrosion Science*, vol. 21, no. 1, pp. 17–22, 1981.

[10] K. Y. Choi and S. S. Kim, "Morphological analysis and classification of types of surface corrosion damage by digital image processing," *Corrosion Science*, vol. 47, no. 1, pp. 1–15, 2005.

[11] F. N. S. Medeiros, G. L. B. Ramalho, M. P. Bento, and L. C. L. Medeiros, "On the evaluation of texture and color features for nondestructive corrosion detection," *EURASIP Journal on Advances in Signal Processing*, article 817473, 2010.

[12] G. Ji, Y. Zhu, and Y. Zhang, "The corroded defect rating system of coating material based on computer vision," in *Transactions on Edutainment VIII*, Z. Pan, Ed., pp. 210–220, Springer, Berlin, Germany, 2012.

[13] S. A. Idris and F. A. Jafar, "Image enhancement based on software filter optimization for corrosion inspection," in *Proceedings of the 2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, vol. 27–29, pp. 345–350, Langkawi, Malaysia, January 2014.

[14] H. Son, N. Hwang, C. Kim, and C. Kim, "Rapid and automated determination of rusted surface areas of a steel bridge for robotic maintenance systems," *Automation in Construction*, vol. 42, pp. 13–24, 2014.

[15] K.-W. Liao and Y.-T. Lee, "Detection of rust defects on steel bridge coatings via digital image recognition," *Automation in Construction*, vol. 71, pp. 294–306, 2016.

[16] L. Petricca, T. Moss, G. Figueroa, and S. Broen, "Corrosion detection using A.I.: a comparison of standard computer vision techniques and deep learning model," in *Proceedings of the Sixth International Conference on Computer Science, Engineering and Information Technology*, vol. 91–99, IEEE, Piscataway, NJ, USA, December 2016.

[17] S. Safari and M. A. Shoorehdeli, "Detection and isolation of interior defects based on image processing and neural networks," *HDPE Pipeline Case Study Journal of Pipeline Systems Engineering and Practice*, vol. 9, no. 2, article 05018001, 2018.

[18] N. Cheriet, N. E. Bacha, and A. Skender, "Knowledge base system (KBS) applied on corrosion damage assessment on metallic structure pipes," *Heliyon*, vol. 4, no. 10, article e00865, 2018.

[19] T. Gibbons, G. Pierce, K. Worden, and I. Antoniadou, "A Gaussian mixture model for automated corrosion detection in remanufacturing," in *Advances in Manufacturing Technology XXXII*, vol. 8, pp. 63–68, IOS Press, Amsterdam, Netherlands, 2018.

[20] S. K. Ahuja and M. K. Shukla, "A survey of computer vision based corrosion detection approaches," in *Proceedings of the Information and Communication Technology for Intelligent Systems (ICTIS 2017)*, vol. 2, pp. 55–63, Springer International Publishing, Cham, Switzerland, August 2018.

[21] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, Hoboken, NJ, USA, 1998, ISBN-10: 0471030031.

[22] G. M. Hadjidemetriou, P. A. Vela, and S. E. Christodoulou, "Automated pavement patch detection and quantification using support vector machines," *Journal of Computing in Civil Engineering*, vol. 32, no. 1, article 04017073, 2018.

[23] M. Wang, Y. Wan, Z. Ye, and X. Lai, "Remote sensing image classification based on the optimal support vector machine

and modified binary coded ant colony optimization algorithm," *Information Sciences*, vol. 402, pp. 50–68, 2017.

[24] Y. Zhou, W. Su, L. Ding, H. Luo, and P. E. D. Love, "Predicting safety risks in deep foundation pits in subway infrastructure projects: support vector machine approach," *Journal of Computing in Civil Engineering*, vol. 31, no. 5, article 04017052, 2017.

[25] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, Cambridge, MA, USA, 2009, ISBN 978-1-59749-272-0.

[26] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992, ISBN 0201569434.

[27] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Cengage Learning, Boston, MA, USA, 2013.

[28] F. Tomita and S. Tsuji, *Computer Analysis of Visual Textures*, Springer, Berlin, Germany, 1990.

[29] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[30] M. M. Galloway, "Texture analysis using gray level run lengths," *Computer Graphics and Image Processing*, vol. 4, no. 2, pp. 172–179, 1975.

[31] B. Abraham and M. S. Nair, "Computer-aided classification of prostate cancer grade groups from MRI images using texture features and stacked sparse autoencoder," *Computerized Medical Imaging and Graphics*, vol. 69, pp. 60–68, 2018.

[32] M. R. K. Mookiah, T. Baum, K. Mei et al., "Effect of radiation dose reduction on texture measures of trabecular bone microstructure: an in vitro study," *Journal of Bone and Mineral Metabolism*, vol. 36, no. 3, pp. 323–335, 2018.

[33] T. Xiaoou, "Texture information in run-length matrices," *IEEE Transactions on Image Processing*, vol. 7, no. 11, pp. 1602–1609, 1998.

[34] A. Chu, C. M. Sehgal, and J. F. Greenleaf, "Use of gray value distribution of run lengths for texture analysis," *Pattern Recognition Letters*, vol. 11, no. 6, pp. 415–419, 1990.

[35] B. V. Dasarathy and E. B. Holder, "Image characterizations based on joint gray level—run length distributions," *Pattern Recognition Letters*, vol. 12, no. 8, pp. 497–502, 1991.

[36] L. Hamel, *Knowledge Discovery with Support Vector Machines*, John Wiley & Sons, Hoboken, NJ, USA, 2009, ISBN: 978-0-470-37192-3.

[37] M.-Y. Cheng and N.-D. Hoang, "Typhoon-induced slope collapse assessment using a novel bee colony optimized support vector classifier," *Natural Hazards*, vol. 78, no. 3, pp. 1961–1978, 2015.

[38] D. Niu and S. Dai, "A short-term load forecasting model with a modified particle swarm optimization algorithm and least squares support vector machine based on the denoising method of empirical mode decomposition and grey relational analysis," *Energies*, vol. 10, no. 3, p. 408, 2017.

[39] D. Prayogo and Y. T. T. Susanto, "Optimizing the prediction accuracy of friction capacity of driven piles in cohesive soil using a novel self-tuning least squares support vector machine," *Advances in Civil Engineering*, vol. 2018, Article ID 6490169, 9 pages, 2018.

[40] T. Yi, H. Zheng, Y. Tian, and J.-p. Liu, "Intelligent prediction of transmission line project cost based on least squares support vector machine optimized by particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5458696, 11 pages, 2018.

[41] N.-D. Hoang, D. Tien Bui, and K.-W. Liao, "Groutability estimation of grouting processes with cement grouts using differential flower pollination optimized support vector machine," *Applied Soft Computing*, vol. 45, pp. 173–186, 2016.

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

[43] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, pp. 240–249, Springer, Berlin, Germany, 2012.

[44] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*, Springer-Verlag, Berlin, Germany, 2005.

[45] V.-H. Nhu, N.-D. Hoang, V.-B. Duong, H.-D. Vu, and D. Tien Bui, "A hybrid computational intelligence approach for predicting soil shear strength for urban housing construction: a case study at Vinhomes Imperia project, Hai Phong City (Vietnam)," *Engineering with Computers*, 2019.

[46] MathWork, *Statistics and Machine Learning Toolbox User's Guide*, MathWork Inc., Natick, MA, USA, 2017, https://www.mathworks.com/help/pdf_doc/stats/stats.pdf.

[47] Z. Guo, X. Shao, Y. Xu, H. Miyazaki, W. Ohira, and R. Shibasaki, "Identification of village building via Google earth images and supervised machine learning methods," *Remote Sensing*, vol. 8, no. 4, p. 271, 2016.

[48] D. G. Altman and J. M. Bland, "Statistics notes: diagnostic tests 2: predictive values," *BMJ*, vol. 309, no. 6947, p. 102, 1994.

[49] T. Kawasaki, T. Iwamoto, M. Matsumoto et al., "A method for detecting damage of traffic marks by half celestial camera attached to cars," in *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Coimbra, Portugal, July2015.

[50] N.-D. Hoang and Q.-L. Nguyen, "Metaheuristic optimized edge detection for recognition of concrete wall cracks: a comparative study on the performances of roberts, prewitt, canny, and sobel algorithms," *Advances in Civil Engineering*, vol. 2018, Article ID 7163580, 16 pages, 2018.

[51] J. Suykens, J. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Publishing Co. Pte. Ltd., Singapore, 2002, ISBN-13: 978-9812381514.

[52] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classifcation and Regression Trees*, Wadsworth and Brooks, Montery, CA, USA, 1984, ISBN-13: 978-0412048418.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[54] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[55] K. Khosravi, B. T. Pham, K. Chapi et al., "A comparative assessment of decision trees algorithms for flash flood susceptibility modeling at Haraz watershed, Northern Iran," *Science of the Total Environment*, vol. 627, pp. 744–755, 2018.

[56] P.-T. Ngo, N.-D. Hoang, B. Pradhan et al., "A novel hybrid swarm optimized multilayer neural network for spatial prediction of flash floods in tropical areas using sentinel-1 SAR imagery and geospatial data," *Sensors*, vol. 18, no. 11, p. 3704, 2018.

[57] Z. Tong, J. Gao, A. Sha, L. Hu, and S. Li, "Convolutional neural network for asphalt pavement surface texture analysis," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1056–1072, 2018.

[58] J. Heaton, *Introduction to Neural Networks for C#*, Heaton Research, Inc., St. Louis, MO, USA, 2008.
[59] MathWorks, *Image Processing Toolbox User's Guide*, MathWork Inc., Natick, MA, USA, 2016, https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf.
[60] S. Sidney, *Non-Parametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York, NY, USA, 1988, ISBN 0070573573.
[61] N.-D. Hoang and D. T. Bui, "Predicting earthquake-induced soil liquefaction based on a hybridization of kernel Fisher discriminant analysis and a least squares support vector machine: a multi-dataset study," *Bulletin of Engineering Geology and the Environment*, vol. 77, no. 1, pp. 191–204, 2018.